Harvard Data Science Capstone: Building a Movie Recommendation System

Jason Baird

April 7, 2021

Contents

```
## Loading required package: tidyverse

## -- Attaching packages --------------------------------------- tidyverse 1.3.0 --

## v ggplot2 3.3.2     v purrr   0.3.4
## v tibble  3.0.4     v dplyr   1.0.2
## v tidyr   1.1.2     v stringr 1.4.0
## v readr   1.4.0     v forcats 0.5.0

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

## Loading required package: caret

## Warning: package 'caret' was built under R version 4.0.4

## Loading required package: lattice

##
## Attaching package: 'caret'

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

## Loading required package: data.table

## Warning: package 'data.table' was built under R version 4.0.4

##
## Attaching package: 'data.table'

## The following objects are masked from 'package:dplyr':
##
##     between, first, last

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

## Warning in set.seed(1, sample.kind = "Rounding"): non-uniform 'Rounding' sampler
## used

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

1 Executive Summary

The purpose of this report is to generate a movie recommendation algorithm that predicts movie ratings based on a subset of the MovieLens dataset which contains 10 million rows. Specifically, the algorithm is judged on how well errors are reduced, and must have a root mean squared error (RMSE) less than 0.86490 to be considered high quality.

In short, I used a **regularized least squares model** that produced an RMSE equal to **0.8626956**. This model leveraged two variables and a tuning parameter: 1) **movieId**, 2) **userId**, 3) **lambda** ¬– a tuning parameter that penalizes large predictions based on a small sample size. The **regularized least square formula** is:

$$\frac{1}{n}\sum_{u,i}(y_{u,i} - \mu - beta_i - beta_u)^2 + \lambda(\sum_i beta_i^2 + \sum_u beta_u^2)$$

Throughout this paper, I will go into more detail about the objectives of the algorithm, formulas used, initial data analysis, exploratory analysis, feature selection process, and tuning parameters used to optimize algorithm performance.

2 Project Outline

2.1 Objective:

Build a movie recommendation algorithm that predicts movie ratings. The model is to be judged on how well it reduces errors. Specifically, a high quality model will have an RMSE less than 0.86490.

$$\text{RMSE} = \sqrt{\frac{1}{n}\sum_{t=1}^{n} e_t^2}$$

2.2 Least Squares Regression Formula

The following least squares regression formula provides the predicted value per category and the residual error we need to minimize from our key metric (loss function).

$$Y_{u,i} = \mu + beta_i + ...beta_n + \epsilon_{u,i}$$

Here are the definitions for the variables in the equation

$Y_{u,i}$ - the predicted rating for each user and film combination

μ - the true rating of each movie (approximated by the mean rating of all movies)

$beta_i$ - the average difference between actual rating and true rating for category combination one.

$beta_n$ – the average difference between the actual ratings and true rating for category combination one.

$\epsilon_{u,i}$ - the independent errors sampled from the same distribution centered at 0

2.3 Feature Selection Process

Since we do not know the true impact each variable of interest will have on our algorithm, we will use **forward selection** to select variables in our model.

Forward selection starts with no selected variables. During subsequent steps, it evaluates if each candidate variable improves RMSE given previously selected variables, and adds the variable that improves the criterion most.

3 Initial Exploratory Data Analysis

3.1 Importing the Data: MovieLens Dataset

For this project, we are using a subset of the MovieLens Data. The data has been split into a training set (edx) and a test set (validation). The code was imported at the beginning of this report.

3.2 Understanding the Data

How many rows and columns in the edx and validation datasets?

```
## [1] 9000055       6
```

```
## [1] 999999      6
```

Take a look at the first 10 rows of each data set.

```
##     userId movieId rating timestamp                                    title
##  1:      1     122      5 838985046                           Boomerang (1992)
##  2:      1     185      5 838983525                            Net, The (1995)
##  3:      1     292      5 838983421                            Outbreak (1995)
##  4:      1     316      5 838983392                            Stargate (1994)
##  5:      1     329      5 838983392             Star Trek: Generations (1994)
##  6:      1     355      5 838984474                   Flintstones, The (1994)
##  7:      1     356      5 838983653                       Forrest Gump (1994)
##  8:      1     362      5 838984885                   Jungle Book, The (1994)
##  9:      1     364      5 838983707                     Lion King, The (1994)
## 10:      1     370      5 838984596 Naked Gun 33 1/3: The Final Insult (1994)
##                                       genres
##  1:                            Comedy|Romance
##  2:                      Action|Crime|Thriller
##  3:                Action|Drama|Sci-Fi|Thriller
##  4:                     Action|Adventure|Sci-Fi
##  5:                Action|Adventure|Drama|Sci-Fi
##  6:                     Children|Comedy|Fantasy
##  7:                    Comedy|Drama|Romance|War
##  8:                  Adventure|Children|Romance
##  9: Adventure|Animation|Children|Drama|Musical
## 10:                              Action|Comedy
```

```
##     userId movieId rating  timestamp
##  1:      1     231    5.0  838983392
##  2:      1     480    5.0  838983653
##  3:      1     586    5.0  838984068
##  4:      2     151    3.0  868246450
##  5:      2     858    2.0  868245645
##  6:      2    1544    3.0  868245920
##  7:      3     590    3.5 1136075494
##  8:      3    4995    4.5 1133571200
##  9:      4      34    5.0  844416936
## 10:      4     432    3.0  844417070
##                                                    title
##  1:                                     Dumb & Dumber (1994)
##  2:                                     Jurassic Park (1993)
##  3:                                        Home Alone (1990)
##  4:                                           Rob Roy (1995)
##  5:                                    Godfather, The (1972)
##  6: Lost World: Jurassic Park, The (Jurassic Park 2) (1997)
##  7:                                 Dances with Wolves (1990)
##  8:                                  Beautiful Mind, A (2001)
##  9:                                              Babe (1995)
## 10:     City Slickers II: The Legend of Curly's Gold (1994)
##                                       genres
##  1:                                   Comedy
##  2:        Action|Adventure|Sci-Fi|Thriller
##  3:                         Children|Comedy
##  4:                 Action|Drama|Romance|War
##  5:                             Crime|Drama
##  6: Action|Adventure|Horror|Sci-Fi|Thriller
##  7:                  Adventure|Drama|Western
##  8:                    Drama|Mystery|Romance
##  9:            Children|Comedy|Drama|Fantasy
## 10:                 Adventure|Comedy|Western
```

How many users and how many movies are in the edx dataset?

```
##   userids movieids
## 1   69878    10677
```

Check for missing data.

```
sum(complete.cases(edx))
```

```
## [1] 9000055
```

```
sum(complete.cases(validation))
```

```
## [1] 999999
```

Determine the class and type of data in the edx dataset

```
## Rows: 9,000,055
## Columns: 6
## $ userId    <int> 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, ...
## $ movieId   <dbl> 122, 185, 292, 316, 329, 355, 356, 362, 364, 370, 377, 42...
## $ rating    <dbl> 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, ...
## $ timestamp <int> 838985046, 838983525, 838983421, 838983392, 838983392, 83...
## $ title     <chr> "Boomerang (1992)", "Net, The (1995)", "Outbreak (1995)",...
## $ genres    <chr> "Comedy|Romance", "Action|Crime|Thriller", "Action|Drama|...
```

4 Preprocessing Data to further Exploratory Data Analysis

Convert review timestamp to date and time for both test and validation set.

```
edx$date <- as.POSIXct(edx$timestamp, origin="1970-01-01")
validation$date <- as.POSIXct(validation$timestamp, origin="1970-01-01")
```

Grab year, month, and hour of day data from review timestamp which was converted to date and time variables

```
edx$rating_year <- as.numeric(format(edx$date, "%Y"))
edx$rating_month <- as.numeric(format(edx$date, "%m"))
edx$rating_hour <- as.numeric(format(edx$date, "%H"))
validation$rating_year <- as.numeric(format(validation$date, "%Y"))
validation$rating_month <- as.numeric(format(validation$date, "%m"))
validation$rating_hour <- as.numeric(format(validation$date, "%H"))
```

Extract the movie's release year from the title for both

```
edx <- edx %>%
  mutate(title = str_trim(title)) %>%
  extract(title,
          into = c("Title", "movie_release"),
          regex = "^(.*) \\(([0-9]{4})\\)$",
          remove = FALSE ) %>%
  mutate(Title = str_trim(Title, side = "both")) %>%
  select(-title)

validation <- validation %>%
  mutate(title = str_trim(title)) %>%
  extract(title,
          into = c("Title", "movie_release"),
          regex = "^(.*) \\(([0-9]{4})\\)$",
          remove = FALSE ) %>%
  mutate(Title = str_trim(Title, side = "both")) %>%
  select(-title)
```

Make sure that the previous separation did not create whitespace in the movie name and title.

```
edx$Title <- str_trim(edx$Title, side = "both")
edx$movie_release <- str_trim(edx$movie_release, side = "both")
validation$Title <- str_trim(validation$Title, side = "both")
validation$movie_release <- str_trim(validation$movie_release, side = "both")
```

Convert movie release date to numeric vector

```
edx$movie_release <- as.numeric(edx$movie_release)
validation$movie_release <- as.numeric(validation$movie_release)
```

Create a new variable indicating how many years between the release of the movie and the movie's review. In the process of feature selection, it is important to implement creative ways that may explain the variability in user ratings. For this variable, I assume that ratings may change based on the length of time between the user watching the movie and when the movie was released. As film graphics and social norms change, the enjoyment of a movie may change.

```
edx <- edx %>%
     mutate(review_minus_release_year = rating_year - movie_release)
validation <- validation %>%
     mutate(review_minus_release_year = rating_year - movie_release)
```

Separate genre rows based on the vertical bar that splits genres. Notice that this preprocessing step dramatically increased the number of rows in our train and test dataset.

```
edx <- edx %>%
       separate_rows(genres,
                     sep = "\\|")

validation <- validation %>%
  separate_rows(genres,
                sep = "\\|")
```

The previous code dramatically increased the number of rows in both the training and test datasets. Lets take a look at how many additional rows were created.

```
## [1] 23371423       12
```

```
## [1] 2595771      12
```

Select data to move forward with

```
edx <- edx %>%
  select(userId, movieId, rating, Title, movie_release, genres, rating_year, rating_month, rating_hour,

validation <- validation %>%
  select(userId, movieId, rating, Title, movie_release, genres, rating_year, rating_month, rating_hour,
```

5 Advanced Exploratory Data Analysis

5.1 Users Effect on Rating

How many users?

```
## [1] 69878
```

What are the median and mean average number of ratings for all users?

```
## [1] 334.4604
```

```
## [1] 164
```

How does the avg rating vary per user?

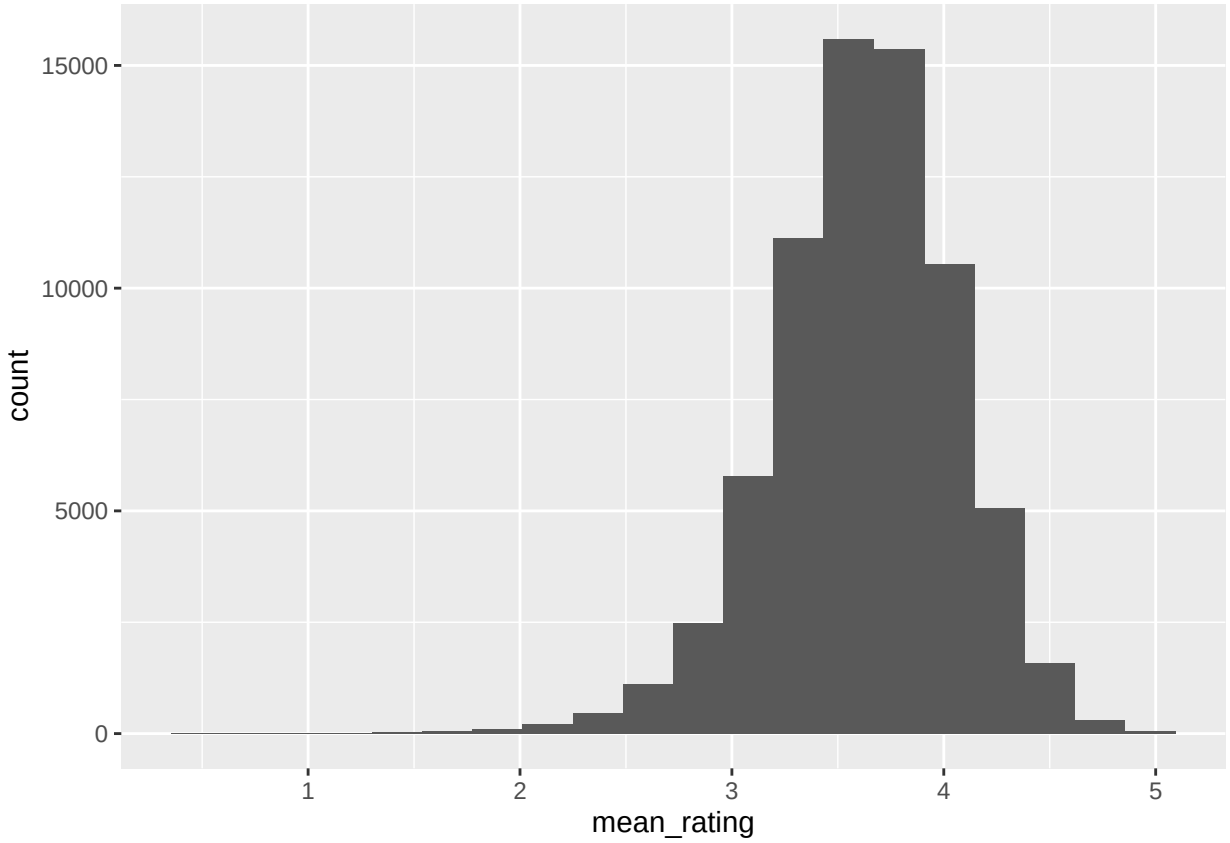

Figure 1: Avg Rating Per User

What is the average rating for the top 35 users in the database?

Mean Rating for Top Users - Table:

```
## # A tibble: 35 x 3
##    userId avg_rating total_ratings
##     <int>      <dbl>         <int>
##  1  59269       3.29         13496
##  2  67385       3.21         13360
##  3  14463       2.46          9121
##  4  27468       3.85          8953
##  5   3817       3.13          8637
##  6  68259       3.57          8519
##  7  19635       3.51          8221
##  8  58357       3.07          7649
##  9  63134       3.32          7521
```

```
## 10   6757       2.82          7087
## # ... with 25 more rows
```

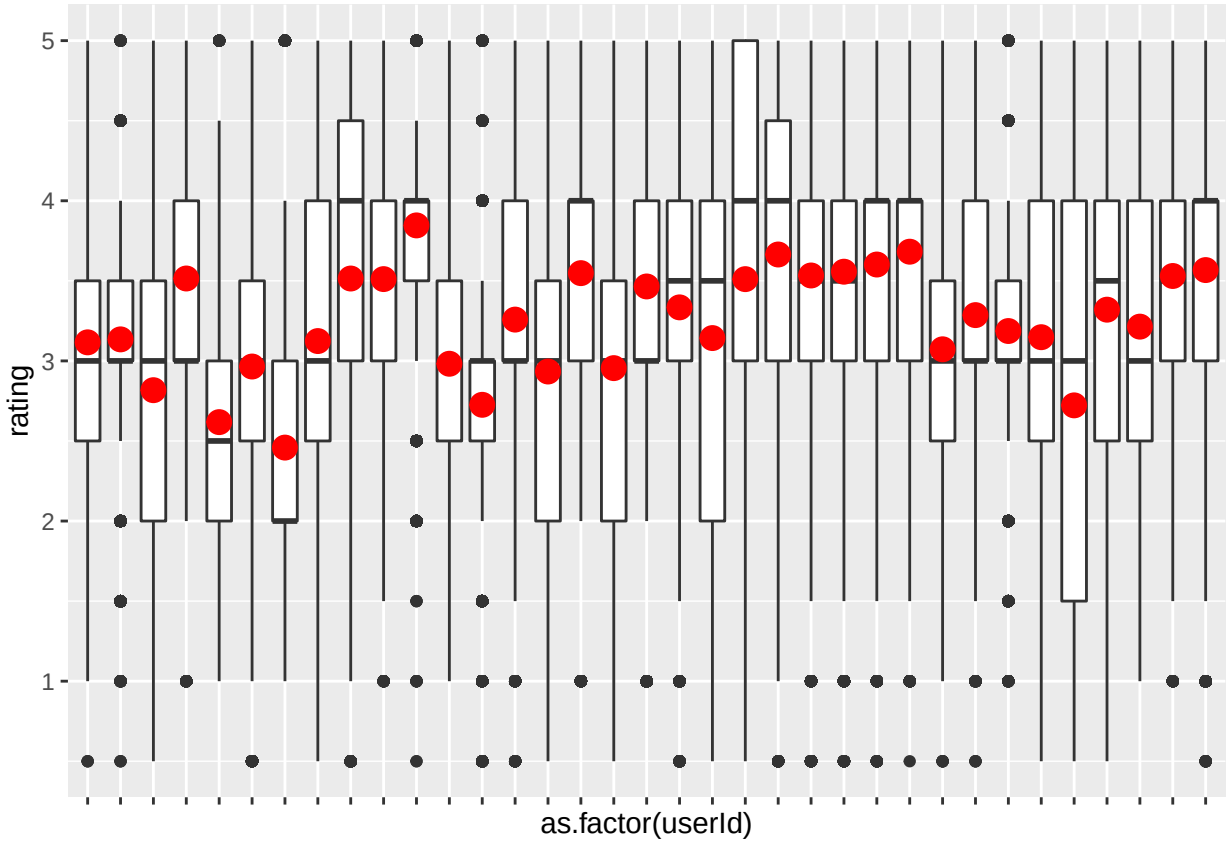

Figure 2: Mean and Median Ratings for Top Users

How do the median and mean rating per top 35 user vary across each user? In this section, we learn a lot of things about the variability associated with each user's rating. For example, we find that the mean value is a more accurate representation of a user's central tendency because it is much more inclusive of when a user has high and low ratings. Meanwhile, median value is not inclusive of those extreme ratings. Additionally, the variance for each user is substantial and person specific.

Lets see if this variance holds for users with a more central number of ratings.

Identify 30 users around the median number of ratings per user

30 users around the median number of ratings - Table:

```
## # A tibble: 30 x 3
##    userId avg_rating total_ratings
##     <int>      <dbl>         <int>
##  1   2815       3.70           165
##  2   3959       3.50           165
##  3   4186       3.91           165
##  4   5605       3.58           165
##  5   5826       3.85           165
##  6   6818       3.43           165
##  7   6870       3.33           165
```

```
##  8   7017       3.73           165
##  9   7761       3.79           165
## 10   8498       4.31           165
## # ... with 20 more rows
```

How do the median and mean ratings for these 30 users vary across each user? As you can see, the variability is consistent across this group of users as well.

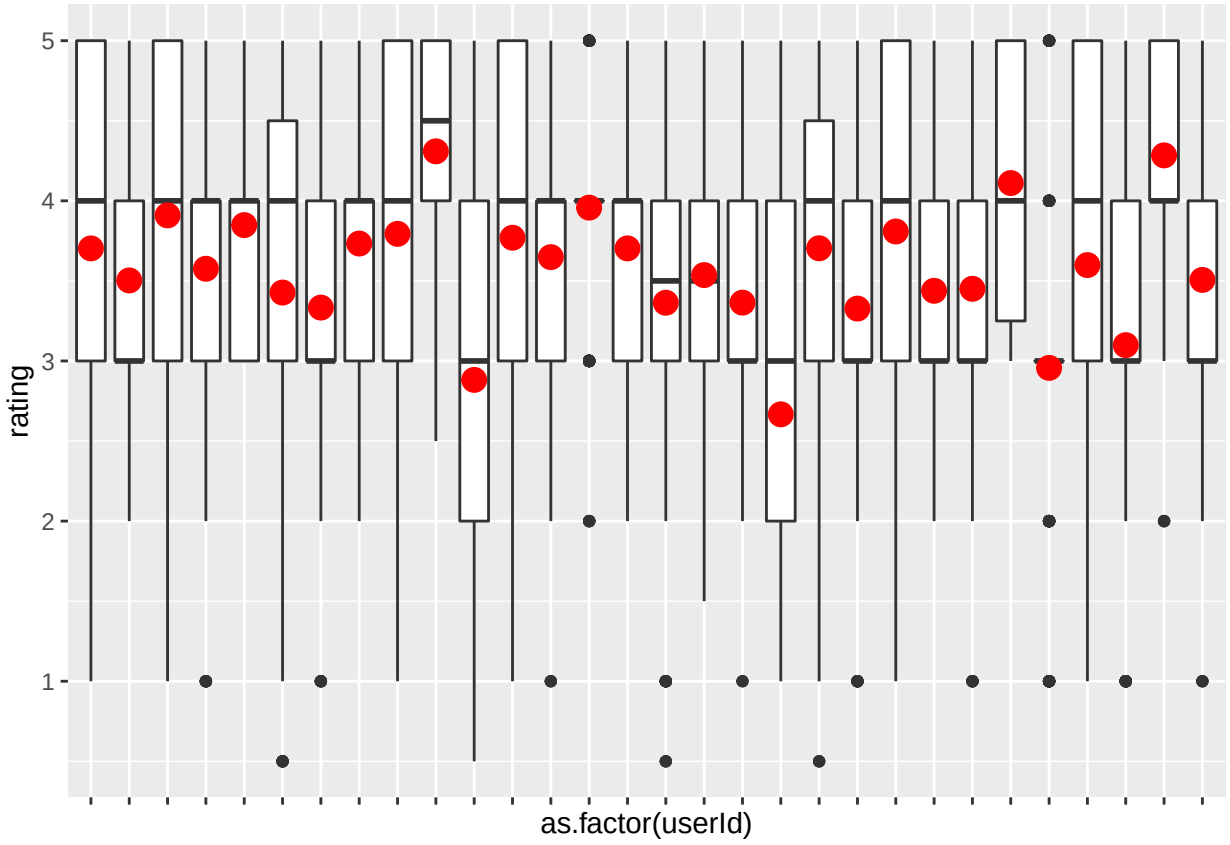

Figure 3: Mean and Median Ratings for 30 users with more common number of ratings

5.2 Movie Effect on Rating

How many movies were rated?

```
## [1] 10677
```

How many ratings per movie?

What are the top 35 most frequently reviewed movies?

Top 35 Movies - Table:

```
## # A tibble: 35 x 3
##    movieId avg_rating total_ratings
##      <dbl>      <dbl>         <int>
```

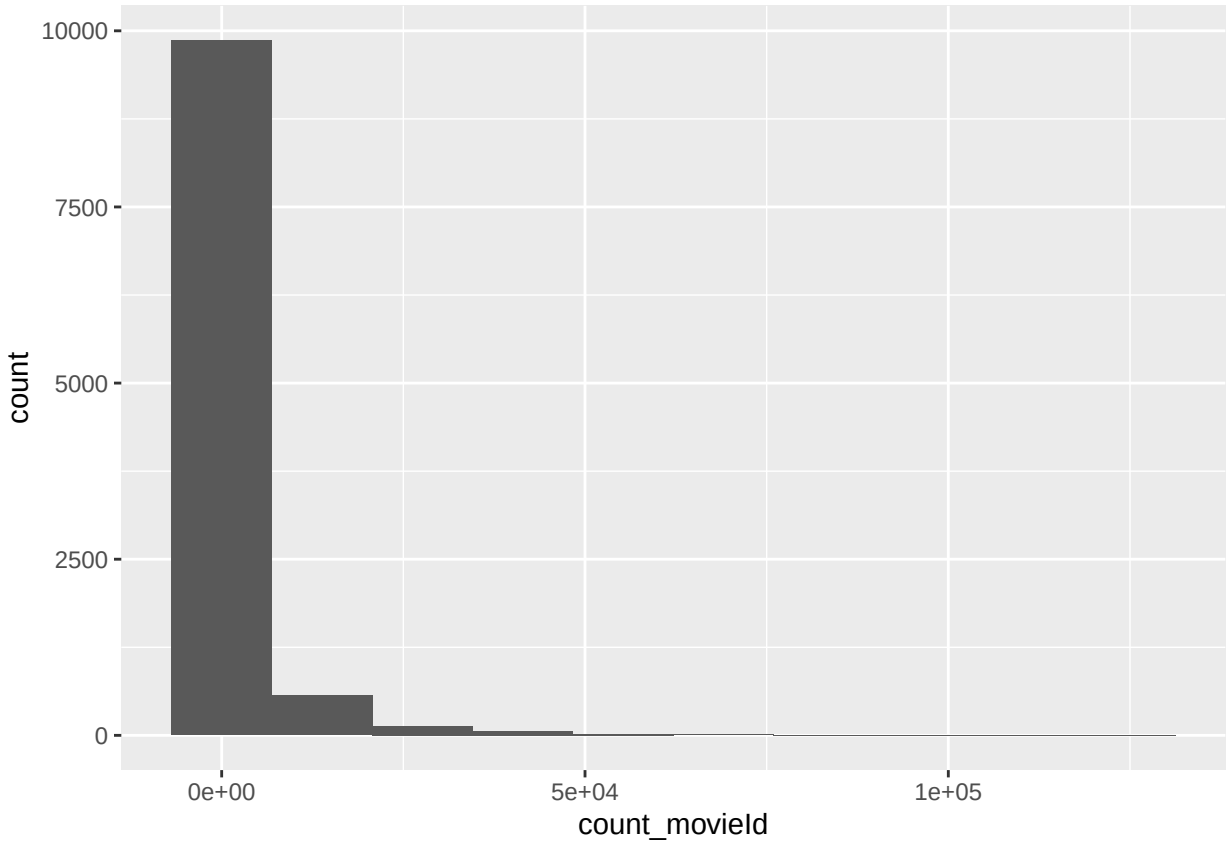

Figure 4: Number of Reviews per Movie

```
##  1     356       4.01        124316
##  2       1       3.93        118950
##  3     480       3.66        117440
##  4     380       3.50        114115
##  5     588       3.67        105865
##  6     592       3.39         97108
##  7     364       3.75         94605
##  8     296       4.15         94086
##  9     780       3.38         93796
## 10     593       4.20         91146
## # ... with 25 more rows
```

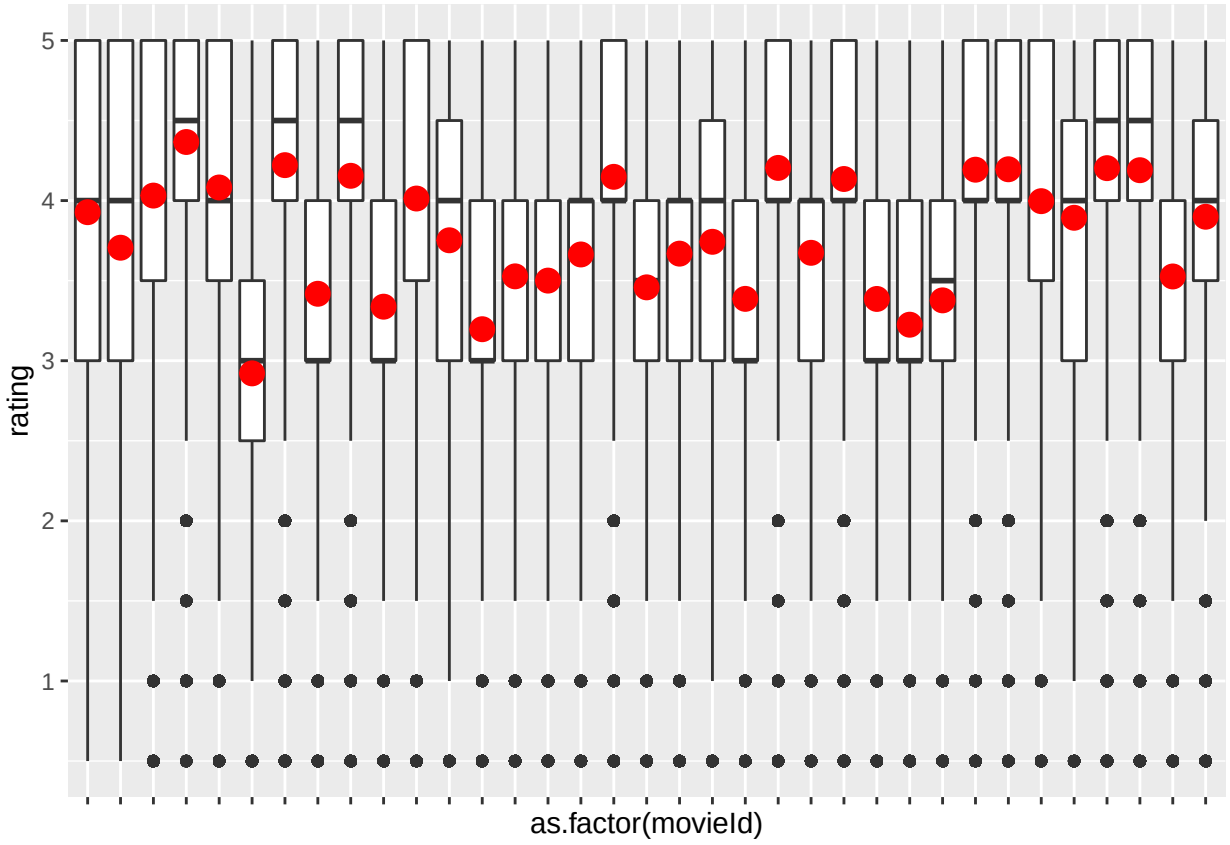

Figure 5: Mean and Median Ratings for the Top 35 Movies

How do the median and mean rating for top 35 most frequently reviewed movies vary across each film? Once again, the film ID/Title showcases a significant amount of variability depending on the movie.

Lets see if this variance holds for movies with a more central number of ratings

What is the median number of ratings per movie?

```
## [1] 204
```

Identify 30 movies around the median number of ratings per movie and check out their average rating.

30 Movies with the median number of ratings - Table

```
## # A tibble: 31 x 3
##    movieId avg_rating total_ratings
##      <dbl>      <dbl>         <int>
##  1     634       2.33           206
##  2    1055       2.71           206
##  3    4429       3.62           206
##  4    5482       1.83           206
##  5    6067       3.32           206
##  6    7493       3.83           206
##  7   39427       3.39           206
##  8   40412       3.74           206
##  9   53974       2.61           206
## 10    3109       3.10           205
## # ... with 21 more rows
```

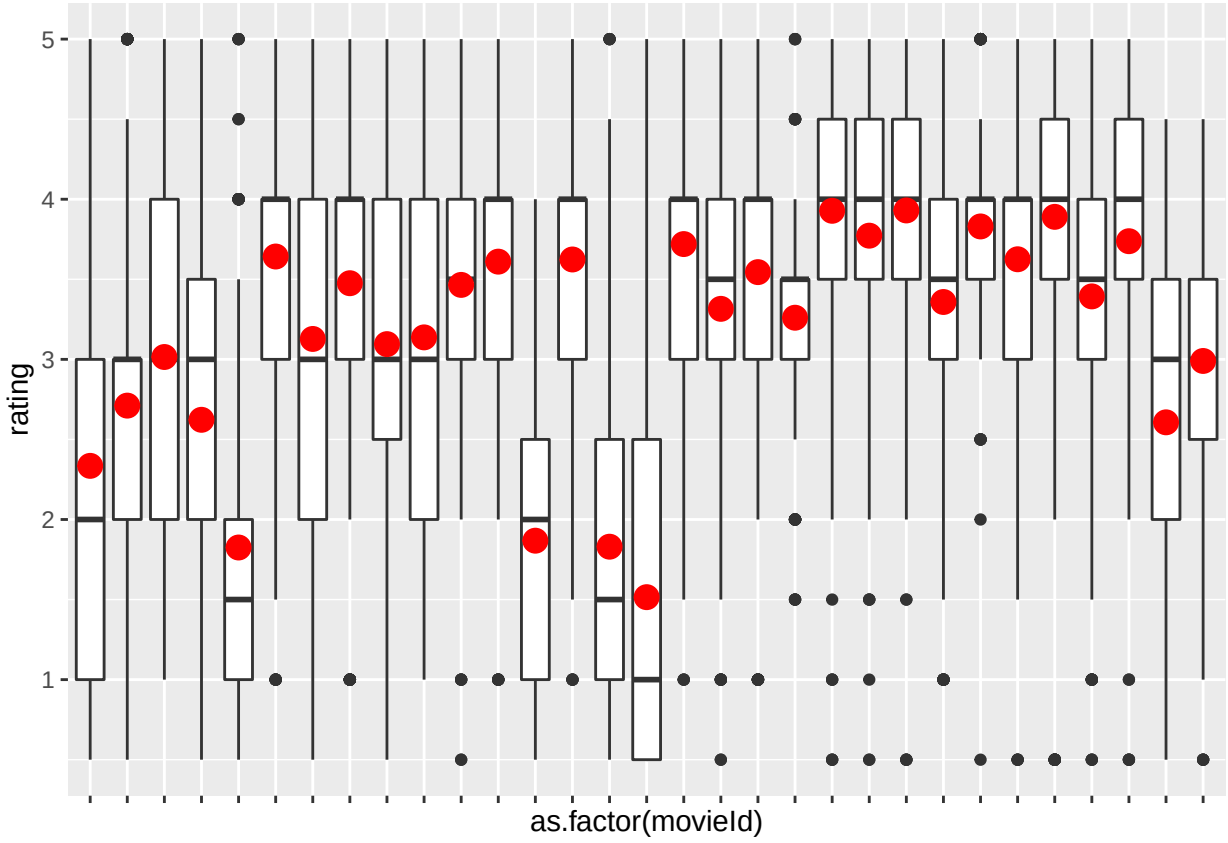

Figure 6: Median and Mean Ratings for 30 Movies

As you can see, the variability is consistent with the findings for the top 35 movies.

5.3 Genre Effect on Rating

How many unique genres in the dataset?

```
## [1] 20
```

How many movie ratings per genre? As you can see, the most frequently rated movies are considered dramas, comedies, and/or action movies.

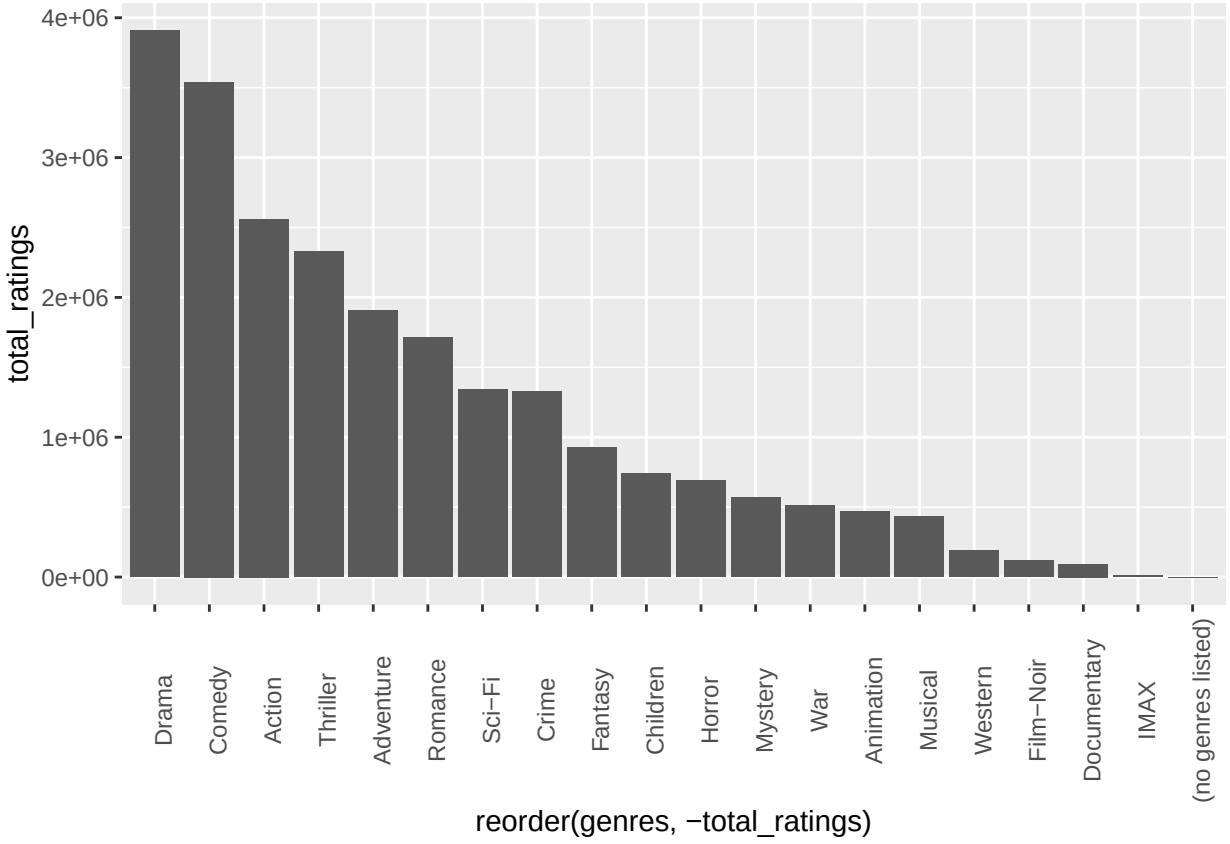

Figure 7: Which Genres Had the most Ratings?

How are the average ratings by genre distributed?

How do the median and mean ratings vary across each movie genre?

5.4 Time Difference Between Rating and Review Effect on Rating

How many unique years between a review and the release of a movie?

```
## [1] 96
```

What is the longest amount of time (in years) between a review and the release of a movie?

```
## [1] 93
```

What is the shortest amount of time between a review and the release of a movie? It is a little odd that some movies were reviewed prior to the release of the movie.

```
## [1] -2
```

What is the median amount of time between a movie review and the movie release?

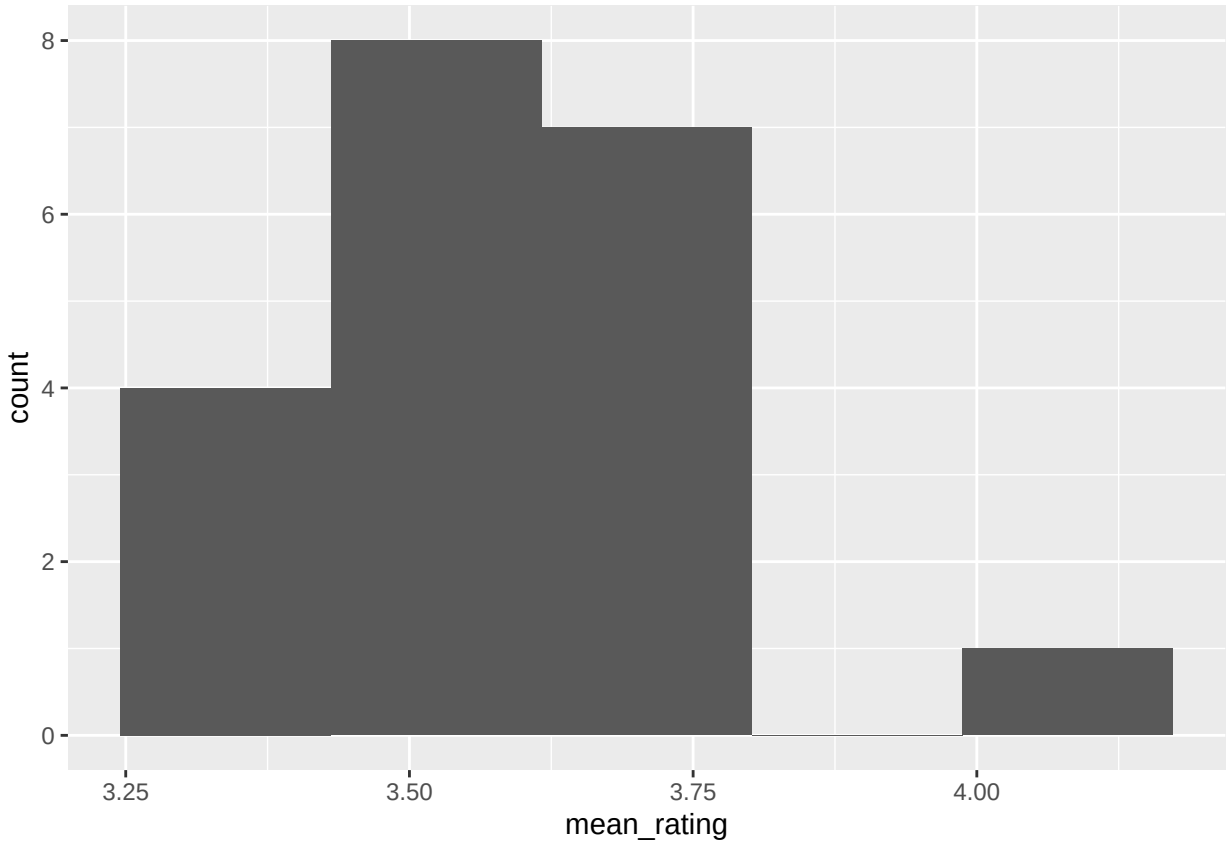

Figure 8: Average Rating By Genre

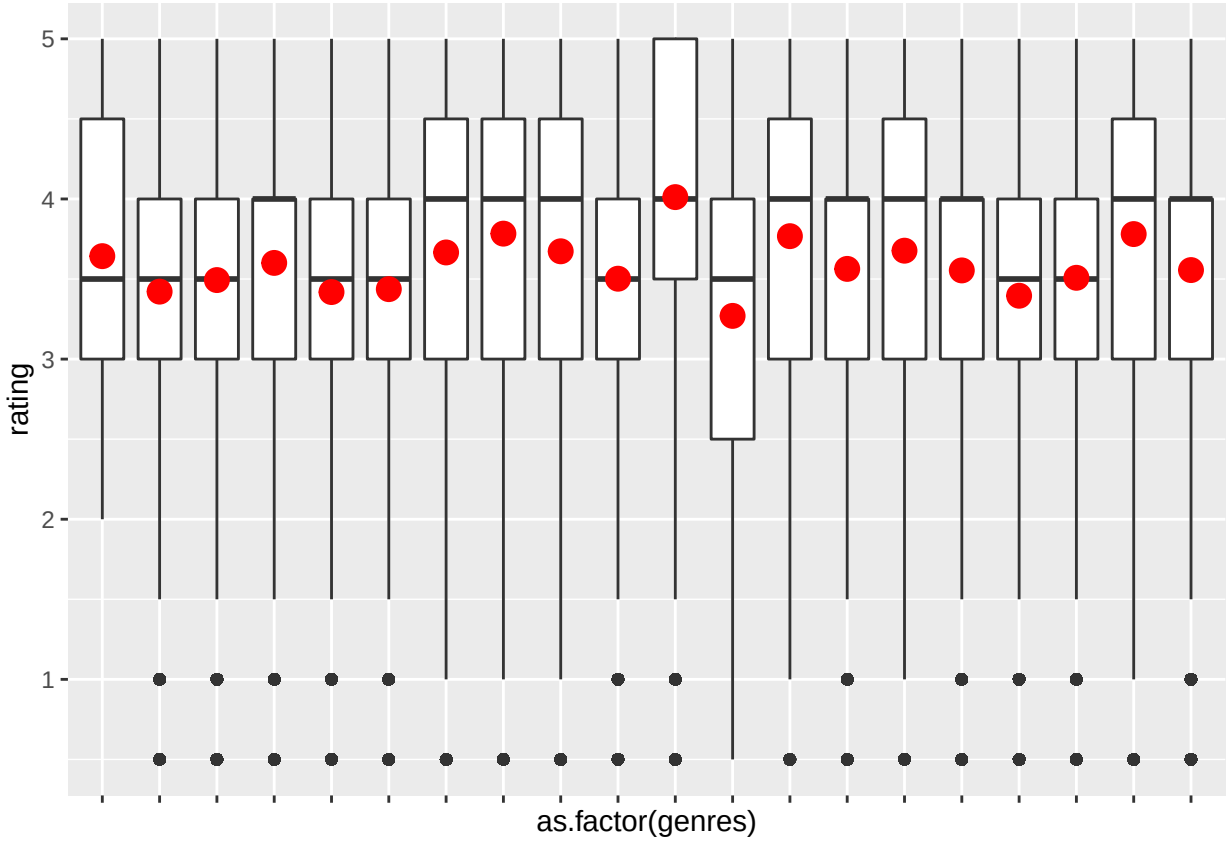

Figure 9: Mean and Median Rating per Genre

```
## [1] 7
```

What are the top 15 years of difference between the review and movie release based on the number of ratings per year?

Are older or newer movies being rated more frequently - Table

```
## # A tibble: 15 x 3
##    review_minus_release_year avg_rating total_ratings
##                        <dbl>      <dbl>         <int>
##  1                         1       3.50       2784621
##  2                         2       3.50       2213834
##  3                         3       3.48       1636497
##  4                         4       3.48       1270263
##  5                         5       3.50       1166893
##  6                         6       3.48       1077322
##  7                         0       3.46       1011978
##  8                         7       3.44        943015
##  9                         8       3.42        834399
## 10                         9       3.41        776558
## 11                        10       3.39        756306
## 12                        11       3.41        701744
## 13                        12       3.41        646390
## 14                        13       3.45        619377
## 15                        14       3.47        571051
```

Visualize how the median and mean rating changes for the top 15 years difference.

6 Algorithm: Reducing RMSE on the Validation Set

6.1 Define RMSE

The following code defines our loss function, RMSE, which is the square root of the mean squared errors.

```
RMSE <- function(true_ratings = NULL, predicted_ratings = NULL) {
  sqrt(mean((true_ratings - predicted_ratings)^2))
}
```

6.2 Naïve Model

The naïve model is the most basic algorithm we can define. It is defined as the mean value of all movie ratings from the training dataset.

```
mu_hat <- mean(edx$rating)
rmse_mean_model <- RMSE(validation$rating, mu_hat)
rmse_mean_model
```

```
## [1] 1.052558
```

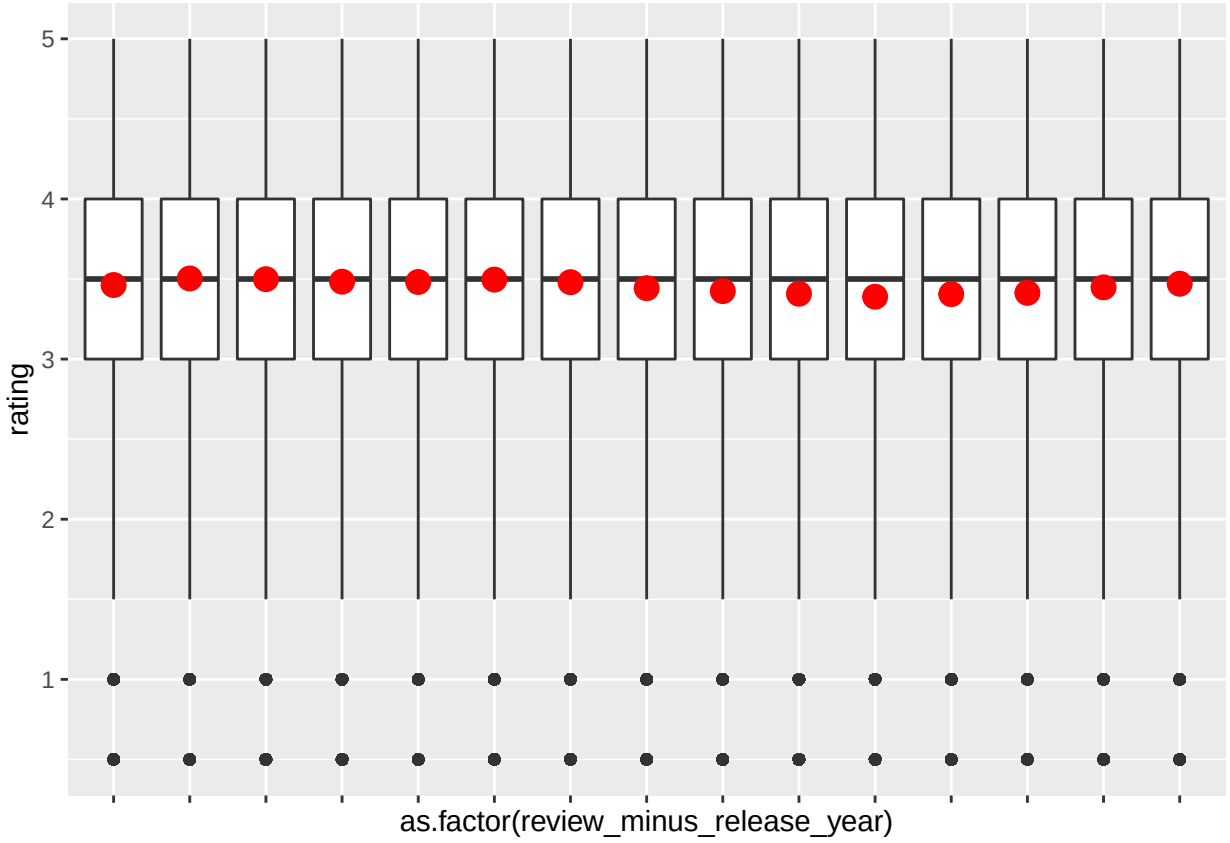

Figure 10: How do the Mean and Median Ratings Vary Among Older and Newer Movies

6.3 Storing Results

I've created two data frames to store the results of our models. The first, results_df contains the RMSE for each variable tested on our training dataset. The second, results_final, contains the variables that improve RMSE the most for each additional Beta included in our model.

7 Forward Selection: Assessing Beta 1

For each of the following variables, I will be assessing the category's grouped effect on the mean prediction. The variable contributing to the lowest RMSE will be locked in for Beta 1.

7.1 UserID's impact on RMSE

```
avg_user_rating <- edx %>%
  group_by(userId) %>%
  summarize(b1 = mean(rating - mu_hat))

rmse_mean_user_model <- validation %>%
  left_join(avg_user_rating, by = "userId") %>%
  mutate(y_hat = mu_hat + b1) %>%
  select(y_hat)

rmse_mean_user_model_result <- RMSE(validation$rating, rmse_mean_user_model$y_hat)

results_df <- results_df %>% add_row(model_name = "mean_userId", RMSE = rmse_mean_user_model_result, be
```

7.2 MovieID's impact on RMSE

```
avg_movie_rating <- edx %>%
  group_by(movieId) %>%
  summarize(b1 = mean(rating - mu_hat))

rmse_mean_movie_model <- validation %>%
  left_join(avg_movie_rating, by = "movieId") %>%
  mutate(y_hat = mu_hat + b1) %>%
  select(y_hat)

rmse_mean_movie_model_result <- RMSE(validation$rating, rmse_mean_movie_model$y_hat)
results_df <- results_df %>% add_row(model_name = "mean_movieId", RMSE = rmse_mean_movie_model_result, 
```

7.3 Genre's impact on RMSE

```
avg_genre_rating <- edx %>%
  group_by(genres) %>%
  summarize(b1 = mean(rating - mu_hat))

rmse_mean_genre_model <- validation %>%
  left_join(avg_genre_rating, by = "genres") %>%
  mutate(y_hat = mu_hat + b1) %>%
  select(y_hat)
```

```
rmse_mean_genres_model_result <- RMSE(validation$rating, rmse_mean_genre_model$y_hat)
results_df <- results_df %>% add_row(model_name = "mean_genre", RMSE = rmse_mean_genres_model_result, be
```

7.4 Film Release Date vs Rating Date's impact on RMSE

```
avg_releasereview_rating <- edx %>%
  group_by(review_minus_release_year) %>%
  summarize(b1 = mean(rating - mu_hat))

rmse_mean_releasereview_model <- validation %>%
  left_join(avg_releasereview_rating, by = "review_minus_release_year") %>%
  mutate(y_hat = mu_hat + b1) %>%
  select(y_hat)

rmse_mean_releasereview_model_result <- RMSE(validation$rating, rmse_mean_releasereview_model$y_hat)
results_df <- results_df %>% add_row(model_name = "mean_releasereview", RMSE = rmse_mean_releasereview_r
```

7.5 Evaluating Results of Beta 1

```
##           model_name      RMSE beta_hat
## 1         Mean Model 1.0525579        0
## 2        mean_userId 0.9729599        1
## 3       mean_movieId 0.9410700        1
## 4         mean_genre 1.0457714        1
## 5 mean_releasereview 1.0446523        1
```

The results stored in results_df for Beta 1 clearly show that movieId reduces RMSE more than any other variable. Our RMSE is now 0.94107, which is well above our threshold for a high-quality movie recommendation model. Therefore, lets lock in movieId as the Beta 1 variable and select another contributing variable.

```
avg_movie_rating <- edx %>%
  group_by(movieId) %>%
  summarize(b1 = mean(rating - mu_hat))

rmse_mean_movie_model <- validation %>%
  left_join(avg_movie_rating, by = "movieId") %>%
  mutate(y_hat = mu_hat + b1) %>%
  select(y_hat)

rmse_mean_movie_model_result <- RMSE(validation$rating, rmse_mean_movie_model$y_hat)

results_final <- results_final %>% add_row(model_name = "mean_movie", RMSE = rmse_mean_movie_model_resul

results_final
```

```
##   model_name     RMSE beta_hat
## 1 Mean Model 1.052558        0
## 2 mean_movie 0.941070        1
```

8 Forward Selection: Assessing Beta 2

Now that our model consists of the average rating and movieId, so we will evaluate the three remaining variables effect on RMSE. The model will include the variable that most dramatically improves RMSE.

8.1 UserID's impact on RMSE

```
avg_user_rating <- edx %>%
  left_join(avg_movie_rating, by = "movieId") %>%
  group_by(userId) %>%
  summarize(b2 = mean(rating - mu_hat - b1))

rmse_mean_movie_user_model_pred <- validation %>%
  left_join(avg_movie_rating, by = "movieId") %>%
  left_join(avg_user_rating, by = "userId") %>%
  mutate(y_hat = mu_hat + b1 + b2) %>%
  select(y_hat)

rmse_mean_movie_user_model_result <- RMSE(validation$rating, rmse_mean_movie_user_model_pred$y_hat)

results_df <- results_df %>% add_row(model_name = "mean_movie_user", RMSE = rmse_mean_movie_user_model_
```

8.2 Genre's impact on RMSE

```
avg_genre_rating <- edx %>%
  left_join(avg_movie_rating, by = "movieId") %>%
  group_by(genres) %>%
  summarize(b2 = mean(rating - mu_hat - b1))

rmse_mean_movie_genre_model_pred <- validation %>%
  left_join(avg_movie_rating, by = "movieId") %>%
  left_join(avg_genre_rating, by = "genres") %>%
  mutate(y_hat = mu_hat + b1 + b2) %>%
  select(y_hat)

rmse_mean_movie_genre_model_result <- RMSE(validation$rating, rmse_mean_movie_genre_model_pred$y_hat)

results_df <- results_df %>% add_row(model_name = "mean_movie_genre", RMSE = rmse_mean_movie_genre_model
```

8.3 Film Release Date vs. Rating Date's impact on RMSE

8.4 Evaluating Beta 2

Evaluate the results for Beta2 in the results_df data frame.

```
##                 model_name      RMSE beta_hat
## 1               Mean Model 1.0525579        0
## 2             mean_userId 0.9729599        1
## 3            mean_movieId 0.9410700        1
## 4              mean_genre 1.0457714        1
## 5      mean_releasereview 1.0446523        1
```

```
## 6          mean_movie_user 0.8633660        2
## 7         mean_movie_genre 0.9410700        2
## 8 mean_movie_releasereview 0.9395407        2
```

For our model that already includes the estimated average rating and our first selected variable movieId, the addition of the userId variable reduces RMSE most significantly. In fact, the RMSE for our model is now 0.8802762. This model is very close to our target RMSE of 0.86490 and is worth a closer inspection.

```
##        model_name     RMSE beta_hat
## 1      Mean Model 1.052558        0
## 2      mean_movie 0.941070        1
## 3 mean_movie_user 0.863366        2
```

9 Tuning Our Model

9.1 Determining Whether Regularization is the Right Approach

Earlier, you saw two data visualizations that showcased the disparity between number of ratings per movie and number of ratings per user. Here they are again:

As you can see, many movies were rated thousands of times, while most movies were rated only a few times. This is due to the popularity of the movie being reviewed.

Similarly, some users have thousands of reviews, while most users have fewer than 250 reviews. This is due to the fact that some users are more active film critics than others.

After seeing this, I assumed that some less popular movies and less active users with fewer ratings may provide more extreme rating values which skew its prediction from the mean average. We can check this theory by running the following code that evaluates which movies have the highest predicted rating and lowest predicted rating as compared to the number of times the movie was actually rated. Specifically, we look at this for the predicted value using only the movieId variable. Here is the code:

** Best Performing Predictions **

```
## # A tibble: 10 x 3
##    Title                                                     b1     n
##    <chr>                                                  <dbl> <int>
##  1 Hellhounds on My Trail                                  1.47     1
##  2 Satan’s Tango (SÃ¡tÃ¡ntangÃ³)                            1.47     2
##  3 Shadows of Forgotten Ancestors                          1.47     2
##  4 Fighting Elegy (Kenka erejii)                           1.47     2
##  5 Sun Alley (Sonnenallee)                                 1.47     2
##  6 Blue Light, The (Das Blaue Licht)                       1.47     3
##  7 Who’s Singin’ Over There? (a.k.a. Who Sings Over There) (Ko to t~  1.22     4
##  8 Human Condition II, The (Ningen no joken II)            1.22     8
##  9 Human Condition III, The (Ningen no joken III)          1.22     8
## 10 Constantine’s Sword                                    1.22     2
```

** Worst Performing Predictions **

```
## # A tibble: 10 x 3
##    Title                                b1     n
##    <chr>                             <dbl> <int>
```

```
##  1 Besotted                              -3.03     2
##  2 Hi-Line, The                          -3.03     1
##  3 Accused (Anklaget)                    -3.03     1
##  4 Confessions of a Superhero            -3.03     1
##  5 War of the Worlds 2: The Next Wave    -3.03     2
##  6 SuperBabies: Baby Geniuses 2          -2.73    56
##  7 Hip Hop Witch, Da                     -2.71    42
##  8 Disaster Movie                        -2.67    32
##  9 From Justin to Kelly                  -2.63   398
## 10 Criminals                             -2.53     2
```

As you can see, a majority of our best and worst rated movies were reviewed very few times. Far below the median rating per movie. Now, it seems essential to use a regularization model that penalizes values when they have a small sample size. This, essentially, does two things: 1) It moves the predicted rating closer to the mean rating, and 2) the less extreme prediction values (due to small sample size) will dramatically improve our RMSE when prediction is incorrect.

9.2 Apply regularization to penalize movies and users with fewer ratings

Note: Since I am applying regularization on variables that do not include genre, I've added the distinct() to reduce rows to the original dataset size.

```
lambdas <- seq(0, 10, by = 0.1)
rmses <- sapply(lambdas, function(lambda) {
  # Calculate the average by movie

  b_m <- edx %>%
    distinct(movieId, userId, .keep_all = TRUE) %>%
    group_by(movieId) %>%
    summarize(b_m = sum(rating - mu_hat) / (n() + lambda))

  # Calculate the average by user

  b_u <- edx %>%
    distinct(movieId, userId, .keep_all = TRUE) %>%
    left_join(b_m, by='movieId') %>%
    group_by(userId) %>%
    summarize(b_u = sum(rating - b_m - mu_hat) / (n() + lambda))

  # Compute the predicted ratings on validation dataset

  predicted_ratings <- validation %>%
    left_join(b_m, by='movieId') %>%
    left_join(b_u, by='userId') %>%
    mutate(pred = mu_hat + b_m + b_u) %>%
    pull(pred)

  # Predict the RMSE on the validation set

  return(RMSE(validation$rating, predicted_ratings))
})
```

9.3 Identify the lowest RMSE for the regularized model

It's effective to plot the results of the lambdas vs RMSE to ensure your sequence of values includes the value that best solves our problem.

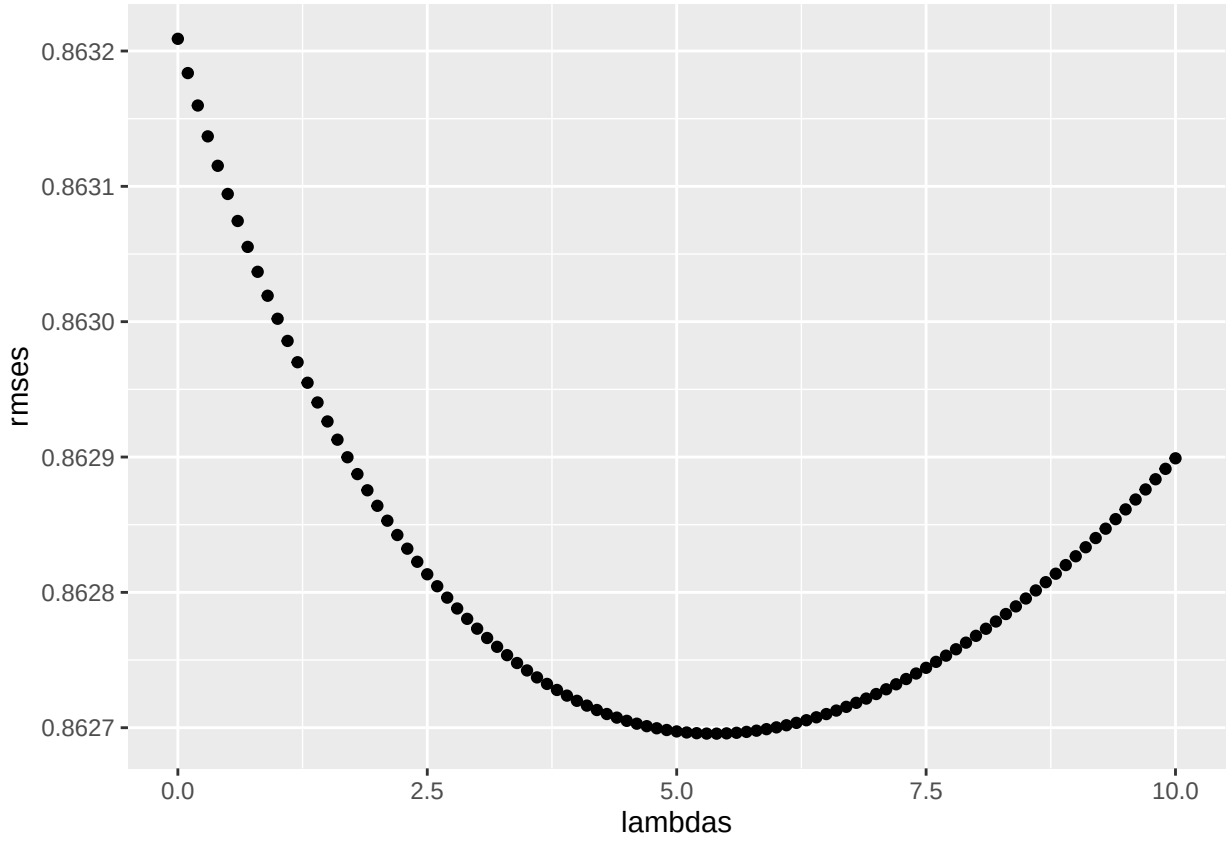

Figure 11: Lambdas vs RMSE score

Now we can select the lambda with the lowest RMSE

```
lambda <- lambdas[which.min(rmses)]
lambda
```

```
## [1] 5.4
```

10 Conclusion

In summary, a regularized least squares model using only two variables, movieId and userId, has enabled us to make an effective movie recommendation model that reduces the root mean squared error to 0.8626956.