MovieLens Data Recommendation System

Carol Riffle

2023-002-27

Contents

Introduction

Recommendation systems are used across many sectors including eCommerce, academia, farming, and other industries to facilitate the recommendation of products and services based on indicators such as personal preferences. The system leverages machine learning techniques applied to large data sets. The process of building a recommendation system uses different strategies to filter data. For example content based filtering (based on item features) and collaborative filtering (based on user preferences or responses) are commonly used to build recommendation systems. The value for these system lies in the ability of the system to accurately recommend products and services ranging from movies, clothes, restaurants, books, and cars to potentially support other applications in industries such as healthcare and precision farming. For companies such as Amazon, recommendation system applications are valuable tools for saving time and money.

In October 2006, Netflix launched a challenge to develop a recommendation system with a $1M prize to the winner who could beat their prediction RMSE of 0.9525 by 10% (reference: https://netflixtechblog.com/netflix-recommendations-beyond-the-5-stars-part-1-55838468f429). This challenge has been the subject of many articles. MovieLens data sets, from GroupLens Research, has been made available for use on many Machine Learning projects.

The goal for this Capstone project is to build and test a recommendation system to predict movie ratings based on features within the data set. The target RMSE for the model is < 0.86490. The data used to build the recommendation system is the MovieLens 10M data set (reference: https://grouplens.org/datasets/movielens/). Code was provided (reference: https://learning.edx.org/course/course-v1:HarvardX+PH125.9x+3T2022/block-v1:HarvardX+PH125.9x+3T2022+type@sequential+block@e8800e37aa444297a3a2f35bf84ce452/block-v1:HarvardX+PH125.9x+3T2022+type@vertical+block@e9abcdd945b1416098a15fc95807b5db)to generate the edx and final_holdout_test data sets.

Methods and Analysis

Preparation of Data Sets

Create edx and final_holdout_test sets

Each of the data sets was established using the code provided by the course for the Capstone project. Prior to pulling the data, several libraries were loaded for analysis including tidyverse, caret, ggplot2, dplyr, dslabs, and lubridate.

```
library(tidyverse)
library(caret)
library(ggplot2)
library(dplyr)
library(dslabs)
library(lubridate)
```

```
##
## Attaching package: 'lubridate'

## The following objects are masked from 'package:base':
##
##     date, intersect, setdiff, union
```

The files were accessed using the following links and data sets generated using the code below. Both the movies_file and ratings_file were accessed. Column names were established and selected characters converted to integers (userId, movieId, timestamp) and rating was converted to a numeric. The MOvieLens data was split (90:10) into a training data set (edx) and a test data set (final_holdout_test). The final_holdout_test data set has the same movidId and userId as the edx data set.

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

Two data sets were established. The edx data set was used for data exploration and model development. The final_holdout_test data set was used for testing the final model.

Split edx Data Set for Model Development

An additional split of the edx data set to prepare an "edx_train" and "edx_test" set was performed using the caret package to allow for testing during model development.

```
set.seed(1, sample.kind = "Rounding" )
```

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

```
edx_index <- createDataPartition(y=edx$rating, times = 1, p = 0.1, list = FALSE)
edx_train <- edx[-edx_index,]
edx_temp <- edx[edx_index,]

# Make sure edx_test data set has the same movies and users as the edx_train data set.
edx_test <- edx_temp %>%
     semi_join(edx_train, by = "movieId") %>%
     semi_join(edx_train, by = "userId")

# Add rows back to edx_train
test_removed <- anti_join(edx_temp, edx_test)
```

```
## Joining with `by = join_by(userId, movieId, rating, timestamp, title, genres)`
```

```
edx_train <- rbind(edx_train, test_removed)

rm(edx_index, edx_temp, test_removed)
```

A data.frame containing the movieId and title was also prepared for supporting analysis during model development.

```
movie_titles_train <- edx_train |>
select(movieId, title) |>
distinct()
```

Data Exploration and Visualization

The entire edx data set was used to explore and visualize the data prior to model development. Simple code functions were first used to understand the structure, dimensions, and features (variables) of the the data set. The basic data structure was determined using the str() function. The edx data set is a data frame with six columns of variables (userId, movieId, rating, timestamp, title, and genres) and 9000055 rows of observations (instances). Each row represents an observation of a unique rating by a user for a given movie. Each variable has the data type (class) defined (e.g. userId is an integer, title is a character).

Data Structure

```
## 'data.frame':    9000055 obs. of  6 variables:
##  $ userId   : int  1 1 1 1 1 1 1 1 1 1 ...
##  $ movieId  : int  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 838984474 838983653 838984885 83
##  $ title    : chr  "Boomerang (1992)" "Net, The (1995)" "Outbreak (1995)" "Stargate (1994)" ...
##  $ genres   : chr  "Comedy|Romance" "Action|Crime|Thriller" "Action|Drama|Sci-Fi|Thriller" "Action|Ad
```

There are a few notable things to mention about the data set from exploring the data structure. The title of the movie also includes the year released in parenthesis at the end of the title. This becomes important to separate if the release year is going to be used during model development.

The summary() function was used to understand some basic statistics around the data set and indicates no data is missing. The average rating across the data set is 3.512 and the median was 4. The ratings values range from 0.5 to 5 indicating no movies were rated with a zero or missing a rating.

```
##      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
##
##
##
```

Data Wrangling and Cleaning

Each variable was checked again for missing values using the code below.

```
# edx |> filter(is.na(variable))
```

In order to generate a year for the Unix timestamp, a new column was formed using mutate() function and the as_datetime() function from the lubridate package. The new column, rating_year was modified to be only the year the movie was rated by a user, using the year() function. Data indicates movies were rated from 1995 through 2009.The rating year is now included in a new column for further exploration.

```
edx_yr <- edx |> mutate(rating_year = year(as_datetime(timestamp)))
```

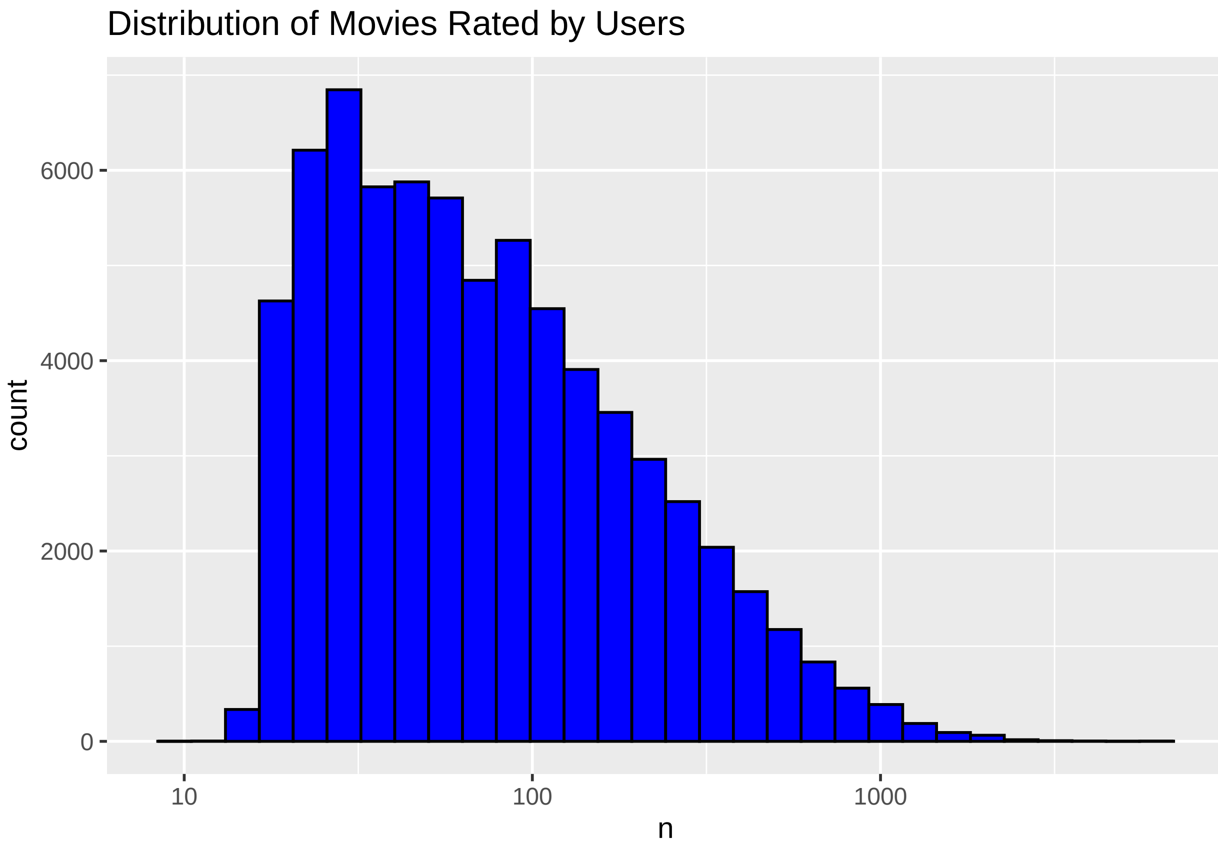
```r
str(edx_yr)
```

```
## 'data.frame':    9000055 obs. of  7 variables:
##  $ userId     : int  1 1 1 1 1 1 1 1 1 1 ...
##  $ movieId    : int  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 838984474 838983653 838984885
##  $ title      : chr  "Boomerang (1992)" "Net, The (1995)" "Outbreak (1995)" "Stargate (1994)" ...
##  $ genres     : chr  "Comedy|Romance" "Action|Crime|Thriller" "Action|Drama|Sci-Fi|Thriller" "Action
##  $ rating_year: num  1996 1996 1996 1996 1996 ...
```

User Data

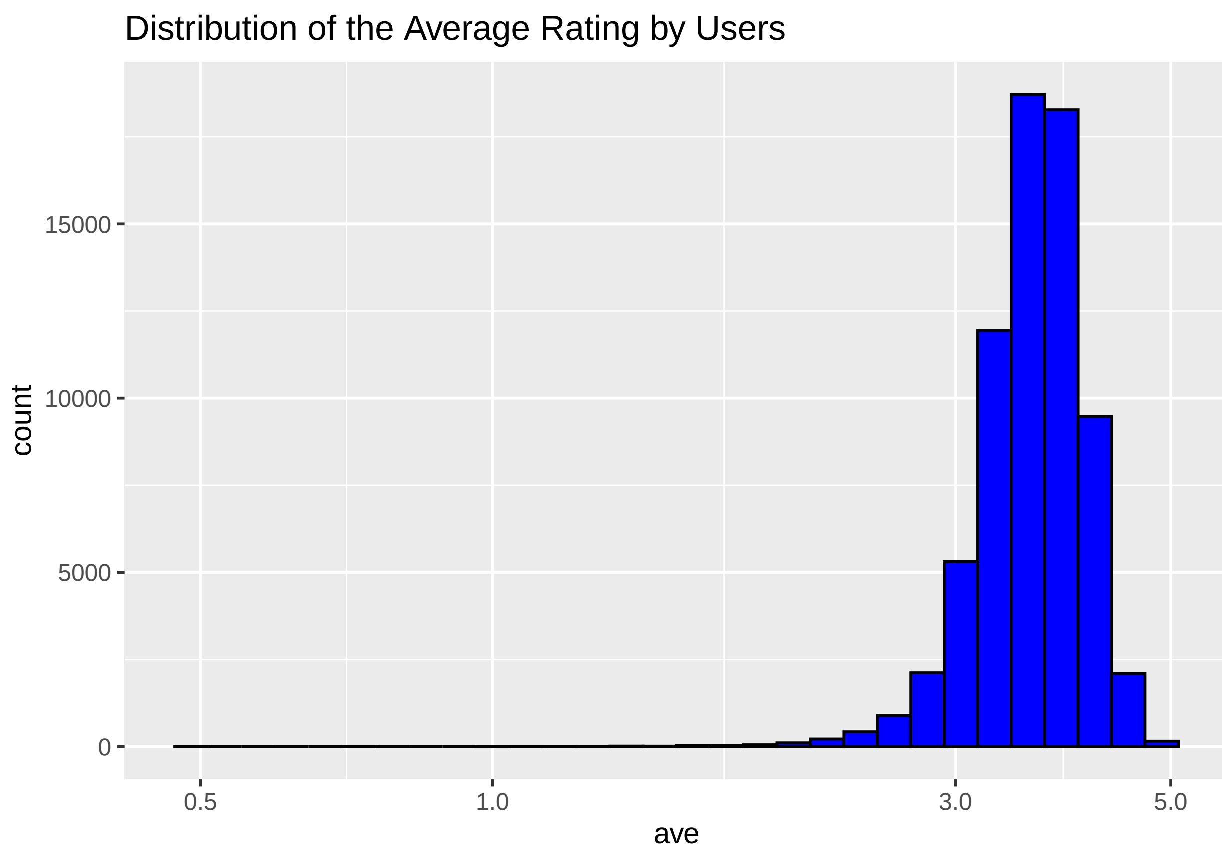
The summarize() function was used to determine there were 69,878 unique users in the edx data set. The distribution plot for users shows how many movies were rated by users. The distribution is skewed right (peak is left of the center) indicating a high percentage of the users rated less than 100 movies. There were some users who rated over 1000 movies. Of the 69,878 unique users, 46,010 rated <= 100 movies (over 50%), 611 users rated 1000 movies or more, and 23,505 users rated 100 to 1000 movies. The variability in the number of movies rated could be do to a number of reasons including personal perferences for example a particular genre, leading actor or actresses;blockbuster movies are typically seen by a large number of people, some movies may be more niche and only are reviewed by a select few individuals.

```r
edx |> dplyr::count(userId) |>
ggplot(aes(n)) +
geom_histogram(bins = 30, color = "black", fill = "blue") +
scale_x_log10() +
ggtitle("Distribution of Movies Rated by Users")
```



The figure below shows the distribution of the average rating for users. The peak is to the far right of the histogram indicating data is skewed left (negative). Since the peak is left of the center, the mean is less than the median.

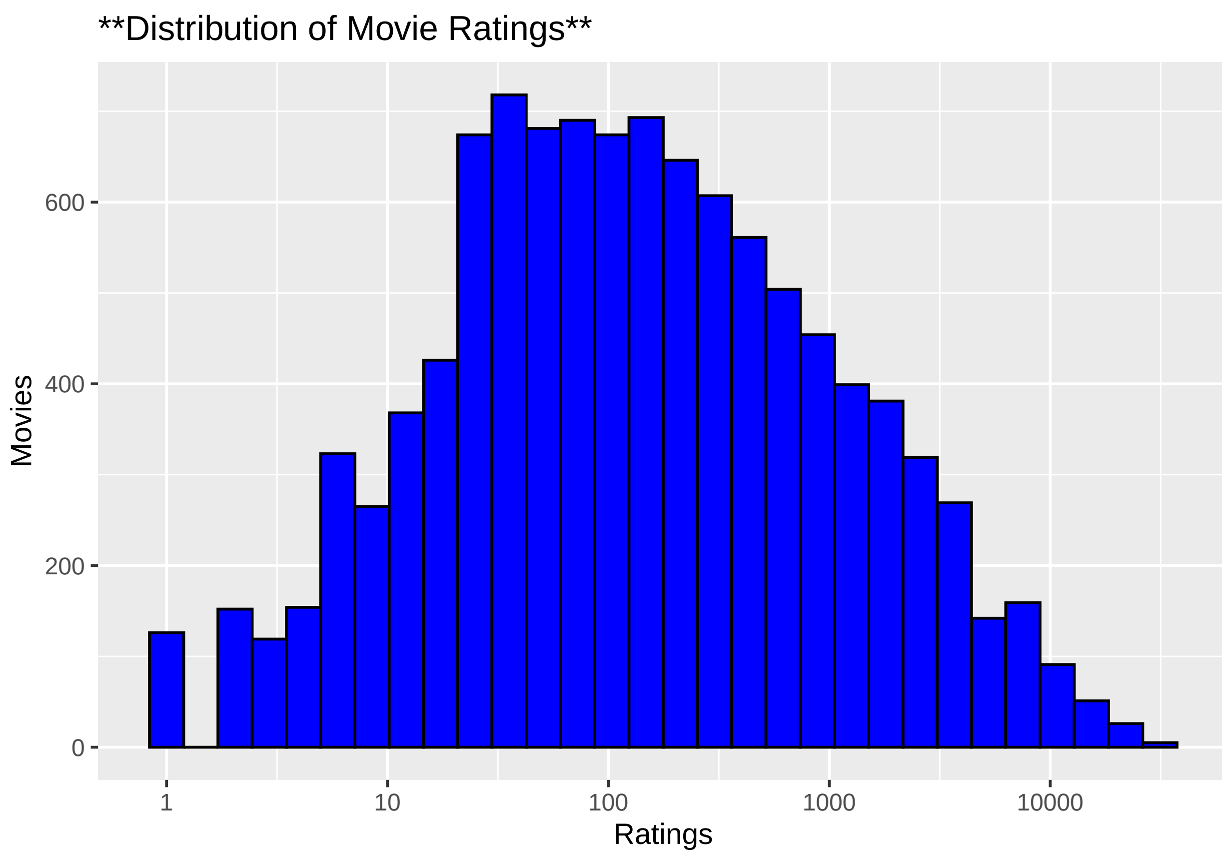
```
edx |>
group_by(userId) |>
summarize( ave = mean(rating)) |>
ggplot(aes(ave)) +
geom_histogram(bins = 30, color = "black", fill = "blue") +
scale_x_log10() +
ggtitle("Distribution of the Average Rating by Users")
```



Movie Data

Basic analysis of the movie data indicates there are 10,677 unique movies in the edx data set. The distribution is normal with the exception of a group of users who only rated one movie. Some movies are rated much more than others as shown by data on the right side of the distribution. Based on the number of users and movies, there would be over 700M observations (rows) if every user rated every movie. The edx data set contains 9M observations indicating not every user rated every movie, which would be expected. Users have movie preferences that can be impacted by the genre and leading actors/actresses,and a number of variables. A matrix created from this data set would be very sparse.

```
##   n_movies
## 1    10677
```



To verify if movies with the highest average ratings are popular, the average rating was calculated for each movie. Table 1 indicates the top 10 movies are relatively unknown and have few ratings. Most likely, the users have a preference for a certain kind of movie.

Table 1: **Top 10 Movies Based on the Average Rating**

Movie Title	Average Rating	Number of Ratings
Hellhounds on My Trail (1999)	5.00	1
Satan's Tango (Sátántangó) (1994)	5.00	2
Shadows of Forgotten Ancestors (1964)	5.00	1
Fighting Elegy (Kenka erejii) (1966)	5.00	1
Sun Alley (Sonnenallee) (1999)	5.00	1
Blue Light, The (Das Blaue Licht) (1932)	5.00	1
Who's Singin' Over There? (a.k.a. Who Sings Over There) (Ko to tamo peva) (1980)	4.75	4
Human Condition II, The (Ningen no joken II) (1959)	4.75	4
Human Condition III, The (Ningen no joken III) (1961)	4.75	4
Constantine's Sword (2007)	4.75	2

Movies with less than 10 ratings were removed to see how the top movies would change. Several cut-off levels were initially evaluated (code not included). The effect of filtering out movies with less than or equal to 10 ratings is shown below. The top movies shifted to well known movies including classics such at "Casablanca".

```
## # A tibble: 9,623 x 3
```

```
##    movieId ave_rating count
##      <int>      <dbl> <int>
##  1       1       3.93 23790
##  2       2       3.21 10779
##  3       3       3.15  7028
##  4       4       2.86  1577
##  5       5       3.07  6400
##  6       6       3.82 12346
##  7       7       3.36  7259
##  8       8       3.13   821
##  9       9       3.00  2278
## 10      10       3.43 15187
## # ... with 9,613 more rows
```

Table 2: **Top 10 Movies Based on the Average Rating When the Number of Ratings was >= 10**

Movie Title	Average Rating	Number of Ratings
Shawshank Redemption, The (1994)	4.455131	28015
Godfather, The (1972)	4.415366	17747
Usual Suspects, The (1995)	4.365854	21648
Schindler's List (1993)	4.363493	23193
Casablanca (1942)	4.320424	11232
Rear Window (1954)	4.318651	7935
Sunset Blvd. (a.k.a. Sunset Boulevard) (1950)	4.315880	2922
Third Man, The (1949)	4.311426	2967
Double Indemnity (1944)	4.310817	2154
Paths of Glory (1957)	4.308721	1571

Ratings

By plotting a bar chart of the ratings you can see 10 distinct ratings for the movies are shown ranging from 0.5 to 5. No movies were rated with a "0" and the rating given the most was "4" with over 2.5M ratings followed by "3" with over 2.1M ratings and "5" with over 1.3M ratings. In total, these three ratings represented over 67% of the total ratings. In addition, the ratings that represented whole numbers were higher than the numeric decimal ratings on either side. The average rating across the edx data set is 3.512. The year_rated data in the next section provides some additional insight into the ratings pattern shown below.The breakout also supports the observations above where users had a tendency to rate movies high.The number of movies and users is known and not all movies were rated by every user.

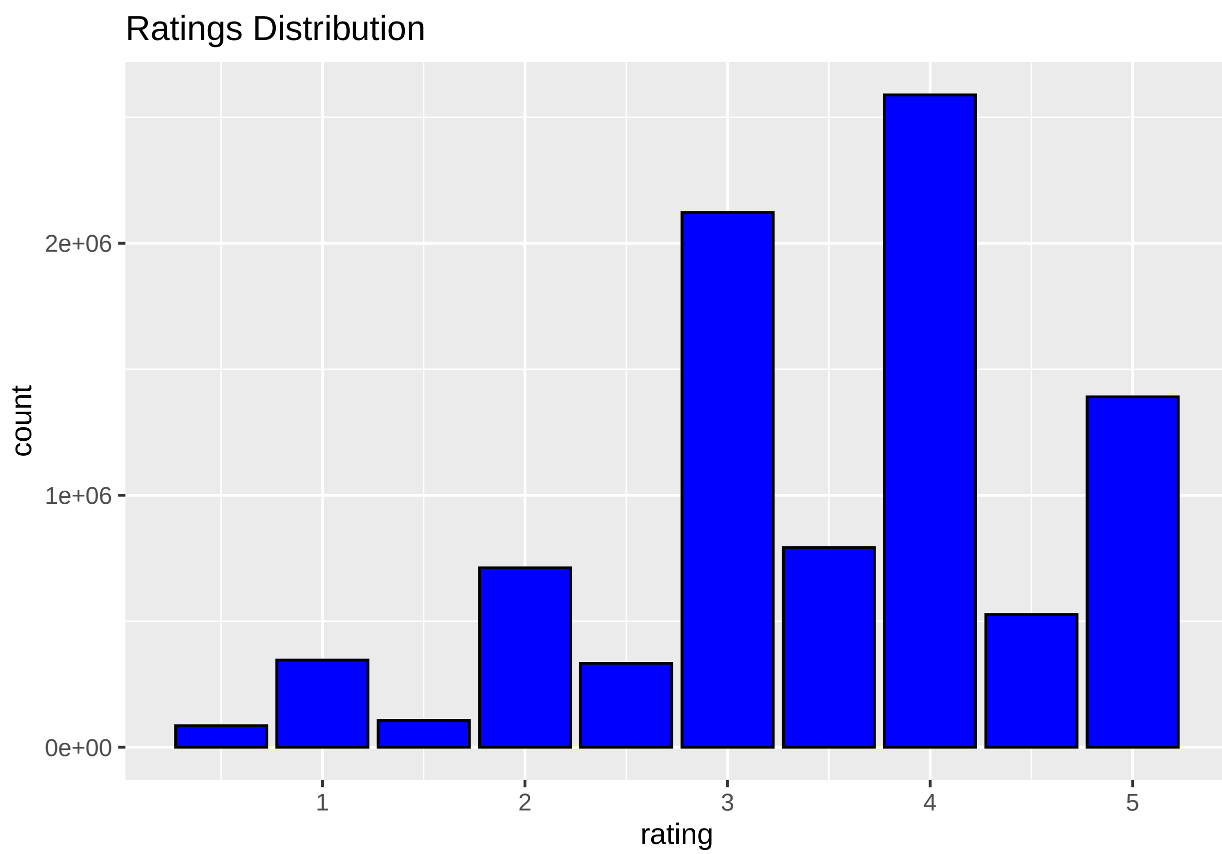
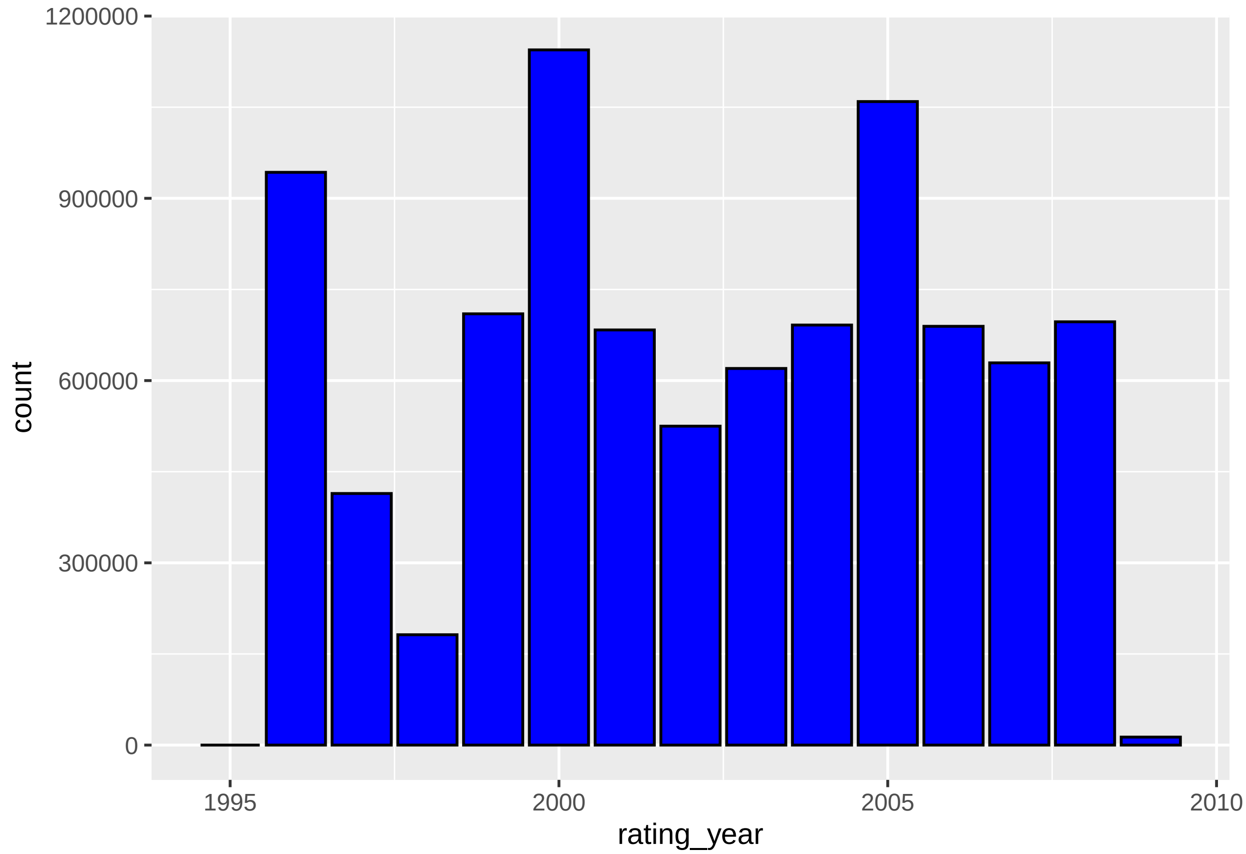


Table 3: **Ratings Distribution Summary Table**

rating	count
4.0	2588430
3.0	2121240
5.0	1390114
3.5	791624
2.0	711422
4.5	526736
1.0	345679
2.5	333010
1.5	106426
0.5	85374

Year Rated

The movie ratings were collected over a period of 15 years ranging from 1995 into 2009. The first and last years ratings were collected (1995 and 2009) had a low number of ratings. Some rating years were more popular than others, both 2000 and 2005 had over 1M ratings.



The distribution of ratings was visualized using a heatmap. The pattern shows only whole number ratings (1 through 5) were used through 2002. The pattern reflects the highest number of ratings were in the 3 to 5 range. The pattern indicates a few years with a high number of ratings. The pattern needs to be evaluated based on the split of when decimal ratings were included. Up through 2002, 1996 and 2000 appear to have the highest number of ratings. After 2002, 2005 appears to have the highest number of ratings.

```
edx_yr |> count(rating_year, rating) |>
  ggplot(aes(x = rating_year, y = rating)) +
  geom_tile(aes(fill = n)) +
geom_text(aes(label = n), color = "white", size = 2) +
   guides(fill = guide_colourbar(barwidth = 1,
                                barheight = 10))
```

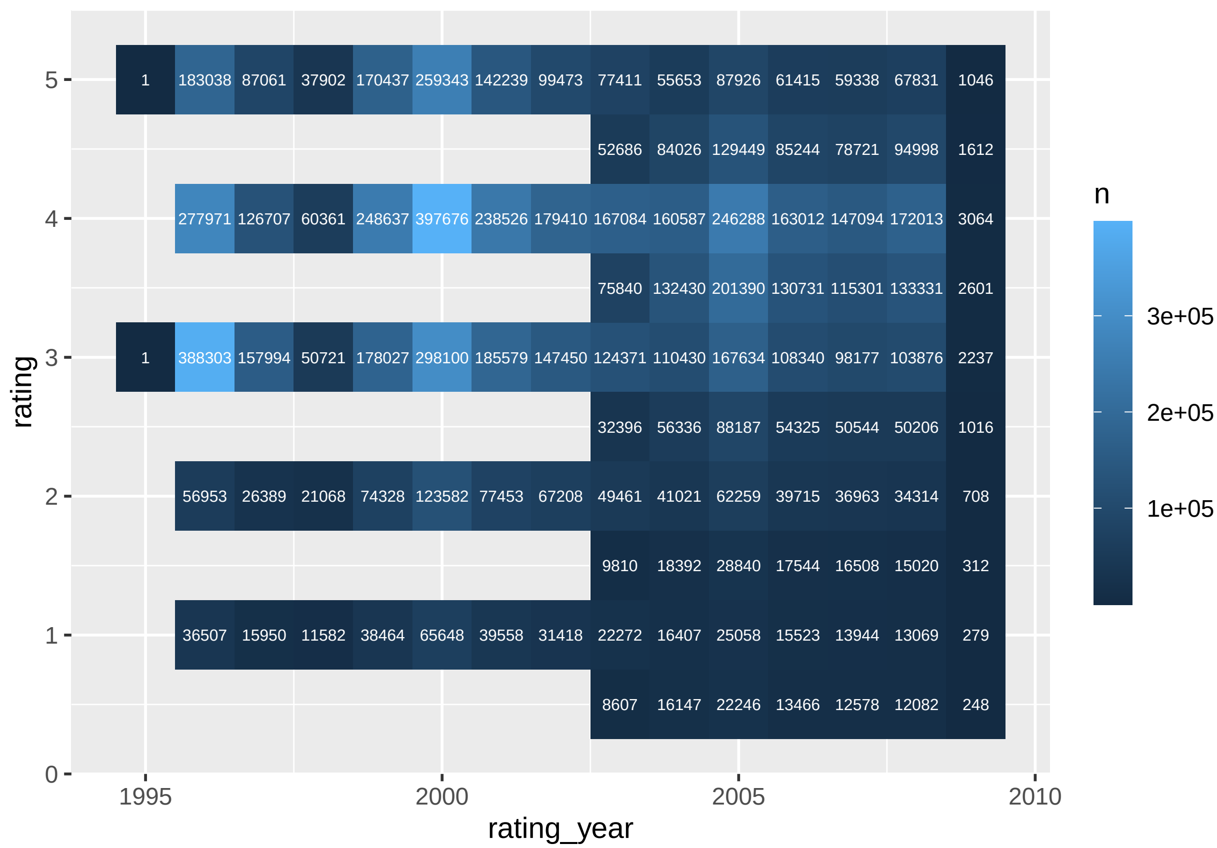


Table 4 summarizes the average rating and number of ratings for each year. The top three years with the highest number of ratings were 2000, 2005, and 1996; all with over 900K ratings.

```
edx_yr |>
group_by(rating_year) |>
summarize( year_ave = mean(rating), count = n()) |>
knitr::kable(col.names = c("Rating Year", "Average Rating", "Number of Ratings"), caption = "**Rating Y
```

Table 4: **Rating Year Summary**

Rating Year	Average Rating	Number of Ratings
1995	4.000000	2
1996	3.545286	942772
1997	3.585703	414101
1998	3.506144	181634
1999	3.617354	709893
2000	3.578044	1144349
2001	3.536229	683355
2002	3.473012	524959
2003	3.471691	619938
2004	3.425479	691429
2005	3.435830	1059277
2006	3.465925	689315
2007	3.469147	629168
2008	3.543312	696740

Rating Year	Average Rating	Number of Ratings
2009	3.458165	13123

Genres

The edx data set contains movieIds where the genre are grouped in a single observation for a given movie. There are 797 unique genre combinations for movies in the data set.

```
genre_number <- summarize(edx, n_genres = n_distinct(genres))
            genre_number
```

```
##   n_genres
## 1      797
```

The top 10 genre combinations are below. Drama and Comedies appear to be rated the most along with various combinations which include them (e.g., Comedy/Romance, Drama/Romance, Comedy/Drama), indicating a preference for these genres. A separate edx_genre data frame was set up to explore genres in the data set.

```
edx_genre <- edx |>
select(genres, rating)
```

```
edx_genre |>
    group_by(genres) |>
    summarize(count = n()) |>
  arrange(desc(count)) |>
  slice(1:20) |>
  knitr::kable(caption = "**Top 20 Genres**")
```

Table 5: **Top 20 Genres**

genres	count
Drama	733296
Comedy	700889
Comedy|Romance	365468
Comedy|Drama	323637
Comedy|Drama|Romance	261425
Drama|Romance	259355
Action|Adventure|Sci-Fi	219938
Action|Adventure|Thriller	149091
Drama|Thriller	145373
Crime|Drama	137387
Drama|War	111029
Crime|Drama|Thriller	106101
Action|Adventure|Sci-Fi|Thriller	105144
Action|Crime|Thriller	102259
Action|Drama|War	99183
Action|Thriller	96535
Action|Sci-Fi|Thriller	95280

genres	count
Thriller	94662
Horror|Thriller	75000
Comedy|Crime	73286

Lets explore the genres in more detail by splitting the combined genres to understand the number of unique genre and summarize to understand the number of ratings for each genre. Since this code would add additional lines to the data set, it was not used for model development as a variable to test. Table 6 shows the top genre based on counts also have the highest average rating bracketing the average across the edx data set of 3.512. Action movies emerged as third followed by Thriller, Adventure, Romance, Sci-Fi and Crime; all with over 1M ratings. Seven movies had no genre listed.

```
edx_genre |> separate_rows(genres, sep = "\\|") |>
group_by(genres) |>
summarize(ave_genre_rating = mean(rating), count = n()) |>
arrange(desc(count)) |>
knitr::kable(caption = "**Genres and Average Ratings**")
```

Table 6: **Genres and Average Ratings**

genres	ave_genre_rating	count
Drama	3.673131	3910127
Comedy	3.436908	3540930
Action	3.421405	2560545
Thriller	3.507676	2325899
Adventure	3.493544	1908892
Romance	3.553813	1712100
Sci-Fi	3.395743	1341183
Crime	3.665925	1327715
Fantasy	3.501946	925637
Children	3.418715	737994
Horror	3.269815	691485
Mystery	3.677001	568332
War	3.780813	511147
Animation	3.600644	467168
Musical	3.563305	433080
Western	3.555918	189394
Film-Noir	4.011625	118541
Documentary	3.783487	93066
IMAX	3.767693	8181
(no genres listed)	3.642857	7

Model Development

The models used for development followed a collaborative filtering approach on selected variables including users (userId) and movies(movieId)to predict ratings(rating). The movie title (title) is included in the data frame to evaluate impact on model recommendations.

```
edx_train <- edx_train |> select(userId, movieId, title, rating)
```

The root-mean-square error (RMSE) was determined for each model using the code below. The lowest RMSE values represent the model with the best fit. As mentioned previously, the target RMSE for the model is <0.86490.

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

Model 1 Average Rating

If we follow a basic system, looking at the average rating across all movies in the edx_train data set gives an RMSE of of 1.06 and an average of 3.512. When the RMSE is over 1, there are errors associated the model which makes sense since it assumes no bias due to personal preference or response to a wide selection of movies. Data exploration in the previous sections indicates there are differences which impact movie ratings.

```
avg <- mean(edx_train$rating)
avg
```

```
## [1] 3.512456
```

```
Model_1_rmse <- RMSE(edx_test$rating, avg)
Model_1_rmse
```

```
## [1] 1.060054
```

```
rmse_values <- data_frame(Method = "Model 1 Average Rating", RMSE = Model_1_rmse)
```

```
## Warning: 'data_frame()' was deprecated in tibble 1.1.0.
## i Please use 'tibble()' instead.
```

```
rmse_values
```

```
## # A tibble: 1 x 2
##   Method                  RMSE
##   <chr>                  <dbl>
## 1 Model 1 Average Rating  1.06
```

Model 2 Movie Effects

The model was updated by including the effects or bias from all of the movies rated.

```
avg <- mean(edx_train$rating)
movie_avgs <- edx_train %>%
     group_by(movieId) %>%
     summarize(bi = mean(rating - avg))
```

```
predicted_ratings <- avg + edx_test |>
     left_join(movie_avgs, by = "movieId") |>
     pull(bi)

Model_2_movie_rmse <- RMSE(predicted_ratings, edx_test$rating)
Model_2_movie_rmse
```

```
## [1] 0.9429615
```

```
Model_Results <- bind_rows(rmse_values, data_frame(Method = "Model 2 Movie Effects", RMSE = Model_2_movi
Model_Results
```

```
## # A tibble: 2 x 2
##   Method                  RMSE
##   <chr>                  <dbl>
## 1 Model 1 Average Rating 1.06
## 2 Model 2 Movie Effects  0.943
```

The RMSE was lowered to 0.9429615, a reduction of approximately 11%. If movies associated with the highest and lowest levels of bias are pulled, the data indicate the movies are not well known and typically have a low number of ratings. In order to further reduce the RMSE, user bias will be included in Model 3.

Model 3 User and Movie Effect

The user and movie effect were determined using the code below. The histogram peak for the user is to the far right of the histogram indicating data is skewed left (negative).

```
user_avgs <- edx_train |>
     left_join(movie_avgs, by='movieId') |>
     group_by(userId) |>
     summarize(bu = mean(rating - avg - bi))
```

```
predicted_ratings <- edx_test |>
     left_join(movie_avgs, by='movieId') |>
     left_join(user_avgs, by='userId') |>
     mutate(pred = avg + bi + bu) |>
     pull(pred)
```

```
Model_3_User_movie_rmse <- RMSE(predicted_ratings, edx_test$rating)
Model_3_User_movie_rmse
```

```
## [1] 0.8646843
```

```
Model_Results <- bind_rows(Model_Results, data_frame(Method = "Model 3 User and Movie Effect", RMSE = Mo
Model_Results
```

```
## # A tibble: 3 x 2
##   Method                         RMSE
##   <chr>                         <dbl>
## 1 Model 1 Average Rating        1.06
## 2 Model 2 Movie Effects         0.943
## 3 Model 3 User and Movie Effect 0.865
```

The addition of user bias lowered the RMSE from 0.9429615 to 0.864843, a reduction of approximately 8%. When there is a small sample size or few users, uncertaintly exists which can impact accuracy of the prediction model.

Model 4 Regularization of Movie and User Bias

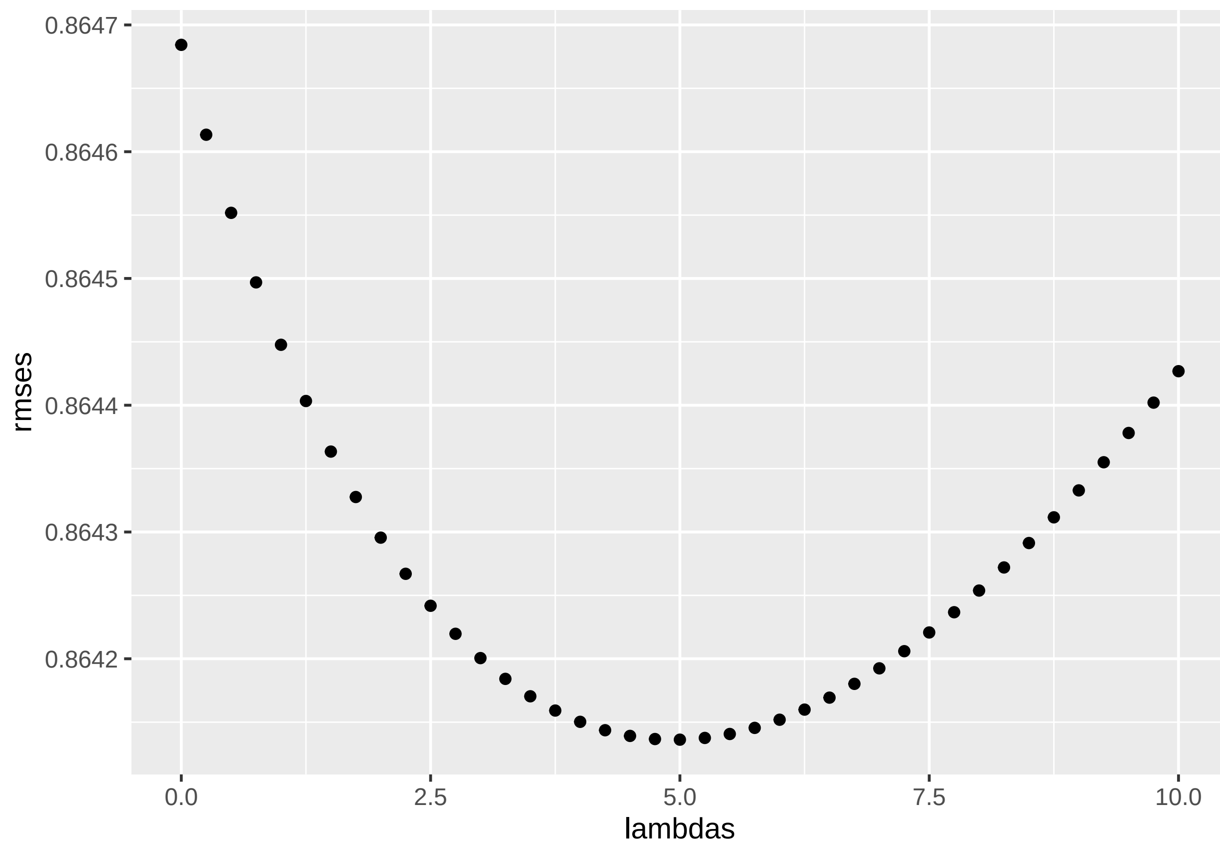
In order to further reduce the RMSE, user and movie bias will be regularized to include a penalty for large bias associated with few ratings. The penalty helps shrink the bias to zero.

The best lambda was selected using cross validation. The edx_test set was used to check the RMSE

```
# Selection of lambda for movie and user bias
lambdas <- seq(0, 10, 0.25)
rmses <- sapply(lambdas, function(l){
     ave_rating <- mean(edx_train$rating)
     bi <- edx_train |>
          group_by(movieId) |>
          summarize(bi = sum(rating - ave_rating)/(n()+l))
     bu <- edx_train |>
          left_join(bi, by="movieId") |>
          group_by(userId) |>
          summarize(bu = sum(rating - bi - ave_rating)/(n()+l))
     predicted_ratings <-
          edx_test |>
          left_join(bi, by = "movieId") |>
          left_join(bu, by = "userId") |>
          mutate(pred = ave_rating + bi + bu) |>
          pull(pred)
     return(RMSE(predicted_ratings, edx_test$rating))
})

qplot(lambdas, rmses)
```

```
## Warning: `qplot()` was deprecated in ggplot2 3.4.0.
```




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

```
## [1] 5
```

```
Model_4_apply_regularization_rmse <- min(rmses)
Model_4_apply_regularization_rmse
```

```
## [1] 0.8641362
```

```
Model_Results <- bind_rows(Model_Results, data_frame(Method = "Model 4 Regularization of Movie and User
Model_Results
```

```
## # A tibble: 4 x 2
##   Method                                          RMSE
##   <chr>                                          <dbl>
## 1 Model 1 Average Rating                         1.06
## 2 Model 2 Movie Effects                          0.943
## 3 Model 3 User and Movie Effect                  0.865
## 4 Model 4 Regularization of Movie and User Bias 0.864
```

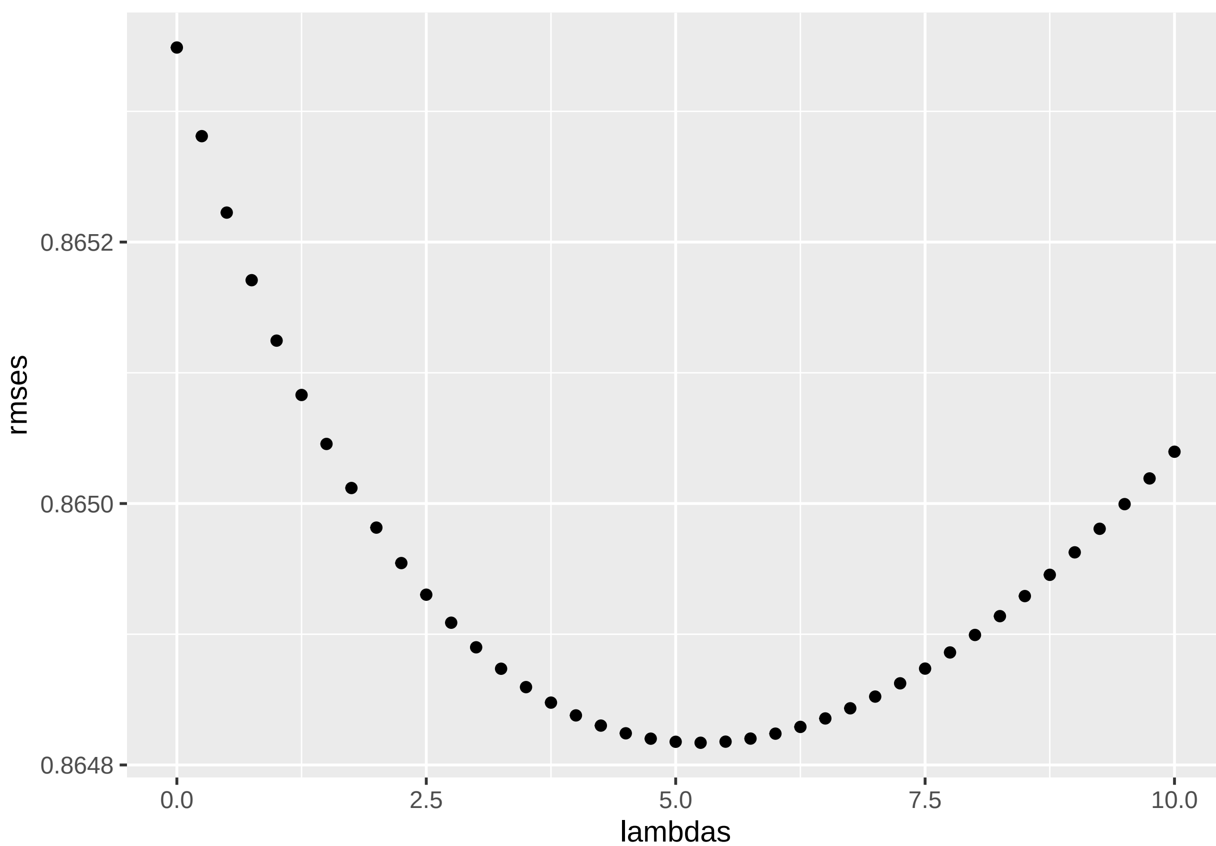
A slight decrease of RMSE, less than 0.1%, was observed when regularization was applied to movie and user bias. Model 4 was tested with the final_holdout_test data set in the "Results" section below.

Results

Model 4 using regularized estimates of move and user bias was selected to evaluate the final_holdout_test data set.

```
lambdas <- seq(0, 10, 0.25)
rmses <- sapply(lambdas, function(l){
     ave_rating <- mean(edx$rating)
     bi <- edx |>
          group_by(movieId) |>
          summarize(bi = sum(rating - ave_rating)/(n()+l))
     bu <- edx |>
          left_join(bi, by="movieId") |>
          group_by(userId) |>
          summarize(bu = sum(rating - bi - ave_rating)/(n()+l))
     predicted_ratings <-
          final_holdout_test |>
          left_join(bi, by = "movieId") |>
          left_join(bu, by = "userId") |>
          mutate(pred = ave_rating + bi + bu) |>
          pull(pred)
     return(RMSE(predicted_ratings, final_holdout_test$rating))
})

qplot(lambdas, rmses)
```



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

```
## [1] 5.25
```

```r
Model_4_results <- min(rmses)
Model_4_results
```

```
## [1] 0.864817
```

There was little change with the RMSE when using the final_holdout_test data set. The models used collaborative filtering based on movie and user effects. No movies or users were removed for this analysis. Data was used to capture content from every type of user for every movie. While removing users who have rated a certain number of movies helps get a better understanding of what the top movies are, it does not preclude the user preference piece when a user may have a passion for an unusual movie. Using ratings across the movies helps balance, to a certain degree, the overzealous user and the grumpy user.

Conclusions

The RMSE target was achieved as the model development progressed. Additional work to look at genres using a content-based approach where the content would be attributes and features of the movie would be interesting to explore. Further exploration of how the genres and other features correlate with each other would be a good next step in continuation of the model development. Two options for generation of correlation coefficients would be using a Pearson Correlation was used to measure the the relationship between variables (Note that Pearson's correlation assumes normal distribution of variables and no outliers so some additional data wrangling would be required). The second option would be the Spearman's rank correlation coefficients. Descriptions of both are included in the following reference. https://www.scribbr.com/statistics/pearson-correlation-coefficient/#:~:text=You%20should%20use%20the%20Pearson,(4)%20have%20no%20outliers).

Other packages to learn and explore for model development would be the the recommenderlab package (https://cran.r-project.org/web/packages/recommenderlab/readme/README.html#:~:text=R%20package%20recommenderlab%20%2D%20Lab%20for,recommendations%2C%20and%20cross%2Dvalidation) and the recosystem package for Matrix Factorization (https://cran.r-project.org/web/packages/recosystem/index.html)

One final note, while there are many options for development of recommendation models, each has advantages and disadvantages that need to be considered prior to implementation.