MovieLens Project

Brendan Ball

7/9/2021

1. Introduction

The MovieLens Dataset, published by F. Maxwell Harper and Joseph A. Konstan, is a compiled data that has 10 million ratings, 100,000 tag applications in relation to 10,000 movies, and reviewed by 72,000 users [1]. The data, published in 2009 will help provide the information necessary to create a recommendation system using the tools learned the HarvardX: Data Science course. This is the goal of the project, and if achieved, can help improve the user experience when selecting a movie from recommendation sources.

The key steps that will be performed includes various features, such as genres, movie ratings, and popular movie titles will to be investigated. This will help provide an informal understanding of the entire data set. Once this has been done, a root mean square error (RMSE) calculation can be calculated to study the accuracy and training of the algorithm.

Coding techniques such as data-wrangling, data visualization, plot generation, and other essential topics will be utilized to analyze the data.

To first operate the R code, the following packages will need to be installed: 1) tidyverse 2) caret 3) data.table 4) kableExtra 5) lubridate

The library packages will then be uploaded into the R code after installation.

```
# Use the library commands
library(tidyverse)
library(caret)
library(data.table)
library(kableExtra)
library(lubridate)
library(tinytex)
```

The movie set is imported from the online website shown below. A 62.5MB file is expected to be downloaded. Please ensure there is enough room to download this large file before running the code.

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

# Download the temporary file (If no file is present)
dl <- tempfile()
download.file("http://files.grouplens.org/datasets/movielens/ml-10m.zip", dl)
```

The movies data that is downloadeed from the internet is manipulated to become a clean data set. Using the following code below:

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

# Create a movies dataframe
movies <- str_split_fixed(readLines(unzip(dl, "ml-10M100K/movies.dat")),
                          "\\::", 3)
colnames(movies) <- c("movieId", "title", "genres")
```

```
# If using R 4.0 or later:
movies <- as.data.frame(movies) %>% mutate(movieId = as.numeric(movieId),
                                            title = as.character(title),
                                            genres = as.character(genres))

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

The validation set for the model is created below:

```
# Validation set will be 10% of MovieLens data

# If using R 3.5 or earlier, use `set.seed(1)`
suppressWarnings(set.seed(1, sample.kind="Rounding"))

test_index <- createDataPartition(y = movielens$rating, times = 1, p = 0.1, list = FALSE)
edx <- movielens[-test_index,]
temp <- movielens[test_index,]

# Make sure userId and movieId in validation set are also in edx 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", "genres")
```

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

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

2. Methods & Analysis

To better understand the current data set, a few preliminary commands are used. The first is the class() command to see the class of the edx data.

```
# Determine the class of the edx data
class(edx)
```

```
## [1] "data.table" "data.frame"
```

The class of edx is a data.frame

The glimpse of the edx data can also be studied. This will show all observations (9,000,055) and variables (6).

```
# Glimpse the data set of "edx"
glimpse(edx)
```

```
## 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, 2, ~
## $ movieId   <dbl> 122, 185, 292, 316, 329, 355, 356, 362, 364, 370, 377, 420, ~
## $ rating    <dbl> 5, 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, 83898~
## $ title     <chr> "Boomerang (1992)", "Net, The (1995)", "Outbreak (1995)", "S~
## $ genres    <chr> "Comedy|Romance", "Action|Crime|Thriller", "Action|Drama|Sci~
```

The summary of the results is shown below. Where different information for each of the variables is listed.

```
# Inspect the result summaries of the results
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
##
##
##
```

After generating plots and learning more about the data, the next step to test the performance of the model is with root mean squared error (RMSE).

To calculate the RMSE value, residuals must be defined. This is the difference between the predicted value, and the actual value in a data set. In the following equation, $\hat{y}_{i,o}$ is denoted as the observed value, whereas $\hat{y}_{u,a}$ is the actual value. Finding the difference, and squaring this term as the sum of all 'u' terms, and dividing by N number of non-missing data points will return the RMSE value. A higher RMSE value will show as a better fit to the data, however; we want to ensure that the model is not overtrained, as this will not be fully reliable in any future movies that may be released.

$$RMSE = \sqrt{(1/N)\sum_{u,i}(\hat{y}_{u,o} - \hat{y}_{u,a})^2}$$

3. Results

To understand the results, a genre_order data is made where the genres are ranked in descending order. This will prepare the data for the following table.

```
# Explore top individual genres
genre_order <- edx %>% separate_rows(genres, sep = "\\|") %>%
  group_by(genres) %>%
  summarize(count = n()) %>%
  arrange(desc(count))
```

The table of the top genres is generated below. The top three genres include drama, comedy, and action in respective order. It is clear that the top three genres are significantly more popular than the following genres. This suggests that the bulk of genres that are enjoyed can be grouped into drama, comedy, and action. Perhaps thrillers and adventure movies can be ranked high as well.

```
# Create a table to list the top genres and the
table_genre <- data.table(genre_order, rownames = FALSE, filter="top")
table_genre <- table_genre[,c("genres","count")]
table_genre
```

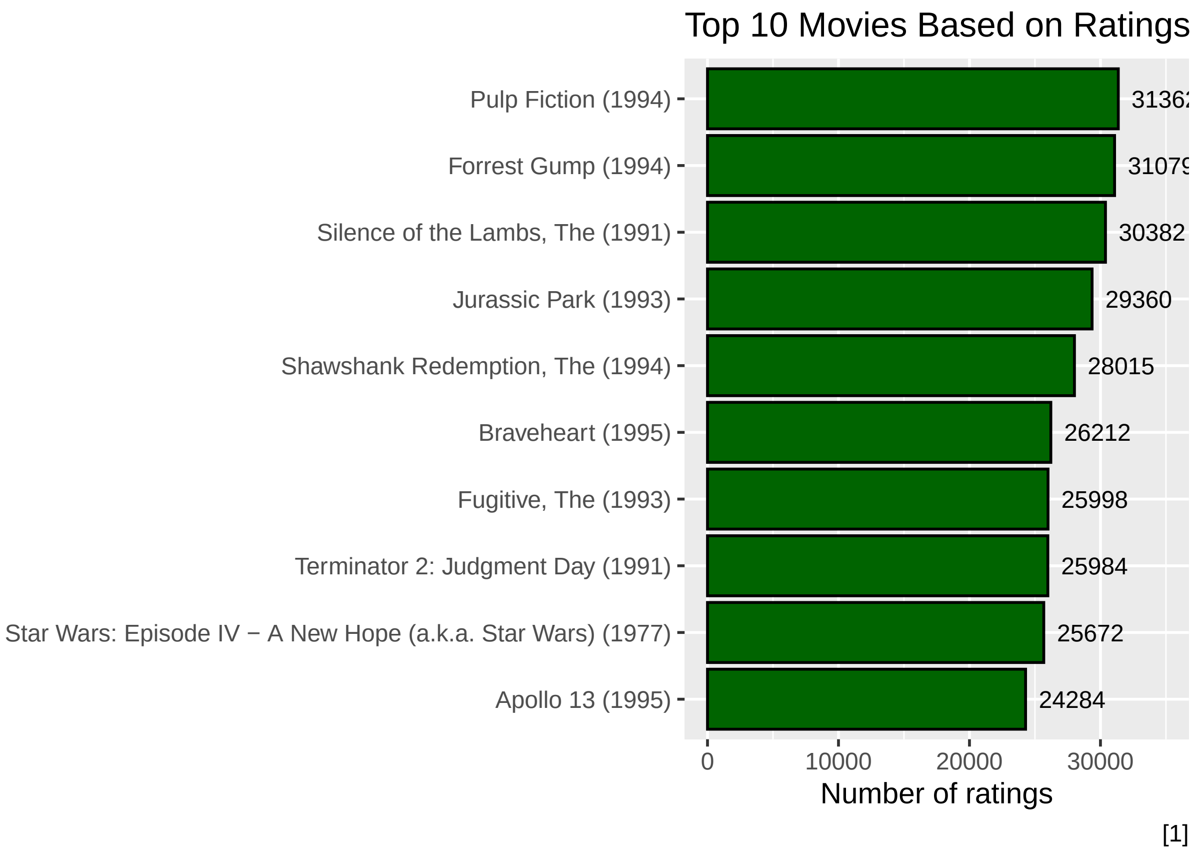
```
##                   genres   count
##  1:                Drama 3910127
##  2:               Comedy 3540930
##  3:               Action 2560545
##  4:             Thriller 2325899
##  5:            Adventure 1908892
##  6:              Romance 1712100
##  7:               Sci-Fi 1341183
##  8:                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
```

The top 10 movie titles are ranked in descending order.

```
# Rank the Top 10 Movie Titles
ranked_titles <- edx %>%
  group_by(title) %>%
  summarize(count=n()) %>%
  top_n(10,count) %>%
  arrange(desc(count))
```

Another visual to compare the top 10 movies based on ratings is shown below. Movies in the late 1990s seemed to be the most popular, with Pulp Fiction (1994) ranking the highest rated movie, following Forrest Gump (1994) in second, and The Silence of the Lambs (1991) ranked third.

```
# Plot a bar graph of the top ranked titles
ranked_titles %>%
  ggplot(aes(x=reorder(title, count), y=count)) +
  geom_bar(stat='identity', fill="dark green", color = "black") +
  coord_flip(y=c(0, 35000)) +
  labs(x="", y="Number of ratings") +
  geom_text(aes(label= count), hjust=-0.2, size=3) +
  labs(title="Top 10 Movies Based on Ratings" , caption = "[1]")
```



Based from the earlier two figures, it can be hypothesized that the Pulp Fiction movie could be a movie with a genre ranked in the top three or five. It is, in fact, confirmed when viewing the table below. Pulp Fiction is comedy, crime, and drama movie, with both comedy and drama ranking in the top three genres. The majority of the top 10 titles also show this similar pattern, with a few exceptions to this pattern.

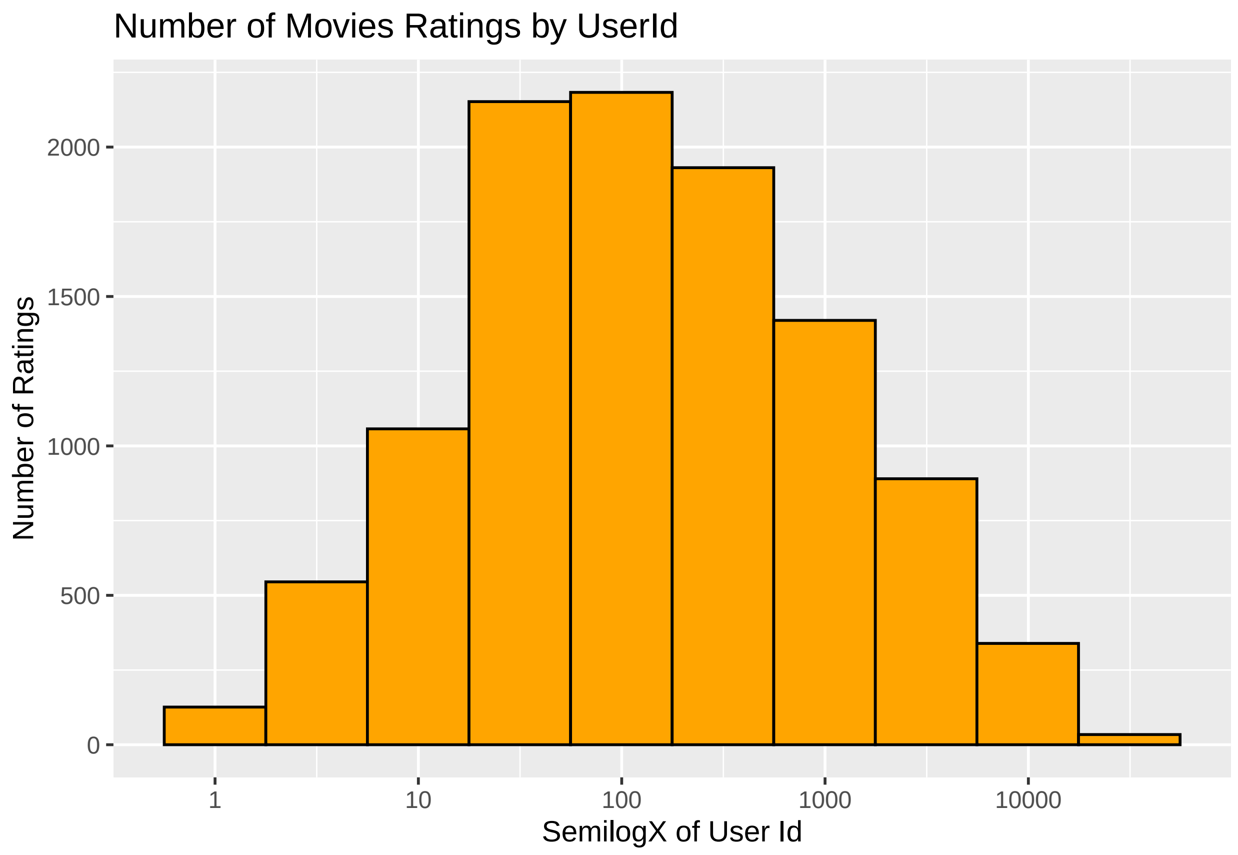
```
# Output the Top 10 Titles, and its Corresponding Titles and Count
kable(head(edx %>% group_by(title,genres) %>%
             summarize(count = n()) %>%
             top_n(10,count) %>%
             arrange(desc(count)) ,
           10)) %>%
  kable_styling(bootstrap_options = "basic", full_width = NULL ,
                position ="center") %>%
```

title	genres	count
Pulp Fiction (1994)	Comedy|Crime|Drama	31362
Forrest Gump (1994)	Comedy|Drama|Romance|War	31079
Silence of the Lambs, The (1991)	Crime|Horror|Thriller	30382
Jurassic Park (1993)	Action|Adventure|Sci-Fi|Thriller	29360
Shawshank Redemption, The (1994)	Drama	28015
Braveheart (1995)	Action|Drama|War	26212
Fugitive, The (1993)	Thriller	25998
Terminator 2: Judgment Day (1991)	Action|Sci-Fi	25984
Star Wars: Episode IV - A New Hope (a.k.a. Star Wars) (1977)	Action|Adventure|Sci-Fi	25672
Apollo 13 (1995)	Adventure|Drama	24284

```
  column_spec(1, bold = T) %>%
  column_spec(2) %>%
  column_spec(3)
```

Based on the histogram below, it seems to indicate that some users had a wider range of movie ratings than other users. Based on this figure, some people rated significantly more movies than other people. If ratings by each individual were to be somewhat equal, we would expect a more linearly-based pattern.

```
# Number of User Ratings by UserId
edx %>% group_by(movieId) %>% summarize(n = n()) %>%
  ggplot(aes(n)) + geom_histogram(fill = "orange", color = "black", bins = 10) +
  scale_x_log10() +
  ggtitle("Number of Movies Ratings by UserId") +
  xlab("SemilogX of User Id") +
  ylab("Number of Ratings")
```

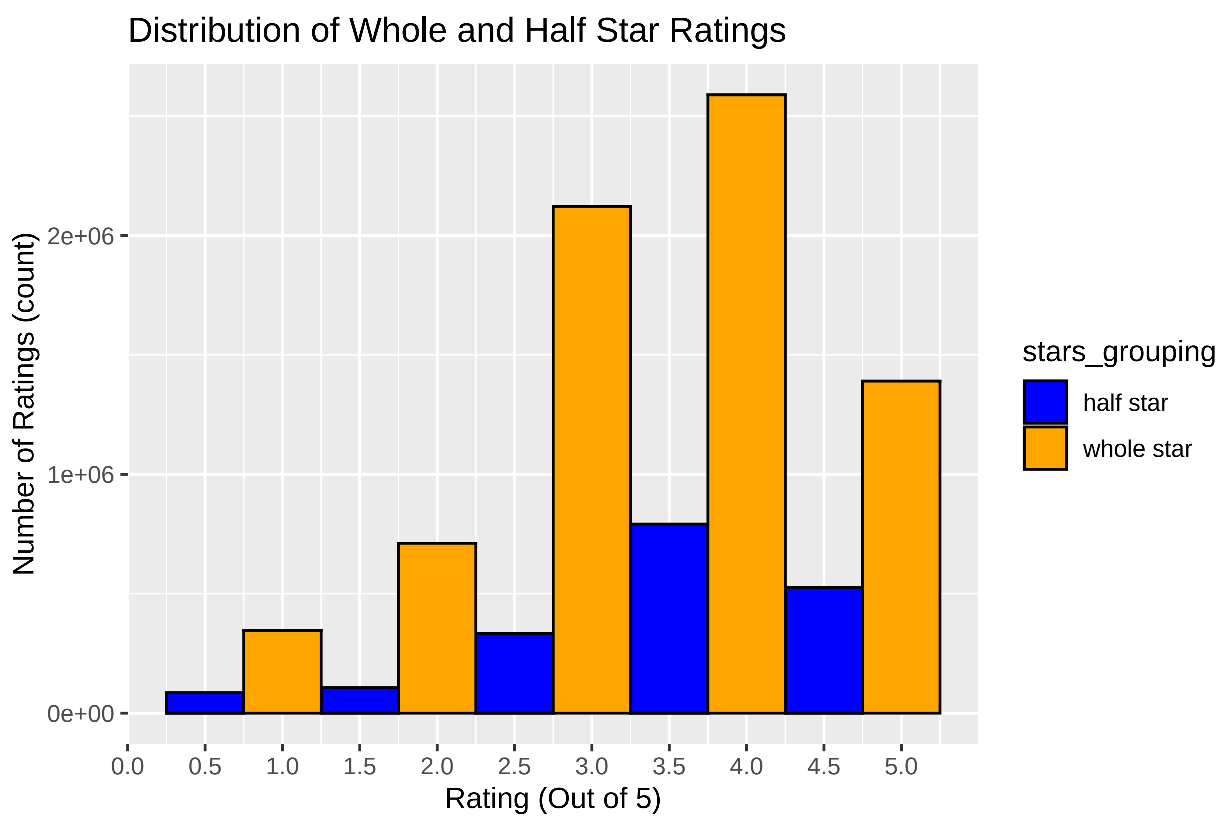


Another question asks the way a person rates a movie. Are whole star ratings more common than half stars? And if so, which rating value (out of five stars) is the most popular selection? Below, the stars grouping is prepared for plotting.

```r
# Generate a group between whole star and half star results
stars_grouping <-  ifelse((edx$rating == 1 |edx$rating == 2 | edx$rating == 3 |
                  edx$rating == 4 | edx$rating == 5) ,
                   "whole star",
                   "half star")

# Create a data frame to use for the plot generation
stars_ratings <- data.frame(edx$rating, stars_grouping)
```

The histogram is plotted below, with 4.0 stars being the most popular choice among users who rate the movies. Following 4.0 stars are 3.0 stars, then 5.0 stars, then two and one stars. No person rated 0 stars on any movie.

```r
# Generating a Histogram comparing the distribution
ggplot(stars_ratings, aes(x= edx.rating, fill = stars_grouping)) +
  geom_histogram( binwidth = 0.5, color = "black") +
  scale_x_continuous(breaks=seq(0, 5, by= 0.5)) +
  scale_fill_manual(values = c("half star"="blue", "whole star"="orange")) +
  labs(x = "Rating (Out of 5)", y = "Number of Ratings (count)",
       caption = '[1]') +
  ggtitle("Distribution of Whole and Half Star Ratings")
```



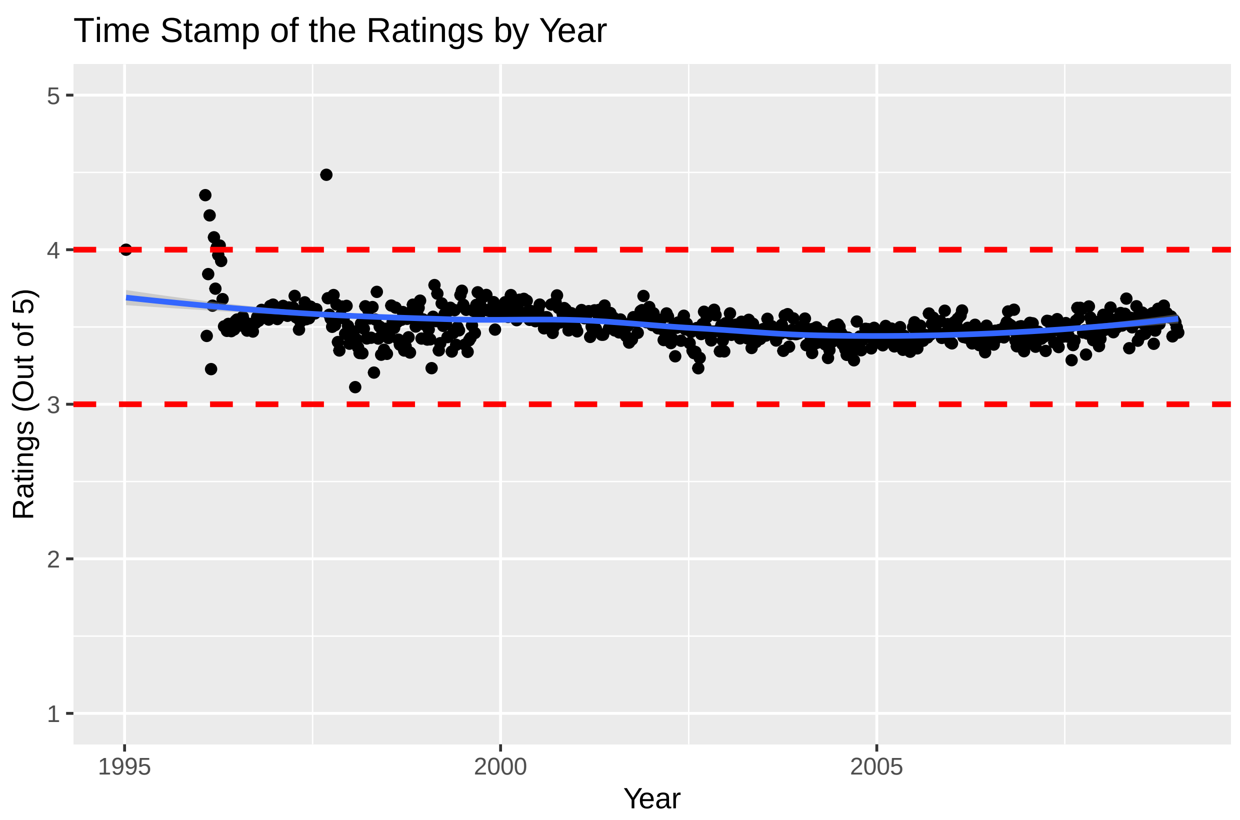
[1]

In both the whole star and half star patterns, a similar trend follows.

Whole Stars Highest Rating: 4 > 3 > 5 > 2 > 1 :Lowest Rating Half Stars Highest Rating: 3.5 > 4.5 > 2.5 > 1.5 > 1.0 :Lowest Rating

To ensure there are no significant changes in rating patterns throughout the years, a time stamp plot is made, following the trends of ratings each year. Based on the results below, it seems that all ratings fell within the 3-4 star threshold. The average stayed stagnant around 3.5 stars, showing that there is little to no effects on the ratings based on time. This assumption will help simplify the understanding of the data significantly.

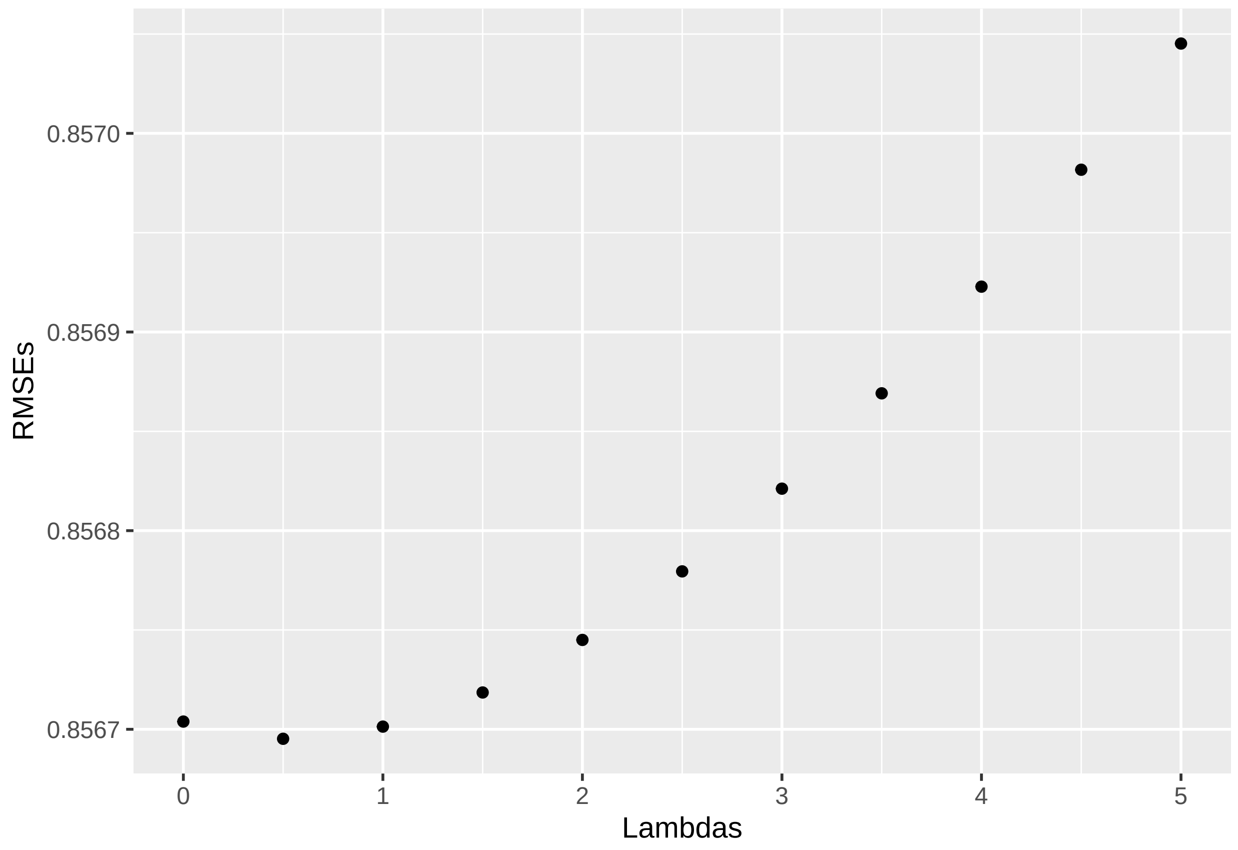
```r
# Generate a plot of the timestamp, with x axis being year, y being ratings
edx %>%
  mutate(date = round_date(as_datetime(timestamp), unit = "week")) %>%
  group_by(date) %>%
  summarize(rating = mean(rating)) %>%
  ggplot(aes(date, rating)) +
  geom_point() +
  geom_smooth(method = 'loess', formula = y ~ x) +
  ggtitle("Time Stamp of the Ratings by Year")+
  labs(x = "Year", y = "Ratings (Out of 5)", caption = "[1]") +
  ylim(1, 5) +
  geom_hline(yintercept=3, linetype="dashed",
               color = "red", size=1) +
  geom_hline(yintercept=4, linetype="dashed",
               color = "red", size=1)
```



[1]

The root mean squared error (RMSE) calculation is next conducted. Based on the plot, the RMSE increases with lambda. There is a minimum value present, when lambda = 0.5. At this respective value, the RMSE is 0.8567.

```
# Define the RMSE Function
RMSE <- function(ratings_T, ratings_P){
  sqrt(mean((ratings_T - ratings_P)^2))
}

# Observing the RMSE based on lambdas value
lambdas <- seq(0,5,.5)
rmses <- sapply(lambdas, function(l){
  mu <- mean(edx$rating)

  b_i <- edx %>%
    group_by(movieId) %>%
    summarize(b_i = sum(rating - mu)/(n() + l))

  b_u <- edx %>%
    left_join(b_i, by='movieId') %>%
    group_by(userId) %>%
    summarize(b_u = sum(rating - b_i - mu)/(n() +l))

  ratings_P <- edx %>%
    left_join(b_i, by = "movieId") %>%
    left_join(b_u, by = "userId") %>%
    mutate(pred = mu + b_i +  b_u) %>% .$pred

  return(RMSE(ratings_P, edx$rating))
})

qplot(lambdas, rmses) + xlab("Lambdas") + ylab("RMSEs")
```



Interpreting the graph, the RMSE value is calculated below:

```
# Using the which.min() command to determine the lambdas value
print(c("The RMSEs value is:", round(min(rmses), 5)))
```

```
## [1] "The RMSEs value is:" "0.8567"
```

```
print(c("The lambdas (l) value is:", lambdas[which.min(rmses)]))
```

```
## [1] "The lambdas (l) value is:" "0.5"
```

After tuning with a new lambdas value, the new RMSE value becomes 0.8258.

```
# Use the Model on the Validation Data for Confirmation
mu <- mean(validation$rating)
l <- 0.5
b_i <- validation %>%
  group_by(movieId) %>%
  summarize(b_i = sum(rating - mu)/(n() + 1))

b_u <- validation %>%
  left_join(b_i, by='movieId') %>%
  group_by(userId) %>%
  summarize(b_u = sum(rating - b_i - mu)/(n() +1))

ratings_P <- validation %>%
  left_join(b_i, by = "movieId") %>%
  left_join(b_u, by = "userId") %>%
  mutate(pred = mu + b_i +  b_u) %>% .$pred

RMSE(ratings_P, validation$rating)
```

```
## [1] 0.8258487
```

4. Conclusion

The MovieLens data set provided countless data interpretations to the user. Significant discoveries include the popular genre selections, distribution of the star ratings, and top movie titles. Additionally, discovering that the rating average remained relatively constant within the past decades has shown that people from different generations tend to rate movies similarly. As a result, it would be of best interest to emphasize areas such as genre, title of the movies.

The RMSE value resulted in a value of 0.8258, a value that is close to 1, but not overtrained such that some potential errors may arise.

There are current limitations based on the userId and the provided details, however; more work is needed to enhance the movie recommendation system. For example, future work could include principal component analysis (PCA), a technique widely used to reduce the dimensionality of a large data set, and to observe any correlating information between different variables that may have been unnoticed in the earlier data interpretation stage. Using this powerful tool, other statistical analyses can be conducted to better suit the users' needs.

5. References

1. F. Maxwell Harper and Joseph A. Konstan. 2015. The MovieLens Datasets: History and Context. ACM Transactions on Interactive Intelligent Systems (TiiS) 5, 4, Article 19 (December 2015), 19 pages. DOI=http://dx.doi.org/10.1145/2827872