MovieLens Project

Justin McQuown

April 2024

Contents

Introduction section

The MovieLens dataset is a database with over 20 million ratings for over 27,000 movies by almost 140,000 users which can be viewed here: https://grouplens.org/datasets/movielens/. A subset of this database will be used to create a movie recommendation system for the Harvardx Data Science capstone project. The subset of the data is provided by the course and can be viewed at the following location: https://grouplens.org/datasets/movielens/10m/. The data used for this project has 10 million ratings from 10,000

movies by 72,000 users and was released in January of 2009. The data set includes a userID, movieID, movie title that includes the year of the release of the movie as well as a unix style timestamp (seconds since January 1, 1970) along with the genre of the movie.

The goal of this project is to predict movie ratings using the techniques discusses in the series of classes that are prerequisites for the capstone course. Code provided by the course created training and validation sets that are to be used for the final accuracy measurement of the generated model. The validation or final_holdout_test set is 10% of the original data and is not used in the construction of the model. As per the instructions the the final_holdout_test data is not to be used during the development of the model and used just for the final validation test so the edx data set was again broken down into a test and validation sets where the validation set is 10% of the edx data. This additional validation set is not used during the development of the models but is used to verify the intermediate steps.

Since the data is set is fairly large fairly simple techniques were used and an *lm* model was not generated. Also, due to the fact that I'm running out of time in this course I didn't go into more advanced prediction techniques and will be saving those for the final create your own project. The analysis followed the general pattern of the text book and the Machine learning course but did not just copy code as expressly written in the instructions.

Setting up the environment

The packages and libraries used throughout the analysis need to be loaded on startup. The analysis was a series of trial and error events so some of these libraries may no longer be used in the final analysis but were need as part of the discovery process.

```
# Note: this process could take a couple of minutes

# Installing required packages
if(!require(tidyverse)) install.packages("tidyverse", repos = "http://cran.us.r-project.org")
if(!require(caret)) install.packages("caret", repos = "http://cran.us.r-project.org")
if(!require(data.table)) install.packages("data.table", repos = "http://cran.us.r-project.org")
if(!require(ggplot2)) install.packages("ggplot2");
if(!require(stringr)) install.packages("string");
if(!require(tidyr)) install.packages("tidyr");
if(!require(data.table)) install.packages("data.table")
if(!require(summarytools)) install.packages("summarytools")
if(!require(gridExtra)) install.packages("gridExtra")
if(!require(kableExtra)) install.packages(("kableExtra"))

# Loading required packages
library(tidyverse)
library(caret)
library(rafalib)
library(ggplot2)
library(knitr)
library(raster)
library(dslabs)
library(data.table)
library(dplyr)
library(summarytools)
library(gridExtra)
library(kableExtra)
```

Helper functions

As part of the development and learning experience several functions were created. These are listed here for reference.

```
#Basic function to report the basic stats of a category
statFunc <- function(inData, group, groupy = TRUE) {
  if( groupy ) {
    localData <- inData %>% group_by(inData[[group]])
  }
  else {
    localData <- inData
  }

  localData %>% summarize(ratings=n(), avg=mean(rating), med=median(rating),
                          stdev=sd(rating), high=max(rating), low=min(rating))
}

# The RMSE function that will be used for calculations
RMSE <- function(verificationRations = NULL, predictedRatings = NULL) {
  sqrt(mean((verificationRations - predictedRatings)^2))
}
```

Loading the data

The initial data loading, data wrangling, and breaking the data into test and validation sets was provided by the course. Here's the provided code a reference.

Data Exploration

In an effort to get a better sense of the data and important features we're going to take some basic statistics and generate some plots of the data that may help provide some insight of what is techniques are best for this data. Here's a snip of the data set used throughout to get a general idea of the data format and important information.

```
#Here's a glimpse of the data set
edx %>% as_tibble()
```

```
## # A tibble: 9,000,055 x 6
##    userId movieId rating timestamp title                                genres
##     <int>   <int>  <dbl>     <int> <chr>                                <chr>
##  1      1     122      5 838985046 Boomerang (1992)                     Comed~
##  2      1     185      5 838983525 Net, The (1995)                      Actio~
##  3      1     292      5 838983421 Outbreak (1995)                      Actio~
##  4      1     316      5 838983392 Stargate (1994)                      Actio~
##  5      1     329      5 838983392 Star Trek: Generations (1994)        Actio~
##  6      1     355      5 838984474 Flintstones, The (1994)              Child~
##  7      1     356      5 838983653 Forrest Gump (1994)                  Comed~
##  8      1     362      5 838984885 Jungle Book, The (1994)              Adven~
##  9      1     364      5 838983707 Lion King, The (1994)                Adven~
## 10      1     370      5 838984596 Naked Gun 33 1/3: The Final Insult (1~ Actio~
## # i 9,000,045 more rows
```

Next we're going to calculate the basic statistics of the data:

```
#In an effort to get a feel for the data I'm going to take a few basic statistics
uniqueUsers = length(unique(edx$userId));
uniqueMovies = length(unique(edx$movieId));
uniqueGenres = str_extract_all(unique(edx$genres), "[^|]+") %>%
               unlist() %>%
               unique()


#Calculate the statistics group by a column (except for total stats)
totalStats = statFunc(edx, FALSE, FALSE);
userIdStats = statFunc(edx, "userId", TRUE);
movieStats = statFunc(edx, "movieId", TRUE);
genreStats = statFunc(edx, "genres", TRUE)

colnames(totalStats) <- c("Ratings", "Avg", "Median", "Std. Dev", "High", "Low")
kable(totalStats) %>%
kable_styling(bootstrap_options = c("striped", "hover", "condensed", "responsive"),
              position = "center",
              font_size = 12)
```

Ratings	Avg	Median	Std. Dev	High	Low
9000055	3.512465	4	1.060331	5	0.5

Next we're going to look at the total counting stats for the edx dataset:

```
tab <- cbind(uniqueUsers,uniqueMovies,length(uniqueGenres), length(unique(edx$genres)))
rownames(tab) <- 'Data Set'
colnames(tab) <- c('Users', 'Movies', 'Genres', 'Total Genres');
kable(tab) %>%
kable_styling(bootstrap_options = c("striped", "hover", "condensed", "responsive"),
              position = "center",
              font_size = 12)
```

	Users	Movies	Genres	Total Genres
Data Set	69878	10677	20	797

Now we're going to go into a little more depth on the data. First, the timestamp of the rating information is going to grabbed from the data sets along with the year the movie was release. The time information is also going to parsed out of the final_holdout_test data set so that it can be used in the final analysis if needed.

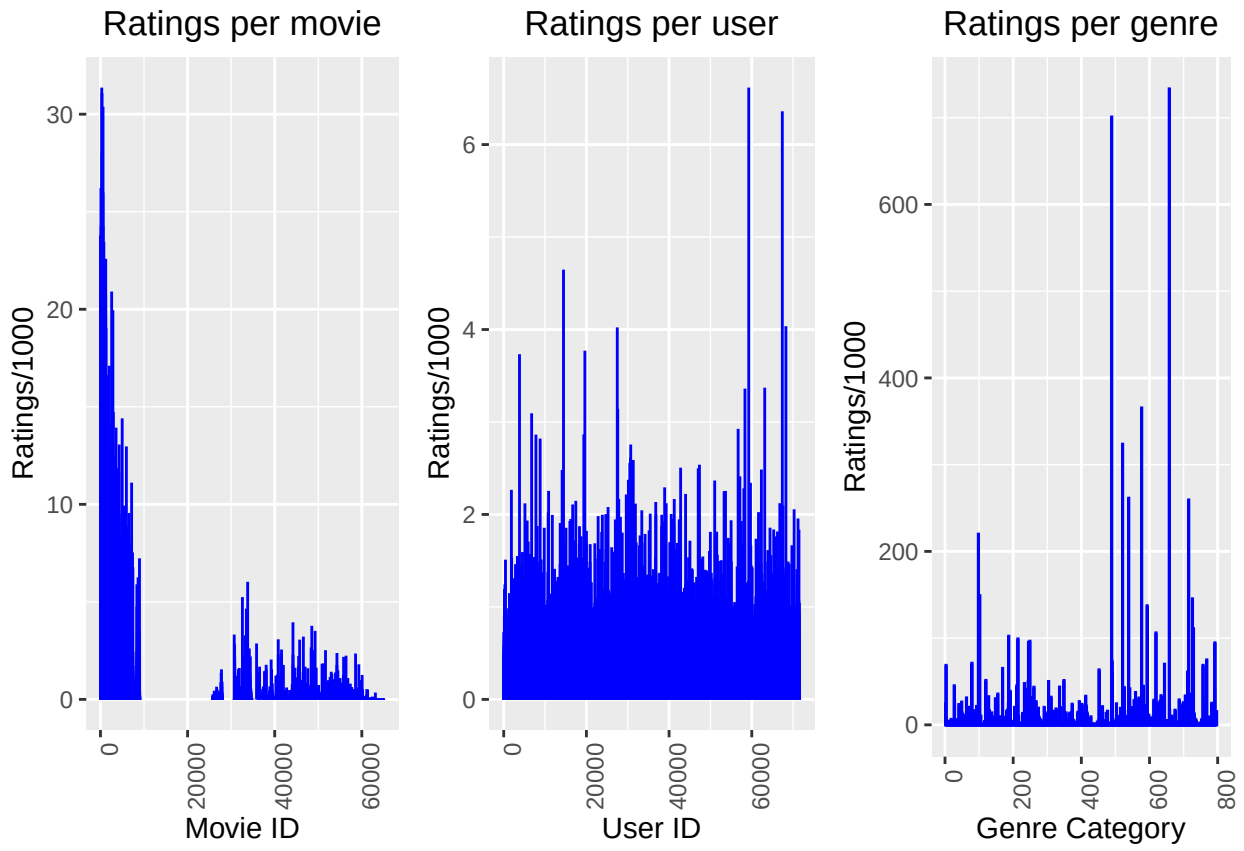
```
#There seems to be more information available by parsing out the timestamp
#Extract the timestamp and convert it to a human readable date
edx$date <- as.POSIXct(edx$timestamp, origin="1970-01-01");

#Pull out the year and month from the timestamp of the rating
edx$ratingYear = as.numeric(format(edx$date, "%Y"));
edx$ratingMonth = as.numeric(format(edx$date, "%m"));
```

```
# Get the year of the movie. Each movie title seemed to have the release year appended to the end
# Get the parenthesis and what is inside
k <- str_extract_all(edx$title, "\\([^()]+\\)$");
# Remove parenthesis
k <- substring(k, 2, nchar(k)-1);
# Remove all unwanted leftover special characters and transpose the data so it will
# be a new column in the data frame
k <- as.data.frame(matrix(as.numeric(gsub(")","", k))));
edx$movieYear = k$V1;
edx$yearDelta = as.numeric(edx$ratingYear)-as.numeric(edx$movieYear);

#Separate the genres, this will make multiple movie entries for titles that have multiple genres listed
edx <- edx %>% separate_rows(genres,sep = "\\|") %>% mutate(value=1)
genresIndependent <- edx %>% group_by(genres) %>% summarize(n=n())

#Do the same thing above for the final_holdout_test data. Should make this a function but that's for ne
#There seems to be more information available by parsing out the timestamp
#Extract the timestamp and convert it to a human readable date
final_holdout_test$date <- as.POSIXct(final_holdout_test$timestamp, origin="1970-01-01");

#Pull out the year and month from the timestamp of the rating
final_holdout_test$ratingYear = as.numeric(format(final_holdout_test$date, "%Y"));
final_holdout_test$ratingMonth = as.numeric(format(final_holdout_test$date, "%m"));

# Get the year of the movie. Each movie title seemed to have the release year appended to the end
# Get the parenthesis and what is inside
k <- str_extract_all(final_holdout_test$title, "\\([^()]+\\)$");
# Remove parenthesis
k <- substring(k, 2, nchar(k)-1);
# Remove all unwanted leftover special characters and transpose the data so it will
# be a new column in the data frame
k <- as.data.frame(matrix(as.numeric(gsub(")","", k))));
final_holdout_test$movieYear = k$V1;
final_holdout_test$yearDelta = as.numeric(final_holdout_test$ratingYear)-as.numeric(final_holdout_test$

#Separate the genres, this will make multiple movie entries for titles that have multiple genres listed
#Separate the genres for the final test data as it will be needed later ;) ;)
final_holdout_test <- final_holdout_test %>% separate_rows(genres,sep = "\\|") %>% mutate(value=1)
```

Now the number of ratings are going to be plotted against the movie IDs, user IDs, and genre categories to see if there's useful information there.

```
#Plotting some of the data to get an idea of how things look to determine what variables
#are important for the prediction process

#Raw data plots before any real analysis
ratingsPerTitle <- ggplot(movieStats, aes(`inData[[group]]`, ratings/1000))+geom_col(col="blue") +
                    labs(
                      title = "Ratings per movie",
                      x = "Movie ID",
                      y = "Ratings/1000"
                    ) +
                    theme(axis.text.x = element_text(angle=90, hjust=1), plot.title = element_text(hjus
```

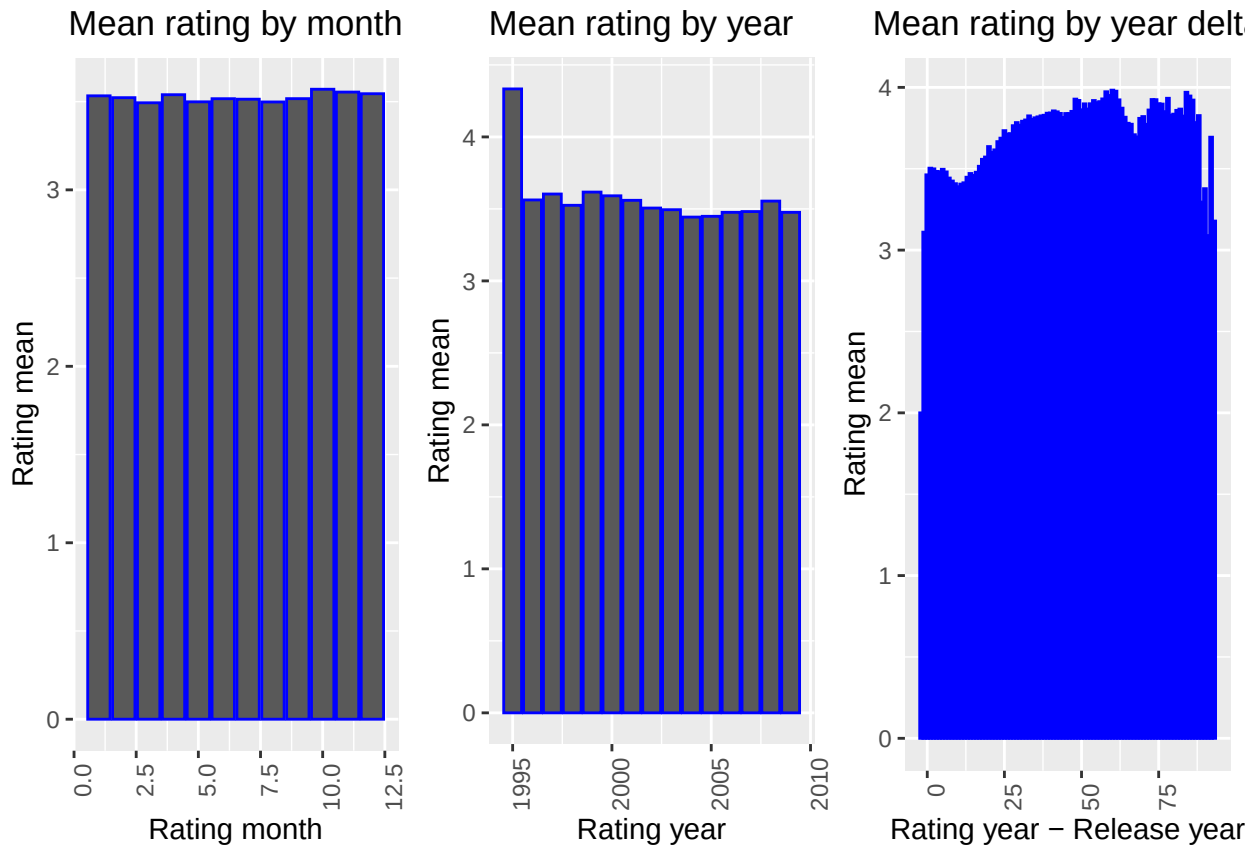
```
ratingsPerUser <- ggplot(userIdStats, aes(`inData[[group]]`, ratings/1000))+geom_col(col="blue") +
                    labs(
                      title = "Ratings per user",
                      x = "User ID",
                      y = "Ratings/1000"
                    ) +
                    theme(axis.text.x = element_text(angle=90, hjust=1), plot.title = element_text(hjust

ratingsPerGenre <- ggplot(genreStats, aes(1:length(genreStats[[1]]), ratings/1000))+geom_col(col="blue"
                    labs(
                      title = "Ratings per genre",
                      x = "Genre Category",
                      y = "Ratings/1000"
                    ) +
                    theme(axis.text.x = element_text(angle=90, hjust=1), plot.title = element_text(hjust

grid.arrange(ratingsPerTitle, ratingsPerUser, ratingsPerGenre, ncol = 3);
```



The counting stats did not provide a whole lot of information other than the data is not evenly distributed. Several of the movies are rated much more frequently than others and some users frequently rate movies while others not so much. The genre information probably doesn't tell us a whole lot because some movies could be in multiple categories if they are individually broken out. We'll look more into this later.

Next, we'll plot the movie ratings mean by the given time/date information.

```r
#More meaningful plots of data after some processing
#Mean of ratings per some feature
meanRatingPerMonth <-  edx %>%
                    group_by(ratingMonth) %>%
                    summarize(meanRating = mean(rating)) %>%
                    ggplot(aes(ratingMonth, meanRating)) +
                    geom_col(col="blue") +
                    labs(
                      title = "Mean rating by month",
                      x = "Rating month",
                      y = "Rating mean"
                    ) +
                    theme(axis.text.x = element_text(angle=90, hjust=1), plot.title = element_text(hjus

meanRatingPerYear <-  edx %>%
  group_by(ratingYear) %>%
  summarize(meanRating = mean(rating)) %>%
  ggplot(aes(ratingYear, meanRating)) +
  geom_col(col="blue") +
  labs(
    title = "Mean rating by year",
    x = "Rating year",
    y = "Rating mean"
  ) +
  theme(axis.text.x = element_text(angle=90, hjust=1))

meanRatingsvsDelta <- edx %>%
                    group_by(yearDelta) %>%
                    summarize(deltamean = mean(rating)) %>%
                    ggplot(aes(yearDelta, deltamean)) +
                    geom_col(col="blue") +
                    labs(
                      title = "Mean rating by year delta",
                      x = "Rating year - Release year",
                      y = "Rating mean"
                    ) +
                    theme(axis.text.x = element_text(angle=90, hjust=1), plot.title = element_text(hjus


grid.arrange(meanRatingPerMonth, meanRatingPerYear, meanRatingsvsDelta, ncol = 3);
```

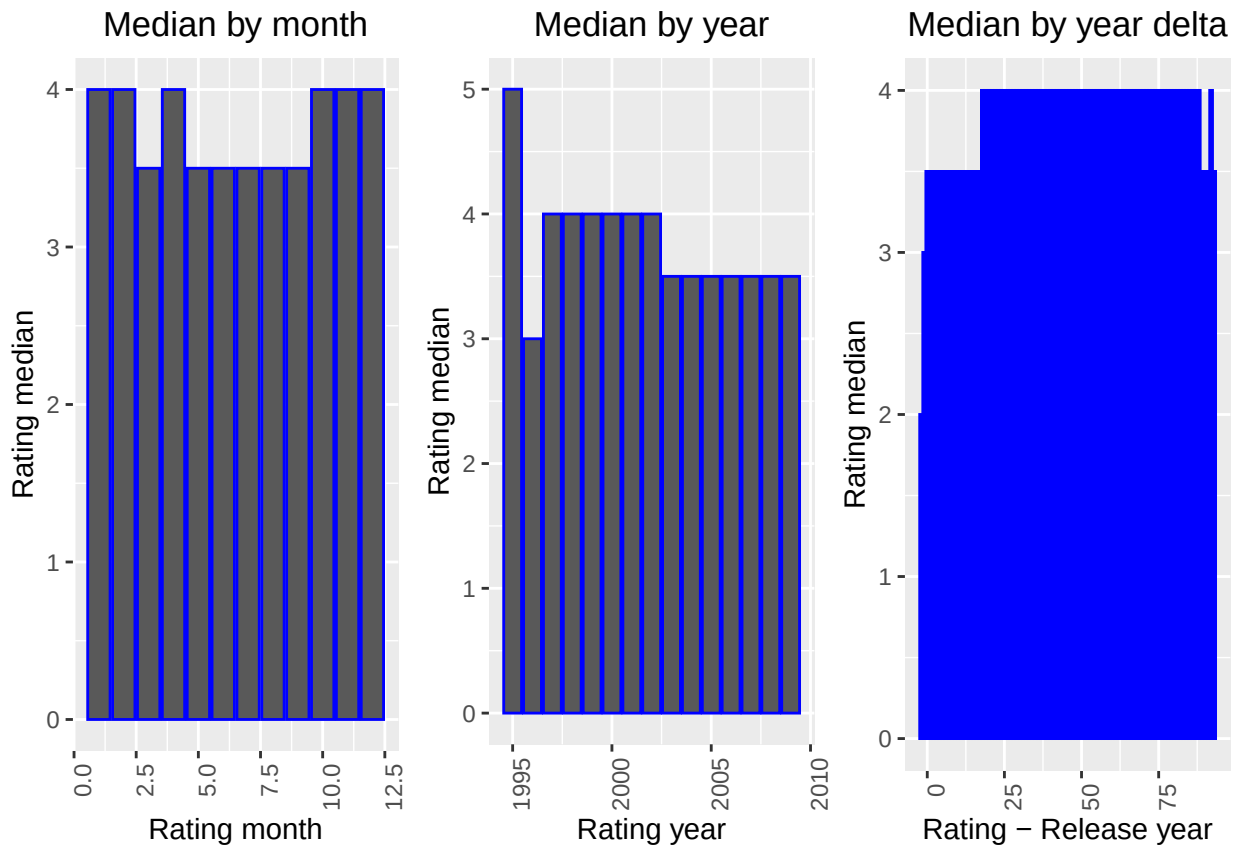


The mean information was somewhat interesting but I'm not sure it benefits us all that much. The movie ratings by month showed slightly higher ratings during the winter months which may be related to the awards season. The gap really isn't that big though so I don't think this warrants additional analysis, the year of the rating also has some highlights like the early to mid 90's had a disproportionately higher ratings but in looking at the movie list there's also a whole lot big hits. It may be coincidence but we'll look into movie year effects in the analysis section. The delta between the rating year and the movie release year shows that if the movie is rate within the first few years, the rating is generally lower, may have to looking into this effect as well.

Next we're going to do a very similar analysis using the median ratings instead of the means.

```
#Median of ratings per some feature
medianRatingPerMonth <- edx %>%
                        group_by(ratingMonth) %>%
                        summarize(medianRating = median(rating)) %>%
                        ggplot(aes(ratingMonth, medianRating)) +
                        geom_col(col="blue") +
                        labs(
                          title = "Median by month",
                          x = "Rating month",
                          y = "Rating median"
                        ) +
                        theme(axis.text.x = element_text(angle=90, hjust=1), plot.title = element_tex

medianRatingPerYear <-edx %>%
                        group_by(ratingYear) %>%
                        summarize(medianRating = median(rating)) %>%
```

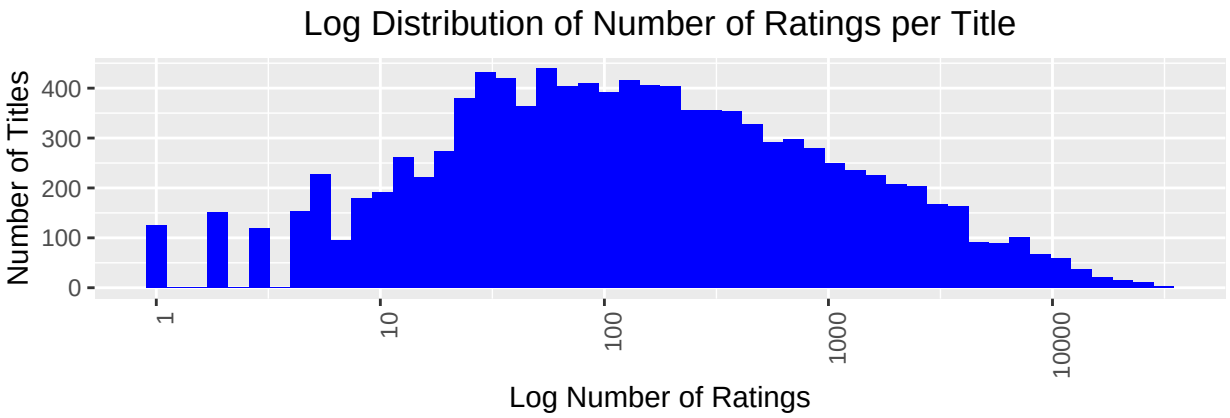
```
                          ggplot(aes(ratingYear, medianRating)) +
                          geom_col(col="blue") +
                          labs(
                            title = "Median by year",
                            x = "Rating year",
                            y = "Rating median"
                          ) +
                          theme(axis.text.x = element_text(angle=90, hjust=1), plot.title = element_tex

medianRatingsvsDelta <- edx %>%
                          group_by(yearDelta) %>%
                          summarize(deltaMedian = median(rating)) %>%
                          ggplot(aes(yearDelta, deltaMedian)) +
                          geom_col(col="blue") +
                          labs(
                            title = "Median by year delta",
                            x = "Rating - Release year",
                            y = "Rating median"
                          ) +
                          theme(axis.text.x = element_text(angle=90, hjust=1), plot.title = element_tex

grid.arrange(medianRatingPerMonth, medianRatingPerYear, medianRatingsvsDelta, ncol = 3);
```

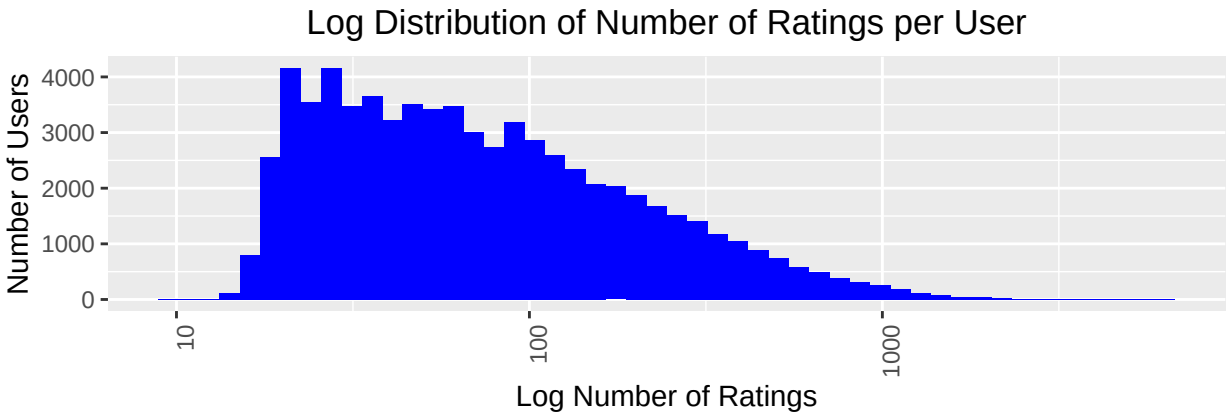


The median data really didn't show much additional information than what was discovered during the mean plots so this as far as we'll go with the median information.

Next, the distribution data will be plotted on a logarithmic scale to see if there's any helpful insight. First we'll plot the total number of users and titles vs. the distribution of ratings.

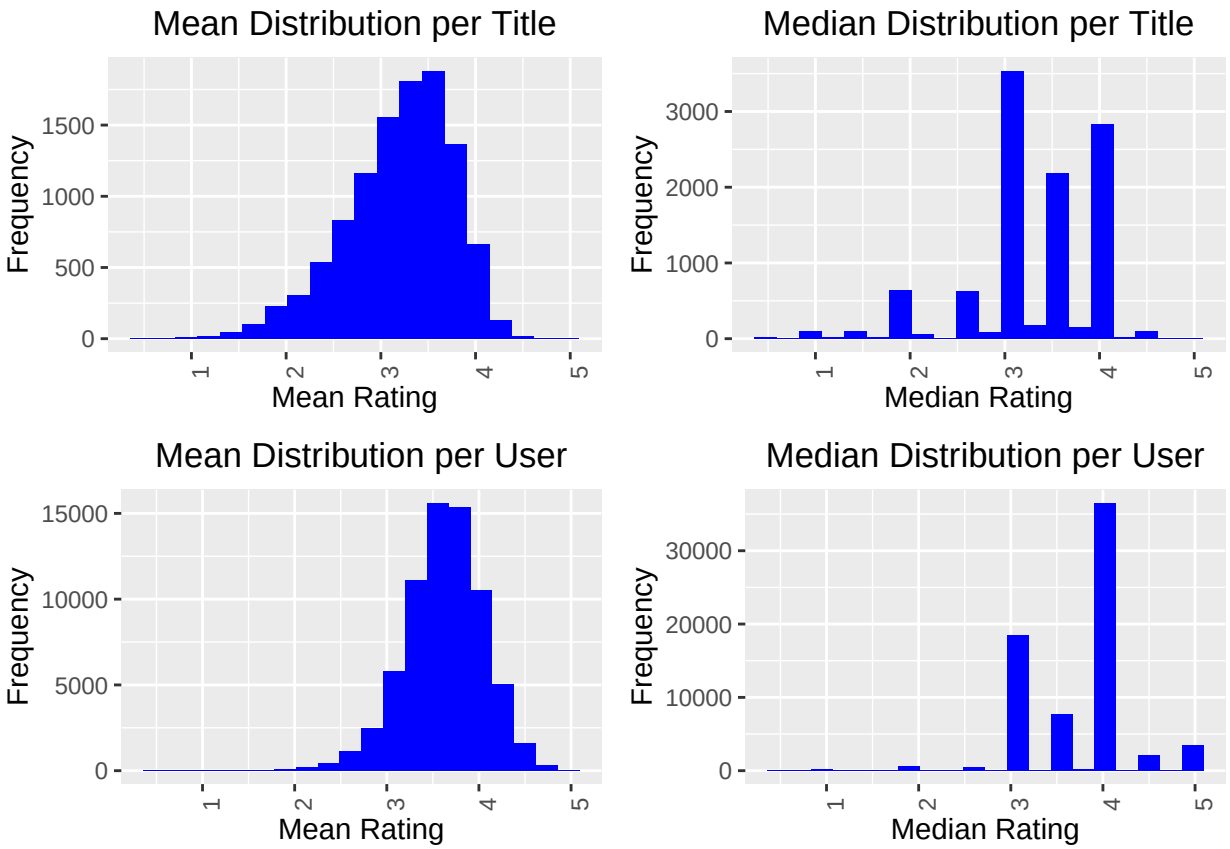
```
#Distribution plots
#Ratings per user histogram
ratingsPerUser <- userIdStats %>%
                    ggplot(aes(x=ratings)) +
                    geom_histogram(bins=50, fill="blue") +
                    scale_x_log10() +
                    labs(title = "Log Distribution of Number of Ratings per User",
                         x = "Log Number of Ratings",
                         y = "Number of Users"
                    )+
                    theme(axis.text.x = element_text(angle=90, hjust=1), plot.title = element_text(hjus

ratingsPerMovie <- movieStats %>%
  ggplot(aes(x=ratings)) +
  geom_histogram(bins=50, fill="blue") +
  scale_x_log10() +
  labs(title = "Log Distribution of Number of Ratings per Title",
       x = "Log Number of Ratings",
       y = "Number of Titles"
  )+
  theme(axis.text.x = element_text(angle=90, hjust=1), plot.title = element_text(hjust = 0.5))

grid.arrange(ratingsPerMovie, ratingsPerUser, nrow = 2);
```





These initial plots seem to confirm what we already now in that some movies/users are much more active on the ratings than others. Next we'll go a little deeper and plot the mean and median distribution data.

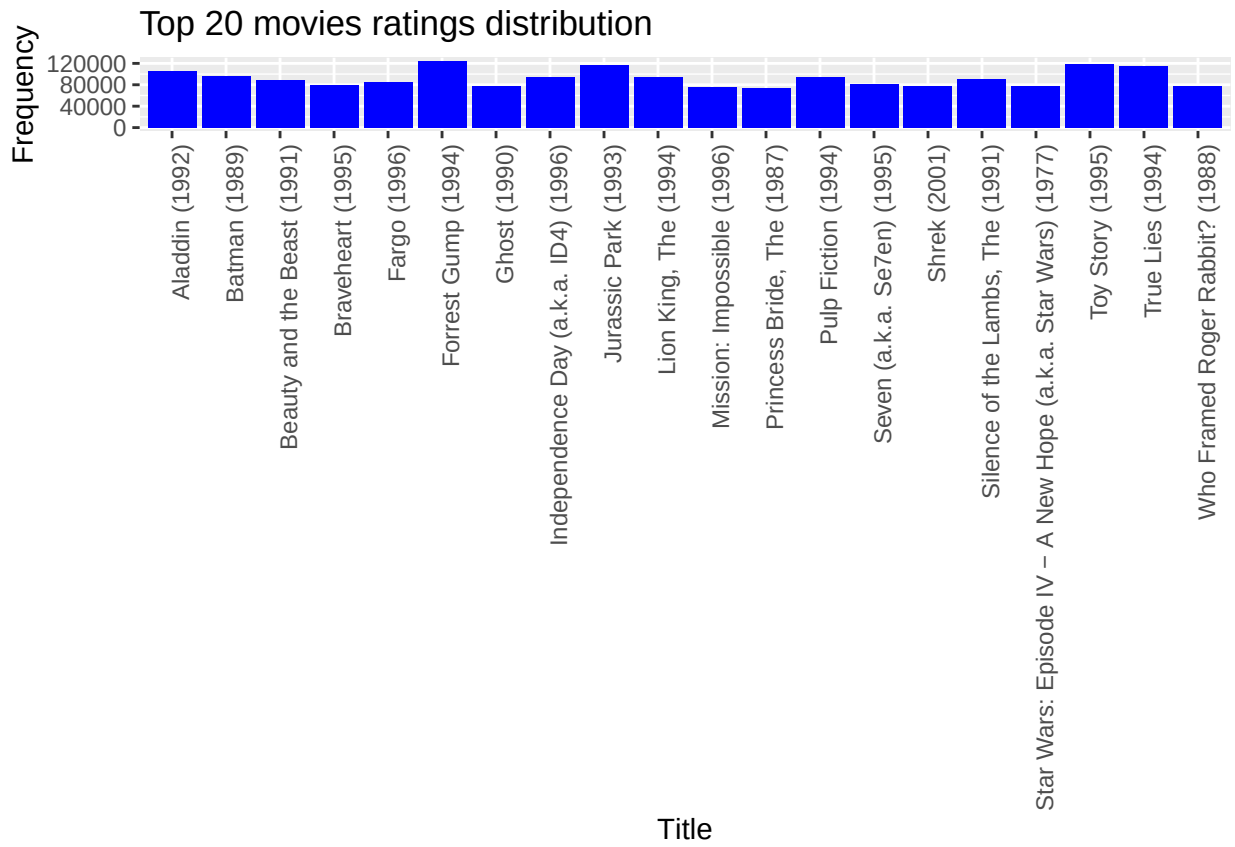
```
# Mean rating per movie
meanPerTitle <- edx %>%
                  group_by(movieId) %>%
                  summarise(meanRating = mean(rating)) %>%
                  ggplot(aes(meanRating)) +
                  geom_histogram(bins=20, fill = "blue") +
                  labs(title = "Mean Distribution per Title",
                       x = "Mean Rating",
                       y = "Frequency"
                  )+
                  theme(axis.text.x = element_text(angle=90, hjust=1), plot.title = element_text(hjust =

medianPerTitle <- edx %>%
                    group_by(movieId) %>%
                    summarise(medianRating = median(rating)) %>%
                    ggplot(aes(medianRating)) +
                    geom_histogram(bins=20, fill = "blue") +
                    labs(title = "Median Distribution per Title",
                         x = "Median Rating",
                         y = "Frequency"
                    )+
                    theme(axis.text.x = element_text(angle=90, hjust=1), plot.title = element_text(hjus
# Mean rating per user
meanPerUser <- edx %>%
                 group_by(userId) %>%
                 summarise(meanRating = mean(rating)) %>%
                 ggplot(aes(meanRating)) +
                 geom_histogram(bins=20, fill = "blue") +
                 labs(title = "Mean Distribution per User",
                      x = "Mean Rating",
                      y = "Frequency"
                 )+
                 theme(axis.text.x = element_text(angle=90, hjust=1), plot.title = element_text(hjust = (

medianPerUser <- edx %>%
                  group_by(userId) %>%
                  summarise(medianRating = median(rating)) %>%
                  ggplot(aes(medianRating)) +
                  geom_histogram(bins=20, fill = "blue") +
                  labs(title = "Median Distribution per User",
                       x = "Median Rating",
                       y = "Frequency"
                  )+
                  theme(axis.text.x = element_text(angle=90, hjust=1),plot.title = element_text(hjust =

grid.arrange(meanPerTitle, medianPerTitle, meanPerUser, medianPerUser, nrow = 2, ncol = 2)
```



Both the mean and median distributions show similar effects so we'll stick to using the mean. This also somewhat confirms our initial findings of the mean being around 3.5 for both movie titles and users. This is kind of interesting that both means are vary similar though the kurtosis on the user distribution is lower and the skewness of the title distribution is more left sided. Not sure if we'll be able to use that in this analysis though that could be something that's looked at in the future.

We'll wrap up the data exploration next by plotting the top 20 movies according to the number of ratings. First it will be shown in a distribution plot and then just in a table.

```
#Top rated movies
topMoviesFrequency <- edx %>%
  group_by(title) %>%
  summarise(count = n()) %>%
  arrange(desc(count)) %>%
  head(n=20) %>%
  ggplot(aes(title, count)) +
  geom_col(fill = "blue") +
  theme(axis.text.x = element_text(angle = 90, hjust = 1)) +
  labs(title = "Top 20 movies ratings distribution",
       x = "Title",
       y = "Frequency")
topMoviesFrequency
```




```
edx %>%
  group_by(title) %>%
  summarise(count = n()) %>%
  arrange(desc(count)) %>%
  head(n=20) %>%
  kable() %>%
  kable_styling(bootstrap_options = c("striped", "hover", "condensed", "responsive"),
                position = "center",
                font_size = 12)
```

title	count
Forrest Gump (1994)	124316
Toy Story (1995)	118950
Jurassic Park (1993)	117440
True Lies (1994)	114115
Aladdin (1992)	105865
Batman (1989)	97108
Lion King, The (1994)	94605
Pulp Fiction (1994)	94086
Independence Day (a.k.a. ID4) (1996)	93796
Silence of the Lambs, The (1991)	91146

Beauty and the Beast (1991)	89145
Fargo (1996)	85580
Seven (a.k.a. Se7en) (1995)	81244
Braveheart (1995)	78636
Shrek (2001)	78378
Ghost (1990)	77440
Star Wars: Episode IV - A New Hope (a.k.a. Star Wars) (1977)	77016
Who Framed Roger Rabbit? (1988)	76825
Mission: Impossible (1996)	75968
Princess Bride, The (1987)	74045

The distribution and plot really only show us that some movies are rated way more frequently than others and the ones that are more frequently rated tend to be box office hits. On to the analysis.

Analysis section

Create development test set

As per the instructions of the course, the final_holdout_test data set should not be used during the model development so here we're creating a new subset of data to develop the model independent on the final_holdout_test data.

```
# The project instructions say to not use the final holdout data until the final model so here we're go
# to make a training and test data set from the edx data
testIndex <-createDataPartition(y = edx$rating, times = 1, p = 0.1, list = F)
trainEdx <-edx[-testIndex,]
tempEdx <-edx[testIndex,]

#Again, Make sure userId and movieId are in both the train and test sets
testEdx <- tempEdx %>%
  semi_join(trainEdx, by = "movieId") %>%
  semi_join(trainEdx, by = "userId")

#Add the Rows removed from the edx_test back into edx_train
removed <-anti_join(tempEdx, testEdx)
trainEdx <-rbind(trainEdx, removed)
rm(tempEdx, testIndex, removed)
```

Analysis precursor

In total there are 69878 unique users providing ratings and 10677 unique movies. That's a lot of combinations between users and movies. The test data has around 9 million rows so this reinforces the distribution plots above that implies users have not rated every movie. This leaves us with a sparse matrix due to only 3.13% of all possible combinations being used.

With respect to genres it can be shown that some genres have a lot more ratings than others and the mean rating is different between genres. As shown in the quiz to start this project the most popular rated genre types are Drama and Comedy. Drama and film-noir are some of the higher rated genre types, while horror is the worst rated.

As described above, the large nature of the dataset makes modeling the data using a function like *lm* to resource intensive for this application. Working around this limitation with the using the least square estimates loss function manually keeps us in the ballpark. The loss function can be found in the helper function is described by the equation: $RMSE = \sqrt{\frac{1}{N}\sum_{u,i}(\hat{y}_{u,i} - y_{u,i})^2}$

Also, remember that we're using a subset of the data from the edx set for development of the model but will be using the whole set in the validation at the end.

Initial model - Simple average

We're starting with the simplest model, a straight mean which ignores the effects of the user, movie, time, and genre. Since none of the data is less than 0.5 or higher than 5 we're going to clamp our results with these values. The initial model will follow the following equation:

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

Where u is the index for users and i is for movies.

The initial estimate for μ is the average of all the train data ratings ratings, which is 3.5270067.

```
mu <- mean(trainEdx$rating)
meanRMSE <- RMSE(testEdx$rating, mu)

resultsMean <- tibble(Method = "Mean", RMSE = meanRMSE)
kable(resultsMean) %>%
  kable_styling(bootstrap_options = c("striped", "hover", "condensed", "responsive"),
                position = "center",
                font_size = 12)
```

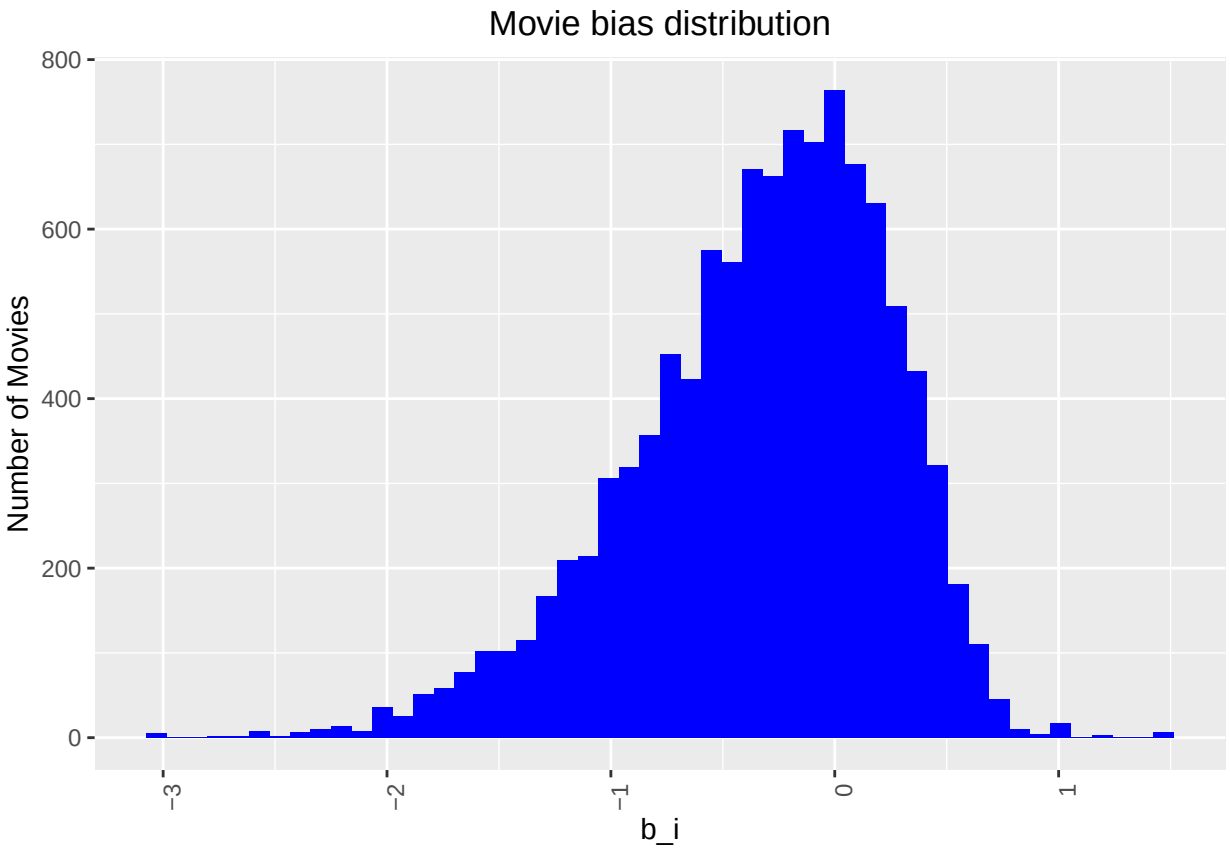
Method	RMSE
Mean	1.051986

The result of 1.0519855 isn't that great but that's to be expected for the initial model.

Movie effect on the model

The next step in improving the model will take into account the idea that some movies are subjectively rated higher than others. We're going to compute the deviation from the movie's mean rating from the total mean of all the movies. The resulting variable be called $b_{i} which will be the movie bias. We can view the movie bias in the following distribution of users:

```
movieBias <- trainEdx %>%
  group_by(movieId) %>%
  summarize(b_i = mean(rating - mu))

ggplot(movieBias, aes(x = b_i)) +
geom_histogram(bins=50, fill="blue") +
                    labs(title = "Movie bias distribution",
                         x = "b_i",
                         y = "Number of Movies"
                    )+
                    theme(axis.text.x = element_text(angle=90, hjust=1), plot.title = element_text(hjus
```



The updated model including the movie bias will have the following equation:

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

```
titleFit <- trainEdx %>%
  group_by(movieId) %>%
  summarize(b_i = mean(rating - mu))

pred <- testEdx %>%
  left_join(titleFit, by = "movieId") %>%
  mutate(y = mu + b_i) %>%
  pull(y)

pred <- clamp(pred, 0.5, 5)

resultsMovieEffect <- RMSE(pred, testEdx$rating)

resultsMean <- bind_rows(
  resultsMean,
  tibble(
    Method = "Movie Effect Model",
    RMSE = resultsMovieEffect
  )
)

kable(resultsMean) %>%
  kable_styling(bootstrap_options = c("striped", "hover", "condensed", "responsive"),
```

```
                position = "center",
                font_size = 12)
```

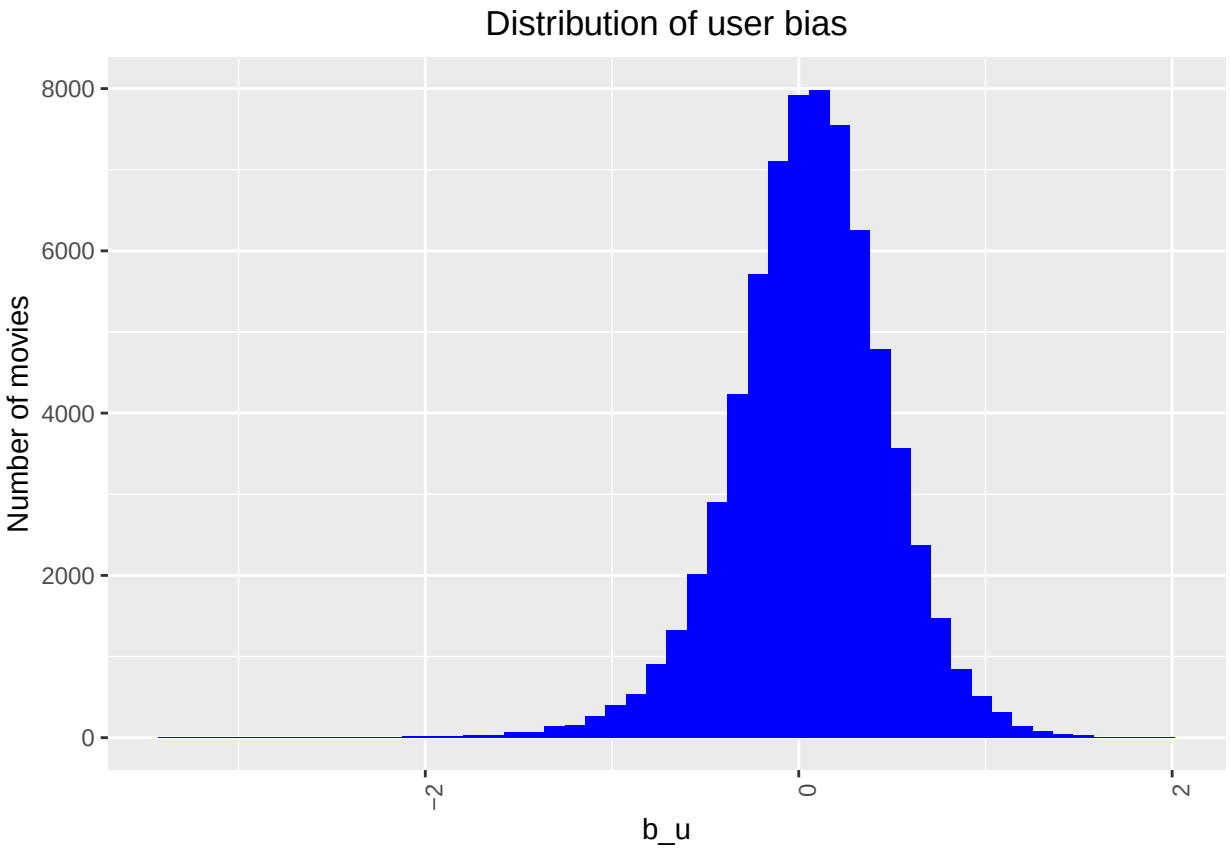
Method	RMSE
Mean	1.0519855
Movie Effect Model	0.9408881

The movie effect model improved the result somewhat significantly but we still aren't getting the end result that we want so on to the next model.

Movie + user effect on the model

Now we're going to add the user effects under the assumption that some users rate movies higher than others. The next model will consider both the movie and the user effect. We estimate the user effect as the average of the ratings per user. For example, a user who typically rates movies higher than the average will have a positive bias. To view the user bias we can look at the distribution below:

```
userBias <- trainEdx %>%
  left_join(movieBias, by = "movieId") %>%
  group_by(userId) %>%
  summarize(b_u = mean(rating - mu - b_i))

ggplot(userBias, aes(x = b_u)) +
geom_histogram(bins=50, fill="blue") +
                    labs(title = "Distribution of user bias",
                         x = "b_u",
                         y = "Number of movies"
                    )+
                    theme(axis.text.x = element_text(angle=90, hjust=1), plot.title = element_text(hjus
```



The updated model including the movie bias will have the following equation:

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

```
userFit <- trainEdx %>%
  left_join(titleFit, by = "movieId") %>%
  group_by(userId) %>%
  summarize(b_u = mean(rating - mu - b_i))

pred <- testEdx %>%
  left_join(titleFit, by = "movieId") %>%
  left_join(userFit, by = "userId") %>%
  mutate(y = mu + b_i + b_u) %>%
  pull(y)

pred <- clamp(pred, 0.5, 5)

resultsMoviePlusUserEffect <- RMSE(pred, testEdx$rating)

resultsMean <- bind_rows(
  resultsMean,
  tibble(
    Method = "Movie + User Effect",
    RMSE = resultsMoviePlusUserEffect
  )
)
```

```
kable(resultsMean) %>%
  kable_styling(bootstrap_options = c("striped", "hover", "condensed", "responsive"),
                position = "center",
                font_size = 12)
```

Method	RMSE
Mean	1.0519855
Movie Effect Model	0.9408881
Movie + User Effect	0.8573795

The model improved the RMSE by a small amount but is not quite there yet. We're going to try a

Movie + user + year delta effect model

Now we're going to add the delta between the year the movie was released and the year that the movie was rated effects. The assumption is that the longer the delta is the higher the rating will be, kind of a nastalgia effect.

The updated model including the movie bias will have the following equation:

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

```
dateDelta <- trainEdx %>%
  left_join(titleFit, by = "movieId") %>%
  left_join(userFit, by = "userId") %>%
  group_by(yearDelta) %>%
  summarize(b_y = mean(rating - mu - b_i - b_u))

pred <- testEdx %>%
  left_join(titleFit, by = "movieId") %>%
  left_join(userFit, by = "userId") %>%
  left_join(dateDelta, by = "yearDelta") %>%
  mutate(y = mu + b_i + b_u + b_y) %>%
  pull(y)

pred <- clamp(pred, 0.5, 5)

resultsMoviePlusUserEffectPluseDateDelta <- RMSE(pred, testEdx$rating)
resultsMean <- bind_rows(
  resultsMean,
  tibble(
    Method = "Movie + User + Year delta Effect",
    RMSE = resultsMoviePlusUserEffectPluseDateDelta
  )
)

kable(resultsMean) %>%
  kable_styling(bootstrap_options = c("striped", "hover", "condensed", "responsive"),
                position = "center",
                font_size = 12)
```

Method	RMSE
Mean	1.0519855
Movie Effect Model	0.9408881
Movie + User Effect	0.8573795
Movie + User + Year delta Effect	0.8568473

Use the delta between the year of the movie release and the year of rating didn't add much benefit, we're going to use it but it just made an incremental benefit.

Movie + user + year release effect model

We're going to try to see if the effect of the movie release year effects the model in any significant way. We're going to go on the assumption that some years the movie reviewers were more harsh or lenient effecting the predictions.

The updated model including the movie bias will have the following equation:

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

```
yearOfMovie <- trainEdx %>%
  left_join(titleFit, by = "movieId") %>%
  left_join(userFit, by = "userId") %>%
  group_by(ratingYear) %>%
  summarize(b_y = mean(rating - mu - b_i - b_u))

pred <- testEdx %>%
  left_join(titleFit, by = "movieId") %>%
  left_join(userFit, by = "userId") %>%
  left_join(yearOfMovie, by = "ratingYear") %>%
  mutate(y = mu + b_i + b_u + b_y) %>%
  pull(y)

pred <- clamp(pred, 0.5, 5)

resultsMoviePlusUserEffectPluseDateDelta <- RMSE(pred, testEdx$rating)
resultsMean <- bind_rows(
  resultsMean,
  tibble(
    Method = "Movie + User + Movie year Effect",
    RMSE = resultsMoviePlusUserEffectPluseDateDelta
  )
)

kable(resultsMean) %>%
  kable_styling(bootstrap_options = c("striped", "hover", "condensed", "responsive"),
                position = "center",
                font_size = 12)
```

Method	RMSE

Mean	1.0519855
Movie Effect Model	0.9408881
Movie + User Effect	0.8573795
Movie + User + Year delta Effect	0.8568473
Movie + User + Movie year Effect	0.8573712

The movie release year made the estimate a bit worse so that will not be in the final model. Moving on to genre effects.

Movie + user + genre effect model

Since the year of the movie release didn't have much of an effect we're going to try out the genre effect. Personally I don't think the genre's as they are in the original data is that useful because I think the data needs to be separated out so a movie will have separate lines for each genre as opposed to listing them in the 797 groups. The data above was separated leaving us with the 20 genres that will be used setup the model. This is the data that will be used in the modeling effort below.

The updated model including the movie bias will have the following equation:

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

```
# Separate the genres out so a movie can be listed by individual genre and not the combined list
genreAvg <- trainEdx %>%
  left_join(titleFit, by = "movieId") %>%
  left_join(userFit, by = "userId") %>%
  group_by(genres) %>%
  summarize(b_g = mean(rating - mu - b_i - b_u))

pred <- testEdx %>%
  left_join(titleFit, by = "movieId") %>%
  left_join(userFit, by = "userId") %>%
  left_join(genreAvg, by = "genres") %>%
  mutate(y = mu + b_i + b_u + b_g) %>%
  pull(y)

pred <- clamp(pred, 0.5, 5)

resultsMoviePlusUserEffectPluseDateDelta <- RMSE(pred, testEdx$rating)
resultsMean <- bind_rows(
  resultsMean,
  tibble(
    Method = "Movie + User + Genre Effect",
    RMSE = resultsMoviePlusUserEffectPluseDateDelta
  )
)

kable(resultsMean) %>%
  kable_styling(bootstrap_options = c("striped", "hover", "condensed", "responsive"),
                position = "center",
                font_size = 12)
```

Method	RMSE
Mean	1.0519855
Movie Effect Model	0.9408881
Movie + User Effect	0.8573795
Movie + User + Year delta Effect	0.8568473
Movie + User + Movie year Effect	0.8573712
Movie + User + Genre Effect	0.8572924

Uses the genre's separated independently didn't make a significant difference, in fact it actually made things worse a bit.

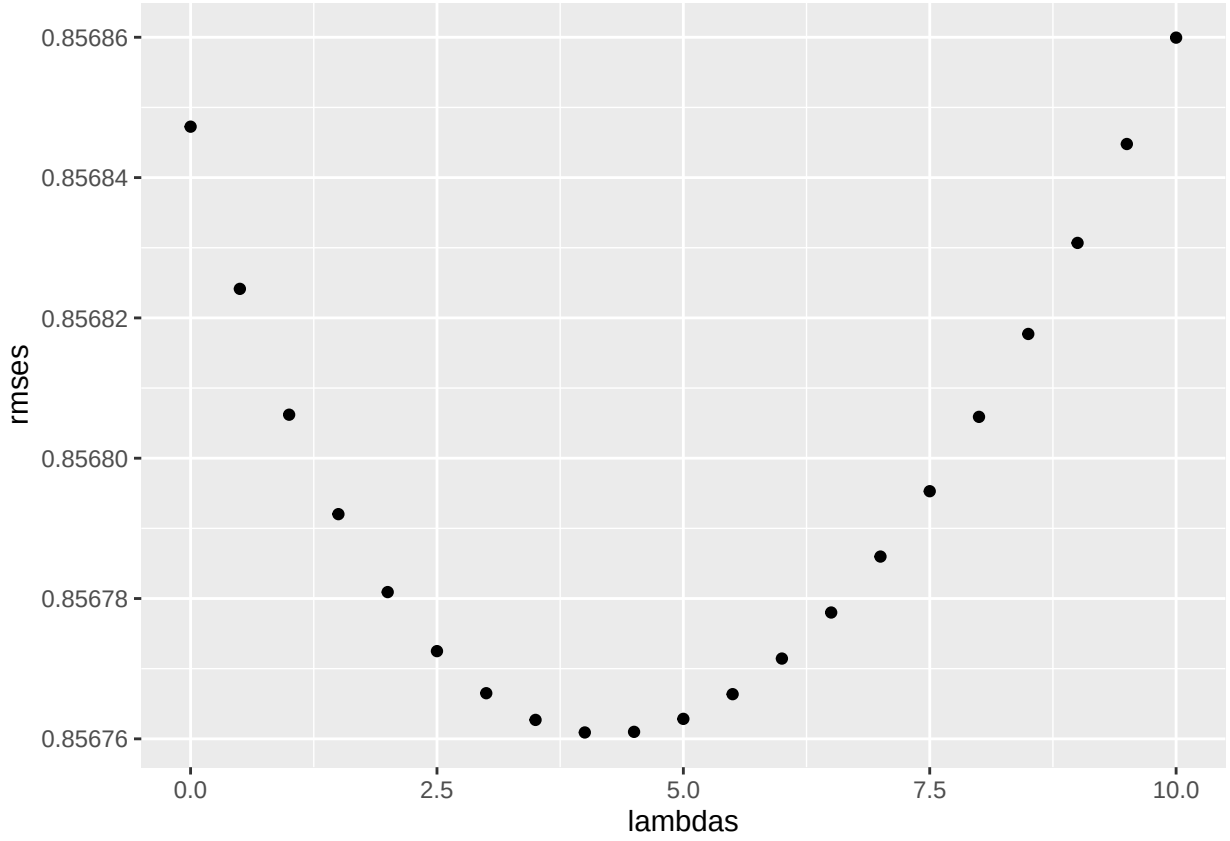
Regularized movie + user + year delta effects model

Using regularization will add the tuning parameter λ to try to further reduce the RMSE. The idea is to punish outliers from the movie and user bias sets which should optimize the recommendation system.

First we need to find the λ that minimizing the error.

```
# Regularized parameter
lambdas <- seq(0, 10, 0.5)

# Grid search to tune the regularized parameter lambda
rmses <- sapply(lambdas, function(l) {
  mu <- mean(trainEdx$rating)

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

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

  b_y <- trainEdx %>%
    left_join(b_i, by = "movieId") %>%
    left_join(b_u, by = "userId") %>%
    group_by(yearDelta) %>%
    summarize(b_y = sum(rating - mu - b_i - b_u) / (n() + l))

  predicted_ratings <- testEdx %>%
    left_join(b_i, by = "movieId") %>%
    left_join(b_u, by = "userId") %>%
    left_join(b_y, by = "yearDelta") %>%
    mutate(pred = mu + b_i + b_u + b_y) %>%
    pull(pred)

  predicted_ratings <- clamp(predicted_ratings, 0.5, 5)

  return(RMSE(predicted_ratings, testEdx$rating))
```

```
})

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



Now that we've identified the λ of 4 that minimizes the model we can print a table of all the results calculated to this point.

```
resultsMean <- bind_rows(
  resultsMean,
  tibble(
    Method = "Regularized Movie + User + Year delta Effect",
    RMSE = min(rmses)
  )
)

kable(resultsMean) %>%
  kable_styling(bootstrap_options = c("striped", "hover", "condensed", "responsive"),
                position = "center",
                font_size = 12)
```

Method	RMSE
Mean	1.0519855

Movie Effect Model	0.9408881
Movie + User Effect	0.8573795
Movie + User + Year delta Effect	0.8568473
Movie + User + Movie year Effect	0.8573712
Movie + User + Genre Effect	0.8572924
Regularized Movie + User + Year delta Effect	0.8567609

The regularization based on the movie effects didn't really help much but there was a slight improvement.

Results section

To predict movie ratings we built several models models that considered the effects of movies, users, independent genres and time features. The best model considered movies, users, and year delta effects Using the movie release year and the genres independently actually made things slightly worse and will be left out of the final model. The movie effect decreased the RMSE the most, suggesting that the movie in itself is of most important feature of those tested.

Now that we've identified the best model of the bunch on the subset of data we'll run that algorithm on the edx data set and compare against the final_holdout_test data.

```
# Regularized parameter
lambdas <- seq(0, 10, 0.5)

# Grid search to tune the regularized parameter lambda
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))

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

  predicted_ratings <- final_holdout_test %>%
    left_join(b_i, by = "movieId") %>%
    left_join(b_u, by = "userId") %>%
    left_join(b_y, by = "yearDelta") %>%
    mutate(pred = mu + b_i + b_u + b_y) %>%
    pull(pred)

  predicted_ratings <- clamp(predicted_ratings, 0.5, 5)
```

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

plot_rmses <- qplot(lambdas, rmses)
lambda <- lambdas[which.min(rmses)]

finalRMSE <- min(rmses)
resultsMean <- tibble(Method = "Movie + User + Year delta Effects", RMSE = finalRMSE)
kable(resultsMean) %>%
  kable_styling(bootstrap_options = c("striped", "hover", "condensed", "responsive"),
                position = "center",
                font_size = 12)
```

Method	RMSE
Movie + User + Year delta Effects	0.8621477

Conclusion section

The goal of this project was to predict movie ratings from a subset of the movie lens data base using features of the dataset such as movie title, ratings, users, genres, and timestamp information. The data was divided into train and validation to avoid over fitting. Since the dataset was fairly large some modeling techniques were avoided due to limitations of time and memory allocation.This left solving the problem using a least squares estimate manually using the above features and some basic regularization. The best-fitted model achieved an RMSE of 0.8621477, which being less than 0.86490 achieved the maximum point allowance according to the grading rubric provided in the course description. Having more information on the users and movie details may allow for improving the model. Also, using some of the more advanced modeling techniques such as distributed random forest, nearest neighbor, or ensemble approach would improve the results but the minimum has been met and time is running short so the model improvements will have to wait for some other assignment.

References

Irizarry, Rafael A., "Introduction to Data Science: Data Analysis and Prediction Algorithms in R" https://grouplens.org/datasets/movielens/10m/

"MovieLens", GroupLens. 2024. https://rafalab.dfci.harvard.edu/dsbook/