HarvardX PH125.9x
Data Science: Capstone
MovieLens Project

Tom Hannigan
January 7, 2021

Introduction

This is the first of two capstone projects for completion of the ninth and final course in HarvardX's multi-part Data Science Professional Certificate series. The scope of this project is to create a movie recommendation system using the MovieLens dataset. Code was provided to generate the datasets for evaluation ("edx" and "validation") from the 10M version of the movie lens dataset. The root mean squared error, RMSE, will be used as the metric to evaluate performance of the movie recommendation system.

The goal of the project is to generate and train a machine learning algorithm using the "edx" dataset to predict ratings in the "validation" dataset. When the machine learning is evaluated on the "validation" subset, the goal is to generate an **RMSE < 0.86490**. The calculation for RMSE is:

RMSE = sqrt (mean ((actual ratings – predicted ratings) ^2)

The steps performed for the project were:

- Download and partition the datasets
- Add additional columns of related data to the data sets
- Visually evaluate the data
- Identify and model the various effects
- Fit the models to the "edx" data set
- Construct the predictors on the validation dataset
- Determine and report the final RMSE

Methods/Analysis

The code provided by the course for retrieval and set up of the “edx” and “validation” datasets was run. The following is the first 6 rows of “edx” at that point: Then, 3 columns of related data were added. First, a “movieYr” column representing the year of the movie was pulled from the “title” column. Second, a “ratingYr” column representing the year of the rating was pulled from the “timestamp” column. Third, a “delay” column representing the year of the rating minus the year of the review was calculated. This was done for the “edx” dataset with the following code (which was duplicated on the “validation” dataset as well):

----- CODE ----

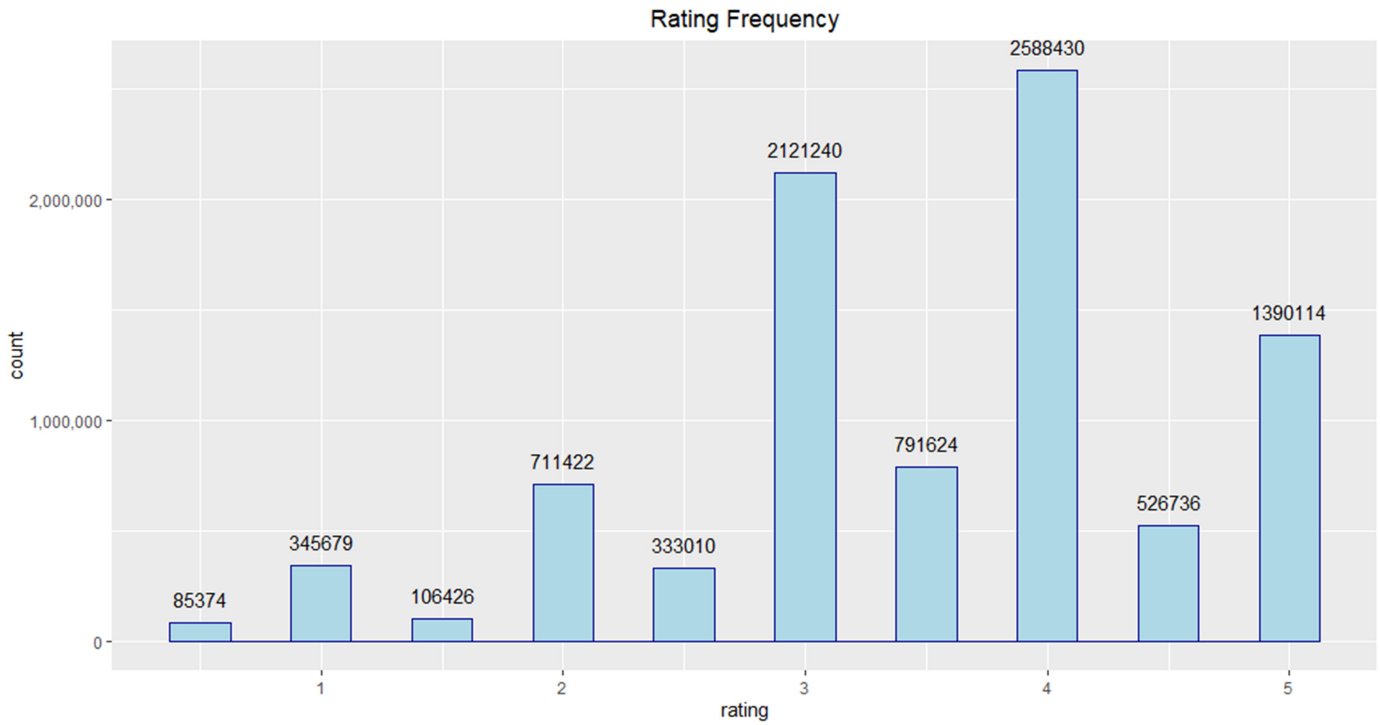
```
edx <- mutate(edx, movieYr =as.numeric(str_sub(title,-5, -2)))
edx <- mutate(edx, ratingYr = year(as_datetime(timestamp)))
edx <- mutate(edx, delay = ratingYr - movieYr)
```

----- RESULT ----

The first 6 rows can be seen with “head(edx)”:

```
> head(edx)
   userId movieId rating timestamp                         title                        genres movieYr ratingYr delay
1:      1     122      5 838985046              Boomerang (1992)                Comedy|Romance    1992     1996     4
2:      1     185      5 838983525               Net, The (1995)         Action|Crime|Thriller    1995     1996     1
3:      1     292      5 838983421               Outbreak (1995)  Action|Drama|Sci-Fi|Thriller    1995     1996     1
4:      1     316      5 838983392               Stargate (1994)       Action|Adventure|Sci-Fi    1994     1996     2
5:      1     329      5 838983392 Star Trek: Generations (1994) Action|Adventure|Drama|Sci-Fi    1994     1996     2
6:      1     355      5 838984474       Flintstones, The (1994)       Children|Comedy|Fantasy    1994     1996     2
>
```

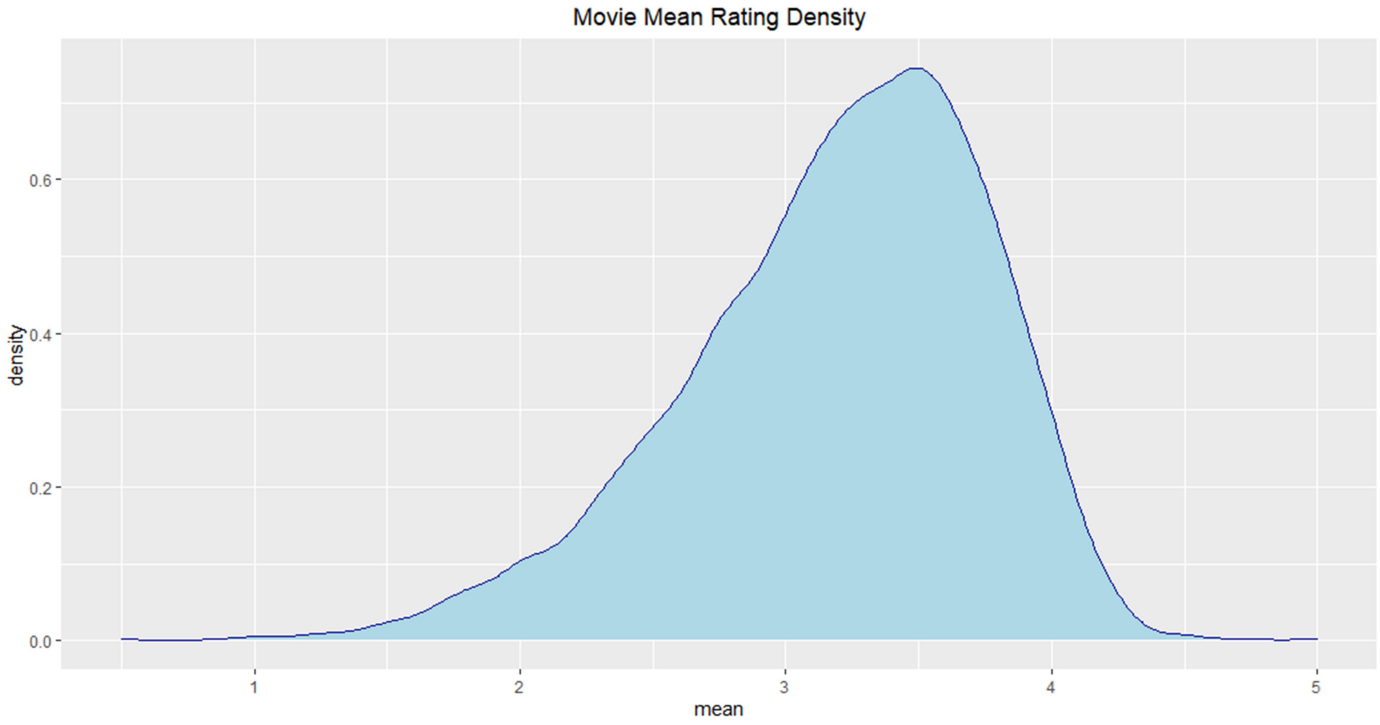
Based on experience, some movies are better than others and would therefore be rated higher on average. An example of this would be comparing “Titanic” with its 14 Oscar nominations and 11 wins, including Best Picture to “Ballistic: Ecks vs. Sever” which has a 0% Tomatometer on 118 critic reviews and 19% on 19% score on 22994 audience reviews. This can be visualized in a graph of rating frequencies:



----- CODE ----

```
edx %>%
  ggplot(aes(rating)) +
  geom_histogram(binwidth = 0.25, color = "darkblue", fill = "lightblue") +
  scale_y_continuous(labels = scales::comma) +
  stat_count(aes(y=..count..,label=..count..),geom="text",vjust=-1)+
  ggtitle("Rating Frequency")+
  theme(plot.title = element_text(hjust = 0.5))
```

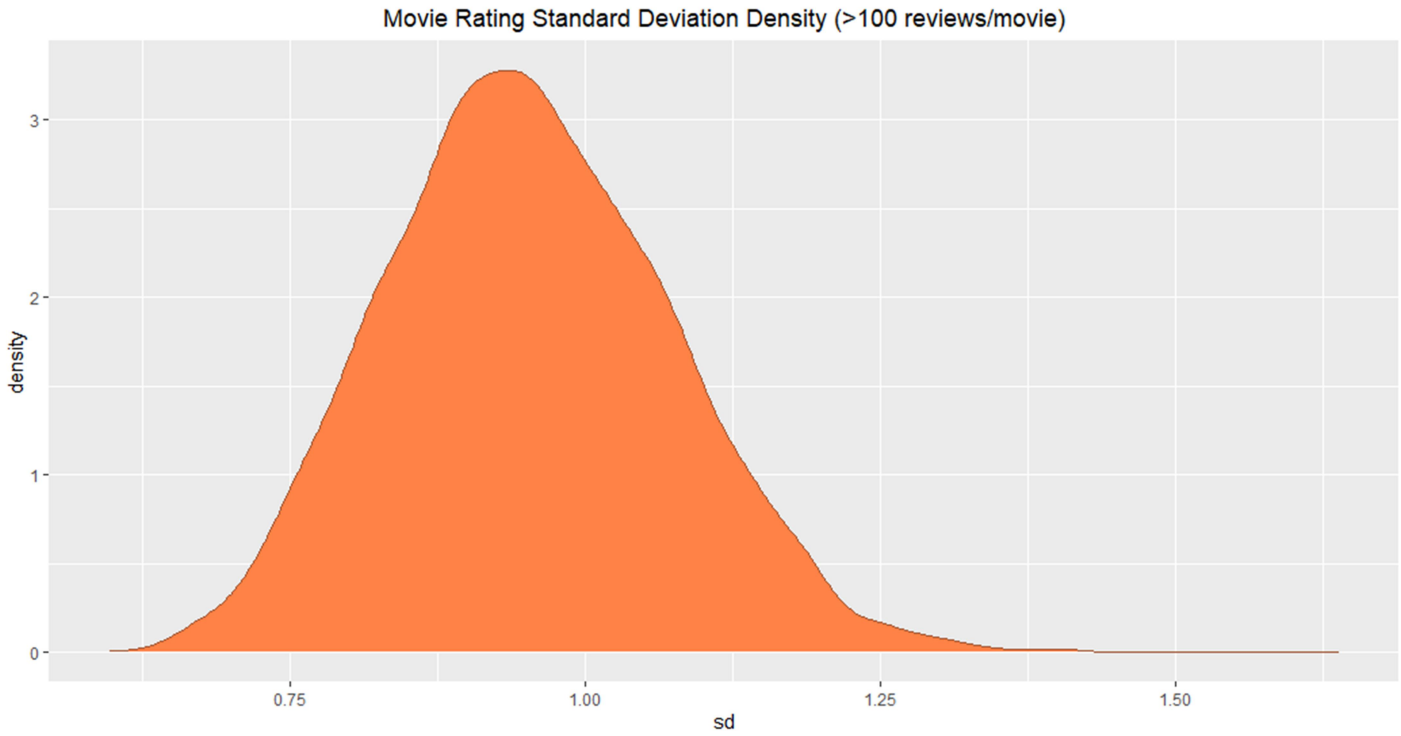
And of the Mean Rating Density by movie:



----- CODE ----

```
edx %>% group_by(movieId) %>% summarize(mean=mean(rating)) %>%
  ggplot(aes(x=mean))+
  geom_density(color="darkblue", fill="lightblue") +
  ggtitle("Movie Mean Rating Density") +
  theme(plot.title = element_text(hjust = 0.5))
```

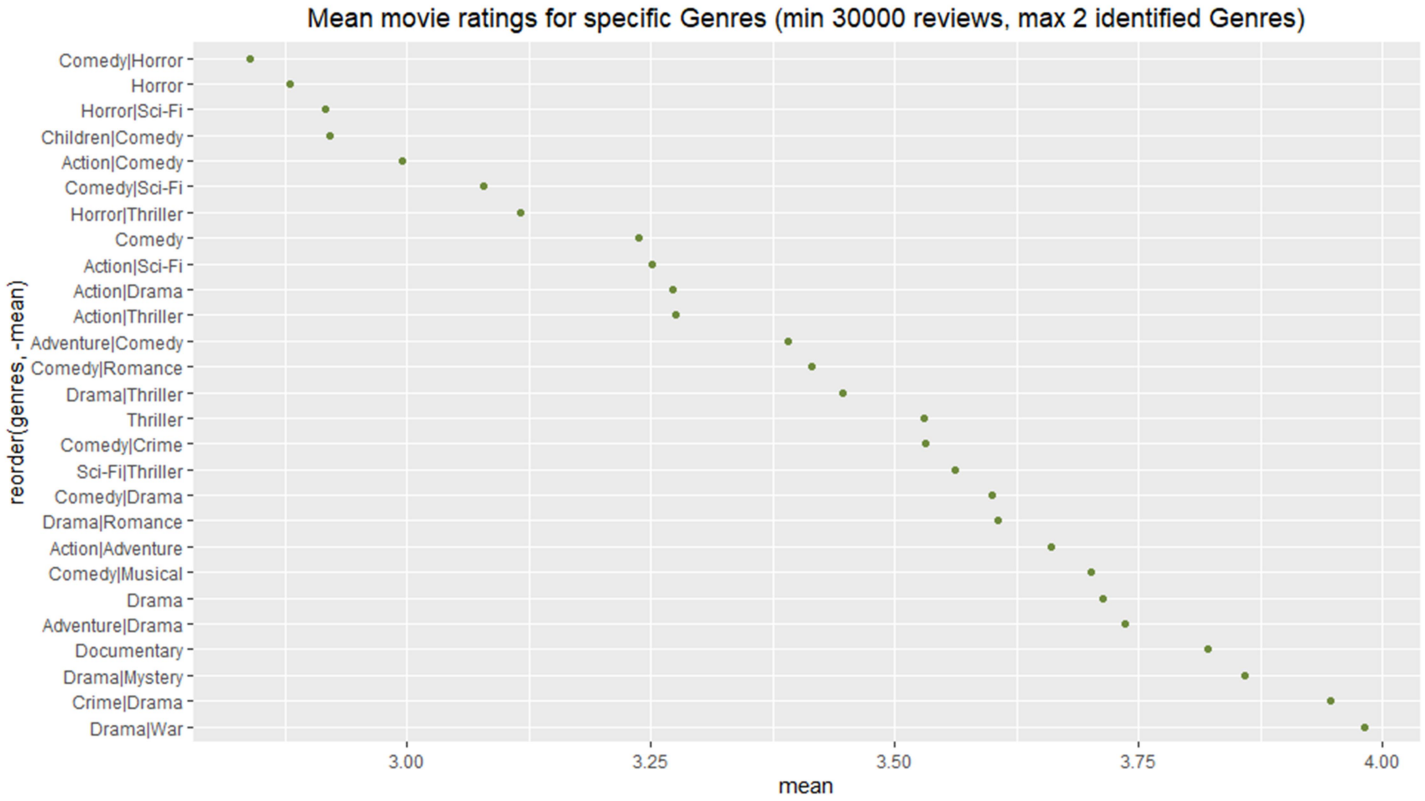
Also based on experience, people will rate differently from each other. As ratings are subjective each person will have differences in likes/dislikes, how they numerically rate a good movie, etc. This will show up visually by graphing the density of the standard deviation of reviews by movie. If there was no variation between users, the standard deviation would be zero. As you can see from the following, that is not the case and there is user bias in the ratings:



----- CODE ----

```
edx %>% group_by(movieId) %>% filter(n() >= 100) %>%
  summarize(sd=sd(rating)) %>%
  ggplot(aes(x=sd))+
  geom_density(color="sienna", fill="sienna1") +
  ggtitle("Movie Rating Standard Deviation Density (>100 reviews/movie)") +
  theme(plot.title = element_text(hjust = 0.5))
```

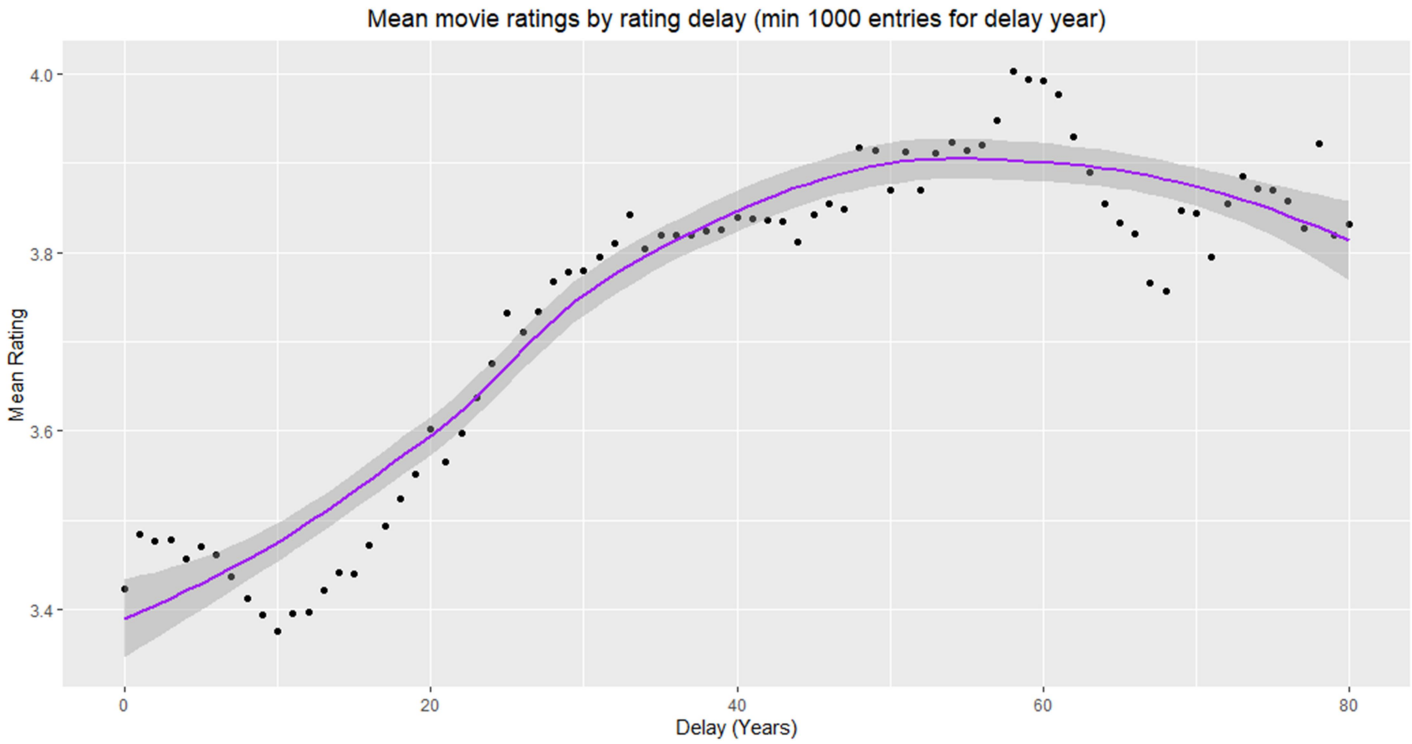
There can also be an impact on ratings based on the genre. To use the Academy Awards example, there have been numerous Drama Best Picture nominees and winners while Horror, Sci-Fi, and Comedies are rarely nominated and have won only a handful of times. You can see this in the ratings in a simplified breakout of genres with only 1 or 2 combinations and greater than 30,000 reviews:



----- CODE ----

```
genre_graph <- edx %>%
  group_by(genres) %>%
  filter(n() >= 30000 & str_count(genres,"\\|")<2) %>%
  summarize(mean = mean(rating)) %>%
  arrange(., mean)

genre_graph %>% ggplot(aes(mean,reorder(genres,-mean))) +
  geom_point(color = "darkolivegreen4") +
  ggtitle("Mean movie ratings for specific Genres (min 30000 reviews, max 2 identified Genres)") +
  theme(plot.title = element_text(hjust = 0.5))
```

Additionally, there can be a crowd mentality and nostalgia about movies. First off, experience would tell you that you are going to watch (and review) movies later that you have heard they are good. You most likely would not watch (and review) a movie much later that had poor reviews. You would expect that this would make your mean ratings increase based on the delay in reviewing the movie. This can be visualized with the following mean movie ratings by rating delay:



----- CODE ----

```
edx %>%
  group_by(delay) %>%
  filter(n() >= 1000) %>%
  summarize(b_d = mean(rating)) %>%
  ggplot(aes(delay,b_d)) +
  geom_point() +
  geom_smooth(method = 'loess', formula = y ~ x, color = "purple") +
  ylab("Mean Rating") +
  xlab("Delay (Years)") +
  ggtitle("Mean movie ratings by rating delay (min 1000 entries for delay year)")+
  theme(plot.title = element_text(hjust = 0.5))
```

To set our baseline, we determine the mean of all of the “edx” ratings, which returns a value of mu = 3.512465. From that we determine the naïve rmse, which returns a value of 1.061202, significantly higher than the goal of < 0.86490. This is generated with the code/formulas below:

----- CODE ----

```
mu <- mean(edx$rating)
naive_rmse <- RMSE(validation$rating, mu)
rmse_results <- tibble(method = "Average movie rating model", RMSE = naive_rmse)
rmse_results %>% knitr::kable()
```

----- RESULT ----

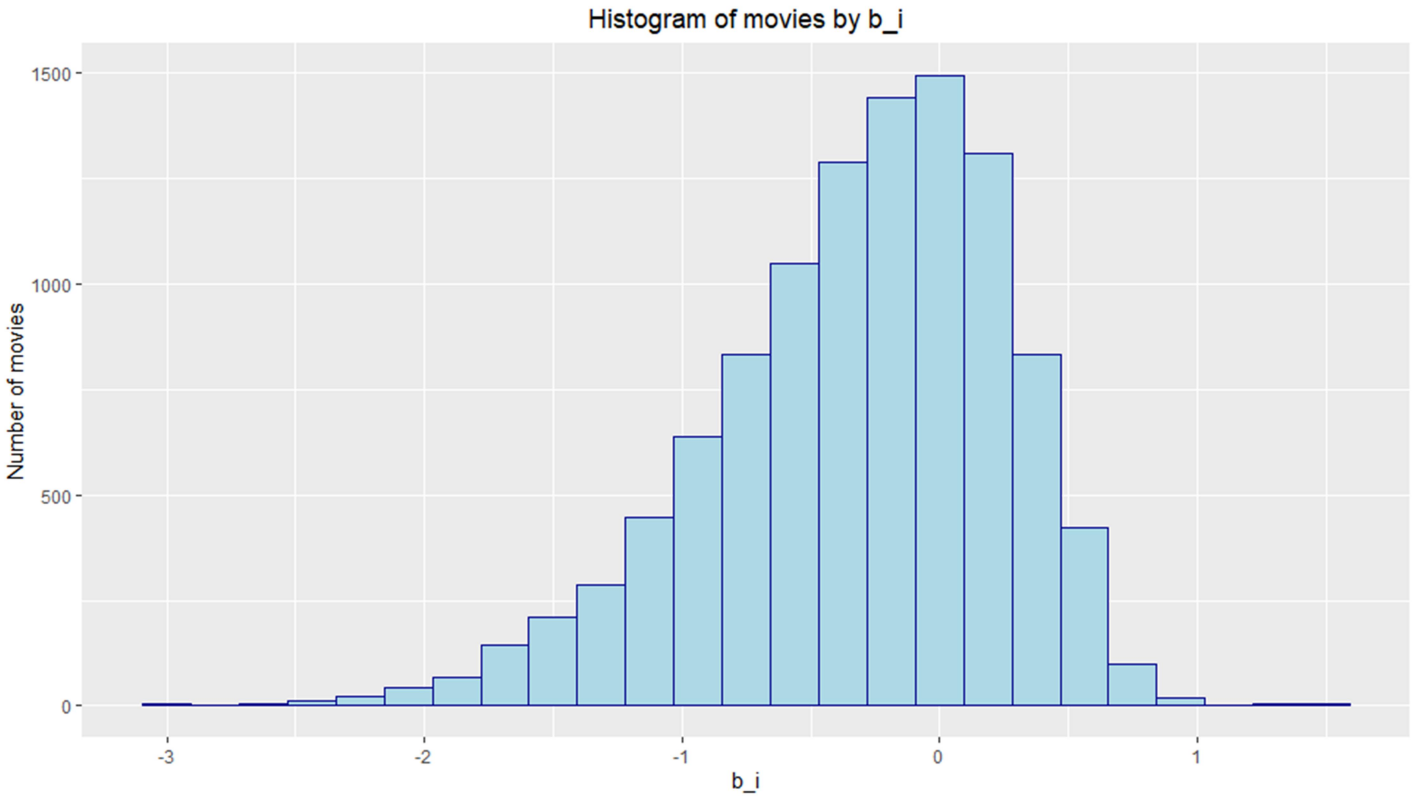
method	RMSE
Average movie rating model	1.061202

Then we can augment the baseline model by factoring in the movie impact. This will be done with estimating the least squares by taking the mean of the rating by movie minus the overall rating mean. This is generated with the code/formulas below:

----- CODE ----

```
movie_avgs <- edx %>%
  group_by(movieId) %>%
  summarize(b_i = mean(rating - mu))
```

The histogram of the estimates can be seen with the following:



----- CODE ----

```
movie_avgs %>% qplot(b_i, geom ="histogram", bins = 25, data = ., fill = I("lightblue"),
                     color = I("darkblue"), ylab = "Number of movies", main = "Histogram of movies by b_i") +
  theme(plot.title = element_text(hjust = 0.5))
```

Then using the movie estimates from the “edx” dataset, we can construct the predictors on the “validation” dataset. This will be done with the code/formulas below:

----- CODE ----

```
predicted_ratings <- mu +  validation %>%
  left_join(movie_avgs, by='movieId') %>%
  pull(b_i)

model_M_rmse <- RMSE(predicted_ratings, validation$rating)

rmse_results <- bind_rows(rmse_results,
                          data_frame(method="Movie effect model",
                                     RMSE = model_M_rmse ))
```

----- RESULT ----

method	RMSE
Average movie rating model	1.0612018
Movie effect model	0.9439087

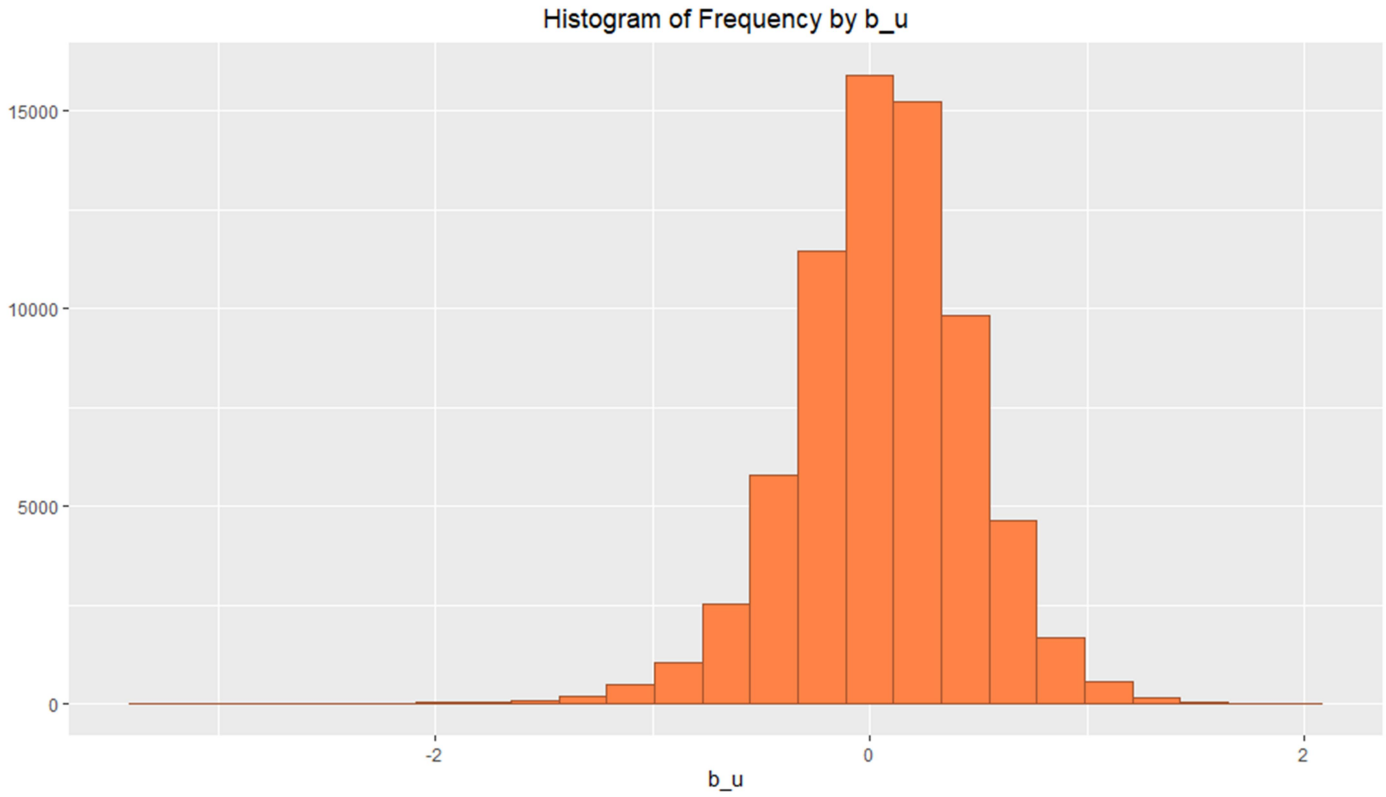
Better, but not there yet.

Then we can augment the movie effect model by factoring in the user impact. This will be done with estimating the least squares by taking the mean of the rating by user minus the overall rating mean minus the previously estimated movie effect. This is generated with the code/formulas below:

----- CODE ----

```
user_avgs <- edx %>%
  left_join(movie_avgs, by='movieId') %>%
  group_by(userId) %>%
  summarize(b_u = mean(rating - mu - b_i))
```

The frequency of the estimates can be seen with the following histogram:



----- CODE ----

```
user_avgs%>% qplot(b_u, geom ="histogram", bins = 25, data = ., color = I("sienna"),
                   fill = I("sienna1"), main = "Histogram of Frequency by b_u") +
  theme(plot.title = element_text(hjust = 0.5))
```

Then using the user estimates from the “edx” dataset, we can construct the predictors on the “validation” dataset. This will be done with the code/formulas below:

----- CODE ----

```
predicted_ratings <- validation %>%
  left_join(movie_avgs, by='movieId') %>%
  left_join(user_avgs, by='userId') %>%
  mutate(pred = mu + b_i + b_u) %>%
  pull(pred)

model_U_rmse <- RMSE(predicted_ratings, validation$rating)

rmse_results <- bind_rows(rmse_results,
                          data_frame(method="Movie and user effect model",
                                     RMSE = model_U_rmse))
```

----- RESULT ----

method	RMSE
Average movie rating model	1.0612018
Movie effect model	0.9439087
Movie and user effect model	0.8653488

Continued progress and closer, but not to the goal yet.

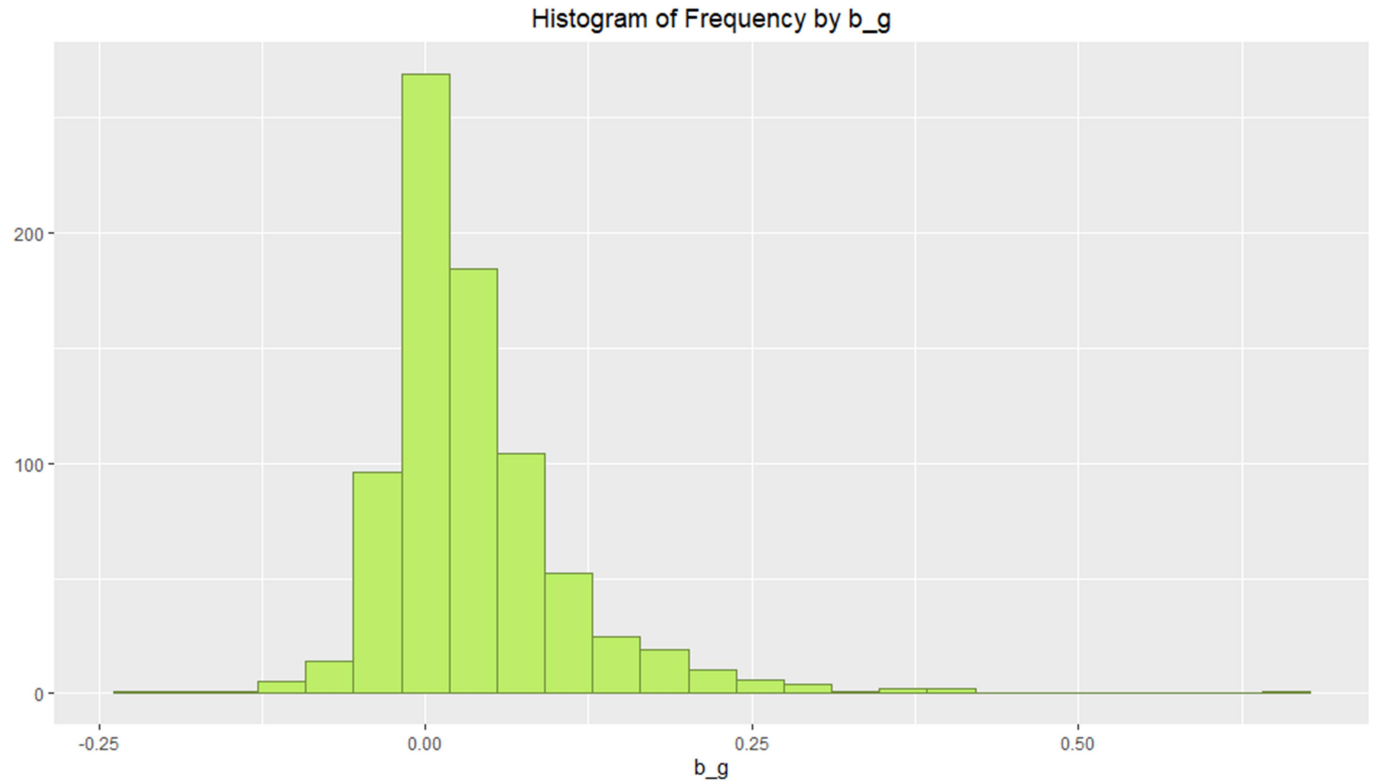
Then we can augment the movie and user effect model by factoring in the genre impact. This will be done with estimating the least squares by taking the mean of the rating by user minus the overall rating mean minus the previously estimated movie effect minus the previously estimated user effect. This is generated with the code/formulas below:

----- CODE ----

```
genre_avgs <- edx %>%
  left_join(movie_avgs, by='movieId') %>%
  left_join(user_avgs, by='userId') %>%
  group_by(genres) %>%
  summarize(b_g = mean(rating - mu - b_i - b_u))
```

The frequency of the estimates can be seen with the following histogram, which shows a narrower spread, so you would expect the impact to improving the RMSE would be less than it was for the movie or user effects

:



----- CODE ----

```
genre_avgs %>% qplot(b_g, geom ="histogram", bins = 25, data = ., color = I("darkolivegreen4"),
                     fill = I("darkolivegreen2"), main = "Histogram of Frequency by b_g") +
  theme(plot.title = element_text(hjust = 0.5))
```

Then using the genre estimates from the “edx” dataset, we can construct the predictors on the “validation” dataset. This will be done with the code/formulas below:

----- CODE ----

```
predicted_ratings <- validation%>%
  left_join(movie_avgs, by='movieId') %>%
  left_join(user_avgs, by='userId') %>%
  left_join(genre_avgs, by='genres') %>%
  mutate(pred = mu + b_i + b_u + b_g) %>%
  pull(pred)

model_G_rmse <- RMSE(predicted_ratings, validation$rating)

rmse_results <- bind_rows(rmse_results,
                          data_frame(method="Movie and user and genre effect model",
                                     RMSE = model_G_rmse))
```

----- RESULT ----

method	RMSE
Average movie rating model	1.0612018
Movie effect model	0.9439087
Movie and user effect model	0.8653488
Movie and user and genre effect model	0.8649469

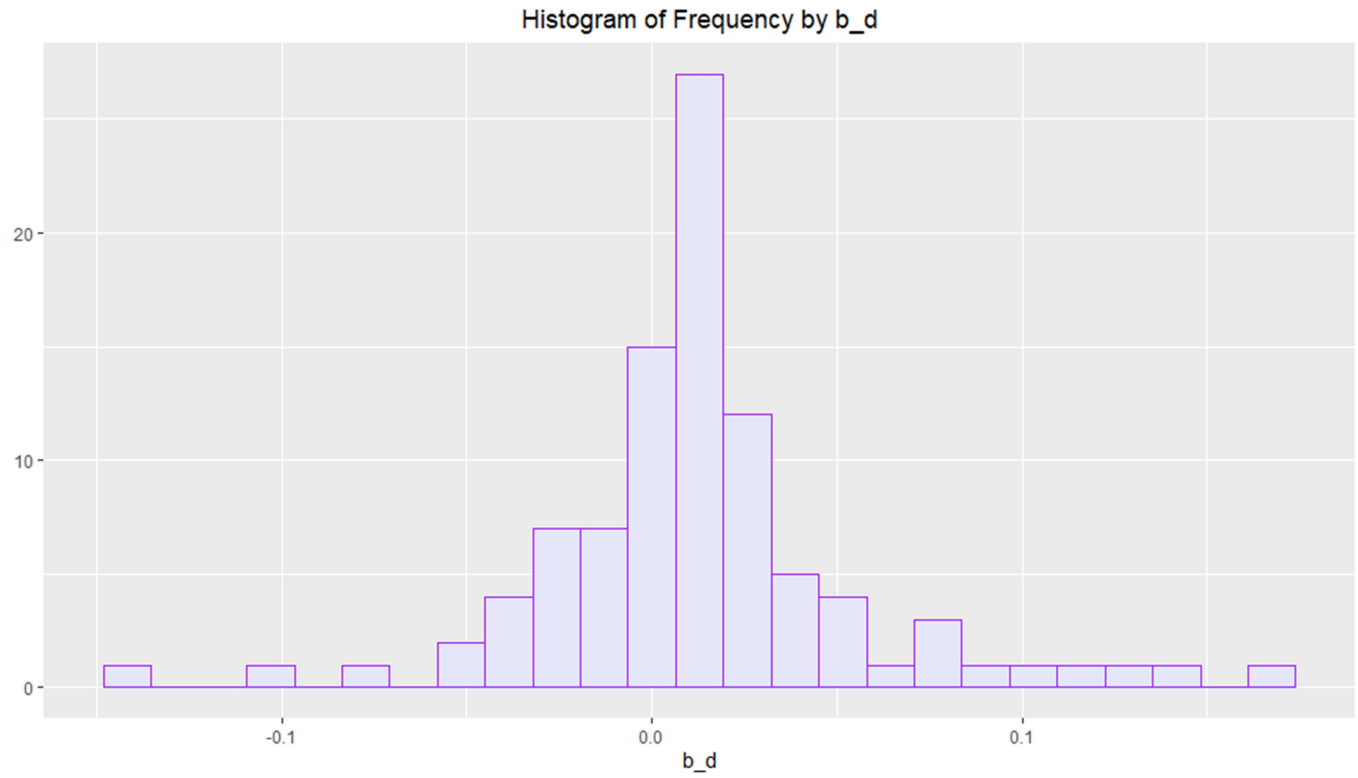
Another improvement and now we are very close to our goal.

Now let's augment the movie and user and genre effect model by factoring in the delay impact. This will be done with estimating the least squares by taking the mean of the rating by user minus the overall rating mean minus the previously estimated movie effect minus the previously estimated user effect minus the previously estimated genre effect. This is generated with the code/formulas below:

----- CODE ----

```
genre_avgs <- edx %>%
  left_join(movie_avgs, by='movieId') %>%
  left_join(user_avgs, by='userId') %>%
  group_by(genres) %>%
  summarize(b_g = mean(rating - mu - b_i - b_u))
```

The frequency of the estimates can be seen with the following histogram, which shows an even narrower spread than the genre spread. But we are so close to our goal, it might be enough.



----- CODE ----

```
delay_avgs %>% qplot(b_d, geom ="histogram", bins = 25, data = ., color = I("purple"),
                     fill = I("lavender"), main = "Histogram of Frequency by b_d") +
  theme(plot.title = element_text(hjust = 0.5))
```

Then using the delay estimates from the “edx” dataset, we can construct the predictors on the “validation” dataset. This will be done with the code/formulas below:

----- CODE ----

```
predicted_ratings <- validation%>%
  left_join(movie_avgs, by='movieId') %>%
  left_join(user_avgs, by='userId') %>%
  left_join(genres_avgs, by='genres') %>%
  left_join(delay_avgs, by='delay') %>%
  mutate(pred = mu + b_i + b_u + b_d + b_g) %>%
  pull(pred)

model_D_rmse <- RMSE(predicted_ratings, validation$rating)

rmse_results <- bind_rows(rmse_results,
                          data_frame(method="Movie and user and genre and delay effect model",
                                     RMSE = model_D_rmse))
```

----- RESULT ----

method	RMSE
Average movie rating model	1.0612018
Movie effect model	0.9439087
Movie and user effect model	0.8653488
Movie and user and genre effect model	0.8649469
Movie and user and genre and delay effect model	0.8645400

Success, the RMSE is now less than our goal of 0.86490, but let’s do one more step to make it better.

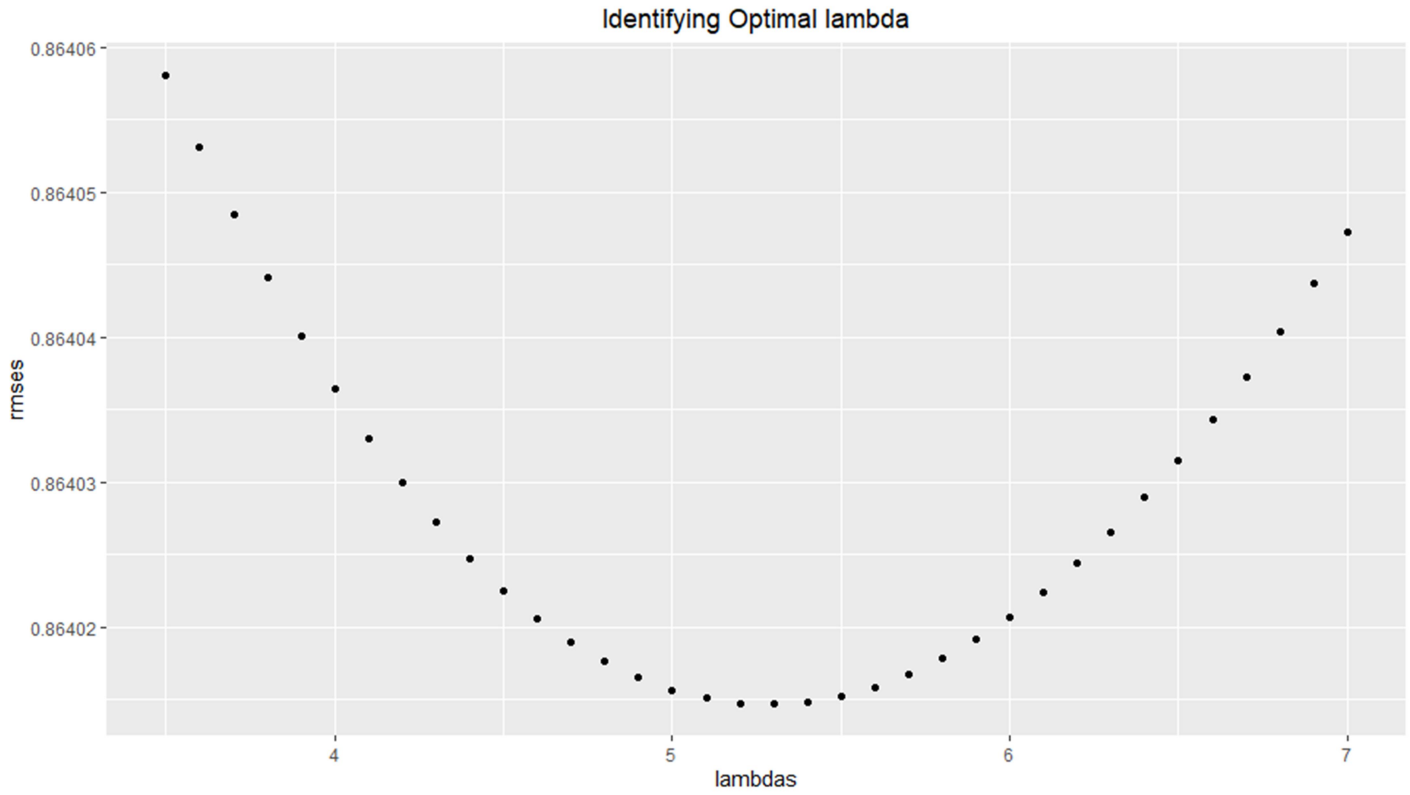
To take it one step further, we will use regularization on the final model to constrain the variability due to mall incident size. For example, you do not want to give the same weight to a movie that has been reviewed twice as to one that has been reviewed 2,000 times. We define the method of doing this by adding a penalty, lambda. When samples are high, the penalty becomes effectively insignificant. To determine the best lambda, we will use cross-validation to use it. This is generated with the code/formulas below:

----- CODE ----

```
lambdas <- seq(3.5, 7, 0.1)

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

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

  predicted_ratings <-
    validation %>%
```

```
    left_join(b_i, by="movieId") %>%
    left_join(b_u, by="userId") %>%
    left_join(b_g, by="genres") %>%
    left_join(b_d, by="delay") %>%
    mutate(pred = mu + b_i + b_u + b_g + b_d) %>%
    pull(pred)

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

You can then plot the rmse vs the lambdas to identify the optimal lambda, which is 5.3.




```
----- CODE ----
qplot(lambdas, rmses, main = "Identifying Optimal lambda") +
  theme(plot.title = element_text(hjust = 0.5))

lambdas[which.min(rmses)]
```

Then with using the optimal lambda from the previous code we can determine the minimal RMSE with the regularized movie/user/genre/delay effect model. It further drops our RMSE to 0.8640147, comfortably past our goal.

method	RMSE
Average movie rating model	1.0612018
Movie effect model	0.9439087
Movie and user effect model	0.8653488
Movie and user and genre effect model	0.8649469
Movie and user and genre and delay effect model	0.8645400
Regularized movie/user/genre/delay effect model	0.8640147

Results

By modeling the effects due to individual movies, individual users, specific genres, and the delay between the year of review and the year the movie came out, we are able to predict effectively enough to meet our goal of an RMSE on the "validation" dataset less than 0.86490. We then regularized the data for further improvement. Below are the results of the process that we followed:

method	RMSE
Average movie rating model	1.0612018
Movie effect model	0.9439087
Movie and user effect model	0.8653488
Movie and user and genre effect model	0.8649469
Movie and user and genre and delay effect model	0.8645400
Regularized movie/user/genre/delay effect model	0.8640147

As would be expected, the effects of individual movies and users had the largest impacts. However, the effects of specific genres and time delay had enough of an impact to meet our goal. The regularization of the data further refined and reduced the RMSE to 0.8640147.

Conclusion

We were able to exceed the project goal of an RMSE < 0.86490. However, in the related Netflix challenge, the RMSE had to be around 0.857 to win the grand prize. Therefore there are more opportunities.

A limitation of this project was that we only used the 10M version of the MovieLens dataset for ease of computing. Utilizing the full data set would be expected to reduce the error.

In order to continue the project and strive towards the grand prize level of RMSE, we would want to expand on the Genre model. There can be 1 to 7 different specific genres identified for each movie and those combinations resulted in 797 unique genres. This project combined them into one iteration per movie. There is an opportunity to investigate how the various combinations and/or the quantity of identified genres affect the RMSE results.

Citations:

MovieLens 10M Dataset: https://grouplens.org/datasets/movielens/10m/
Acadamy awards data: https://www.filmsite.org/
Rotten Tomatoes data: https://www.rottentomatoes.com/m/ballistic_ecks_vs_sever
Initial Code provided as part of HarvardX PH125.9x Capstone Course:
https://www.edx.org/professional-certificate/harvardx-data-science
Textbook: "Introduction to Data Science", by Rafael A. Irizarry: https://rafalab.github.io/dsbook/