MovieLens Project Report

John Vincent Landingin

11/7/2021

Contents

I. Overview

The goal of this project is to create a movie recommendation system with a model using the MovieLens dataset. The entire dataset is found in the following website: https://grouplens.org/datasets/movielens/latest/. However, only the 10M version of the MovieLens dataset (https://grouplens.org/datasets/movielens/10m/) is used for this project to make the computations easier.

The dataset is initially divided into the edx set and the validation set. The algorithm is developed using the edx set, while the final test of the algorithm will be conducted using the validation set (the final hold-out test set) as if this was unknown. RMSE is used to evaluate how close the predictions are to the true values in the validation set (the final hold-out test set).

The following are the first six rows of the edx dataset:

```
head(edx)
```

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

Below are the number of entries for the edx and validation datasets. 9000055 rows for the edx set and 999999 for the validation set.

```
nrow(edx)
```

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

```
nrow(validation)
```

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

We use the following libraries:

```
library(tidyverse)
library(caret)
library(stringr)
library(lubridate)
```

Furthermore, the edx set is divided into train_set and test_set. This is for designing and testing the algorithm. Below is the code to do that:

```
test_index <- createDataPartition(y = edx$rating, times = 1, p = 0.1, list = F)
train_set <- edx[-test_index,]
temp <- edx[test_index,]
test_set <- temp %>%
  semi_join(train_set, by = 'movieId') %>%
  semi_join(train_set, by = 'userId')
```

The test_set and train_set is 10% and 90% of the edx set, respectively. Below are the number of entries for the test_set and train_set.

```
nrow(test_set)
```

```
## [1] 899985
```

```
nrow(train_set)
```

```
## [1] 8100048
```

To summarize, only the edx set is used to train the algorithm. In building the model, the features used to define the biases/effects are the following:
1. Movie - mean rating per movie
2. User - mean rating of each user
3. Genre - mean rating of each user for a particular genre
4. Year Movie was released

Later, we describe how the mentioned biases were used.

II. Analysis

Measuring model performance

The residual mean squared error (RMSE) is used to measure how close the model predictions are to the true values in the validation set (the final hold-out test set). This will also be used in building the model and testing the RMSE of multiple models, as well as tuning parameters and doing regularization but only on the test_set from the edx dataset.

The following is the function that computes for the RMSE:

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

Building the model

To start our model, we begin with just the average of all recorded ratings. This can be interpreted as a model that assumes the same rating for all movies and users with all the differences explained by random variation. The model would look like the following:

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

with $\varepsilon_{u,i}$ independent errors sampled from the same distribution centered at 0 and μ the "true" rating for all movies.

We find the mean using the following code:

```
mu <- mean(train_set$rating)
mu
```

```
## [1] 3.512492
```

To test this model with just the average, we use the following code:

```
rmse_average <- RMSE(test_set$rating, mu)
rmse_average
```

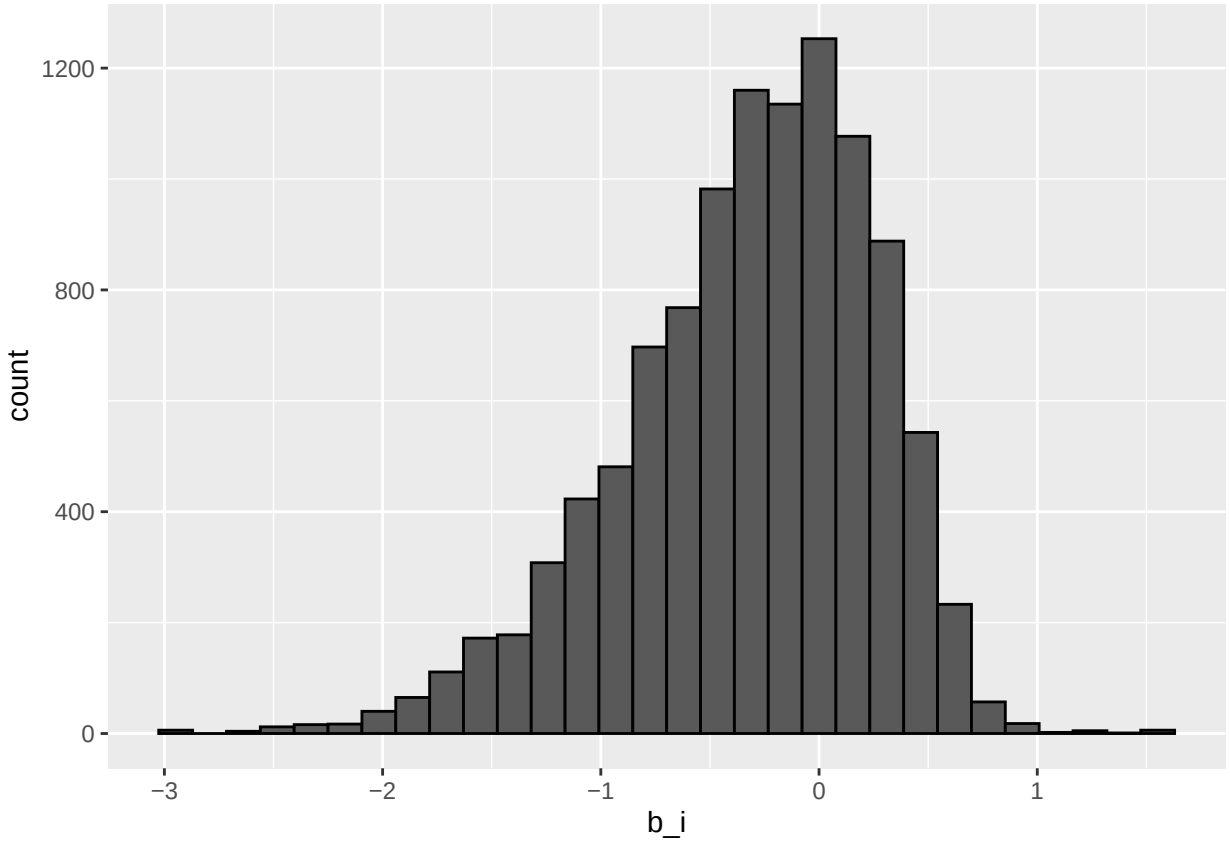
```
## [1] 1.060503
```

To improve our initial model with just the average, we check the distribution of the residuals of the predicted ratings of the model we have so far with the different biases/effects. This is so we can visualize if there is a universal pattern for each bias that we can find wherein we could fit our model to predict ratings. We do this step by step after each additional bias/effect added to the model.

1. Movie effect

```
library(tidyverse)
movie_effect <- train_set %>%
  group_by(movieId) %>%
  summarize(b_i = mean(rating - mu))

movie_effect %>%
  ggplot(aes(b_i)) +
  geom_histogram(bins = 30, color = "black")
```



Based from the plot, we observe that there is substantial variation in the ratings, depending on what movie it is. It just goes to show that some movies are generally rated higher than others. Thus, we may add this to our model. The RMSE for this new model is as follows:

```
pred <- test_set %>%
  left_join(movie_effect, by='movieId') %>%
  mutate(pred = mu + b_i) %>%
  .$pred

rmse_m <- RMSE(pred, test_set$rating)
rmse_m
```

```
## [1] 0.9435731
```

The model would look like the following:

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

where b_i represents average ranking for movie i

So far, below are our models and their RMSEs.

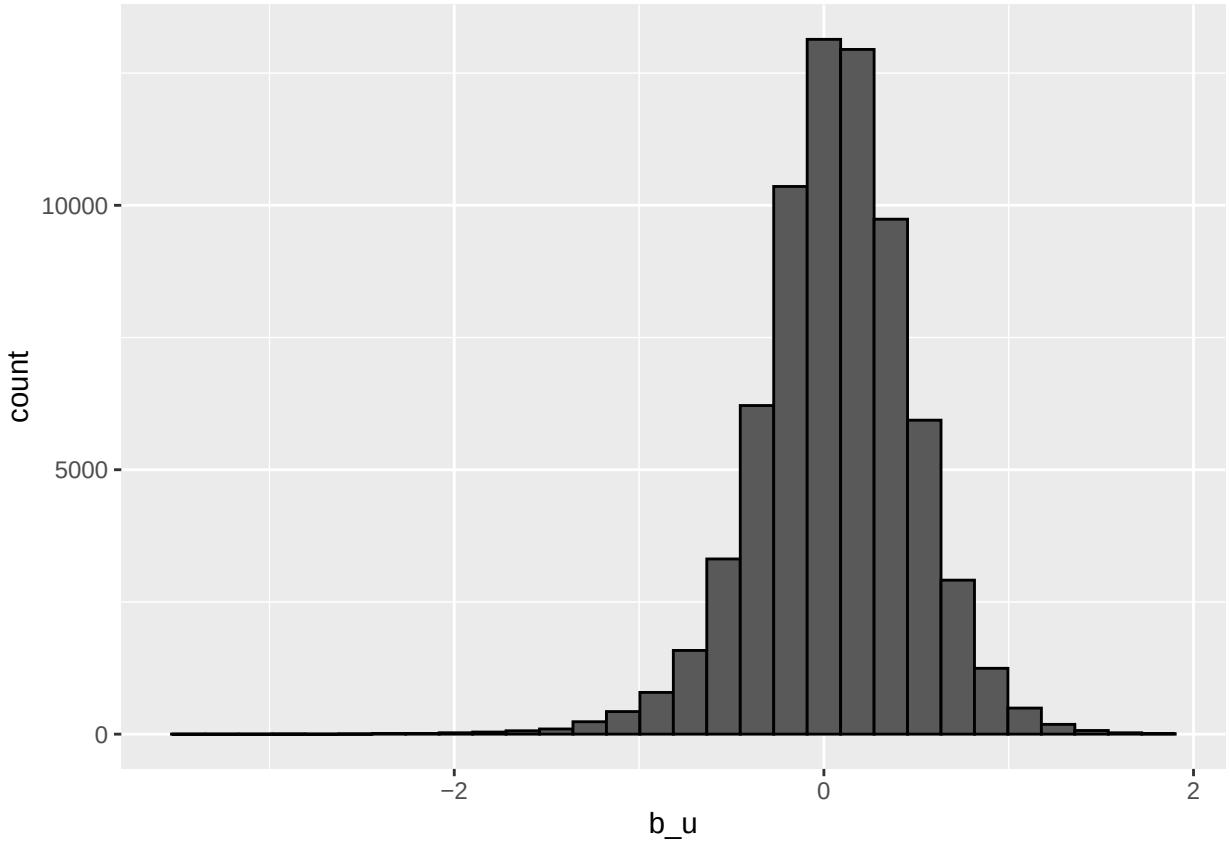
```
##               method      RMSE
## 1   Just the average 1.0605032
## 2 Movie Effect Model 0.9435731
```

2. User effect

Next we check for user effect which is the average rating for each user. Below shows the average residuals of ratings from our previous model for each user.

```
user_effect <- train_set %>%
  left_join(movie_effect, by = 'movieId') %>%
  group_by(userId) %>%
  summarize(b_u = mean(rating - mu - b_i))

user_effect %>%
  ggplot(aes(b_u)) +
  geom_histogram(bins = 30, color = "black")
```



We can observe from the plot that there is variation in the rating for each user. Thus, we add that to the model and get the following RMSE:

```
pred <- test_set %>%
  left_join(movie_effect, by = 'movieId') %>%
  left_join(user_effect, by = 'userId') %>%
  mutate(pred = mu + b_i + b_u) %>%
  .$pred

rmse_mu <- RMSE(pred, test_set$rating)
rmse_mu
```

```
## [1] 0.8653122
```

The model would look like the following:

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

where b_u represents a user-specific effect

So far, below are our models and their RMSEs.

```
##                       method      RMSE
## 1          Just the average 1.0605032
## 2        Movie Effect Model 0.9435731
## 3 Movie + User Effect Model 0.8653122
```

3. Genre effect

Below are all the genres in the edx set and their frequencies.

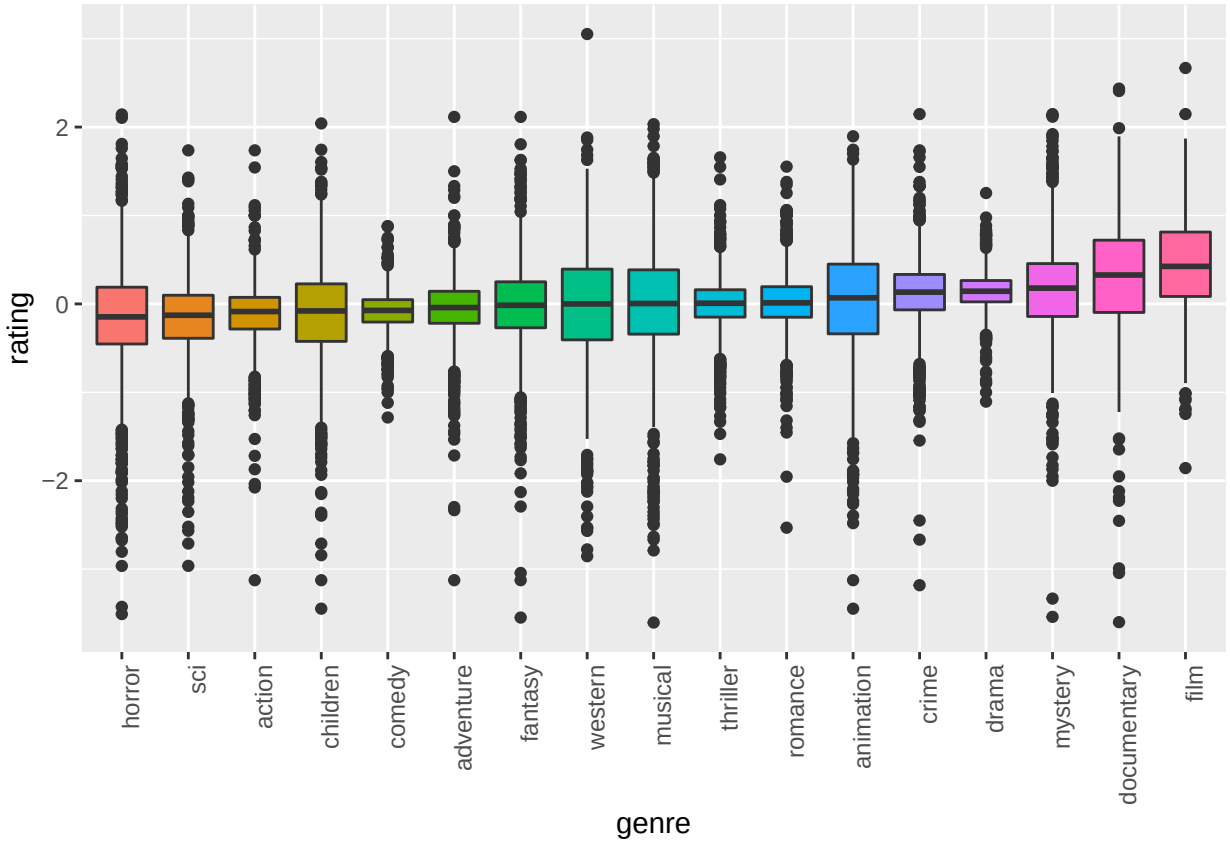
```
genres <- unique(str_match(edx$genres, "[a-zA-Z]+"))[,1]

genre_count <- sapply(genres, function(x){
  y <- sum(str_detect(edx$genres, x))
  return(y)
})

data.frame(genre_count) %>%
  arrange(-genre_count)
```

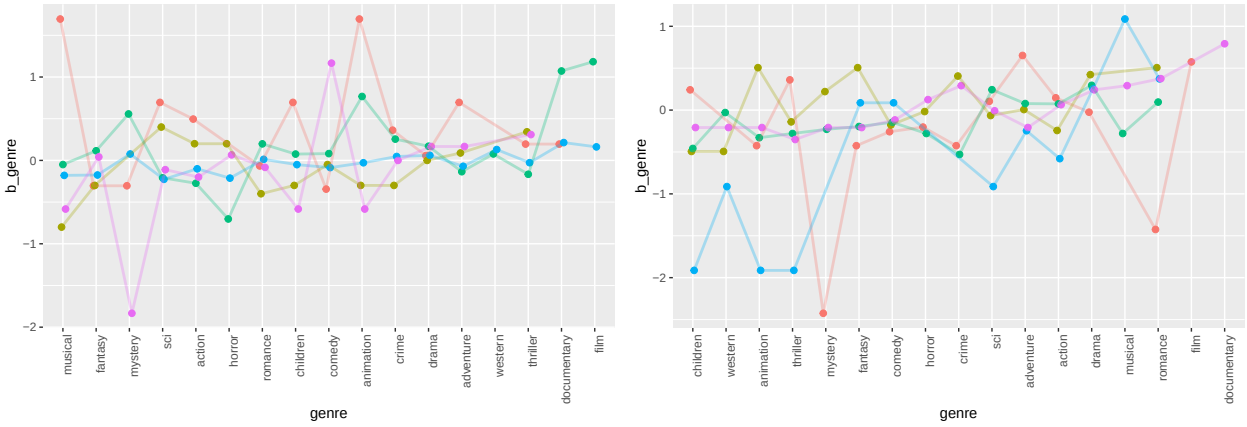
```
##             genre_count
## Drama           3910127
## Comedy          3540930
## Action          2560545
## Thriller        2325899
## Adventure       1908892
## Romance         1712100
## Sci             1341183
## Crime           1327715
## Fantasy          925637
## Children         737994
## Horror           691485
## Mystery          568332
## War              511147
## Animation        467168
## Musical          433080
## Western          189394
## Film             118541
## Documentary       93066
## IMAX               8181
## no                    7
```

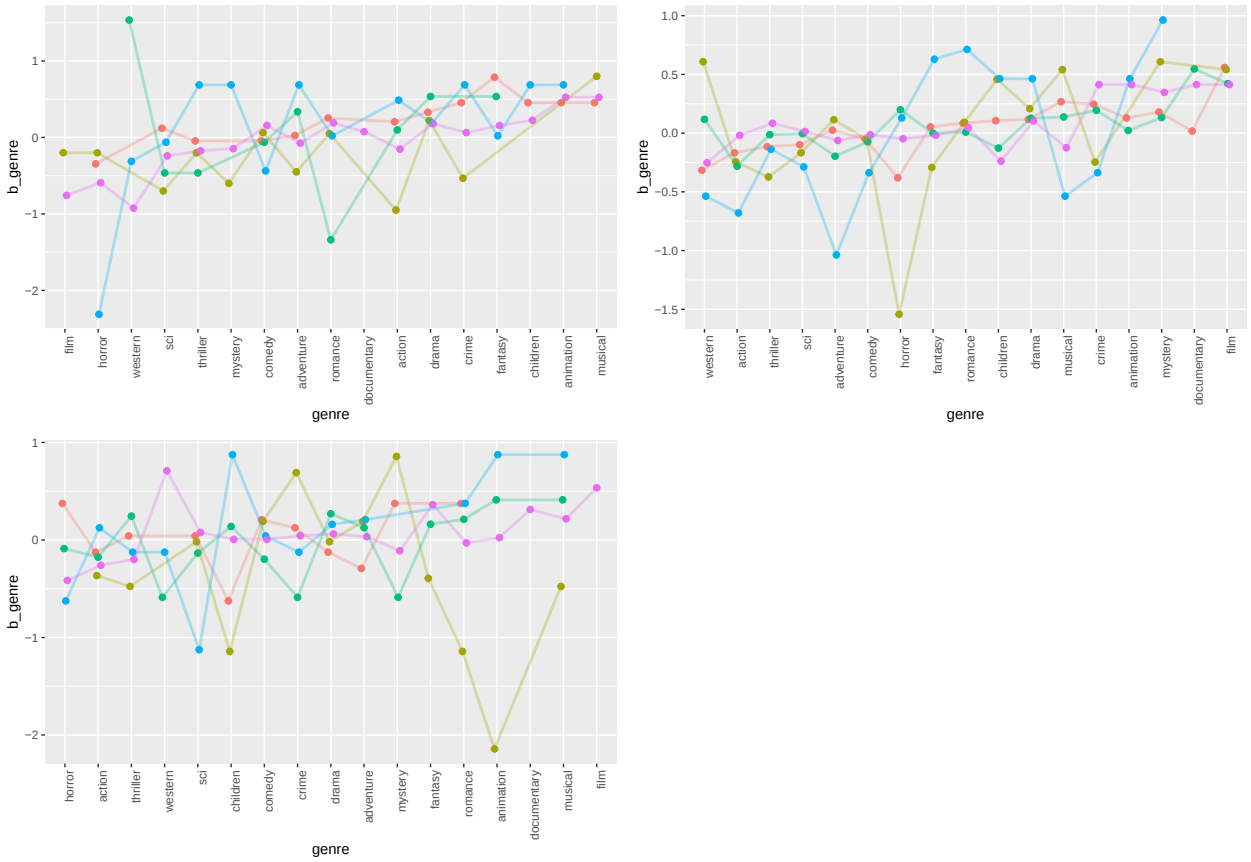
Next we check for the genre effect. First we visualize if there is variation in the rating based from genre. We will not consider the genre 'no' since it is not a genre.



As seen from the plot, there is only very slight differences in the average rating from genre to genre. Which may not give too much to our model.

Below we try to look at average rating from genre to genre but for each user. We produce 5 plots sampling 5 users for each.





As observed from the plots, there is variability in the ratings of each user depending on the genre. This shows that some users may have different tastes in genres.

So adding genre effects, the model would look like the following:

$$Y_{u,i} = \mu + b_i + b_u + \sum_{k=1}^{K} b_{k,u} + \varepsilon_{u,i}$$

where k is a genre and $b_{k,u}$ represents a genre-user effect

We add the genre effects to our model with the following code:

```
comedy_effect <- train_set %>%
  left_join(movie_effect, by = 'movieId') %>%
  left_join(user_effect, by = 'userId') %>%
  mutate(comedy = ifelse(str_detect(genres, "Comedy"), "1", "0"), pred = mu + b_u + b_i) %>%
  filter(comedy == "1") %>%
  group_by(userId) %>%
  summarize(b_comedy = mean(rating - pred))


drama_effect  <- train_set %>%
  left_join(movie_effect, by = 'movieId') %>%
  left_join(user_effect, by = 'userId') %>%
  left_join(comedy_effect, by = 'userId') %>%
  mutate(drama = ifelse(str_detect(genres, "Drama"), "1", "0"),
         pred = mu + b_u + b_i +
```

```
          ifelse(str_detect(genres, "Comedy") & !is.na(b_comedy), b_comedy, 0)) %>%
  filter(drama == "1") %>%
  group_by(userId) %>%
  summarize(b_drama = mean(rating - pred))
```

We continue the above code by adding the rest of the genres one by one.

We check the RMSE of the new model with genre effect with the following code:

Upon completing and running the code, it can be observed that the RMSE for the model with the added genre effect is 0.8757.

Below are the RMSEs of all the models so far:

```
##                               method      RMSE
## 1                  Just the average 1.0605032
## 2                Movie Effect Model 0.9435731
## 3         Movie + User Effect Model 0.8653122
## 4 Movie + User + Genre Effect Model 0.8757116
```

The RMSE for Movie + User + Genre Effect Model is higher than our previous model. This may be due to some users having only rated a certain genre a few times. These would result in noisy estimates that we should not trust, and may result in overfitting. Thus, we perform regularization with our current model. During the process, we also include the movie effect and user effect.

To find lambda for regularization, we divide again the train_set into two for cross validation, namely: train_set_2 and test_set_2. They are 90% and 10% of the train_set, respectively. We get them using the following code:

```
test_index_2 <- createDataPartition(train_set$rating, times = 1, p = 0.1, list = F)
train_set_2 <- train_set[-test_index_2,]
temp2 <- train_set[test_index_2,]
test_set_2 <- temp2 %>%
  semi_join(train_set_2, by = 'movieId') %>%
  semi_join(train_set_2, by = 'userId')
```

We also get a new average rating.

```
mu2 <- mean(train_set_2$rating)
mu2
```

Below is a sample of the code on how the penalty terms lambda was chosen

```
lambdas <- seq(0,30)

rmses <- sapply(lambdas, function(l){
movie_effect <- train_set_2 %>%
  group_by(movieId) %>%
  summarize(b_i = sum(rating-mu)/(n() + l))

pred <- test_set_2 %>%
  left_join(movie_effect, by='movieId') %>%
  mutate(pred = mu + b_i) %>%
  .$pred
```

```
return(RMSE(pred, test_set_2$rating))
})
```

The lambda with the lowest RMSE is chosen. That procedure is done for each effect/bias added to the model, one by one.

After getting the lambda for each effect which gets the lowest RMSE for test_set_2, we use those lambdas for our current model. We get the new bias/effect values where we apply regularization:

We find the RMSE for our new model with the following code:

```
pred <- test_set %>%
  left_join(movie_effect, by = 'movieId') %>%
  left_join(user_effect, by='userId') %>%
  left_join(drama_effect, by='userId') %>%
  left_join(comedy_effect, by='userId') %>%
  left_join(action_effect, by='userId') %>%
  left_join(thriller_effect, by='userId') %>%
  left_join(adventure_effect, by='userId') %>%
  left_join(romance_effect, by='userId') %>%
  left_join(sci_effect, by='userId') %>%
  left_join(crime_effect, by='userId') %>%
  left_join(fantasy_effect, by='userId') %>%
  left_join(children_effect, by='userId') %>%
  left_join(horror_effect, by='userId') %>%
  left_join(mystery_effect, by='userId') %>%
  left_join(war_effect, by='userId') %>%
  left_join(animation_effect, by='userId') %>%
  left_join(musical_effect, by='userId') %>%
  left_join(western_effect, by='userId') %>%
  left_join(film_effect, by='userId') %>%
  left_join(documentary_effect, by='userId') %>%
  left_join(imax_effect, by='userId') %>%
  mutate(pred = mu + b_i + b_u +
           ifelse(str_detect(genres, "Drama") & !is.na(b_drama), b_drama, 0) +
           ifelse(str_detect(genres, "Comedy") & !is.na(b_comedy), b_comedy, 0) +
           ifelse(str_detect(genres, "Action") & !is.na(b_action), b_action, 0) +
           ifelse(str_detect(genres, "Thriller") & !is.na(b_thriller), b_thriller, 0) +
           ifelse(str_detect(genres, "Adventure") & !is.na(b_adventure), b_adventure, 0) +
           ifelse(str_detect(genres, "Romance") & !is.na(b_romance), b_romance, 0) +
           ifelse(str_detect(genres, "Sci") & !is.na(b_sci), b_sci, 0) +
           ifelse(str_detect(genres, "Crime") & !is.na(b_crime), b_crime, 0) +
           ifelse(str_detect(genres, "Fantasy") & !is.na(b_fantasy), b_fantasy, 0) +
           ifelse(str_detect(genres, "Children") & !is.na(b_children), b_children, 0) +
           ifelse(str_detect(genres, "Horror") & !is.na(b_horror), b_horror, 0) +
           ifelse(str_detect(genres, "Mystery") & !is.na(b_mystery), b_mystery, 0) +
           ifelse(str_detect(genres, "War") & !is.na(b_war), b_war, 0) +
           ifelse(str_detect(genres, "Animation") & !is.na(b_animation), b_animation, 0) +
           ifelse(str_detect(genres, "Musical") & !is.na(b_musical), b_musical, 0) +
           ifelse(str_detect(genres, "Western") & !is.na(b_western), b_western, 0) +
           ifelse(str_detect(genres, "Film") & !is.na(b_film), b_film, 0) +
           ifelse(str_detect(genres, "Documentary") & !is.na(b_documentary),
                  b_documentary, 0) +
           ifelse(str_detect(genres, "IMAX") & !is.na(b_imax), b_imax, 0)
```

```
  ) %>%
    .$pred

  rmse_Rmug <- RMSE(pred, test_set$rating)
  rmse_Rmug
```

```
## [1] 0.8461327
```

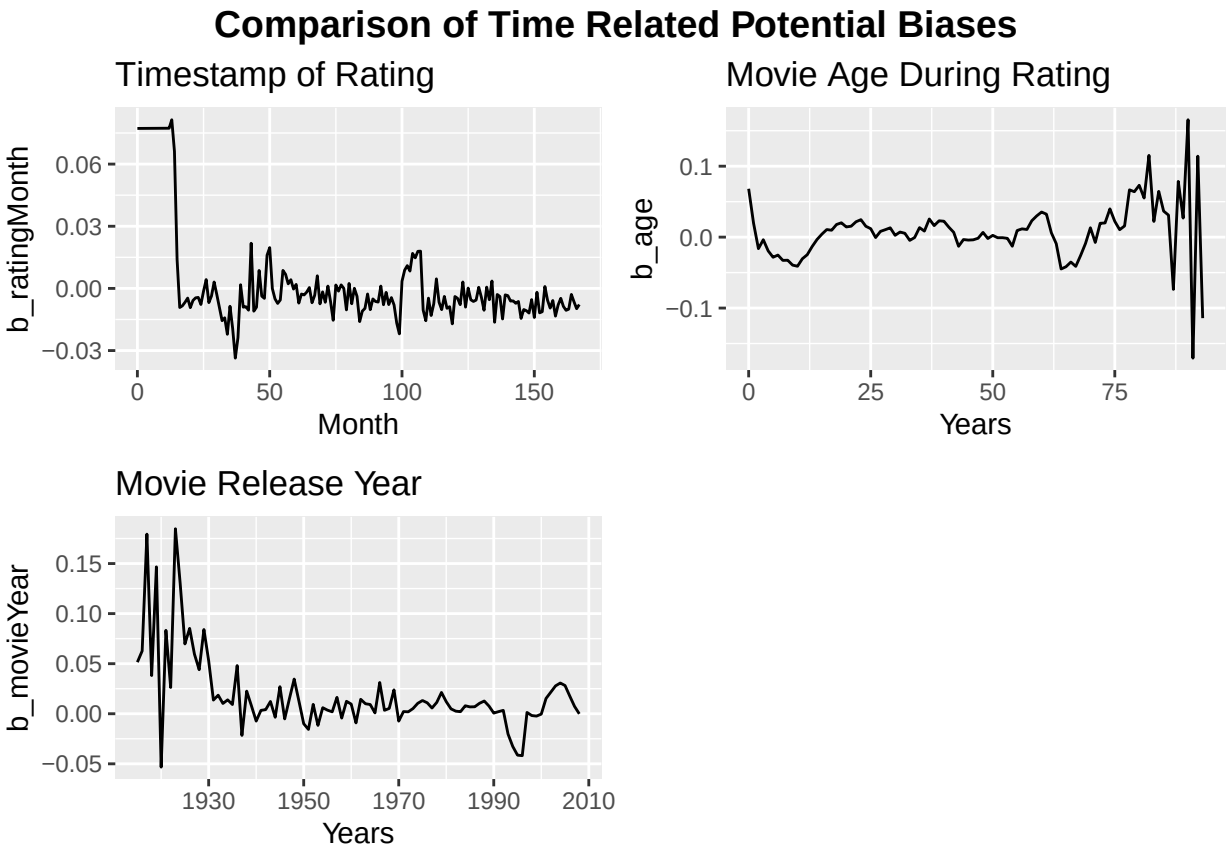
The new model that includes genre effect is now better than the previous models.

```
##                                     method      RMSE
## 1                          Just the average 1.0605032
## 2                        Movie Effect Model 0.9435731
## 3                 Movie + User Effect Model 0.8653122
## 4          Movie + User + Genre Effect Model 0.8757116
## 5 Regularized Movie-User-Genre Effect Model 0.8461327
```

4. Movie release year effect

For the last effect, before movie release year effect was chosen, three potential biases were tested. First was the timestamp of the rating, second was the age of the movie during the time of rating, and lastly the movie release year.

Below we compare the three time related potential biases for the model.



Upon inspection of the graphs, firstly, timestamp of rating seems to just be noise with no significant pattern

or trend. Movie age during rating looks to have some small pattern but not easily explainable which may just be noise. Ratings however, grouped by movie release year, seems to tell a story. It indicates that movies released around before 1930 seemed to have better ratings. It also tells that movies released around the 90's are not usually received well by the users, while movies in the 2000s are rated higher. Based from this observation, movie release year was chosen to be a potential bias that could improve the current model.

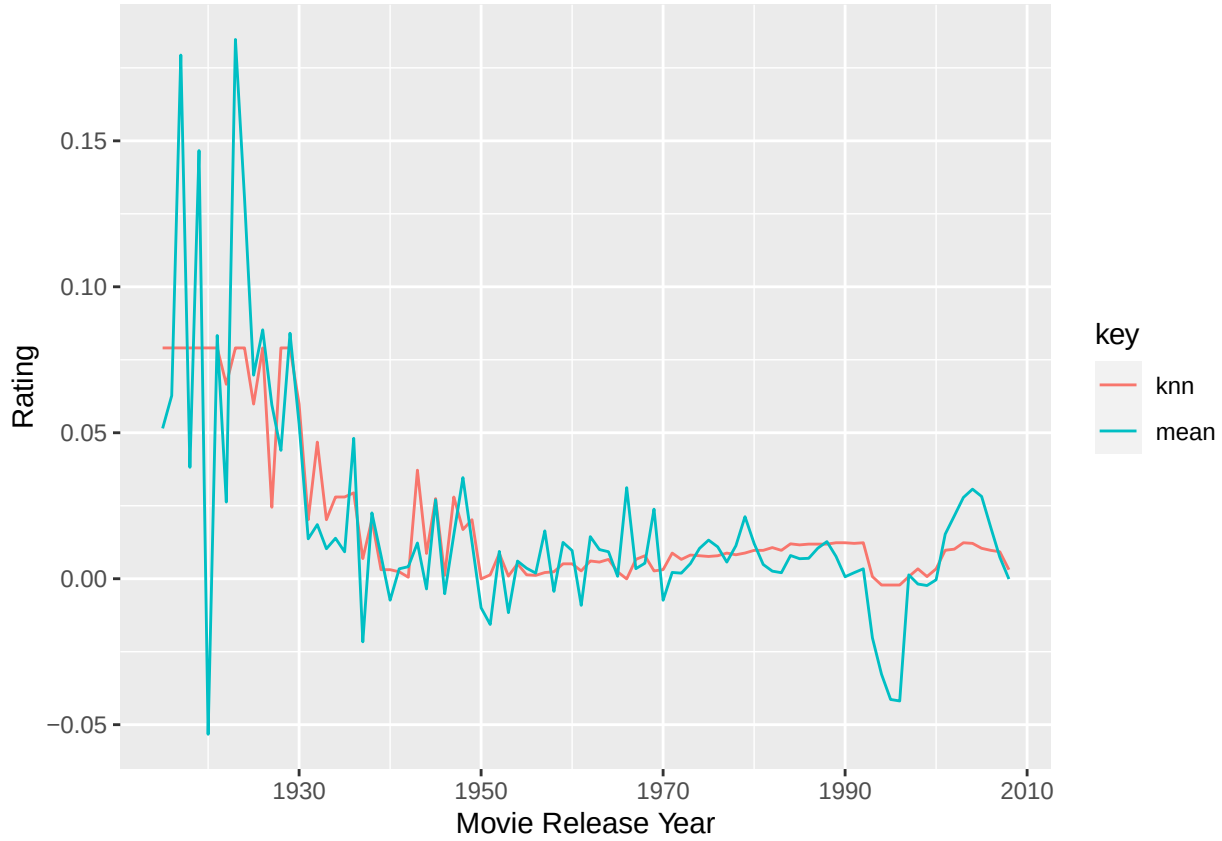
We run a k-nearest neighbor algorithm to get a line that would fit the trend instead of the specific ups and down of the line, which could result in overfitting.

```
library(caret)
train_knn <- train(b_movieYear ~ ., method = "knn",
                   tuneGrid = data.frame(k = seq(1,51,2)),
                   data = movieYear)

predict_knn <- predict(train_knn, movieYear)
```

Below is a visualization of our fitted line.

```
movieYear %>%
  mutate(knn = predict_knn) %>%
  rename(mean = b_movieYear) %>%
  gather(mean, knn, key = key,value = value) %>%
  ggplot(aes(movieYear, value, group = key, color = key)) +
  geom_line() +
  xlab("Movie Release Year") +
  ylab("Rating")
```



Below is the code for our final bias.

```
movieYear_effect <- movieYear %>%
  mutate(knn = predict_knn)
```

The final model would look like the following:

$$Y_{u,i} = \mu + b_i + b_u + \sum_{k=1}^{K} b_{k,u} + f(d_i) + \varepsilon_{u,i}$$

where d_i is the year the movie was released, with f a function of d_i

5. Testing the model with test_set

```
pred <- test_set %>%
  left_join(movie_effect, by = 'movieId') %>%
  left_join(user_effect, by = 'userId') %>%
  left_join(comedy_effect, by = 'userId') %>%
  left_join(drama_effect, by = 'userId') %>%
  left_join(action_effect, by = 'userId') %>%
  left_join(thriller_effect, by = 'userId') %>%
  left_join(adventure_effect, by = 'userId') %>%
  left_join(romance_effect, by = 'userId') %>%
  left_join(sci_effect, by = 'userId') %>%
  left_join(crime_effect, by = 'userId') %>%
  left_join(fantasy_effect, by = 'userId') %>%
  left_join(children_effect, by = 'userId') %>%
  left_join(horror_effect, by = 'userId') %>%
  left_join(mystery_effect, by = 'userId') %>%
  left_join(war_effect, by = 'userId') %>%
  left_join(animation_effect, by = 'userId') %>%
  left_join(musical_effect, by = 'userId') %>%
  left_join(western_effect, by = 'userId') %>%
  left_join(film_effect, by = 'userId') %>%
  left_join(documentary_effect, by = 'userId') %>%
  left_join(imax_effect, by = 'userId') %>%
  mutate(pred = mu + b_i + b_u,
         newpred = pred +
           ifelse(str_detect(genres, "Comedy") & !is.na(b_comedy), b_comedy, 0) +
           ifelse(str_detect(genres, "Drama") & !is.na(b_drama), b_drama, 0) +
           ifelse(str_detect(genres, "Action") & !is.na(b_action), b_action, 0) +
           ifelse(str_detect(genres, "Thriller") & !is.na(b_thriller), b_thriller, 0) +
           ifelse(str_detect(genres, "Adventure") & !is.na(b_adventure), b_adventure, 0) +
           ifelse(str_detect(genres, "Romance") & !is.na(b_romance), b_romance, 0) +
           ifelse(str_detect(genres, "Sci") & !is.na(b_sci), b_sci, 0) +
           ifelse(str_detect(genres, "Crime") & !is.na(b_crime), b_crime, 0) +
           ifelse(str_detect(genres, "Fantasy") & !is.na(b_fantasy), b_fantasy, 0) +
           ifelse(str_detect(genres, "Children") & !is.na(b_children), b_children, 0) +
           ifelse(str_detect(genres, "Horror") & !is.na(b_horror), b_horror, 0) +
           ifelse(str_detect(genres, "Mystery") & !is.na(b_mystery), b_mystery, 0) +
           ifelse(str_detect(genres, "War") & !is.na(b_war), b_war, 0) +
           ifelse(str_detect(genres, "Animation") & !is.na(b_animation), b_animation, 0) +
           ifelse(str_detect(genres, "Musical") & !is.na(b_musical), b_musical, 0) +
```

```
          ifelse(str_detect(genres, "Western") & !is.na(b_western), b_western, 0) +
          ifelse(str_detect(genres, "Film") & !is.na(b_film), b_film, 0) +
          ifelse(str_detect(genres, "Documentary") & !is.na(b_documentary),
                 b_documentary, 0) +
          ifelse(str_detect(genres, "IMAX") & !is.na(b_imax), b_imax, 0)
  ) %>%
  mutate(movieYear = as.numeric(str_match(title, "(\\d{4})\\)$")[,2])) %>%
  left_join(movieYear_effect) %>%
  mutate(newpred = newpred + knn) %>%
  .$newpred
```

```
## Joining, by = "movieYear"
```

```
rmse_Rmug_my <- RMSE(pred, test_set$rating)
rmse_Rmug_my
```

```
## [1] 0.8460816
```

The new model that includes movie year effect is now better than the previous models.

```
##                                                    method      RMSE
## 1                                       Just the average 1.0605032
## 2                                    Movie Effect Model 0.9435731
## 3                                Movie + User Effect Model 0.8653122
## 4                         Movie + User + Genre Effect Model 0.8757116
## 5                 Regularized Movie-User-Genre Effect Model 0.8461327
## 6 Movie Year + Regularized Movie-User-Genre Effect Model 0.8460816
```

III. Results

For the final test of our model, we finally use the validation set.

```
pred <- validation %>%
  left_join(movie_effect, by = 'movieId') %>%
  left_join(user_effect, by = 'userId') %>%
  left_join(comedy_effect, by = 'userId') %>%
  left_join(drama_effect, by = 'userId') %>%
  left_join(action_effect, by = 'userId') %>%
  left_join(thriller_effect, by = 'userId') %>%
  left_join(adventure_effect, by = 'userId') %>%
  left_join(romance_effect, by = 'userId') %>%
  left_join(sci_effect, by = 'userId') %>%
  left_join(crime_effect, by = 'userId') %>%
  left_join(fantasy_effect, by = 'userId') %>%
  left_join(children_effect, by = 'userId') %>%
  left_join(horror_effect, by = 'userId') %>%
  left_join(mystery_effect, by = 'userId') %>%
  left_join(war_effect, by = 'userId') %>%
  left_join(animation_effect, by = 'userId') %>%
  left_join(musical_effect, by = 'userId') %>%
```

```
  left_join(western_effect, by = 'userId') %>%
  left_join(film_effect, by = 'userId') %>%
  left_join(documentary_effect, by = 'userId') %>%
  left_join(imax_effect, by = 'userId') %>%
  mutate(b_u = ifelse(is.na(b_u), 0, b_u),
         b_i = ifelse(is.na(b_i), 0, b_i),
         pred = mu + b_i + b_u,
         newpred = pred +
           ifelse(str_detect(genres, "Comedy") & !is.na(b_comedy), b_comedy, 0) +
           ifelse(str_detect(genres, "Drama") & !is.na(b_drama), b_drama, 0) +
           ifelse(str_detect(genres, "Action") & !is.na(b_action), b_action, 0) +
           ifelse(str_detect(genres, "Thriller") & !is.na(b_thriller), b_thriller, 0) +
           ifelse(str_detect(genres, "Adventure") & !is.na(b_adventure), b_adventure, 0) +
           ifelse(str_detect(genres, "Romance") & !is.na(b_romance), b_romance, 0) +
           ifelse(str_detect(genres, "Sci") & !is.na(b_sci), b_sci, 0) +
           ifelse(str_detect(genres, "Crime") & !is.na(b_crime), b_crime, 0) +
           ifelse(str_detect(genres, "Fantasy") & !is.na(b_fantasy), b_fantasy, 0) +
           ifelse(str_detect(genres, "Children") & !is.na(b_children), b_children, 0) +
           ifelse(str_detect(genres, "Horror") & !is.na(b_horror), b_horror, 0) +
           ifelse(str_detect(genres, "Mystery") & !is.na(b_mystery), b_mystery, 0) +
           ifelse(str_detect(genres, "War") & !is.na(b_war), b_war, 0) +
           ifelse(str_detect(genres, "Animation") & !is.na(b_animation), b_animation, 0) +
           ifelse(str_detect(genres, "Musical") & !is.na(b_musical), b_musical, 0) +
           ifelse(str_detect(genres, "Western") & !is.na(b_western), b_western, 0) +
           ifelse(str_detect(genres, "Film") & !is.na(b_film), b_film, 0) +
           ifelse(str_detect(genres, "Documentary") & !is.na(b_documentary),
                  b_documentary, 0) +
           ifelse(str_detect(genres, "IMAX") & !is.na(b_imax), b_imax, 0)
  ) %>%
  mutate(movieYear = as.numeric(str_match(title, "(\\d{4})\\)$")[,2])) %>%
  left_join(movieYear_effect) %>%
  mutate(newpred = newpred + knn) %>%
  .$newpred
```

```
## Joining, by = "movieYear"
```

```
rmse_val <- RMSE(pred, validation$rating)
rmse_val
```

```
## [1] 0.8464914
```

The RMSE of our model to the validation set is 0.8465.

IV. Conclusion

Using the 10M version of the MovieLens dataset, we were able to create a movie recommendation system. The effects/biases that were used for the algorithm was the user effect, movie effect, genre effect, and year movie released effect. Regularization was done on the user effect, movie effect, and genre effect to avoid noisy estimates and overfitting. All training and model comparisons was done solely on the edx set. The validation set was used in the end to check the RMSE of the final model. The final RMSE of our model to the validation set is 0.8465. The model may possibly be improved if principal component analysis could be utilized to detect potential structures from the data to fit the model into.