Movielens Project

Robert Young

2/5/2021

** Introduction **

In the edX course “Data Science: Machine Learning”, a model was developed which predicts movie ratings based on the movies reviewed and the ratings of various users. The goal of this project is to improve on this model by taking into account the genre of each movie.

The following code generates the two datasets which will be used in this project. The “edx” dataset will be used to develop the model and the “validation” dataset will be used to evaluate the final model produced. Here is the code used to generate these datasets:

```
##########################################################
# Create edx set, validation set (final hold-out test set)
##########################################################

# Note: this process could take a couple of minutes
options(warn = -1)
if(!require(tidyverse)) install.packages("tidyverse", repos =
"http://cran.us.r-project.org")

## Loading required package: tidyverse

## -- Attaching packages --------------------------------------- tidyverse
1.3.0 --

## v ggplot2 3.3.3     v purrr   0.3.4
## v tibble  3.0.1     v dplyr   1.0.0
## v tidyr   1.1.0     v stringr 1.4.0
## v readr   1.3.1     v forcats 0.5.0

## -- Conflicts ------------------------------------------
tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()

if(!require(caret)) install.packages("caret", repos = "http://cran.us.r-
project.org")

## Loading required package: caret

## Loading required package: lattice

##
## Attaching package: 'caret'
## The following object is masked from 'package:purrr':
##
##     lift
```

```r
if(!require(data.table)) install.packages("data.table", repos =
"http://cran.us.r-project.org")
```

```
## Loading required package: data.table
```

```
##
## Attaching package: 'data.table'
```

```
## The following objects are masked from 'package:dplyr':
##
##     between, first, last
```

```
## The following object is masked from 'package:purrr':
##
##     transpose
```

```r
library(tidyverse)
library(caret)
library(data.table)

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

dl <- tempfile()
download.file("http://files.grouplens.org/datasets/movielens/ml-10m.zip", dl)

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

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

# if using R 3.6 or earlier:
#movies <- as.data.frame(movies) %>% mutate(movieId =
as.numeric(levels(movieId))[movieId],
                                            #title = as.character(title),
                                            #genres = as.character(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")
```

```
# Validation set will be 10% of MovieLens data
set.seed(1, sample.kind="Rounding") # if using R 3.5 or earlier, use
`set.seed(1)`
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)
```

In both the “edx” and the “validation” datasets, there is a variable called “genres” which contains the genre of each movie. Each observation for this variable consists of a single genre like “Drama” or a combination of genres like “Drama | Romance | Mystery”. Both datasets also contain the variables movieId, userId, and rating which will be used in the development of the final model.

We will partition the “edx” dataset into a training set and a test set and the genre variable will be examined using both data visualization and numeric summaries. The function RMSE (root mean squared error) will be defined which will be used to evaluate the accuracy of a model. The models used in the Machine Language course will be reviewed since the new model will be based on them. Then the new model will be developed using the information about the genre of each movie. After the new model is finalized, the “validation” dataset will be used to evaluate the model using the RMSE.

** Methods / Analysis **

The first step will be to partition the “edx” dataset into a train_set and a test_set using the following code:

```
#partition the data set into train_set and test_set
dat<-edx
set.seed(1, sample.kind ="Rounding")
test_index <- createDataPartition(y = dat$rating, times = 1,
                                  p = 0.2, list = FALSE)
train_set <- dat[-test_index, ]
test_set <- dat[test_index, ]
```

The main focus of the project will be analyzing and using the information provided about the genres of the movies. In order to accomplish this, a table of ratings for each individual genre will be created using the following code:

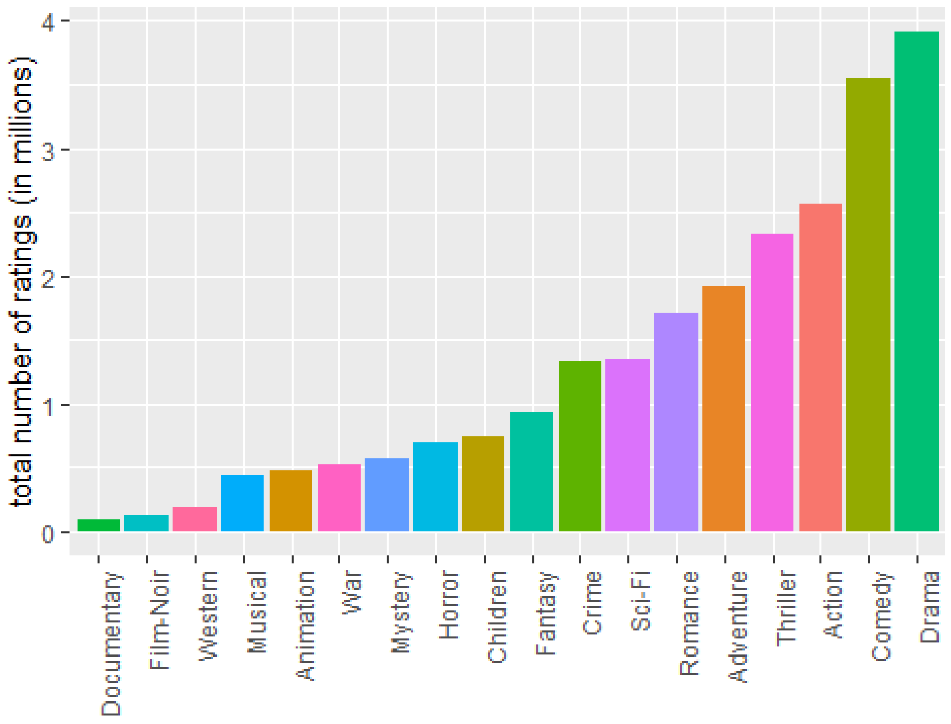
```
#create a table of ratings for each genre
movie_genres <- c("Action", "Adventure", "Animation", "Children", "Comedy",
                  "Crime", "Documentary", "Drama", "Fantasy", "Film-Noir",
                  "Horror", "Musical", "Mystery", "Romance", "Sci-Fi",
                  "Thriller", "War", "Western")
#the following function returns all the ratings for a specified genre
get_ratings <- function(genre) {
    dat %>%
    #find all the observations for the specified genre
    filter(str_detect(genres, genre)) %>%
    #replace the genres field with the name of the specified genre
    mutate(genres = genre) %>%
    #return the movie Id, user ID, genre, and ratings for the specified genre
    select(movieId, userId, genres, rating)
}
#create a table of ratings for each genre in movie_genres
ratings_table<-data.frame(movieId = integer(), userId = integer(),
                          genres = character(), rating = numeric())
for (i in 1:length(movie_genres)) {
  result <- get_ratings(movie_genres[i])
  ratings_table <- bind_rows(ratings_table, result)
}
```

We can get a listing of the number of observations for each genre using this code:

```
#display the number of observations for each genre
ratings_table %>%
  group_by(genres) %>%
  summarize(frequency = n(), .groups = "drop") %>%
  arrange(desc(frequency))

## # A tibble: 18 x 2
##    genres      frequency
##    <chr>           <int>
##  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
```

And here is a plot of the frequency of each genre:

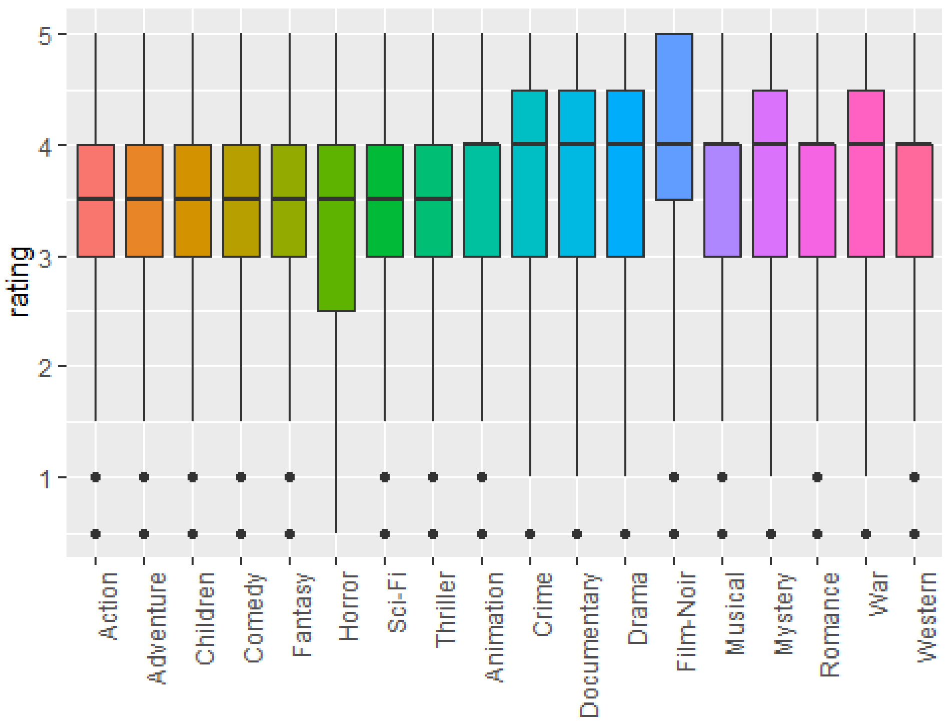
```
#plot the frequency of the ratings for each genre
ratings_table %>%
  group_by(genres) %>%
  summarize(num_ratings = n(), .groups = "drop") %>%
  ggplot(aes(reorder(genres, num_ratings),
             num_ratings / 1000000,
             fill = genres))+
  geom_bar(stat = "identity", width = 0.9) +
  theme(legend.position = "none") +
  xlab("") +
  theme(axis.text.x = element_text(angle = 90, hjust = 1)) +
  ylab("total number of ratings (in millions)")
```



We can see from both the numerical summary and the frequency plot that the number of observations differ from one genre to another. Perhaps this is due to the degree of popularity of each genre - the more popular a particular genre is, the more ratings are probably given for that genre. This would suggest that there is a “genre effect” on the prediction of the ratings.

The movies are rated on a scale of 0 to 5. Let's take a look at the distributions of the ratings for each genre.

```
#plot the distribution of the ratings for each genre
ratings_table %>%
  mutate(genres = fct_reorder(genres, rating, .fun = 'median' )) %>%
  ggplot(aes(genres, rating, fill = genres)) +
  geom_boxplot() +
  xlab("") +
  theme(axis.text.x = element_text(angle = 90, hjust = 1)) +
  theme(legend.position = "none")
```

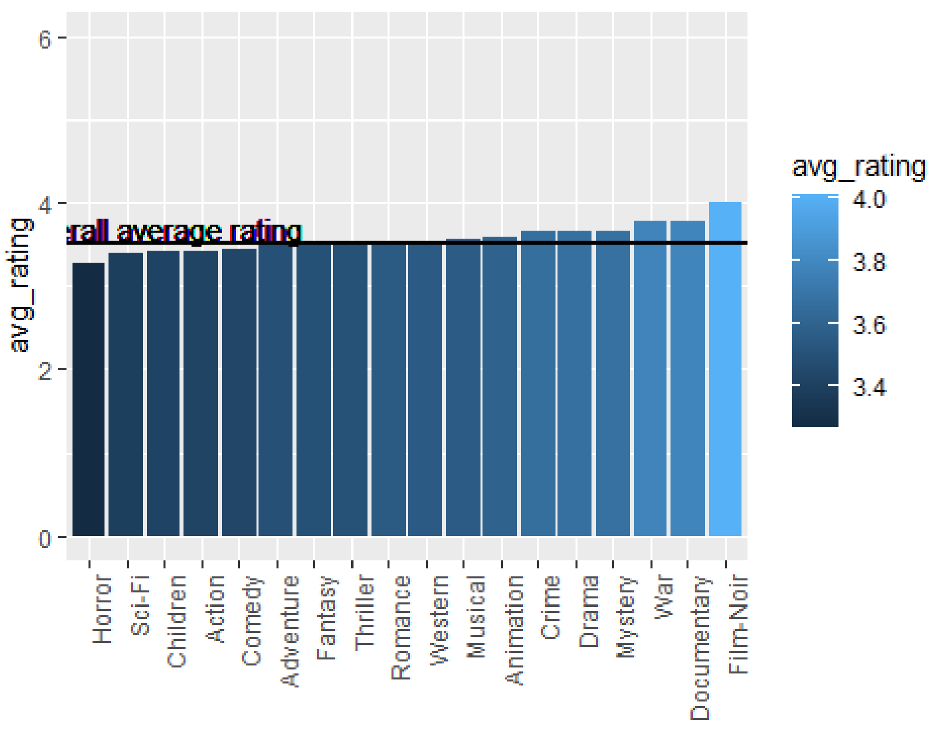


```
#print numerical summaries of the ratings data by genre
ratings_table %>%
  group_by(genres) %>%
  summarize(min = min(rating),
            Q1 = quantile(rating, 0.25),
            median = median(rating),
            mean=mean(rating),
            Q3 = quantile(rating, 0.75),
            max = max(rating), .groups = "drop") %>%
  arrange(median)

## # A tibble: 18 x 7
##    genres        min    Q1 median  mean    Q3   max
##    <chr>       <dbl> <dbl>  <dbl> <dbl> <dbl> <dbl>
##  1 Action        0.5   3      3.5  3.42   4       5
##  2 Adventure     0.5   3      3.5  3.49   4        5
##  3 Children      0.5   3      3.5  3.42   4        5
##  4 Comedy        0.5   3      3.5  3.44   4        5
##  5 Fantasy       0.5   3      3.5  3.50   4        5
##  6 Horror        0.5   2.5    3.5  3.27   4        5
##  7 Sci-Fi        0.5   3      3.5  3.40   4        5
##  8 Thriller      0.5   3      3.5  3.51   4        5
##  9 Animation     0.5   3      4    3.60   4        5
## 10 Crime         0.5   3      4    3.67   4.5      5
## 11 Documentary   0.5   3      4    3.78   4.5      5
## 12 Drama         0.5   3      4    3.67   4.5      5
## 13 Film-Noir     0.5   3.5    4    4.01   5        5
## 14 Musical       0.5   3      4    3.56   4        5
## 15 Mystery       0.5   3      4    3.68   4.5      5
## 16 Romance       0.5   3      4    3.55   4        5
## 17 War           0.5   3      4    3.78   4.5      5
## 18 Western       0.5   3      4    3.56   4        5
```

Notice that the genres fall into two distinct groups. Eight of the genres have a median rating of 3.5 while the remaining ten genres have a median ratings of 4. Most of the distributions are left skewed. The "Horror" genre has the largest range and "Film-Noir" has the largest proportion of ratings of 5. Again, the differences suggest that there may be a "genre effect" on the predicted ratings.

Now, let's take a look at the average rating for each genre and how that compares to the overall average rating for all of the genres.

```
avg_rating<-ratings_table %>%
  group_by(genres) %>%
  summarize(avg_rating = mean(rating), .groups = "drop")

overall_average_rating <- mean(ratings_table$rating)

avg_rating %>%
  ggplot(aes(reorder(genres, avg_rating), avg_rating, fill = avg_rating))+
  geom_bar(stat = "identity", width = 0.9) +
  geom_hline(yintercept = overall_average_rating, lwd = 1, color = "black") +
  geom_text(aes(3, overall_average_rating + 0.2,
                label = "overall average rating")) +
  ylim(0, 6)+
  xlab("") +
  theme(axis.text.x = element_text(angle = 90, hjust = 1))
```



All of the genres have an average rating close to the overall average. Some are below the overall average and some are above.

Now that we have examined the genre data more closely, we are ready to review the models that have been developed so far.

First we will define a function to calculate the RMSE (root mean squared error) which will be used to evaluate the accuracy of our models. The RMSE can be viewed as the average error in the prediction of the ratings, in other words the average difference between the actual, true ratings and the predicted ratings. Here is the code:

```
#define a function to calculate the RMSE
RMSE <- function(true_ratings, predicted_ratings) {
  sqrt(mean((true_ratings - predicted_ratings) ^ 2))
}
```

A reasonable starting point for a ratings model is to let the average rating be our initial prediction for each of the ratings. We will also create a table to keep track of the results and allow for comparisons among the various models.

```
#calculate the initial prediction
initial_prediction <- mean(train_set$rating)

#calculate the RMSE for model one
rmse <- RMSE(test_set$rating, initial_prediction)

#create a table to keep track of the results
```

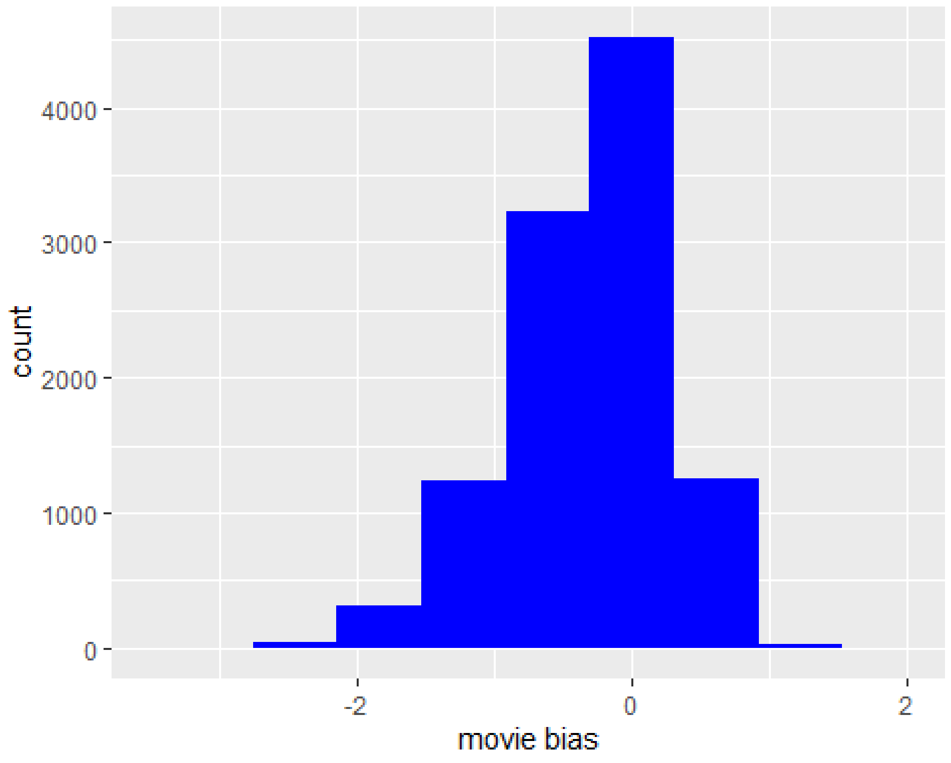
```
results <- tibble(method = "model_one", rmse=rmse)
results

## # A tibble: 1 x 2
##   method     rmse
##   <chr>     <dbl>
## 1 model_one  1.06
```

Now we will develop our second model. To improve the performance of our first model, we will add a movie bias, which is the average prediction error for each movie. The code is as follows:

```
#find the average prediction error for each movie
movie_bias <- train_set %>%
  group_by(movieId) %>%
  summarize(mbias = mean(rating - initial_prediction), .groups = "drop")
```

Let's take a look at the distribution of the movie_bias:

```
#plot the distribution of the movie bias
movie_bias %>% ggplot(aes(mbias)) +
  geom_histogram(bins = 10, fill = "blue") +
  xlim(-3.5, 2) +
  xlab("movie bias")
```



The plot is centered slightly to the left of 0 and is skewed left. There appears to be quite a bit of variability in the movie bias.

Let's make our predictions of the ratings taking the movie bias into account. Notice that the model includes some cleaning of data to replace any resulting NAs in the movie bias with the average movie bias.

```
#calculate predicted ratings using model two
predicted_ratings <- test_set %>%
  select(movieId) %>%
  left_join(movie_bias, by = "movieId") %>%
  mutate(mbias = ifelse(is.na(mbias), mean(mbias, na.rm = TRUE), mbias)) %>%
  summarize(pred = initial_prediction + mbias) %>%
  pull(pred)
```

Finally, we calculate the RMSE and compare the result to the previous model.

```
#calculate the RMSE for model two
rmse <- RMSE(predicted_ratings, test_set$rating)

#add the results to the results table and compare the results
results <- bind_rows(results,
                     tibble(method="model_two", rmse=rmse))
results

## # A tibble: 2 x 2
##   method     rmse
##   <chr>     <dbl>
## 1 model_one 1.06
## 2 model_two 0.944
```

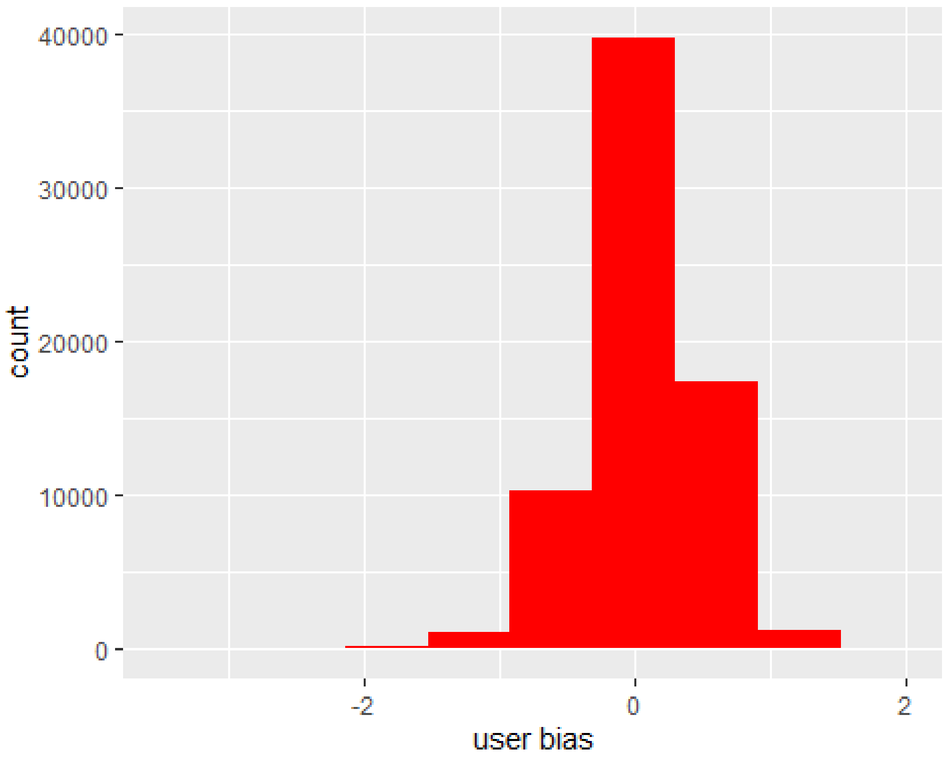
We see that model two is an improvement over model one.

Now, for our third model, we will take into account the user effect on our ratings predictions. First, we calculate the user bias which is the average error in the rating predicted by each user. Here is the code:

```
#find the average prediction error for each user
user_bias <- train_set %>%
  select(movieId, rating, userId) %>%
  left_join(movie_bias, by = "movieId") %>%
  group_by(userId) %>%
  summarize(ubias = mean(rating - initial_prediction - mbias), .groups =
"drop")
```

Let's take a look at the distribution of the user bias.

```
#show variability of the user_bias
user_bias %>% ggplot(aes(ubias)) +
  geom_histogram(bins = 10, fill = "red") +
  xlim(-3.5, 2) +
  xlab("user bias")
```



We see that the distribution is centered at approximately 0 and is approximately symmetrically distributed. There is some variation in the user bias, although it dows no appear to be as much as we had with the movie bias.

Now we make the predictions of the ratings using model three. Notice that there is some cleaning of the movie bias and the user bias to replace any NAs with the average movie bias and the average user bias respectively.

```
#calculate predicted ratings for model three
predicted_ratings <- test_set %>%
  select(movieId, userId) %>%
  left_join(movie_bias, by="movieId") %>%
  mutate(mbias = ifelse(is.na(mbias), mean(mbias, na.rm=TRUE), mbias)) %>%
  left_join(user_bias, by = "userId") %>%
  mutate(ubias = ifelse(is.na(ubias), mean(ubias, na.rm=TRUE), ubias))%>%
  summarize(pred = initial_prediction + mbias + ubias) %>%
  pull(pred)
```

We calculate the RMSE for model three and compare it to the results from the previous models.

```
#calculate the RMSE for model three
rmse <- RMSE(predicted_ratings, test_set$rating)

#compare the results to the results of the previous models
results <- bind_rows(results,
                     tibble(method = "model_three", rmse = rmse))
results
```

```
## # A tibble: 3 x 2
##   method       rmse
##   <chr>       <dbl>
## 1 model_one   1.06
## 2 model_two   0.944
## 3 model_three 0.866
```

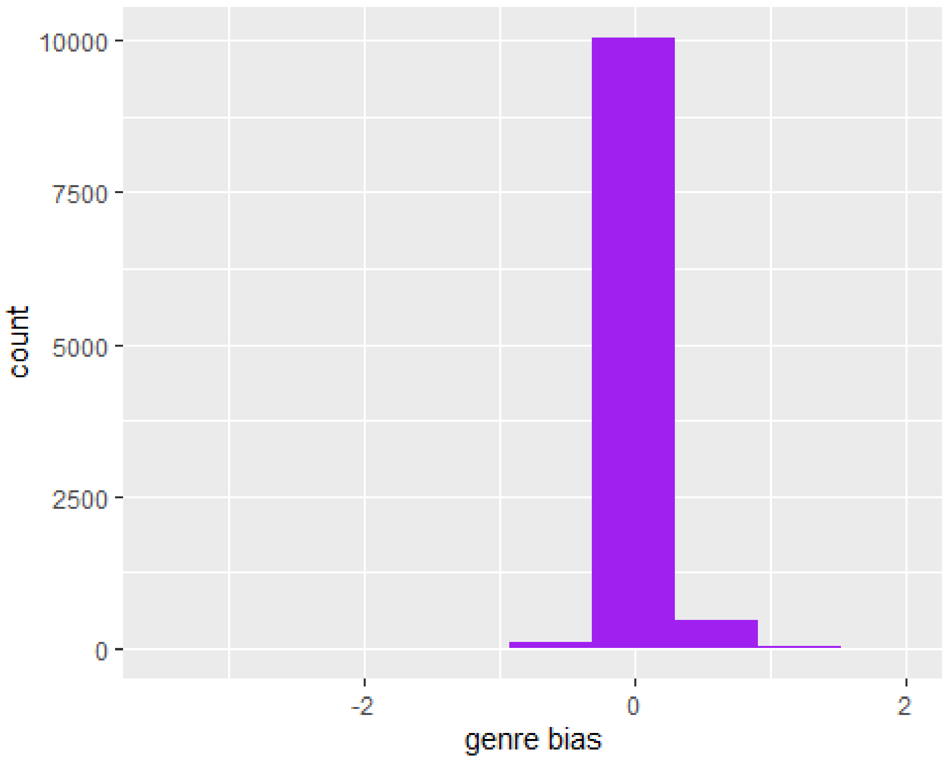
We see a definite improvement over model two.

Finally, for our fourth and final model, we use the genre effects to improve our results. First, we compute the genre bias which is the average prediction error for each genre.

```
#find the average prediction error for each genre
genre_bias <- ratings_table %>%
  left_join(movie_bias, by = "movieId") %>%
  left_join(user_bias, by = "userId") %>%
  group_by(movieId) %>%
  summarize(gbias = mean(rating - initial_prediction - mbias - ubias),
            .groups = "drop")
```

Let’s take a look at the distribution of the genre bias.

```
#show variability of the genre_bias
genre_bias %>% ggplot(aes(gbias)) +
  geom_histogram(bins = 10, fill = "purple") +
  xlim(-3.5, 2) +
  xlab("genre bias")
```



We see that the genre bias is centered at 0 and is approximately symmetrically distributed. However, it has a lot less variability than the movie bias or the user bias.

Next, we make our predictions using this model. Notice that there is some cleaning of the movie bias, the user bias, and the genre bias to replace any NAs with the average movie bias, the average user bias, and the average genre bias respectively.

```r
#determine predicted ratings for model four
predicted_ratings <- test_set %>%
  left_join(movie_bias, by = "movieId") %>%
  mutate(mbias = ifelse(is.na(mbias), mean(mbias, na.rm=TRUE), mbias)) %>%
  left_join(user_bias, by = "userId") %>%
  mutate(ubias = ifelse(is.na(ubias), mean(ubias, na.rm=TRUE), ubias)) %>%
  select(movieId, mbias, ubias) %>%
  left_join(genre_bias, by = "movieId") %>%
  mutate(gbias = ifelse(is.na(gbias), mean(gbias, na.rm = TRUE), gbias)) %>%
  summarize(pred = initial_prediction + mbias +ubias +gbias) %>%
  pull(pred)
```

Finally, we find the RMSE for this model and compare the results to the previous models.

```r
#calculate the RMSE for model four
rmse <- RMSE(predicted_ratings, test_set$rating)

#compare the results
results <- bind_rows(results,
                     tibble(method = "model_four", rmse = rmse))
results

## # A tibble: 4 x 2
##   method       rmse
##   <chr>       <dbl>
## 1 model_one   1.06
## 2 model_two   0.944
## 3 model_three 0.866
## 4 model_four  0.864
```

We see that there is a small amount of improvement over the previous models.

** Results **

We now use the validation dataset to evaluate the performance of the final model we just developed.

```
predicted_ratings <- validation %>%
  left_join(movie_bias, by = "movieId") %>%
  mutate(mbias = ifelse(is.na(mbias), mean(mbias, na.rm = TRUE), mbias)) %>%
  left_join(user_bias, by = "userId") %>%
  mutate(ubias = ifelse(is.na(ubias), mean(ubias, na.rm = TRUE), ubias)) %>%
  select(movieId, mbias, ubias) %>%
  left_join(genre_bias, by = "movieId") %>%
  mutate(gbias = ifelse(is.na(gbias), mean(gbias, na.rm = TRUE), gbias)) %>%
  summarize(pred = initial_prediction + mbias +ubias +gbias) %>%
  pull(pred)

#Use the RMSE to determine how close our predicted ratings are
#to the true values in the validation set
rmse <- RMSE(predicted_ratings, validation$rating)
rmse

## [1] 0.8650756
```

The resulting RMSE is consistent with the result we obtained when we were developing the final model. At that time, we saw that the model gave us a slight improvement in the prediction of the movie ratings over previous models. There remains, however, a fairly large error in our predictions.

** Conclusion **

Even though we saw only a small improvement in the ratings predictions made by the final model, we did see some improvement. The only variable in the edx dataset that we have not yet used is the "timestamp" variable. We know from experience that certain types of movies change in popularity over time. For example, there was a time when musicals and westerns were very popular, although they have not been as popular in recent years. So, finding a "time bias" may help to reduce the RMSE and improve the performance of our model. A basic limitation in our efforts tp improve the accuracy of our model is that the edx dataset has only six variables, namely movieId, userId, genres, title, rating, and timestamp. There may be other factors that we could take into consideration to improve the accuracy of our model.