Movie Lens Report

Tamia van Geloven

Introduction

Many streaming platforms, including Netflix, use a movie recommendation system. The recommendation systems are based on movie ratings given by viewers. This report builds a movie recommendation system using a data set that includes over 10,000 movies and almost 70,000 users who rated those movies. The ratings are out of 5, where 5 is the best and 0.5 is the worst. Based on those ratings a model is built to recommend movies. Before modeling , the data is imported, cleaned, and organized so that it is usable. The data is then split into a training set that is held in the object edx, and a final holdout test set that will only be used to evaluate the final model.

First, the edx data set is examined to determine a starting point to build a recommendation model. Then the edx data set is split into a train set and a test set. Then a function is created to calculate the residual mean squared error (RMSE). The goal is to create a model that results in an RMSE value that is less than 0.86490. The preliminary model is a random sample. The RMSE from the random sample is used as a baseline to be improved. After the random sample, the overall mean rating is used to create a model. In order to further reduce the RMSE, the movie effect and the user effect are added to the mean model. The movie effect uses the mean rating for each movie and the user effect uses the mean rating that each user gives. Finally, the mean plus movie effect plus user effect model is regularized to account for movies with few ratings and users that gave few ratings. After the regularized model is created, it is tested against the final holdout set.

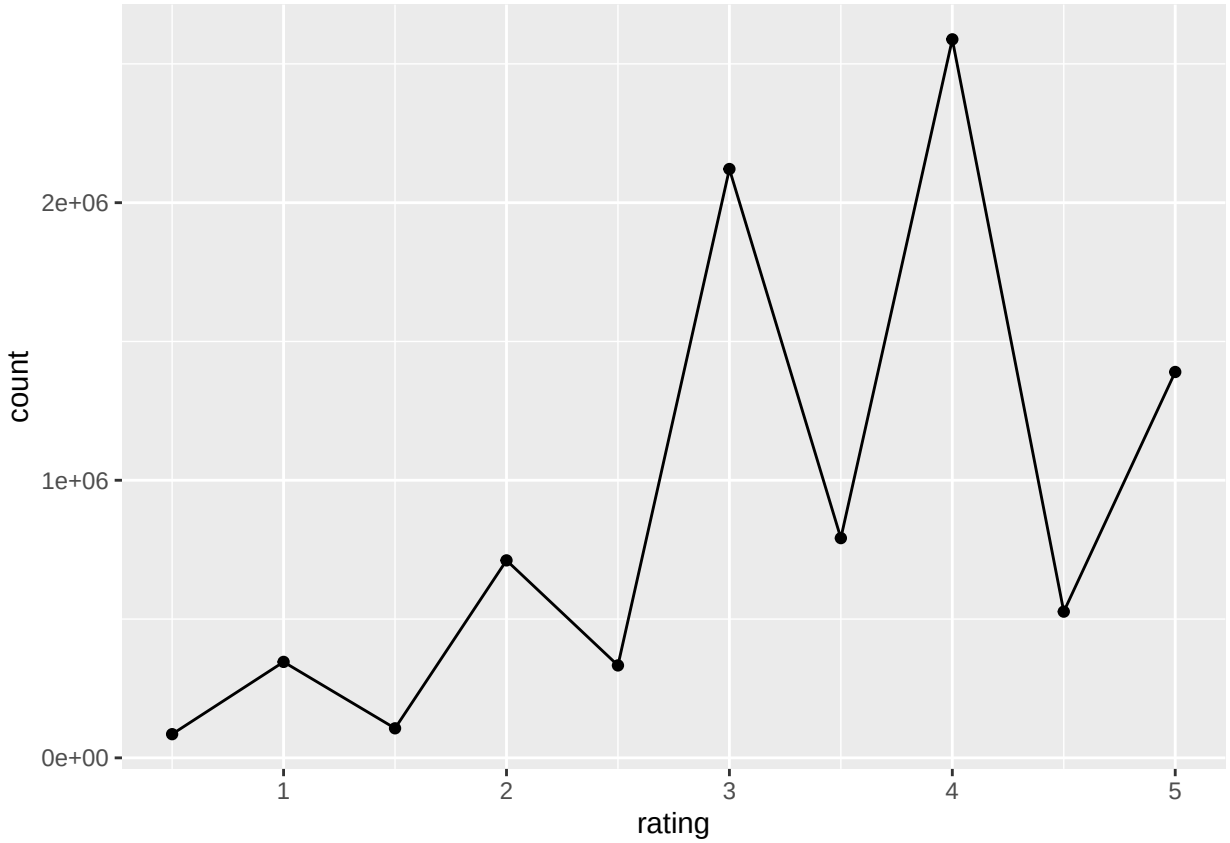
Methods and Analysis

Using the tidyverse package and the caret package in R, the movie lens data set is imported. The movie lens data set is then organized and column names are assigned. The movie lens data set is then split into the edx data set and the final holdout test. The final holdout test set is not used until the final model has been evaluated.

Before splitting the edx data set into a train set and a test set, it is examined to determine the best modeling approach. The edx data set has just over 9 million rows and 6 columns. Each row represents a single rating for a movie. The column names are userId, movieId, rating, timestamp, title, and genres. Each user has a unique user ID and each movie has a unique movie ID. All of these columns can be used to create a recommendation model. However, I decided to use only the user ID, the movie ID, the rating, and the title column, because this approach is the most simple.

The most common rating is 4 closely followed by 3. I used a graph of the ratings to visualize their distribution.

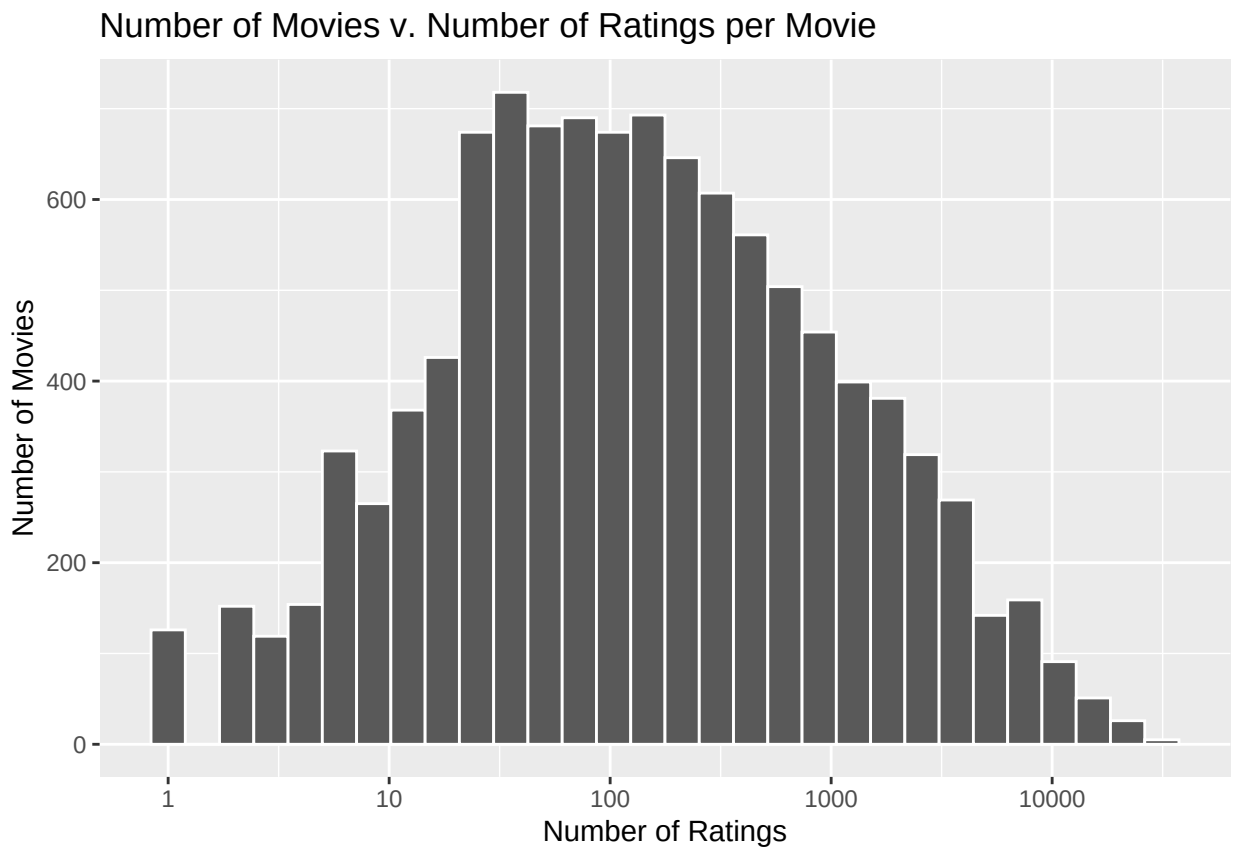
```
edx %>% group_by(rating) %>%
  summarize(count = n()) %>%
  ggplot(aes(rating, count)) +
  geom_line() +
  geom_point()
```



I used a histogram to visualize how many ratings each movie received.

```
edx %>% group_by(movieId) %>%
  summarize(count = n()) %>%
  ggplot(aes(count)) +
  geom_histogram(color = "white") +
  scale_x_log10() +
  ggtitle("Number of Movies v. Number of Ratings per Movie") +
  xlab("Number of Ratings") +
  ylab("Number of Movies")
```

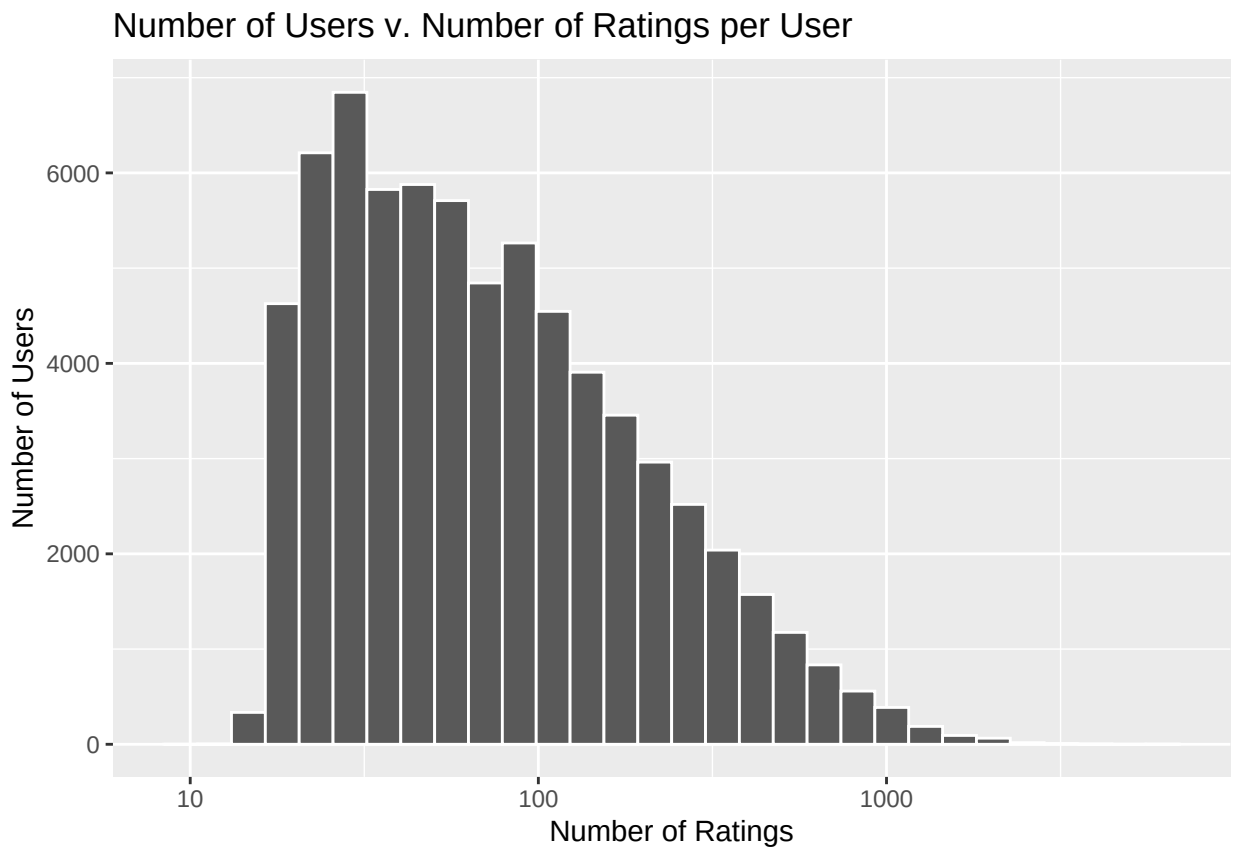
```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



I used another histogram to visualize the amount of times each user rated a movie.

```
edx %>% group_by(userId) %>%
  summarize(count = n()) %>%
  ggplot(aes(count)) +
  geom_histogram(color = "white") +
  scale_x_log10() +
  ggtitle("Number of Users v. Number of Ratings per User") +
  xlab("Number of Ratings") +
  ylab("Number of Users")
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



The movies with the most ratings are popular movies such as Pulp Fiction, Forrest Gump, and Jurassic Park. I used the following code to view the top 10 most rated movies.

```
edx %>% group_by(movieId) %>%
  summarize(num_ratings = n(), movieTitle = first(title)) %>%
  arrange(desc(num_ratings)) %>%
  top_n(10, num_ratings)
```

```
## # A tibble: 10 x 3
##    movieId num_ratings movieTitle
##      <int>       <int> <chr>
##  1     296       31362 Pulp Fiction (1994)
##  2     356       31079 Forrest Gump (1994)
##  3     593       30382 Silence of the Lambs, The (1991)
##  4     480       29360 Jurassic Park (1993)
##  5     318       28015 Shawshank Redemption, The (1994)
##  6     110       26212 Braveheart (1995)
##  7     457       25998 Fugitive, The (1993)
##  8     589       25984 Terminator 2: Judgment Day (1991)
##  9     260       25672 Star Wars: Episode IV - A New Hope (a.k.a. Star Wars) (1~
## 10     150       24284 Apollo 13 (1995)
```

The most highly rated movies are not popular movies. This is because niche films are reviewed by few people and so the sample size of ratings is too small to be reliable. I used the following code to view the best rated movies.

```
best_rating <- edx %>%
  group_by(movieId) %>%
  filter(mean(rating) == 5) %>%
  select(movieId, rating, title)
head(best_rating)
```

```
## # A tibble: 6 x 3
## # Groups:   movieId [5]
##   movieId rating title
##     <int>  <dbl> <chr>
## 1   53355      5 Sun Alley (Sonnenallee) (1999)
## 2   51209      5 Fighting Elegy (Kenka erejii) (1966)
## 3   33264      5 Satan's Tango (Sátántangó) (1994)
## 4   42783      5 Shadows of Forgotten Ancestors (1964)
## 5   33264      5 Satan's Tango (Sátántangó) (1994)
## 6    3226      5 Hellhounds on My Trail (1999)
```

Before modeling the edx data set is split into a train set and a test set. The test set uses a sample that is only 20 percent of the edx data set.

```
#Split the edx data set into a train set and a test set
indexes <- split(1:nrow(edx), edx$userId)
test_ind <- sapply(indexes, function(ind) sample(ind, ceiling(length(ind)*.2))) %>%
  unlist(use.names = TRUE) %>%
  sort()
test_set <- edx[test_ind,]
train_set <- edx[-test_ind,]

#Make sure that the same movies are in the test set and the train set
test_set <- test_set %>%
  semi_join(train_set, by = "movieId")
train_set <- train_set %>%
  semi_join(test_set, by = "movieId")
```

Then the function is created to calculate the residual mean squared error (RMSE). The residual mean squared error represents the average error of a given model. The smaller the average error, the better the recommendation system works. When the RMSE function is applied to a model is produces the average error of that model. The goal is to create a model that shrinks that error to 0.86490.

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

To create a baseline RMSE to improved upon I created a random sample using a Monte Carlo simulation and the frequency with which each rating occurs. The random sample produced an RMSE of 1.500844. To improve on this RMSE I started with an overall average model.

```
p <- function(x, y) mean(y == x)
rating <- seq(0.5, 5, 0.5)

B <- 10000
M <- replicate(B, {
```

```
  s <- sample(train_set$rating, 100, replace = TRUE)
  sapply(rating, p, y = s)
})

prob <- sapply(1:nrow(M), function(x) mean(M[x,]))

y_hat_random <- sample(rating, size = nrow(test_set),
                       replace = TRUE, prob = prob)
result_random <- RMSE(test_set$rating, y_hat_random)
result_random
```

```
## [1] 1.500411
```

While examining the edx data set I noticed that most of the ratings are between 3 and 4. The overall average rating among all the users and movies is 3.512205. So I used the overall average rating to create a model. The average model predicts that every movie will receive the overall average rating. The object mu holds the overall average rating. The RMSE from the overall mean model is 1.060296.

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

```
## [1] 3.512205
```

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

```
## [1] 1.060296
```

To improve on this model the movie effect is added. The movie effect uses the average rating that each individual movie received. The movie effect is calculated by subtracting the overall mean from the mean rating for each movie and then taking the average of that equation. The object bi holds the movie effect. The object y_hat_bi holds the original mean model plus the movie effects tested against the test_set. The RMSE from the mean + movie effects model is 0.9441517.

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

y_hat_bi <- mu + test_set %>%
  left_join(bi, by = "movieId") %>%
  .$b_i

result_mu_bi <- RMSE(test_set$rating, y_hat_bi)
result_mu_bi
```

```
## [1] 0.9441517
```

Because the RMSE is not yet at the target 0.86490, the model is improved by including the user effect. The user effect takes the average rating that each user gives into account. The user effect is calculated by taking the mean of the average rating that each user gave, minus the overall average, minus the movie effect. This is then added to the model. The object bu holds the user effect. The object y_hat_bi_bu holds the mean + the movie effect + the user effect model. The RMSE of the mean + movie effect +user effect model is 0.8664269.

```
bu <- train_set %>%
  left_join(bi, by = "movieId") %>%
  group_by(userId) %>%
  summarize(b_u = mean(rating - mu - b_i))

y_hat_bi_bu <- test_set %>%
  left_join(bi, by = "movieId") %>%
  left_join(bu, by = "userId") %>%
  mutate(pred = mu + b_i + b_u) %>%
  .$pred

result_mu_bi_bu <- RMSE(test_set$rating, y_hat_bi_bu)
result_mu_bi_bu
```

```
## [1] 0.8664269
```

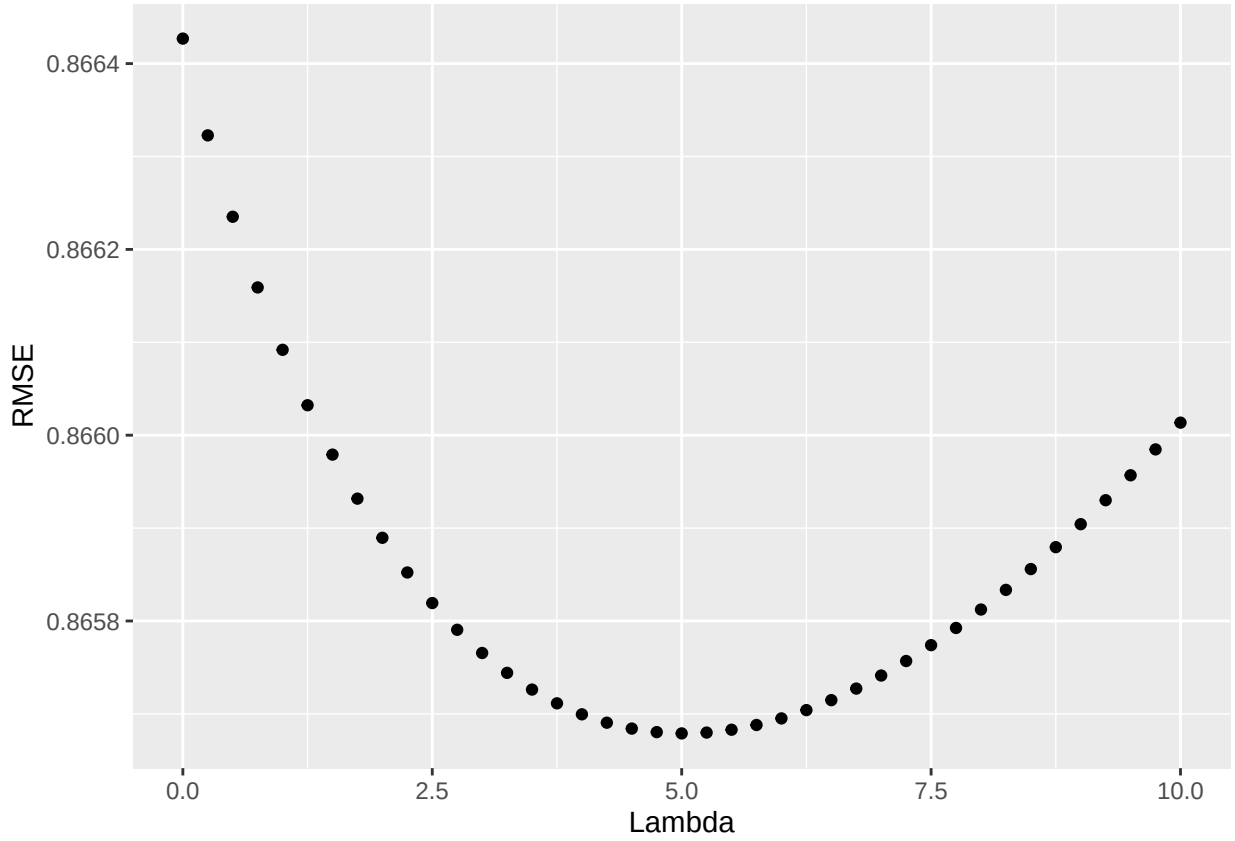
The RMSE is still not at the desired 0.86490 so the whole model is regularized. While examining the data I noticed that the most rated movies are popular, well known movies, while the movie with the best ratings are less known, niche movies. This occurs because some users rate few movies and some movies received few ratings. To take this into account I regularized the mean + movie effect + user effect model. To regularize the model a function is created that divides the movie effect and the user effect by the number of ratings per movie or user respectively plus a variable lambda.

```
#Create a function that uses possible lambdas to regularize the data set
regularized <- function(lambda, trainset, testset){
  mu <- mean(trainset$rating)
  b_i <- trainset %>%
    group_by(movieId) %>%
    summarize(b_i = sum(rating - mu)/(n() + lambda))
  b_u <- trainset %>%
    left_join(b_i, by = "movieId") %>%
    filter(!is.na(b_i)) %>%
    group_by(userId) %>%
    summarize(b_u = sum(rating - b_i - mu)/(n() + lambda))
  predicted_ratings <- testset %>%
    left_join(b_i, by = "movieId") %>%
    left_join(b_u, by = "userId") %>%
    filter(!is.na(b_i), !is.na(b_u)) %>%
    mutate(pred = mu + b_i + b_u) %>%
    pull(pred)
  return(RMSE(predicted_ratings, testset$rating))
}

#Create a set of lambdas to try in the function
lambdas <- seq(0, 10, 0.25)

#Test the lambdas in the function
rmses <- sapply(lambdas,
                regularized,
                trainset = train_set,
                testset = test_set)
```

The more ratings a movie has or the more movies a user rates, the more a regularization will not effect the model. Regularization gives less weight to movies with few ratings and the users that rated few movies. One

specific value of lambda will result in the lowest RMSE. The function tries out values of lambda that range from 0 to 10 at intervals of 0.25. As shown in the graph, the most effective lambda is 5.

```
#Create a graph of lambdas against the RMSE that they produce
tibble(Lambda = lambdas, RMSE = rmses) %>%
  ggplot(aes(x = Lambda, y = RMSE)) +
  geom_point()
```




```
#Determine which lambda creates the minimum RMSE
lambda <- lambdas[which.min(rmses)]
lambda
```

```
## [1] 5
```

The lambda that reduces the RMSE the most is 5. Using lambda, the previous mean + movie effect + user effect model is regularized. The movie effect is divided by the sum of ratings for each movie plus the lambda. The user effect is divided by the sum of ratings that each user gave plus the lambda. Using the regularized movie effect and user effect a model is created and stored in y_hat_reg. It is then tested using the RMSE function. The resulting RMSE is 0.865679.

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

reg_bi <- train_set %>%
  group_by(movieId) %>%
  summarize(b_i = sum(rating - mu)/(n() + lambda))
```

```
reg_bu <- train_set %>%
  left_join(reg_bi, by = "movieId") %>%
  group_by(userId) %>%
  summarize(b_u = sum(rating - b_i - mu)/(n() + lambda))

y_hat_reg <- test_set %>%
  left_join(reg_bi, by = "movieId") %>%
  left_join(reg_bu, by = "userId") %>%
  mutate(pred = mu + b_i + b_u) %>%
  pull(pred)

result_reg <- RMSE(test_set$rating, y_hat_reg)
result_reg
```

```
## [1] 0.865679
```

Results

Finally the regularized model is tested against the final holdout test. To do this, the mean, the movie effect, and the user effect are recalculated using the edx data set instead of the train set, before being tested against the final holdout test. The final RMSE is 0.8648178 which, when rounded is slightly less than the desired 0.86490.

```
mu_final <- mean(edx$rating)

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

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

final_y_hat_reg <- final_holdout_test %>%
  left_join(bi_final, by = "movieId") %>%
  left_join(bu_final, by = "userId") %>%
  mutate(pred = mu + b_i + b_u) %>%
  pull(pred)

final_result <- RMSE(final_holdout_test$rating, final_y_hat_reg)
final_result
```

```
## [1] 0.8648178
```

Conclusion

After examining the data, producing a random sample, and using the average model, the movie effects model, and the user effects model, it is clear that the overall average rating, the movie effect, and the user effect must be used together in order to create a viable model. However, this model does not result in a low enough RMSE. I used regularization to improve this model. Regularization accounts for the movies with few ratings and the users who rated few movies. There are more possible methods that could lower the RMSE further. This model could be improved using genres and time stamps. Movies that came out around the 1990's have

the most ratings because they have been out for long enough to be rated a lot and they came out after movies were consistently rated. Genres also have distinct rating patterns. The genre and the time stamp could be used to improve the model in the future. Because I reached the desired RMSE without using these methods I stopped at a regularized model that uses the movie effect and the user effect and the average overall rating.