MovieLens RMSE Analysis

Dennis Holeski

2020-11-12

INTRODUCTION

This analysis attempts to create a movie recommendation system using public movie and rating data from GroupLens. This specific dataset contains 10 million observations consisting of 100,000 tag applications applied to 10,000 movies by 72,000 users. An ideal algorithm will produce a low RMSE relative to the ratings system ranked 1 through 5. Our goal is to be below **0.86500**.

The data will be downloaded and train/test sets will be created as per course instructions. This was copied and pasted from the assignment, ensuring accuracy of datasets. This would be the initial step for pre-processing and cleaning the data.

METHODS/ANALYSIS The approach in this analysis started with the mean rating only, and increasingly added other features to improve results. The features selected were the mean ratings of users, movies, release years, and timestamps. Regularization was implemented with various ranges of lambda. Data distribution was visualized to ensure model integrity.

Linear models were also attempted, on both of the full datasets and within specified quantiles of users' number of ratings. For demonstration, a lower quantile of 30 was decided on. Not shown here, other values of quantiles were attempted with unimproved results. Including the timestamp feature was also attempted in the linear models, however was removed when it failed to improve the model. This was mainly done to provide a clear analysis and to avoid causing any performance issues when the script is executed.

Start analysis, beginning with the mean rating only. Then create the RMSE comparison table that will be updated after each RMSE is predicted predicted.

```
#average rating of edx ratings
mu <- mean(edx$rating)

#RMSE of average only
avgonly <- RMSE(validation$rating,mu)

#create RMSE table
RMSEs <- data.frame(model = "avg_only", RMSE = avgonly)
```

Prediction based on movie-only grouping.

```
#movie average rating
movie_avg <- edx %>%
  group_by(movieId) %>%
  summarize(bi = mean(rating - mu),n_mov_rat = n())


#predict movie effect
pred_moveffect <- mu + validation %>%
  left_join(movie_avg, by = 'movieId') %>%
  .$bi
```

```
#movie only RMSE
mov_only_model <- RMSE(pred_moveffect, validation$rating)

#add movie only RMSE to RMSE table
RMSEs <- bind_rows(RMSEs, data.frame(model = "movie_effect", RMSE = mov_only_model))
```

Extract movie year release for future analysis.

```
#extract movie year release
movie_yr <- edx %>%
  group_by(movieId) %>%
  summarize(year = as.factor(str_sub(first(title),-5,-2)))
```

Movie and user prediction

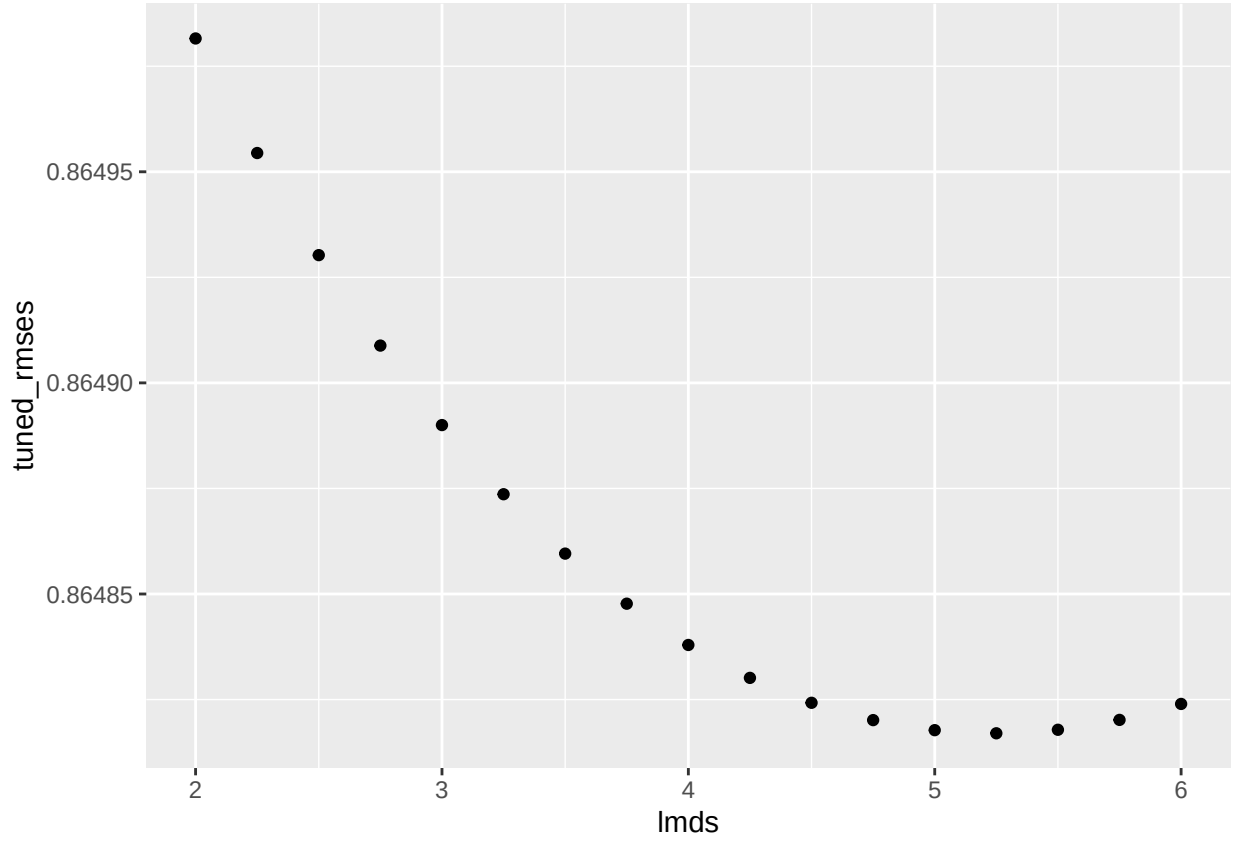
```
#user average
usr_avg <- edx %>%
  left_join(movie_avg, by = 'movieId') %>%
  group_by(userId) %>%
  summarize(bu = mean(rating - mu - bi),n_usr_rat = n())


#movie and user prediction
pred_usrmov <- validation %>%
  left_join(movie_avg, by = 'movieId') %>%
  left_join(usr_avg, by = 'userId') %>%
  mutate(predicted = mu + bi + bu) %>%
    .$predicted

#movie & user RMSE
mov_usr_model <- RMSE(pred_usrmov, validation$rating)

#add movie & user to RMSE table
RMSEs <- bind_rows(RMSEs,data.frame(model = "movie+usr", RMSE = mov_usr_model ))
```

Perform analysis on movie + user with regularization. Started with a larger range of lambdas 'seq(0,12, .5)', however I was able to improve the RMSE by narrowing my regularization parameter.

```
#movie and usr average with tuning parameter, "lambda".
#started with larger lambas and narrowed it down to improve RMSE.
#lmds <- seq(0,12, .5)
lmds <- seq(2,6, .25)

tuned_rmses <- sapply(lmds, function(l) {

  bi_l <-
    edx %>%
    group_by(movieId) %>%
    summarize(biL = sum(rating - mu)/(n()+l))

  bu_l <- edx %>%
  left_join(bi_l, by = 'movieId') %>%
  group_by(userId) %>%
  summarize(buL = sum(rating - mu - biL)/(n()+l))
```

```
preds_usrmov <- validation %>%
  left_join(bi_l, by = 'movieId') %>%
  left_join(bu_l, by = 'userId') %>%
  mutate(predicted = mu + biL + buL) %>%
  .$predicted


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

Plot RMSE vs Lambdas



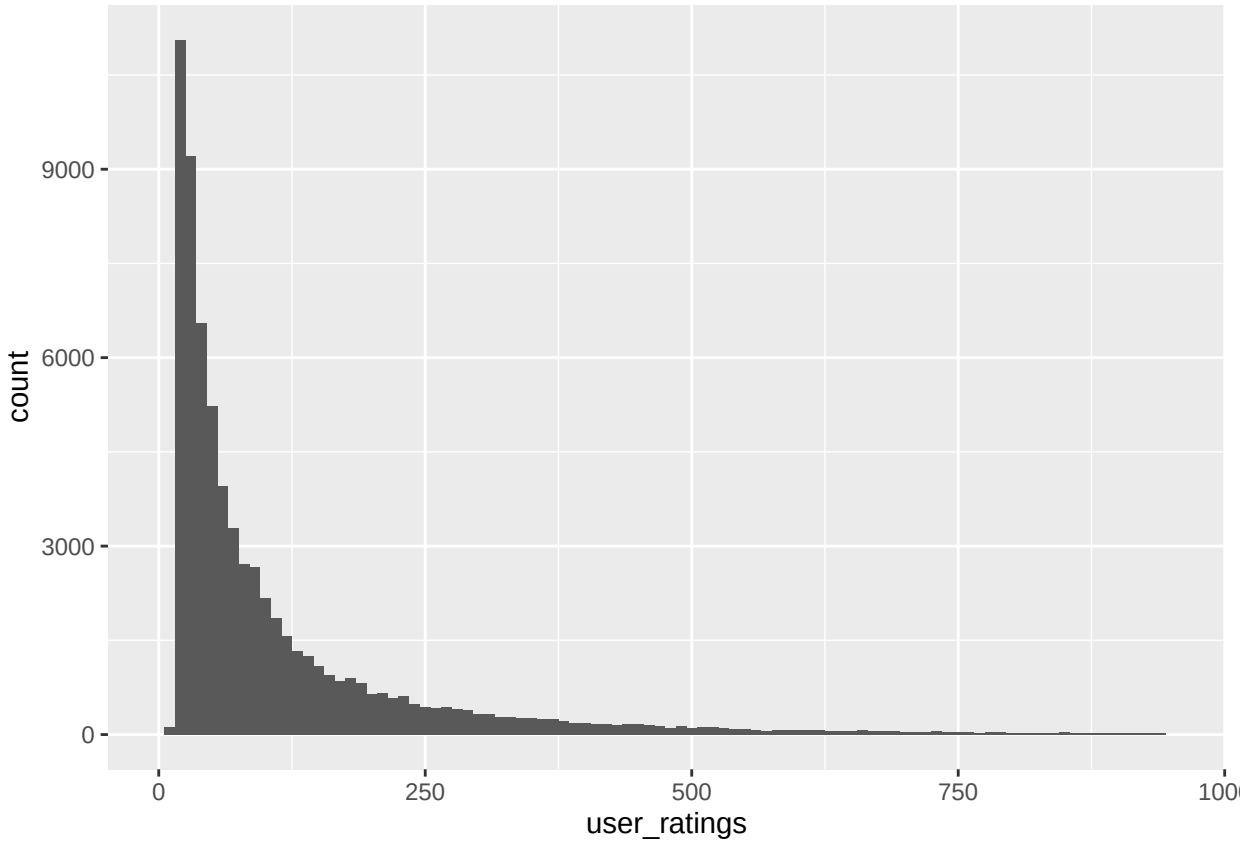
Choose the lamba with the lowest RMSE and add that model to the comparison table.

```
#tuned rmse model with lowest lamda
usr_mov_lambda_model <- tuned_rmses[which.min(lmds)]

#add lamba model to RMSE comparison table
RMSEs <- bind_rows(RMSEs,data_frame(model = "movie+usr+lambda", RMSE = usr_mov_lambda_model))
```

Analyze the distribution of the user rating counts to identify possible anomalies that could affect the model

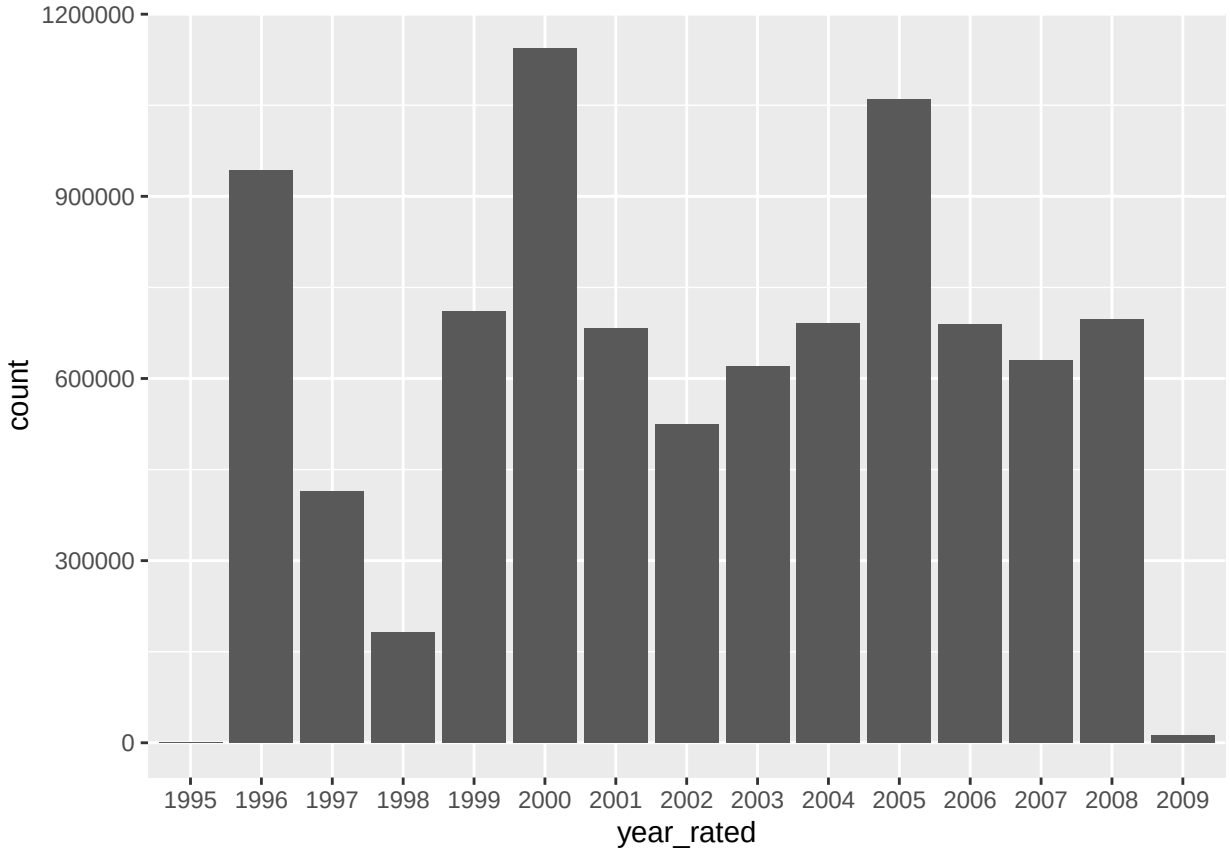
Table 1: number of ratings vs. quantile

	x
0%	10
25%	32
50%	62
75%	141
100%	6616



Verify that the distribution for records per year rated is reasonable, attempting to identify any anomalies to ensure model accuracy.

```
#visualize the distribution of number of records per year the movie was rated
edx %>% mutate(year_rated = format(as.POSIXct(.$timestamp,origin = "1970-01-01",tz = "UTC"),"%Y")) %>% g
```



Further analysis on the movie release year.

```
#rating aggregated by movie year
movie_yr_avg <- edx %>%
  left_join(movie_avg, by = 'movieId') %>%
  left_join(usr_avg, by = 'userId') %>%
  left_join(movie_yr, by = 'movieId') %>%
  group_by(year) %>%
  summarize(bt = mean(rating - mu - bi - bu))


#movie, year, and user prediction
pred_yrmovusr <- validation %>%
  mutate(year = as.factor(str_sub(first(title), -5, -2))) %>%
  left_join(movie_avg, by = 'movieId') %>%
  left_join(usr_avg, by = 'userId') %>%
  left_join(movie_yr_avg, by = 'year') %>%
  mutate(preds = mu + bi + bu + bt) %>%
  .$preds


#movie, user, and year RMSE
```

```r
mov_usr_yr_model <- RMSE(pred_yrmovusr, validation$rating)

#update RMSE table
RMSEs <- bind_rows(RMSEs,data_frame(model = "movie+usr+year", RMSE = mov_usr_yr_model ))
```

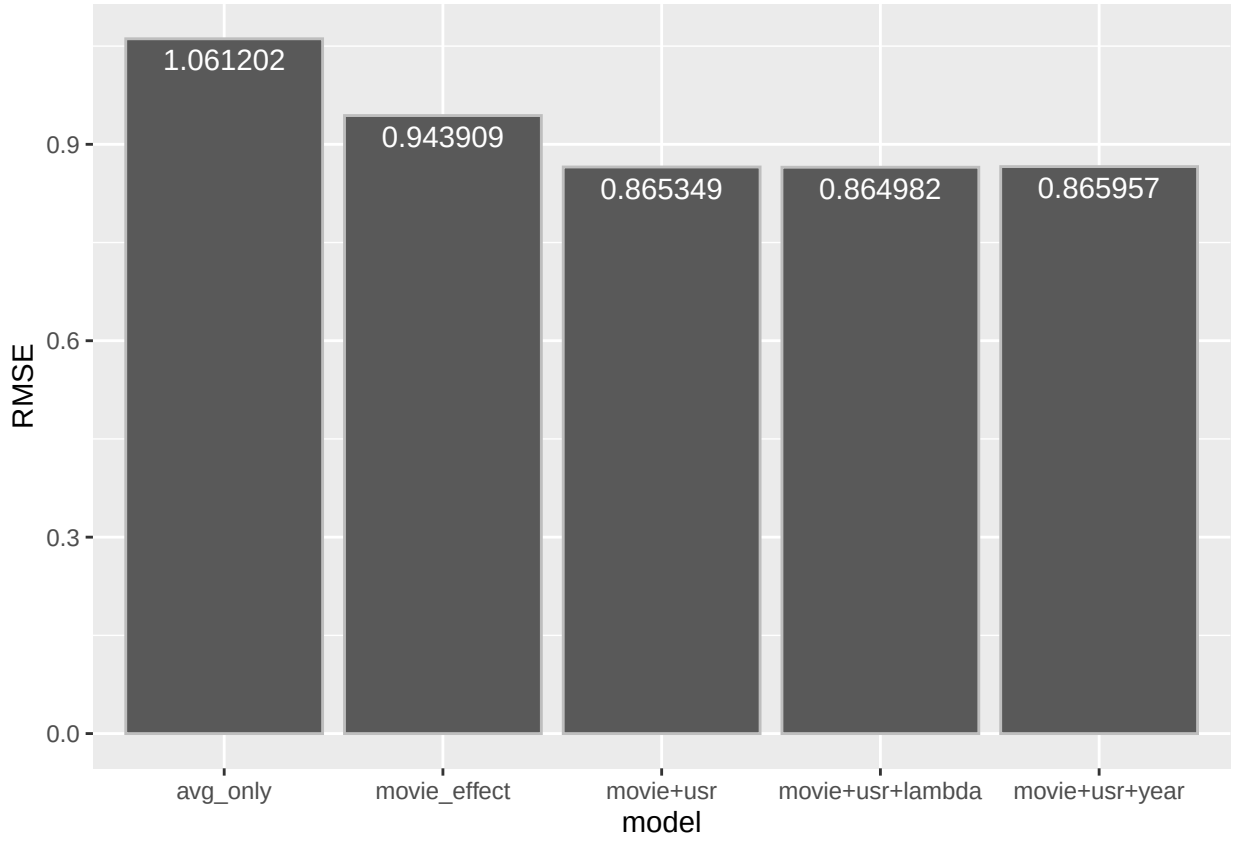
```r
#group datasets by user ratings to eliminate values less than the 30th quantile.
edx_usr_30qnt <- edx %>% group_by(userId) %>% mutate(num_user_ratings = n()) %>% ungroup() %>% filter(nu

vali_usr_30qnt <- validation %>% group_by(userId) %>% mutate(num_user_ratings = n()) %>% ungroup() %>% f

###Linear model using filtered data to only include users who rated more movies than the lower 30 quant
fit_lm30 <- lm(rating~movieId+userId,data=edx_usr_30qnt)

prd_lm30 <-  predict(fit_lm30, vali_usr_30qnt)

usr_30qnt <- RMSE(pred = prd_lm30, obs = vali_usr_30qnt$rating)

lm_RMSEs <- data.frame(model = "Linear Model_User Ratings > 30qnt", RMSE = usr_30qnt )


#regular linear model using movieId and userId.
fit_lm <- lm(rating~movieId+userId,data=edx)

prd_lm <-  predict(fit_lm, validation)

lm_rmse <- RMSE(pred = prd_lm, obs = validation$rating)

lm_RMSEs <- bind_rows(lm_RMSEs,data.frame(model = "Linear Model", RMSE = lm_rmse ))
```

RESULTS

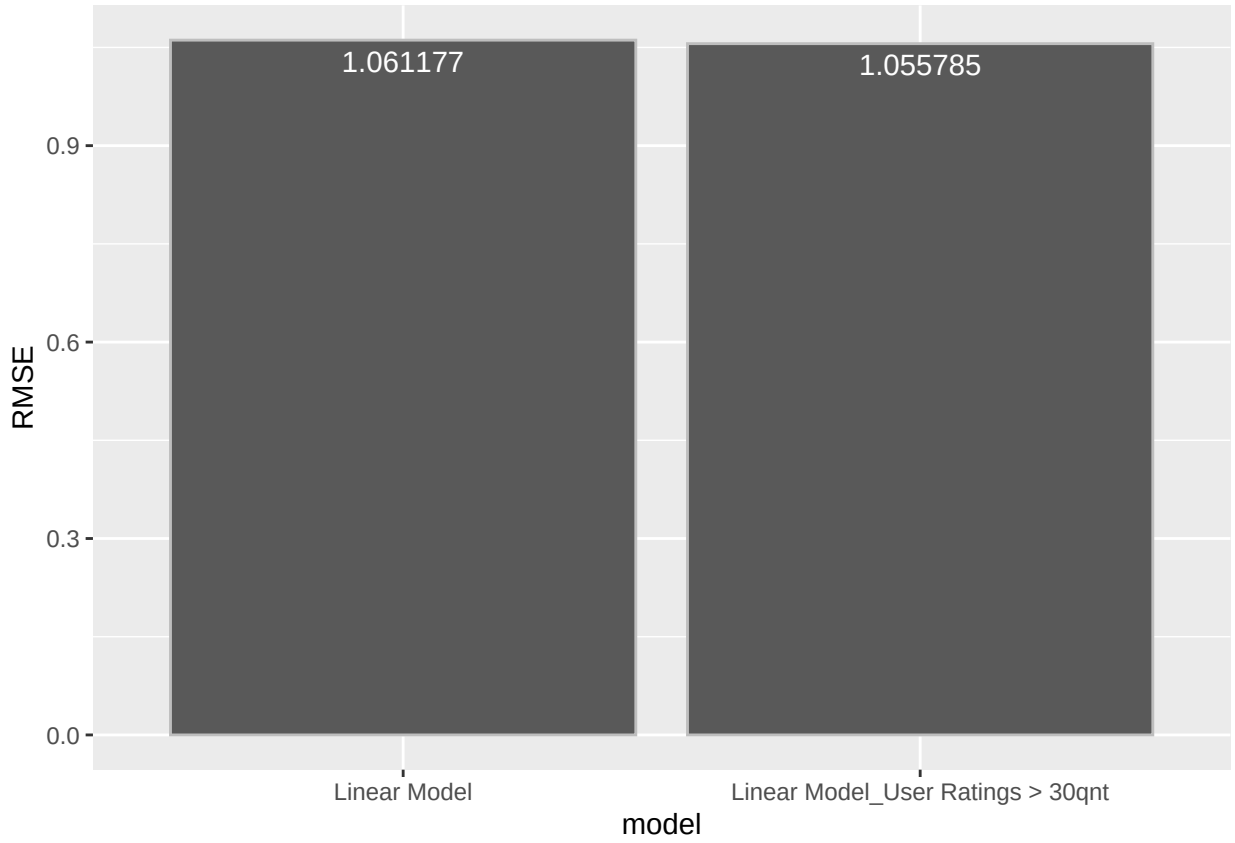
The table below outlines the analyzed models and their respective RMSE results. The graph shows a visual representation of how the RMSE was changed with the different models. Improvements were seen as features were added, however improvements quickly tapered off as the numbers of features increased. The best model included the regularization of user and movie features with the lowest end resulting RMSE of **0.864982**. This is an accepted result and should perform well as a basic use algorithm.

The linear models' results are also displayed below in the lower table. These methods did not produce significant results. This held true even with the addition of several features and also removing lower quantiles of the number of ratings for the respective users.

model	RMSE
avg_only	1.0612018
movie_effect	0.9439087
movie+usr	0.8653488
movie+usr+lambda	0.8649816
movie+usr+year	0.8659570



model	RMSE
Linear Model_User Ratings > 30qnt	1.055786
Linear Model	1.061177



CONCLUSION

This completed RMSE table shows that the best RMSE is from the movie + user with regularization model. Interestingly enough, adding the year that the movie was released as a predictor showed that the RMSE was minimally affected, actually causing a slight increase. This was further proof of what I had learned in the course that adding additional predictors does not mean you will gain improvements on your model. While the best model produced an acceptable result, I believe there is room for improvement. Further development would most likely include more complex algorithms and ensembles. I would also include a breakdown and analysis of the individual genres.