Capstone Project - Movielens

Michael Murray

May 1, 2019

Executive Summary

This project uses a User-Based Collaborative Filtering method to make predictions about how users will rate certain movies. The model is based on the Movielens 10M data-set and residual mean square error (RMSE) is being used for performance valuation of the model.

Methods/Analysis

The MovieLens 10M dataset was used for this analysis to develop a movie reccomendation model. The raw dataset was downloaded from grouplens.org and the data was processed using Data Wrangling techniques to include specific ratings and movie data. The ratings and movie data was then joined to form the movielens dataset.

Download MovieLens 10M dataset and format for processing

Note: this process could take a couple of minutes

MovieLens 10M dataset:

https://grouplens.org/datasets/movielens/10m/

http://files.grouplens.org/datasets/movielens/ml-10m.zip

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

```
## Loading required package: tidyverse

## Warning: package 'tidyverse' was built under R version 3.5.3

## -- Attaching packages ---------------------------------------------------------------------------------

## v ggplot2 3.1.0     v purrr   0.2.5
## v tibble  1.4.2     v dplyr   0.7.8
## v tidyr   0.8.2     v stringr 1.3.1
## v readr   1.3.0     v forcats 0.3.0

## Warning: package 'ggplot2' was built under R version 3.5.2

## Warning: package 'tibble' was built under R version 3.5.1

## Warning: package 'tidyr' was built under R version 3.5.2

## Warning: package 'readr' was built under R version 3.5.1

## Warning: package 'purrr' was built under R version 3.5.1

## Warning: package 'dplyr' was built under R version 3.5.2

## Warning: package 'stringr' was built under R version 3.5.1

## Warning: package 'forcats' was built under R version 3.5.1

## -- 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

## Warning: package 'caret' was built under R version 3.5.1

## Loading required package: lattice

##
## Attaching package: 'caret'

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

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

ratings <- read.table(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")

movies <- as.data.frame(movies) %>% mutate(movieId = as.numeric(levels(movieId))[movieId],
                                           title = as.character(title),
                                           genres = as.character(genres))
```

```
## Warning: package 'bindrcpp' was built under R version 3.5.1
```

```
movielens <- left_join(ratings, movies, by = "movieId")
```

Creating the test and train data sets to perform the analysis. The following code creates edx(train__set) and validation(test__set) ensuring both datasets are consistent and can be used for a valid analysis.

```
set.seed(1)
test_index <- createDataPartition(y = movielens$rating, times = 1, p = 0.1, list = FALSE)
edx <- movielens[-test_index,]
temp <- movielens[test_index,]

validation <- temp %>%
     semi_join(edx, by = "movieId") %>%
     semi_join(edx, by = "userId")

removed <- anti_join(temp, validation)
```

```
## Joining, by = c("userId", "movieId", "rating", "timestamp", "title", "genres")
```

```
edx <- rbind(edx, removed)
```

Function for residual mean squared error(RMSE), in this analysis RMSE will represent our error when making a movie prediction.

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

Results

Initial simple prediction model used to establish a baseline. The code below generates an RMSE of 1.061 using the mean rating of the edx dataset of 3.51. Code used to verify using mu_hat input greater than or less that derived mu_hat for the dataset generates a larger RMSE.

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

```
## [1] 3.512465
```

```
naive_rmse <- RMSE(validation$rating, mu_hat)
naive_rmse
```

```
## [1] 1.061202
```

```
predictions <- rep(2.5, nrow(validation))
RMSE(validation$rating, predictions)
```

```
## [1] 1.46641
```

```
predictions <- rep(4.5, nrow(validation))
RMSE(validation$rating, predictions)
```

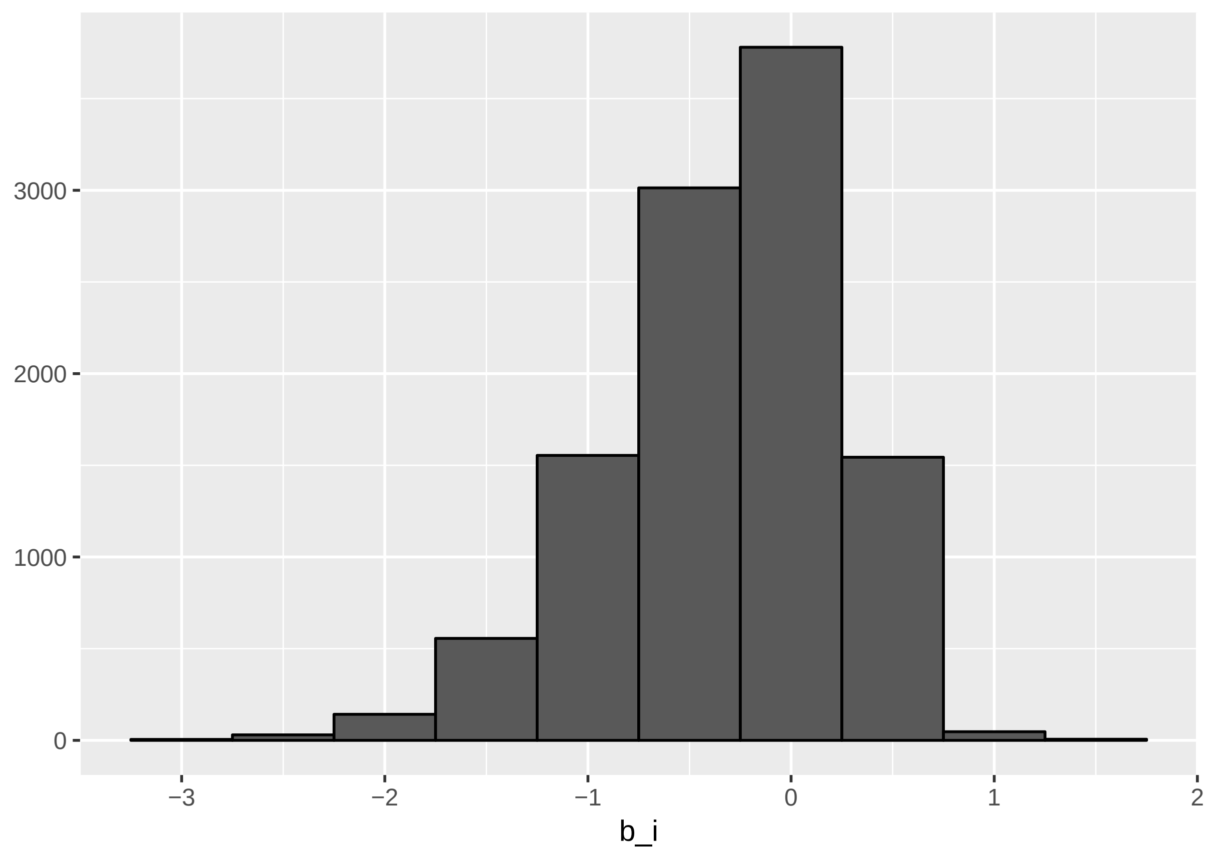
```
## [1] 1.449906
```

```
rmse_results <- data_frame(method = "Just the average", RMSE = naive_rmse,
                    Accuracy = mean(predictions==validation$rating))
rmse_results %>% knitr::kable()
```

method	RMSE	Accuracy
Just the average	1.061202	0.0582861

Enhancing the model to determine movie rating effects due to bias attributed to the movie itself. Graph shows rating estimates can very alot grouping by movie id. Adding the movie effects bias to the model lowers the RMSE to 0.9439. The model below includes added movie effects.

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

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

movie_avgs %>% qplot(b_i, geom ="histogram", bins = 10, data = ., color = I("black"))
```

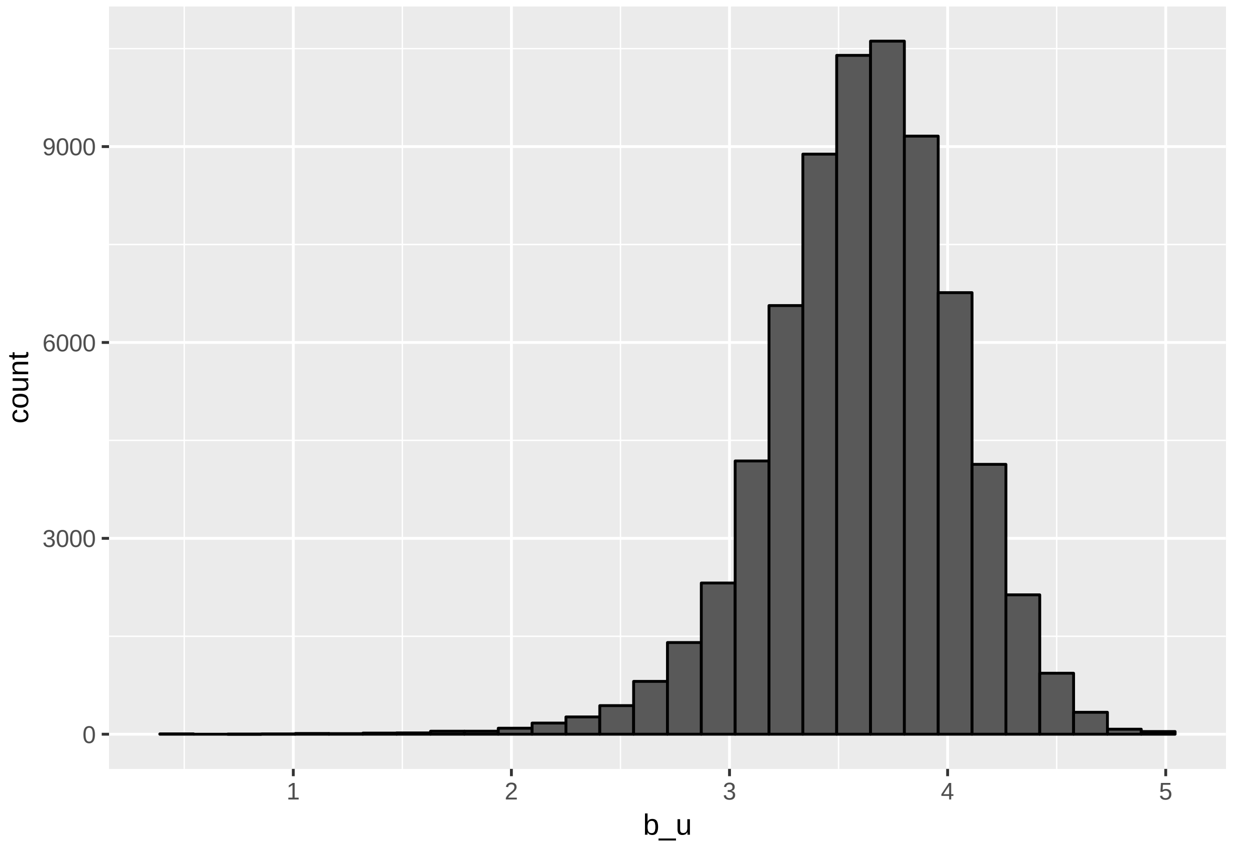


```
predicted_ratings <- mu + validation %>%
    left_join(movie_avgs, by='movieId') %>%
    .$b_i

model1_rmse <- RMSE(predicted_ratings, validation$rating)
rmse_results <- bind_rows(rmse_results,
                          data_frame(method="Movie Effect Model",
                                     RMSE = model1_rmse,
                    Accuracy = mean(round(predicted_ratings/0.5)*0.5==validation$rating)))
rmse_results %>% knitr::kable()
```

method	RMSE	Accuracy
Just the average	1.0612018	0.0582861
Movie Effect Model	0.9439087	0.2257952

Enhancing the model to determine movie rating effects due to user bias. Graphing rating estimates grouped by user id also shows a great degree of variability. Adding the user effects bias to the model reduces the RMSE to 0.8653. The model below includes movie effects and user effects.

```
edx %>%
  group_by(userId) %>%
  summarize(b_u = mean(rating)) %>%
  filter(n()>=100) %>%
  ggplot(aes(b_u)) +
  geom_histogram(bins = 30, color = "black")
```



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

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

model2_rmse <- RMSE(predicted_ratings, validation$rating)
rmse_results <- bind_rows(rmse_results,
                          data_frame(method="Movie + User Effects Model",
                                     RMSE = model2_rmse,
                    Accuracy = mean(round(predicted_ratings/0.5)*0.5==validation$rating)))
rmse_results %>% knitr::kable()
```

method	RMSE	Accuracy
Just the average	1.0612018	0.0582861
Movie Effect Model	0.9439087	0.2257952
Movie + User Effects Model	0.8653488	0.2479812

Considering movie effects and user effects in our model has achieved the required RMSE for this analysis, but let's try using regularization to see if we can improve the model further.

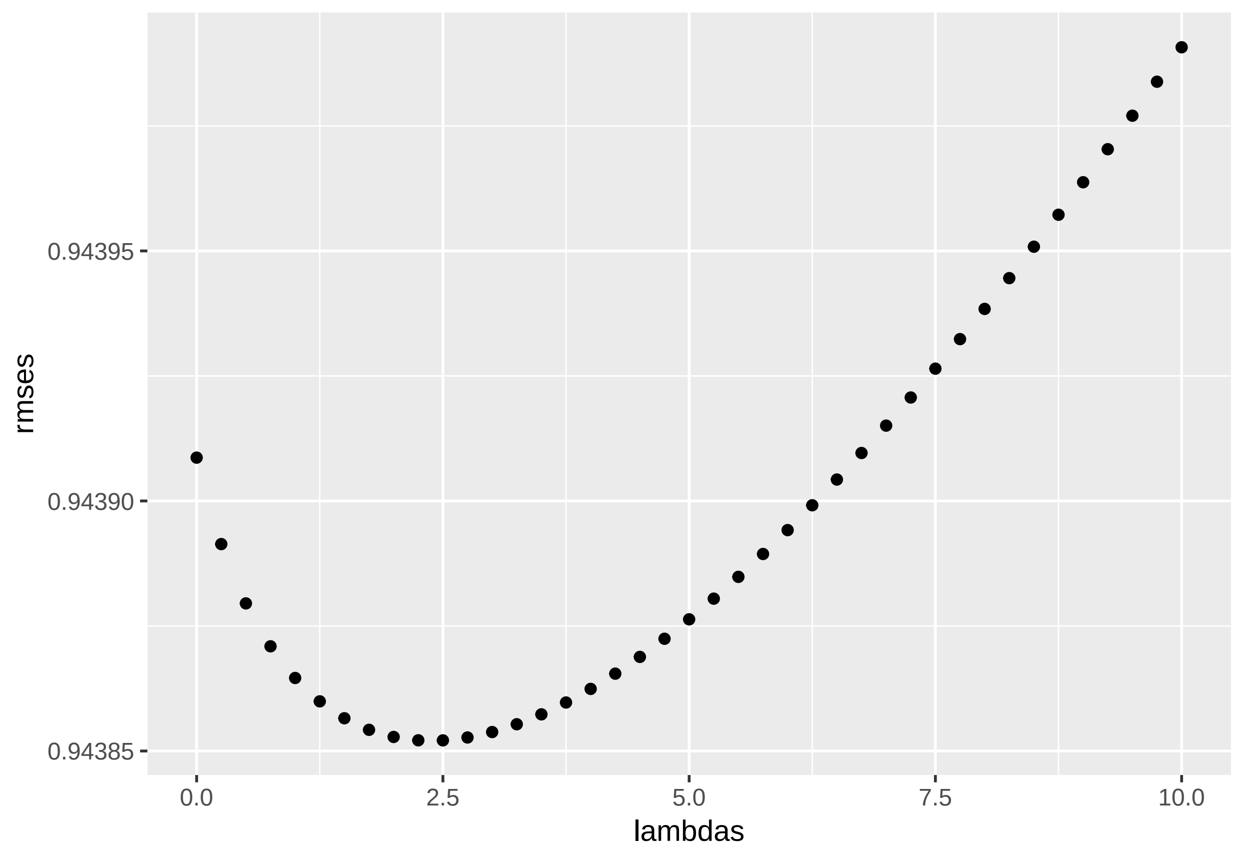
Implement regularization for estimating movie effects. Regularization logic added to model grouped by movie id.

```r
lambda <- 3

mu <- mean(edx$rating)

movie_reg_avgs <- edx %>%
  group_by(movieId) %>%
  dplyr::summarize(b_i = sum(rating - mu)/(n()+lambda), n_i = n())

predicted_ratings <- validation %>%
  left_join(movie_reg_avgs, by='movieId') %>%
  mutate(pred = mu + b_i) %>%
  .$pred

model3_rmse <- RMSE(predicted_ratings, validation$rating)
rmse_results <- bind_rows(rmse_results,
                          data_frame(method="Regularized Movie Effect Model",
                                     RMSE = model3_rmse,
                    Accuracy = mean(round(predicted_ratings/0.5)*0.5==validation$rating)))
rmse_results %>% knitr::kable()
```

method	RMSE	Accuracy
Just the average	1.0612018	0.0582861
Movie Effect Model	0.9439087	0.2257952
Movie + User Effects Model	0.8653488	0.2479812
Regularized Movie Effect Model	0.9438538	0.2259532

Implement regularization for estimating movie effects and user effects. Regularization code added to model grouped by user id.

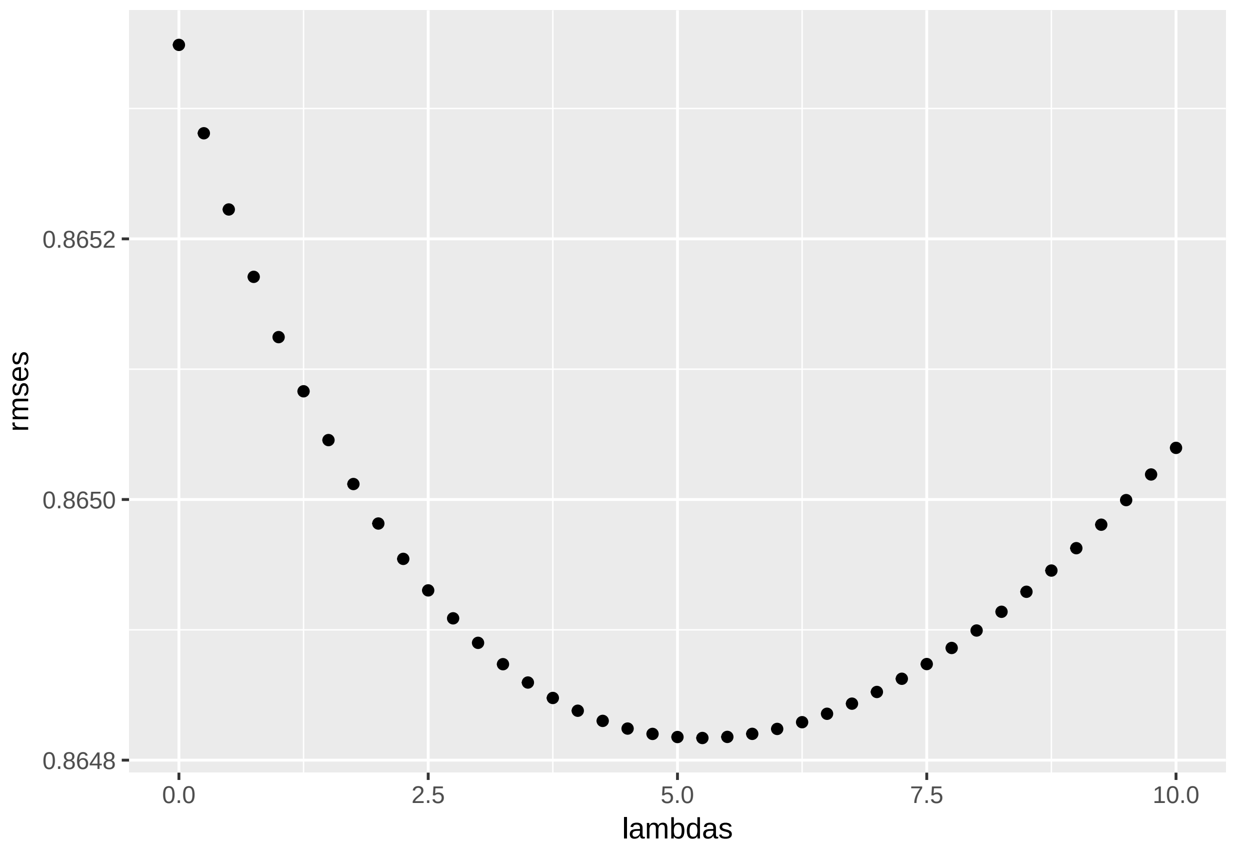
```r
lambdas <- seq(0, 10, 0.25)

mu <- mean(edx$rating)
just_the_sum <- edx %>%
  group_by(movieId) %>%
  dplyr::summarize(s = sum(rating - mu), n_i = n())

rmses <- sapply(lambdas, function(l){
  predicted_ratings <- validation %>%
    left_join(just_the_sum, by='movieId') %>%
    mutate(b_i = s/(n_i+l)) %>%
    mutate(pred = mu + b_i) %>%
    .$pred

  return(RMSE(predicted_ratings, validation$rating))
})
qplot(lambdas, rmses)
```



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

```
## [1] 2.5
```

```r
lambdas <- seq(0, 10, 0.25)

rmses <- sapply(lambdas, function(l){

  mu <- mean(edx$rating)

  b_i <- edx %>%
    group_by(movieId) %>%
    dplyr::summarize(b_i = sum(rating - mu)/(n()+l))

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

  predicted_ratings <-
    validation %>%
    left_join(b_i, by = "movieId") %>%
    left_join(b_u, by = "userId") %>%
    mutate(pred = mu + b_i + b_u) %>%
    .$pred

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

qplot(lambdas, rmses)
```



```
lambda <- lambdas[which.min(rmses)]
lambda
```

```
## [1] 5.25
```

```
rmse_results <- bind_rows(rmse_results,
                          data_frame(method="Regularized Movie + User Effect Model",
                                     RMSE = min(rmses),
                    Accuracy = mean(round(predicted_ratings/0.5)*0.5==validation$rating)))
rmse_results %>% knitr::kable()
```

method	RMSE	Accuracy
Just the average	1.0612018	0.0582861
Movie Effect Model	0.9439087	0.2257952
Movie + User Effects Model	0.8653488	0.2479812
Regularized Movie Effect Model	0.9438538	0.2259532
Regularized Movie + User Effect Model	0.8648170	0.2259532

Code to create output file with predicted ratings.

validation <- validation %>% mutate(pred_rating = predicted_ratings, pred_rating_rnd = round((predicted_ratings/0.5)*0.5)) write.csv(validation %>% select(userId, movieId) %>% mutate(rating = round(predicted_ratings/0.5)*0.5), "submission.csv", na = "", row.names=FALSE)

Conclusion

An acceptable RMSE was achieved by taking into consideration bias due to movie effects and user effects. Regularization improved the RMSE slightly for both scenarios in the analysis. The final model produced a RMSE of 0.8648.