Capstone Project: MovieLens

Johannes Le Blanc

February 8, 2019

1. Introduction

This project is the final part of the capstone project for the Data Science course by HarvardX. In this project, I create a movie recommendation system using the MovieLens dataset. The dataset used is the 10M version of the MovieLens dataset created by GroupLens research lab. The proceeding is based on the sections "recommendation systems" and "regularization" of the machine learning module as described in the textbook Introduction to Data Science.

The recommendation system employs ratings that users have given to movies to predict the users' future preferences. On this basis, the best fitting movies can be recommended to the users. A key challenge of this project is the fact that different predictors determine each outcome.

The goal of this project is to minimize the residual mean squared error (RMSE) on the test set, in this case, the set "validation". RMSE can be interpreted similar to the classic standard deviation, in this case, the error of the prediction made. Ideally, the resulting RMSE is a value below 0.87750.

The key steps are loading the dataset, excluding NAs, conducting first data exploration, creating a simple model, and finally refining that model to get to the final result.

2. Analysis Section

In the first step, the data is loaded and two sets are created by the code provided.

```r
#################################################################
# Create edx set, validation set, and submission file
#################################################################

# Note: this process could take a couple of minutes

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

```
## package 'tidyverse' successfully unpacked and MD5 sums checked
##
## The downloaded binary packages are in
##  C:\Users\Jo\AppData\Local\Temp\Rtmpi2VdxE\downloaded_packages
```

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

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

```r
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))

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

# Validation set will be 10% of MovieLens data

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)
edx <- rbind(edx, removed)

rm(dl, ratings, movies, test_index, temp, movielens, removed)
```

Characteristics of the data

The test set "validation" is significantly smaller than the training set "edx" and contains only 10% of the observations. Both sets contain the six variables "userID", "movieID", "rating", "timestamp", "title", and "genres".

```r
summary(edx)
```

```
##      userId         movieId          rating        timestamp
##  Min.   :    1   Min.   :    1   Min.   :0.500   Min.   :7.897e+08
##  1st Qu.:18124   1st Qu.:  648   1st Qu.:3.000   1st Qu.:9.468e+08
##  Median :35738   Median : 1834   Median :4.000   Median :1.035e+09
##  Mean   :35870   Mean   : 4122   Mean   :3.512   Mean   :1.033e+09
##  3rd Qu.:53607   3rd Qu.: 3626   3rd Qu.:4.000   3rd Qu.:1.127e+09
##  Max.   :71567   Max.   :65133   Max.   :5.000   Max.   :1.231e+09
##     title              genres
##  Length:9000055     Length:9000055
##  Class :character   Class :character
##  Mode  :character   Mode  :character
##
##
##
```

```r
summary(validation)
```

```
##      userId         movieId          rating        timestamp
##  Min.   :    1   Min.   :    1   Min.   :0.500   Min.   :7.897e+08
```

```
##  1st Qu.:18096   1st Qu.:  648   1st Qu.:3.000   1st Qu.:9.467e+08
##  Median :35768   Median : 1827   Median :4.000   Median :1.035e+09
##  Mean   :35870   Mean   : 4108   Mean   :3.512   Mean   :1.033e+09
##  3rd Qu.:53621   3rd Qu.: 3624   3rd Qu.:4.000   3rd Qu.:1.127e+09
##  Max.   :71567   Max.   :65133   Max.   :5.000   Max.   :1.231e+09
##     title              genres
##  Length:999999      Length:999999
##  Class :character   Class :character
##  Mode  :character   Mode  :character
##
##
##
```

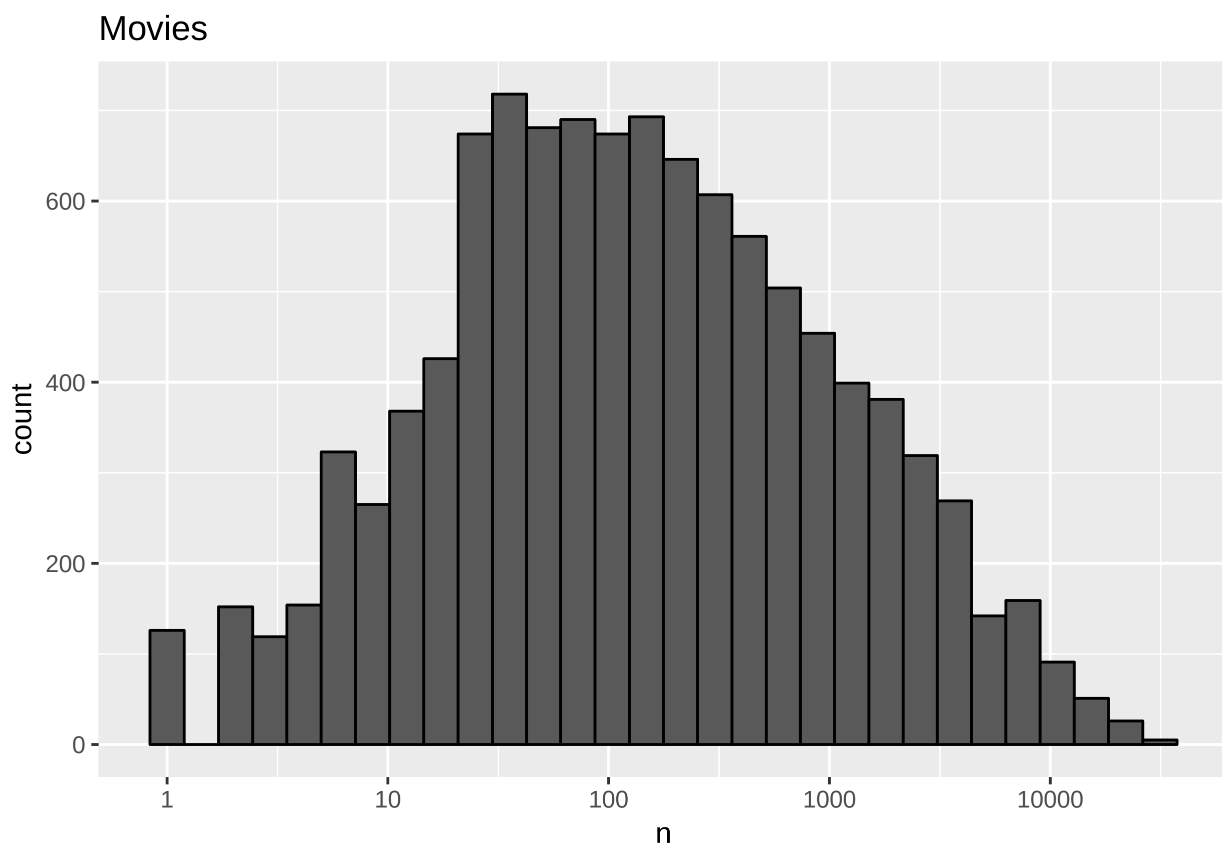
The training set contains the following number of unique users and movies:

```
edx %>%
  summarize(n_users = n_distinct(userId),
            n_movies = n_distinct(movieId))
```

```
##   n_users n_movies
## 1   69878    10677
```

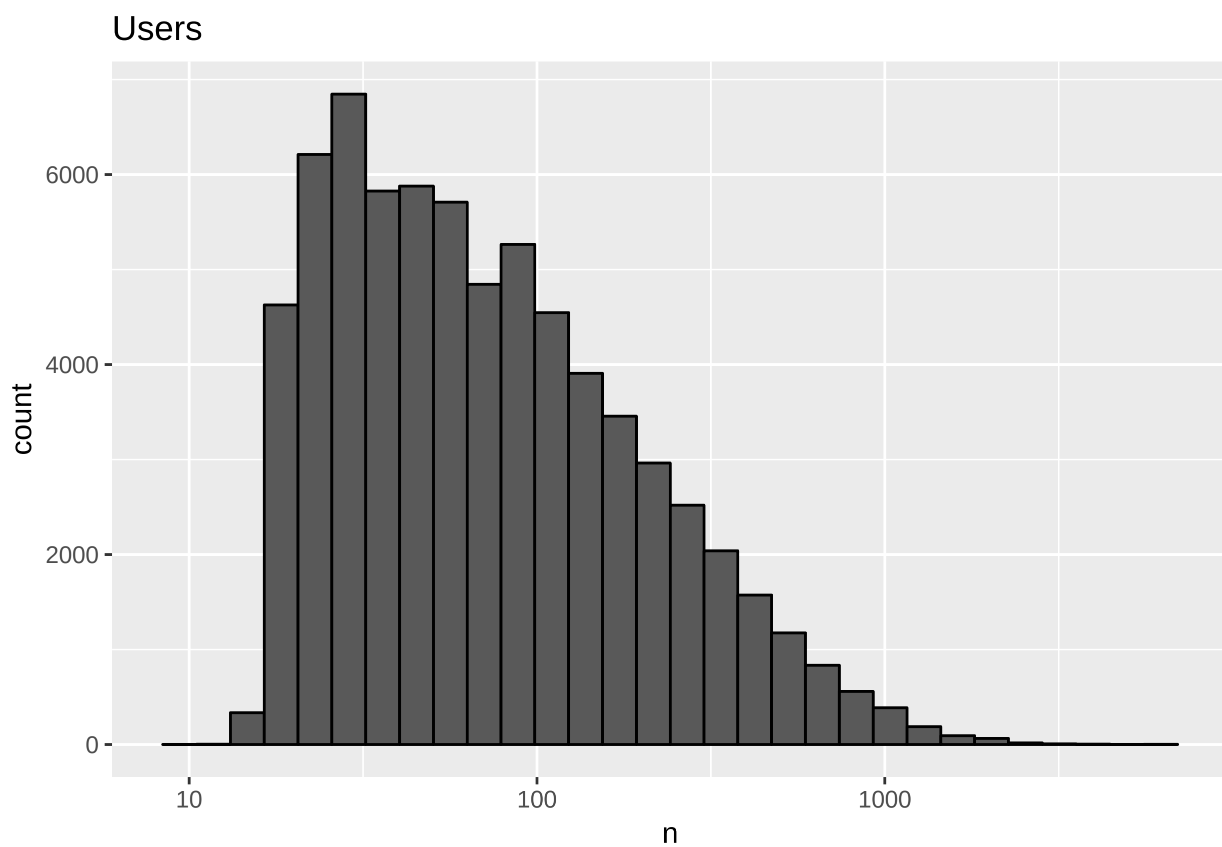
For the analysis it is furthermore important to notice that the number of recommendations per movie differs as can be shown by this plot:

```
edx %>%
  count(movieId) %>%
  ggplot(aes(n)) +
  geom_histogram(bins = 30, color = "black") +
  scale_x_log10() +
  ggtitle("Movies")
```



The same is true for users. While some users only rate a few movies, others are very active.

```
edx %>%
  count(userId) %>%
  ggplot(aes(n)) +
  geom_histogram(bins = 30, color = "black") +
  scale_x_log10() +
  ggtitle("Users")
```



The variable "title" is essential for modeling. Therefore, I want to test the variable for NAs before I proceed to the modeling:

```
sum(is.na(edx$title))
```

```
## [1] 0
```

No NAs are detected in the training-set. I repeat the procedure for the test-set validation:

```
sum(is.na(validation$title))
```

```
## [1] 0
```

The first naive model

The first basic model predicts the same rating for all movies. Possible effects from different user preferences or other factors are being ignored at this point, and it is assumed that variations are random. For this basic model, I use the mean of the ratings.

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

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

From the mean of all ratings I calculate a first naive RMSE:

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

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

Here, I construct a results table with the first RMSE:

```
rmse_results <- data_frame(method = "Just the average", RMSE = naive_rmse)

## Warning: `data_frame()` is deprecated, use `tibble()`.
## This warning is displayed once per session.
```

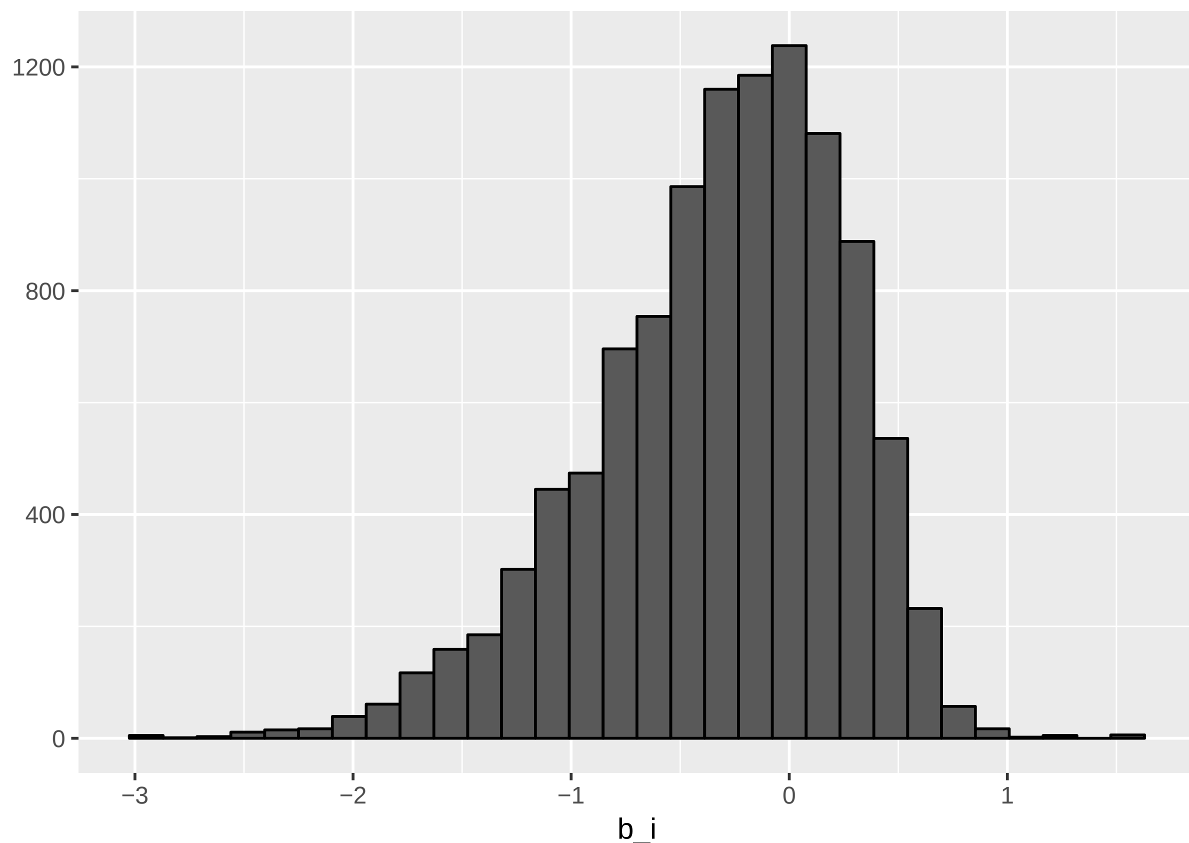
The movie effect

Movies are rated differently. For this reason, I include this difference in the model. Instead of using a least squares approach to estimate the model, I use the difference between Y and mu head to estimate the movie effects:

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

I furthermore plot the estimates to get an impression of their distribution:

```
movie_avgs %>% qplot(b_i, geom ="histogram", bins = 30, data = ., color = I("black"))
```



Now, I calculate a new prediction with the movie effect to show the difference to the naive approach:

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

model_1_rmse <- RMSE(predicted_ratings, validation$rating)
```

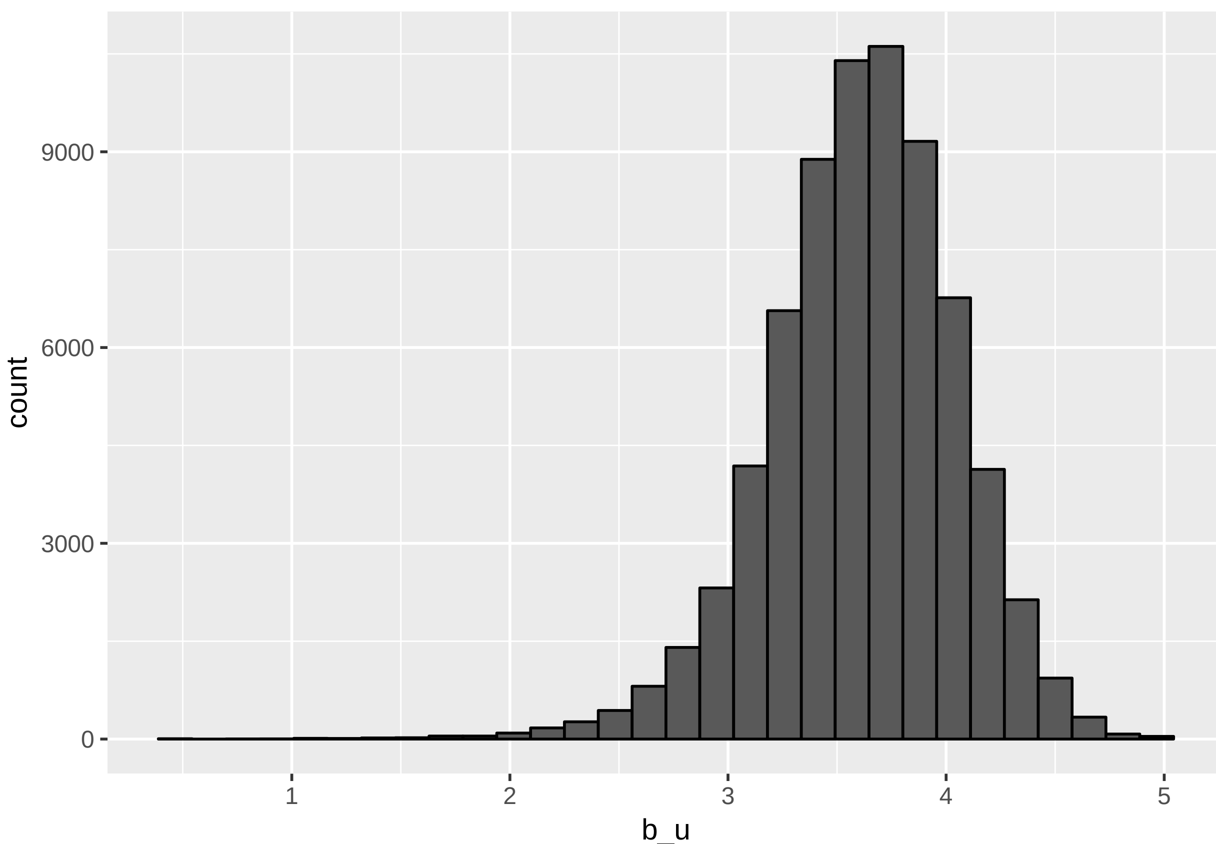
```r
rmse_results <- bind_rows(rmse_results,
                          data_frame(method="Movie Effect Model",
                                     RMSE = model_1_rmse ))
rmse_results %>% knitr::kable()
```

method	RMSE
Just the average	1.0612018
Movie Effect Model	0.9439087

The user effect

Besides the movie effect, the differences between users need to be taken into account. Therefore, I model the user effect to get the average rating for a user:

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



The distribution shows that users differ in their rating.

Again, I do not use the least squared estimate but compute an approximation of user effect:

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

Now, I create predictors to show how RMSE has improved. Again a table is constructed to show the movie and the user effect:

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


model_2_rmse <- RMSE(predicted_ratings, validation$rating)
rmse_results <- bind_rows(rmse_results,
                          data_frame(method="Movie + User Effects Model",
                                     RMSE = model_2_rmse ))
rmse_results %>%
  knitr::kable()
```

method	RMSE
Just the average	1.0612018
Movie Effect Model	0.9439087
Movie + User Effects Model	0.8292477

Refining the model

After constructing a first model, I now want to improve the results to lower the RMSE further. For this, I need to assess the mistakes of the model so far. In a first step, I check for the 10 most massive errors:

```
validation %>%
  left_join(movie_avgs, by="movieId") %>%
  mutate(residual = rating - (mu + b_i)) %>%
  arrange(desc(abs(residual))) %>%
  dplyr::select(title, residual) %>%
  distinct() %>%
  slice(1:10) %>%
  knitr::kable()
```

title	residual
PokÃ©mon Heroes (2003)	3.970803
Shawshank Redemption, The (1994)	-3.955131
Godfather, The (1972)	-3.915366
Usual Suspects, The (1995)	-3.865854
Schindler's List (1993)	-3.863493
Pokemon 4 Ever (a.k.a. PokÃ©mon 4: The Movie) (2002)	3.821782
Casablanca (1942)	-3.820424
Rear Window (1954)	-3.818652
Third Man, The (1949)	-3.811426
Seven Samurai (Shichinin no samurai) (1954)	-3.806744

To get a better impression of the 10 highest and 10 lowest rated movies, I connect "movieID" with "titles".

```
movie_titles <- movielens %>%
  filter(!is.na(title)) %>%
  dplyr::select(movieId, title) %>%
  distinct()
```

First the 10 best rated movies:

```
movie_avgs %>% left_join(movie_titles, by="movieId") %>%
  arrange(desc(b_i)) %>%
  filter(!is.na(title)) %>%
  dplyr::select(title, b_i) %>%
  slice(1:10) %>%
  knitr::kable()
```

title	b_i
Shadows of Our Forgotten Ancestors (Tini zabutykh predkiv)	1.4875348
More	1.2018205
Class, The (Entre les murs)	1.1542015
Power of Nightmares, The: The Rise of the Politics of Fear	0.9875348
End of Summer, The (Early Autumn) (Kohayagawa-ke no aki)	0.9875348
Tokyo!	0.9875348
Shawshank Redemption, The	0.9426660
Godfather, The	0.9029008
Usual Suspects, The	0.8533885
Schindler's List	0.8510281

The list shows several less known movies, why this is hardly the correct result.

Now, I try the same procedure for the 10 worst movies without correcting for the number of ratings:

```
movie_avgs %>% left_join(movie_titles, by="movieId") %>%
  arrange(b_i) %>%
  filter(!is.na(title)) %>%
  dplyr::select(title, b_i) %>%
  slice(1:10) %>%
  knitr::kable()
```

title	b_i
SuperBabies: Baby Geniuses 2	-2.717822
Hip Hop Witch, Da	-2.691037
Disaster Movie	-2.653090
Carnosaur 3: Primal Species	-2.424230
Crossover	-2.387465
Glitter	-2.336949
Pokemon 4 Ever (a.k.a. Pokémon 4: The Movie)	-2.334247
Barney's Great Adventure	-2.324965
Gigli	-2.319174
Horrors of Spider Island (Ein Toter Hing im Netz)	-2.312465

Again, almost all movies in this list are NA´s and therefore no likely result.

To get a better understanding of why these movies are listed as the best and the worst, I explore the number of ratings per movie. First the best movies:

```
edx %>% count(movieId) %>%
  left_join(movie_avgs) %>%
  left_join(movie_titles, by="movieId") %>%
  arrange(desc(b_i)) %>%
  filter(!is.na(title)) %>%
  dplyr::select(title, b_i, n) %>%
  slice(1:10) %>%
  knitr::kable()
```

```
## Joining, by = "movieId"
```

title	b_i	n
Shadows of Our Forgotten Ancestors (Tini zabutykh predkiv)	1.4875348	1
More	1.2018205	7
Class, The (Entre les murs)	1.1542015	3
Power of Nightmares, The: The Rise of the Politics of Fear	0.9875348	4
End of Summer, The (Early Autumn) (Kohayagawa-ke no aki)	0.9875348	3
Tokyo!	0.9875348	1
Shawshank Redemption, The	0.9426660	28015
Godfather, The	0.9029008	17747
Usual Suspects, The	0.8533885	21648
Schindler's List	0.8510281	23193

Number of ratings of the worst movies:

```
edx %>% count(movieId) %>%
  left_join(movie_avgs) %>%
  left_join(movie_titles, by="movieId") %>%
  arrange(b_i) %>%
  filter(!is.na(title)) %>%
  dplyr::select(title, b_i, n) %>%
  slice(1:10) %>%
  knitr::kable()
```

```
## Joining, by = "movieId"
```

title	b_i	n
SuperBabies: Baby Geniuses 2	-2.717822	56
Hip Hop Witch, Da	-2.691037	14
Disaster Movie	-2.653090	32
Carnosaur 3: Primal Species	-2.424230	68
Crossover	-2.387465	4
Glitter	-2.336949	339
Pokemon 4 Ever (a.k.a. Pokémon 4: The Movie)	-2.334247	202
Barney's Great Adventure	-2.324965	208
Gigli	-2.319174	313
Horrors of Spider Island (Ein Toter Hing im Netz)	-2.312465	30

Regularization

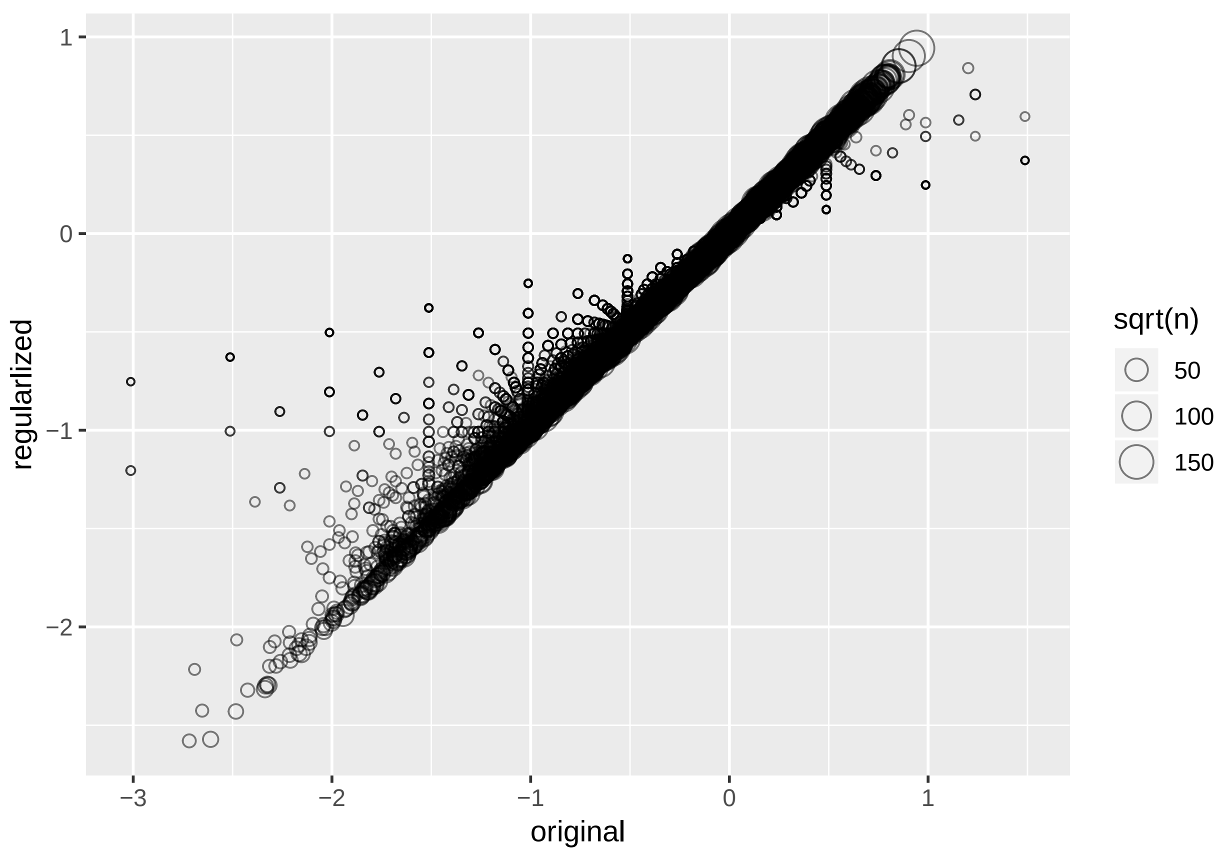
It turns out that the 10 best and the 10 worst movies are rated by only a meager number of users. Such low numbers of users can increase the uncertainty of our predictions. By using regularization, I exclude high ratings coming from small samples. To eliminate small samples, I include the cut-off value lambda. By using lambda, it is possible to penalize small samples without affecting larger ones.

In the regularized estimates, I exclude movies with less than 3 ratings.

```
lambda <- 3
mu <- mean(edx$rating)
movie_reg_avgs <- edx %>%
  group_by(movieId) %>%
  summarize(b_i = sum(rating - mu)/(n()+lambda), n_i = n())
```

I create a plot to show the regularized estimates vs. least squares estimates:

```
data_frame(original = movie_avgs$b_i,
           regularlized = movie_reg_avgs$b_i,
           n = movie_reg_avgs$n_i) %>%
  ggplot(aes(original, regularlized, size=sqrt(n))) +
  geom_point(shape=1, alpha=0.5)
```



After including lambda into the model, I again create a list of the top 10 movies:

```
edx %>%
  count(movieId) %>%
  left_join(movie_reg_avgs) %>%
  left_join(movie_titles, by="movieId") %>%
```

```r
  arrange(desc(b_i)) %>%
  filter(!is.na(title)) %>%
  dplyr::select(title, b_i, n) %>%
  slice(1:10) %>%
  knitr::kable()
```

```
## Joining, by = "movieId"
```

title	b_i	n
Shawshank Redemption, The	0.9425650	28015
Godfather, The	0.9027482	17747
Usual Suspects, The	0.8532702	21648
Schindler's List	0.8509180	23193
More	0.8412744	7
Casablanca	0.8077428	11232
Rear Window	0.8058817	7935
Sunset Blvd. (a.k.a. Sunset Boulevard)	0.8025903	2922
Third Man, The	0.7981535	2967
Double Indemnity	0.7972415	2154

And a list of the 10 worst movies based on lambda:

```r
edx %>%
  count(movieId) %>%
  left_join(movie_reg_avgs) %>%
  left_join(movie_titles, by="movieId") %>%
  arrange(b_i) %>%
  filter(!is.na(title)) %>%
  dplyr::select(title, b_i, n) %>%
  slice(1:10) %>%
  knitr::kable()
```

```
## Joining, by = "movieId"
```

title	b_i	n
SuperBabies: Baby Geniuses 2	-2.579628	56
Disaster Movie	-2.425683	32
Carnosaur 3: Primal Species	-2.321798	68
Glitter	-2.316449	339
Pokemon 4 Ever (a.k.a. Pokémon 4: The Movie)	-2.300088	202
Gigli	-2.297157	313
Barney's Great Adventure	-2.291909	208
Hip Hop Witch, Da	-2.216148	14
Yu-Gi-Oh!	-2.198762	80
Carnosaur 2	-2.143651	92

Now, I create a table to show how the results have changed compared to previous estimates of RMSE:

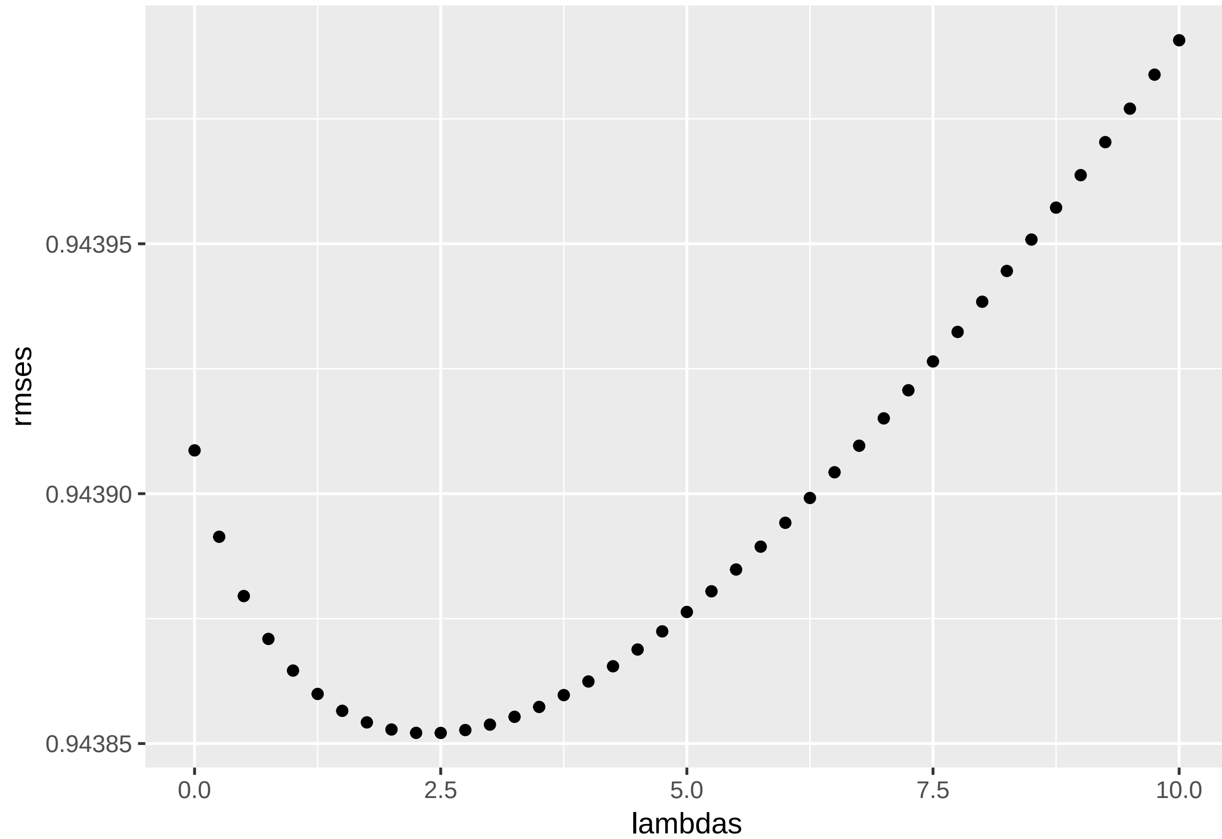
```r
predicted_ratings <- validation %>%
  left_join(movie_reg_avgs, by='movieId') %>%
  mutate(pred = mu + b_i) %>%
  .$pred
```

```r
model_3_rmse <- RMSE(predicted_ratings, validation$rating)
rmse_results <- bind_rows(rmse_results,
                          data_frame(method="Regularized Movie Effect Model",
                                     RMSE = model_2_rmse ))
rmse_results %>%
  knitr::kable()
```

method	RMSE
Just the average	1.0612018
Movie Effect Model	0.9439087
Movie + User Effects Model	0.8292477
Regularized Movie Effect Model	0.8292477

The lambda of 3 was just a random number that might not lead to the optimal output. To find the optimal lambda, I can use cross-validation and plot the resulting lambdas:

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

mu <- mean(edx$rating)
just_the_sum <- edx %>%
  group_by(movieId) %>%
  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)
```




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

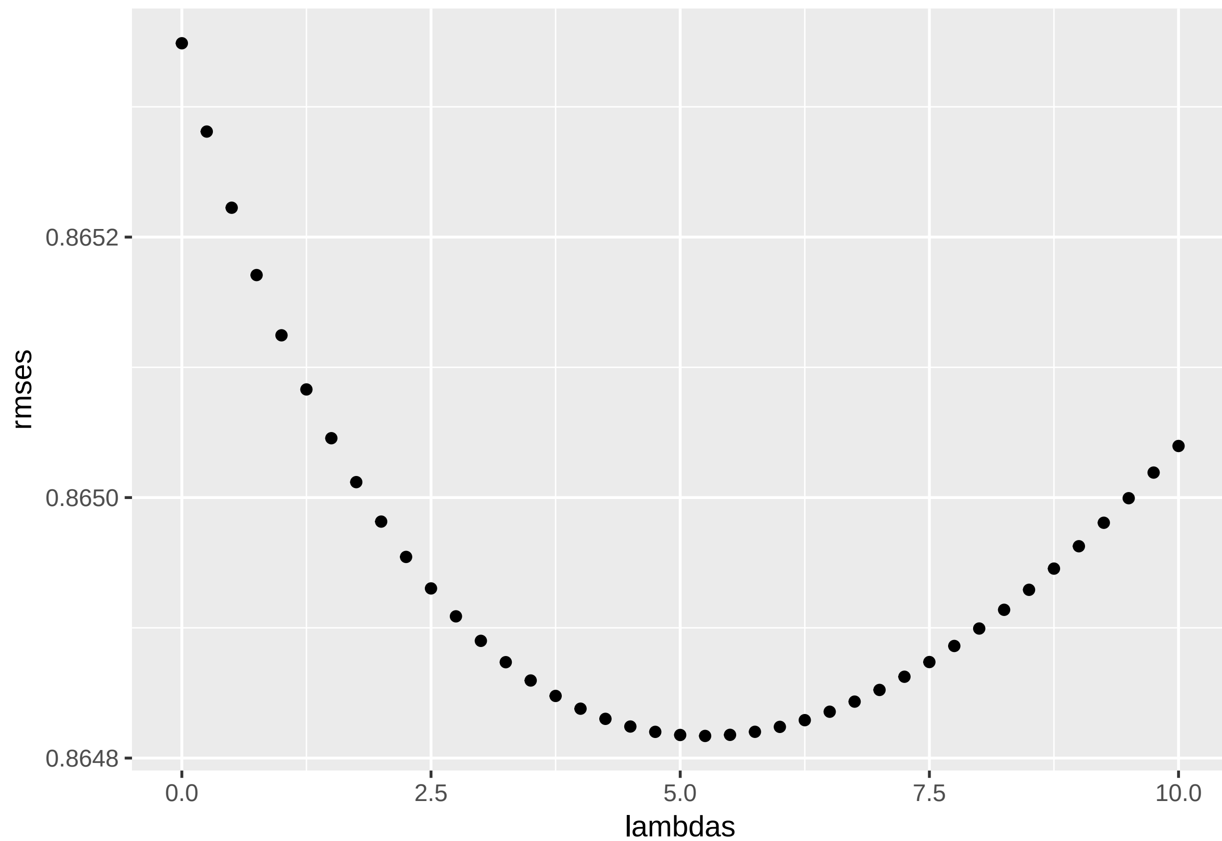
```
## [1] 2.5
```

The optimal lambda is actually at 2.5.

Calculate minimum of the cross-validation:

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

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

  mu <- mean(edx$rating)

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

  b_u <- edx %>%
    left_join(b_i, by="movieId") %>%
    group_by(userId) %>%
    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)
```



Now, I can calculate the optimal lambda for the whole model:

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

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

3. Results

Finally, I create a table to show the results of the basic model and the different outcomes of RSME that I found by including the movie and user effects as well as regularizing.

```
rmse_results <- bind_rows(rmse_results,
                          data_frame(method="Regularized Movie + User Effect Model",
                                     RMSE = min(rmses)))
rmse_results %>%
  knitr::kable()
```

method	RMSE
Just the average	1.0612018

method	RMSE
Movie Effect Model	0.9439087
Movie + User Effects Model	0.8292477
Regularized Movie Effect Model	0.8292477
Regularized Movie + User Effect Model	0.8648170

The RMSE was reduced from 1.06 to 0.86, which is a significant reduction.

4. Conclusion

After an initial data exploration, I constructed a simple model that I refined in several steps. I included the movie and the user effects and used regularization to improve the outcome. As shown in the results section, the model decreased the RMSE in the first step to 0.9439087 by including the movie effect. The movie and the user effect together reduced the value of the RMSE in a second step to even 0.8292477. The same result was reached with the regularized movie effect model. Surprisingly, the inclusion of the user effect into the regularized movie effect model increased the value of the RMSE again to 0.8648170. However, the goal of pushing the RMSE below 0.87750 was achieved.