Harvard Capstone Project

Benno Weber

10 4 2020

Introduction

This report describes the analysis steps performed on data containing movie ratings. The report consists of a methods section that explains the process and techniques used including data cleaning, data exploration and visualization, insights gained, and modeling approaches, results section that presents the modeling results and discusses the model performance and a conclusion section that gives a brief summary of the report, its limitations and future work. Goal of the methods applied is to predict ratings a user might give to a movie. This prediction might be used for instance for a movie recommandation system used by Netflix or other streaming portals. Two datasets were obtained from the 'MovieLens 10M dataset', the 'edx' dataset containing 9000055 observations of different ratings and the 'validation' datatset which consists of nearly 1 Million observations. It should be noted that the only use of the 'validation' data is to obtain the final scores of the algorithms. The metric of choice is RMSE which is computed as follows.

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

Training of algorithms and fitting the models was exclusivly done using the 'edx' dataset. The goal of the analysis is to find a model that minimizes the RMSE.

This is how the 'edx' and the 'validation' datasets were obtained:

```r
################################
# Create edx set, validation set
################################
# 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")
 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")
```

```r
# Validation set will be 10% of MovieLens data
set.seed(1, sample.kind="Rounding")
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)
```

Methods

Basis information about the ‘edx’ dataset can be obtained es follows:

```r
head(edx)
```

```
##   userId movieId rating timestamp                         title
## 1      1     122      5 838985046              Boomerang (1992)
## 2      1     185      5 838983525               Net, The (1995)
## 4      1     292      5 838983421               Outbreak (1995)
## 5      1     316      5 838983392               Stargate (1994)
## 6      1     329      5 838983392 Star Trek: Generations (1994)
## 7      1     355      5 838984474       Flintstones, The (1994)
##                           genres
## 1                 Comedy|Romance
## 2          Action|Crime|Thriller
## 4   Action|Drama|Sci-Fi|Thriller
## 5        Action|Adventure|Sci-Fi
## 6 Action|Adventure|Drama|Sci-Fi
## 7        Children|Comedy|Fantasy
```

```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
```

```
##
##
##
```

As the models used in this analysis are chosen on the assumption that there is an effect of users, movies and genres which influences the specific rating of a movie, here's a short overview of these variables:

```
#Number of different users
length(unique(edx$userId))
```

```
## [1] 69878
```

```
#Number of different movies
length(unique(edx$movieId))
```

```
## [1] 10677
```

Here's short overview of how many ratings we are dealing with for some of the genres:

```
sum(str_detect(edx$genres,"Drama"))
```

```
## [1] 3910127
```

```
sum(str_detect(edx$genres,"Comedy"))
```

```
## [1] 3540930
```

```
sum(str_detect(edx$genres,"Thriller"))
```

```
## [1] 2325899
```

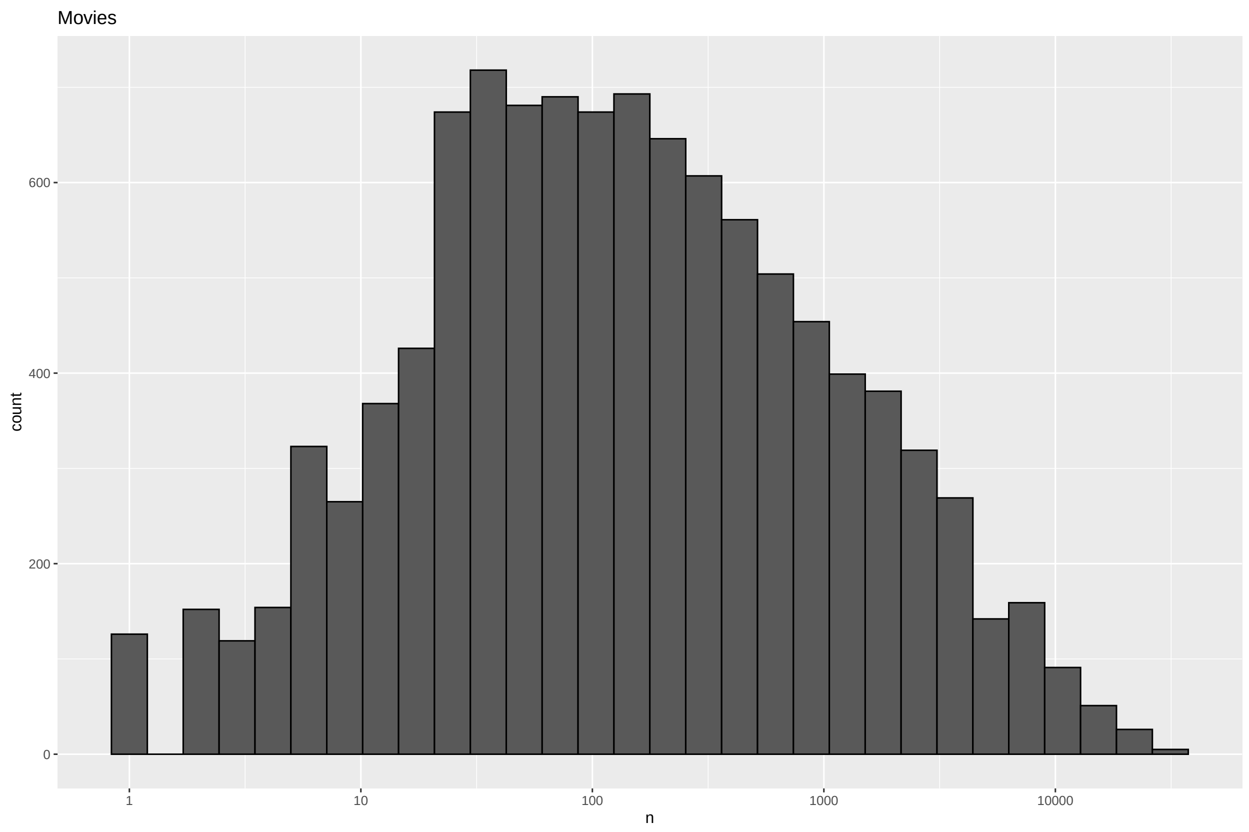
```
sum(str_detect(edx$genres,"Romance"))
```

```
## [1] 1712100
```

It was mentioned before that it is assumed that there are effects of users, movies an genres biasing the prediction of ratings. In order to prove this let us look at the following plots and see how different movies are rated different than others (e.g. more often) and how different users have there own behavior of rating movies (e.g. some of them rate multiple movies, others don't rate any).

Movie effect

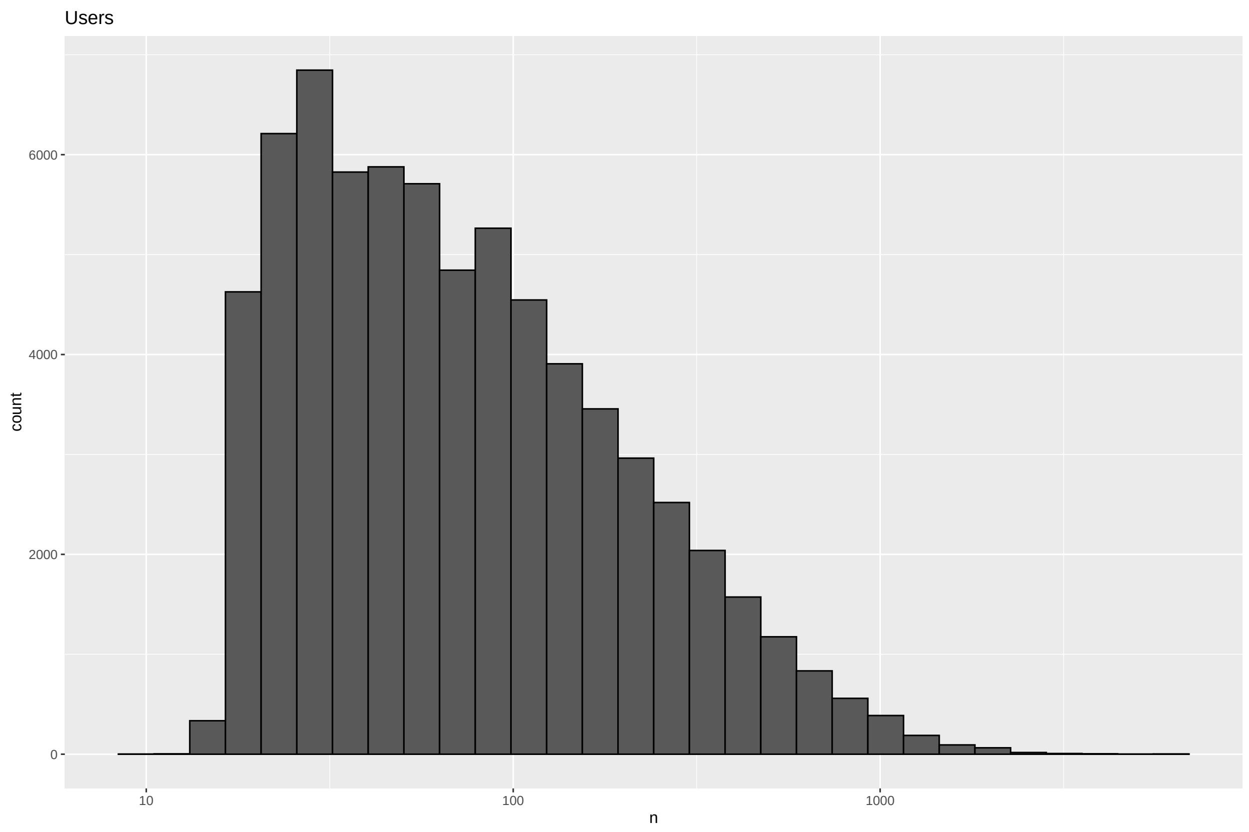
```
#visualizing the movie effect
edx %>%
     dplyr::count(movieId) %>%
     ggplot(aes(n)) +
     geom_histogram(bins = 30, color = "black") +
     scale_x_log10() +
     ggtitle("Movies")
```



The movie bias should account for movies which don't get rated a lot in general where as movies with multiple ratings (e.g. Blockbusters) should be considered more important.

User effect

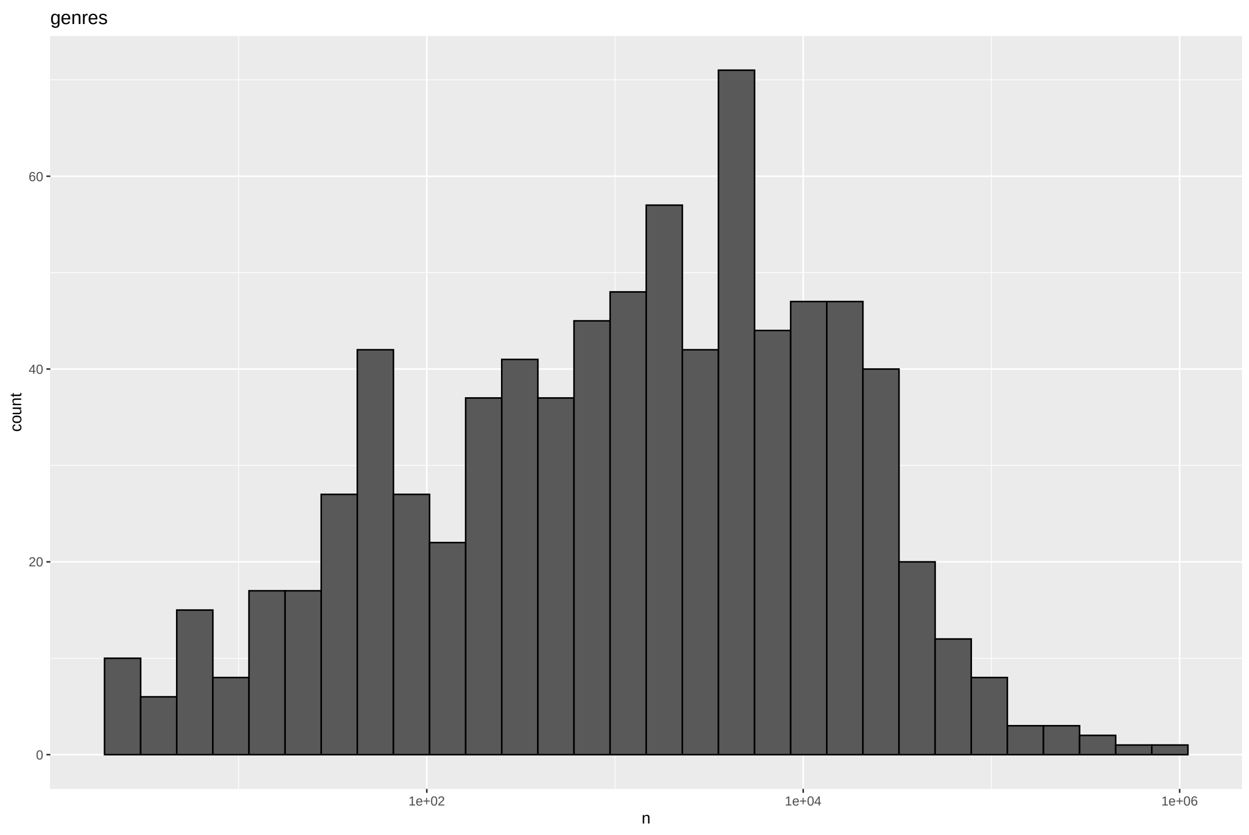
```r
#visualize User effect
edx %>% dplyr::count(userId) %>%
  ggplot(aes(n)) +
  geom_histogram(bins = 30, color = "black") +
  scale_x_log10() +
  ggtitle("Users")
```



The user bias should account for users who rate very few movies and therefor there opinion on these movies should not influence predictions as much as ratings from users rating multiple movies.

Genre effect

```r
#visualize genre effect
edx %>% dplyr::count(genres) %>%
  ggplot(aes(n)) +
  geom_histogram(bins = 30, color = "black") +
  scale_x_log10() +
  ggtitle("genres")
```



Results

Baseline model

The prediction of a movie rating is simply the average of all ratings. This is the baseline to which other models will be compared ragarding RMSE.

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

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

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

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

```
rmse_results <- data_frame(method = "Naive model", RMSE = naive_rmse)
knitr::kable(rmse_results)
```

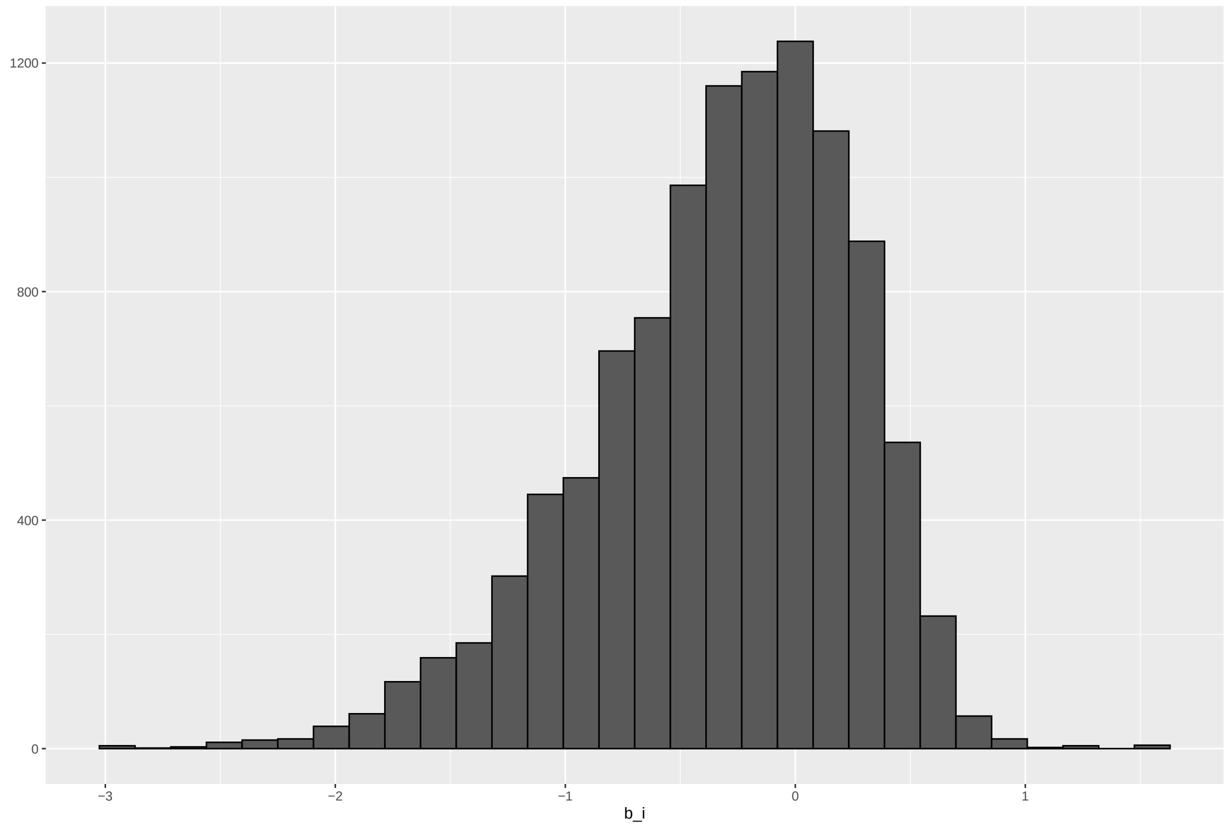
method	RMSE
Naive model	1.061202

As expected - this score is rather poor. More than 1 star deviation from the real rating is hardly better than guessing on a scale of 1-5 stars.

Movie effect model

To account for the movie effect a penalty term b__i is introduced here. It will be used to improve the baseline model by adding it to the equation for prediction.

```
#Movie effect
movie_avgs <- edx %>%
  group_by(movieId) %>%
  summarize(b_i = mean(rating - mu))
movie_avgs %>% qplot(b_i, geom ="histogram", bins = 30, data = ., color = I("black"))
```



```
#Build model
predicted_ratings <- mu + validation %>%
  left_join(movie_avgs, by='movieId') %>%
  .$b_i
model_movie_rmse <- RMSE(validation$rating,predicted_ratings)
rmse_results <- bind_rows(rmse_results,
                          data_frame(method="Movie Effect Model",
                                     RMSE = model_movie_rmse ))
knitr::kable(rmse_results)
```

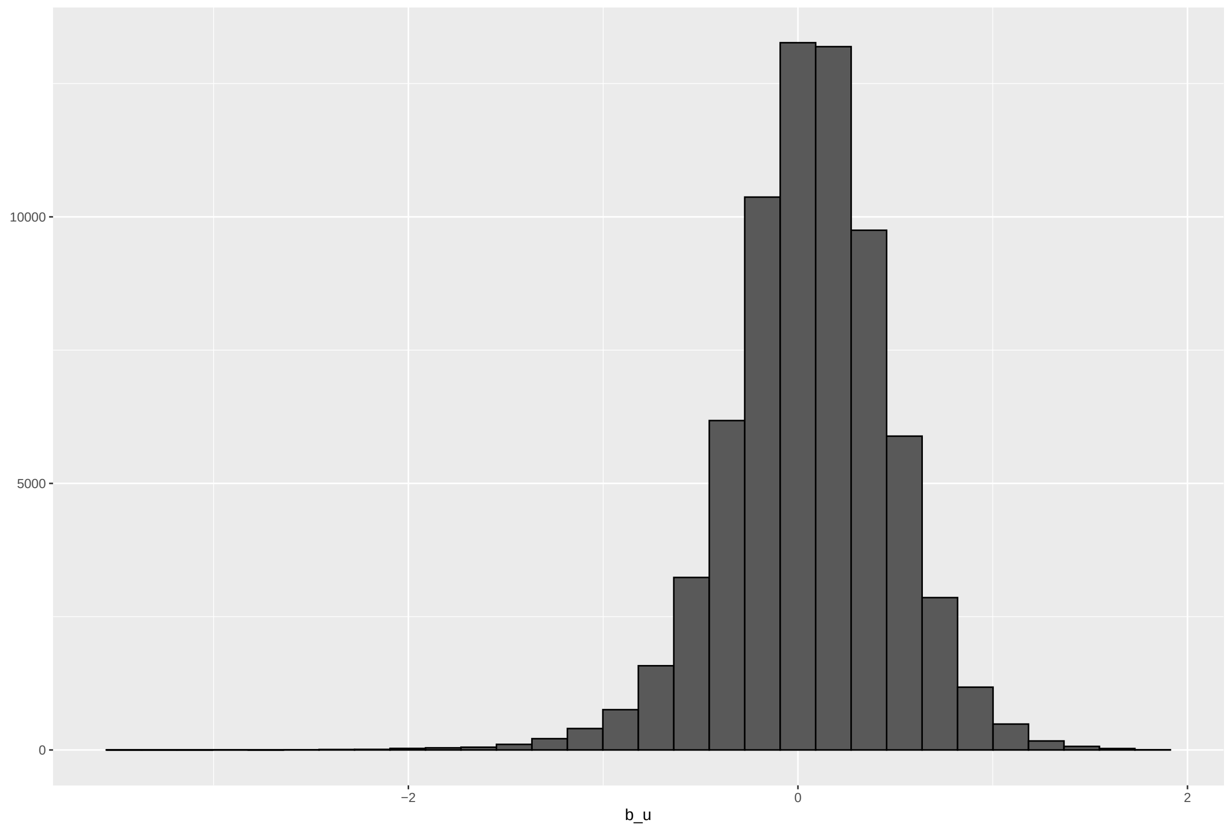
method	RMSE
Naive model	1.0612018
Movie Effect Model	0.9439087

By accounting for a movie effect the model improves its RMSE by over 10%. This is already quite good.

Movie and User effect model

b_u is our user effect which will be included to the model to account for individual user behavior.

```
#User effect
user_avgs <- edx %>%
  left_join(movie_avgs, by='movieId') %>%
  group_by(userId) %>%
  summarize(b_u = mean(rating - mu - b_i))
user_avgs %>% qplot(b_u, geom="histogram", bins = 30, data= ., color = I("black"))
```



```r
#Build model
predicted_ratings_user_movie <- validation %>%
  left_join(movie_avgs, by='movieId') %>%
  left_join(user_avgs,  by='userId') %>%
  mutate(pred = mu + b_i + b_u)
# test and save rmse results
model_user_movie_rmse <- RMSE(validation$rating,predicted_ratings_user_movie$pred)
rmse_results <- bind_rows(rmse_results,
                          data_frame(method="Movie and User Effect Model",
                                     RMSE = model_user_movie_rmse ))
knitr::kable(rmse_results)
```

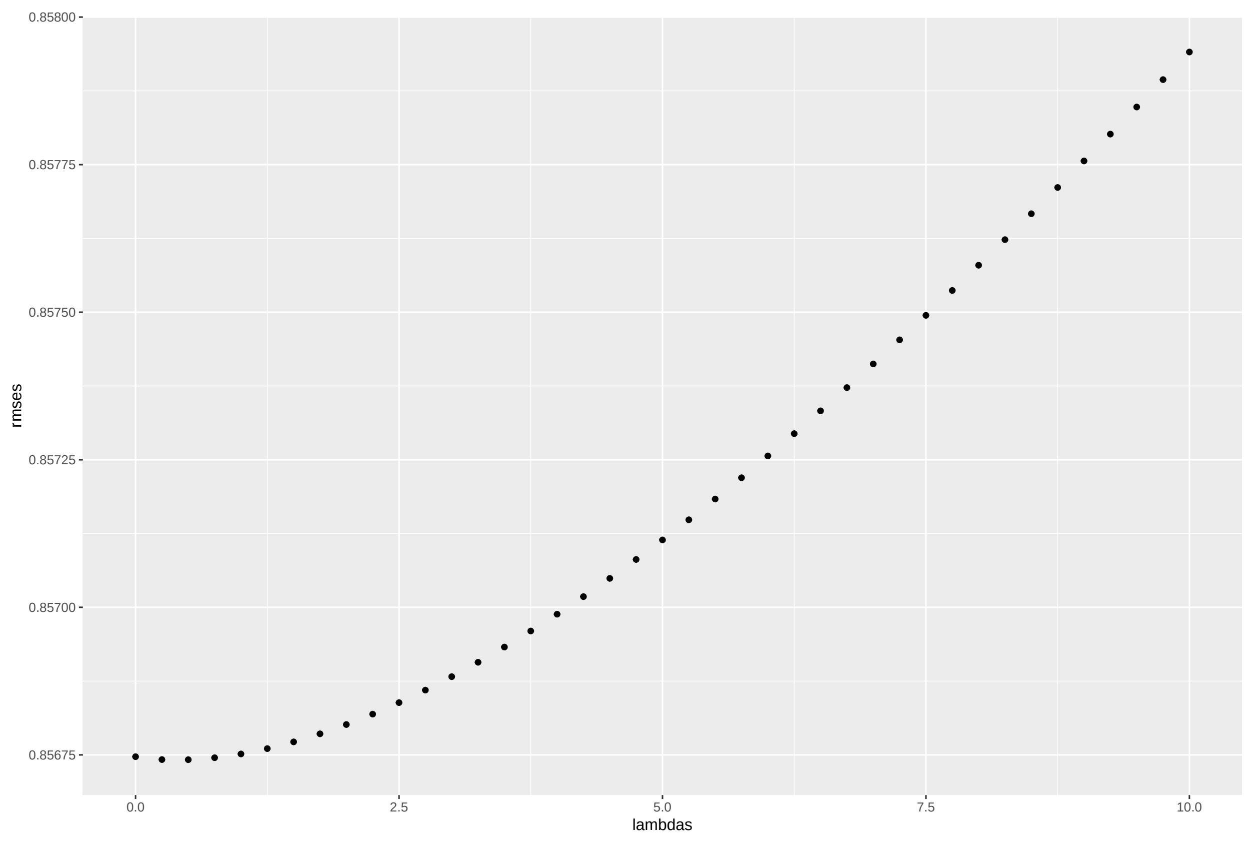
method	RMSE
Naive model	1.0612018
Movie Effect Model	0.9439087
Movie and User Effect Model	0.8653488

A further 8% decrease of the RSME can be achieved by accounting for movie and user effect.

Regularizing Movie and User model

To improve our results, we will use regularization. Regularization constrains the total variability of the effect sizes by penalizing large estimates that come from small sample sizes. With regularization a penalty term is added to the RMSE. The penalty term gets larger when many b's (effect terms from users or movies) are large. With this approach a tuning parameter lambda is introduced. It is chosen by cross-validation on the training dataset edx which was partitioned in an additional training and test dataset.

```r
#Create an additional partition of training and test sets from the provided edx dataset
#Cross validation must not be performed on the validation dataset!
test_index <- createDataPartition(y = edx$rating, times = 1,
                                  p = 0.2, list = FALSE)
train_set <- edx[-test_index,]
test_set <- edx[test_index,]
# Choose lambda by cross validation.
lambdas <- seq(0, 10, 0.25)
rmses <- sapply(lambdas, function(lambda){
  b_i <- edx %>%
    group_by(movieId) %>%
    summarize(b_i = sum(rating - mu)/(n()+lambda))

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

  predicted_ratings <- test_set %>%
    left_join(b_i, by = "movieId") %>%
    left_join(b_u, by = "userId") %>%
    mutate(pred = mu + b_i + b_u) %>%
    .$pred
  return(RMSE(test_set$rating,predicted_ratings))
})
# Plot rmses vs lambdas to see which lambda minimizes rmse
qplot(lambdas, rmses)
```



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

So the optimal value for the tuning parameter lambda is 0.5 which will be used to re-calculate the movie-user model with regularization.

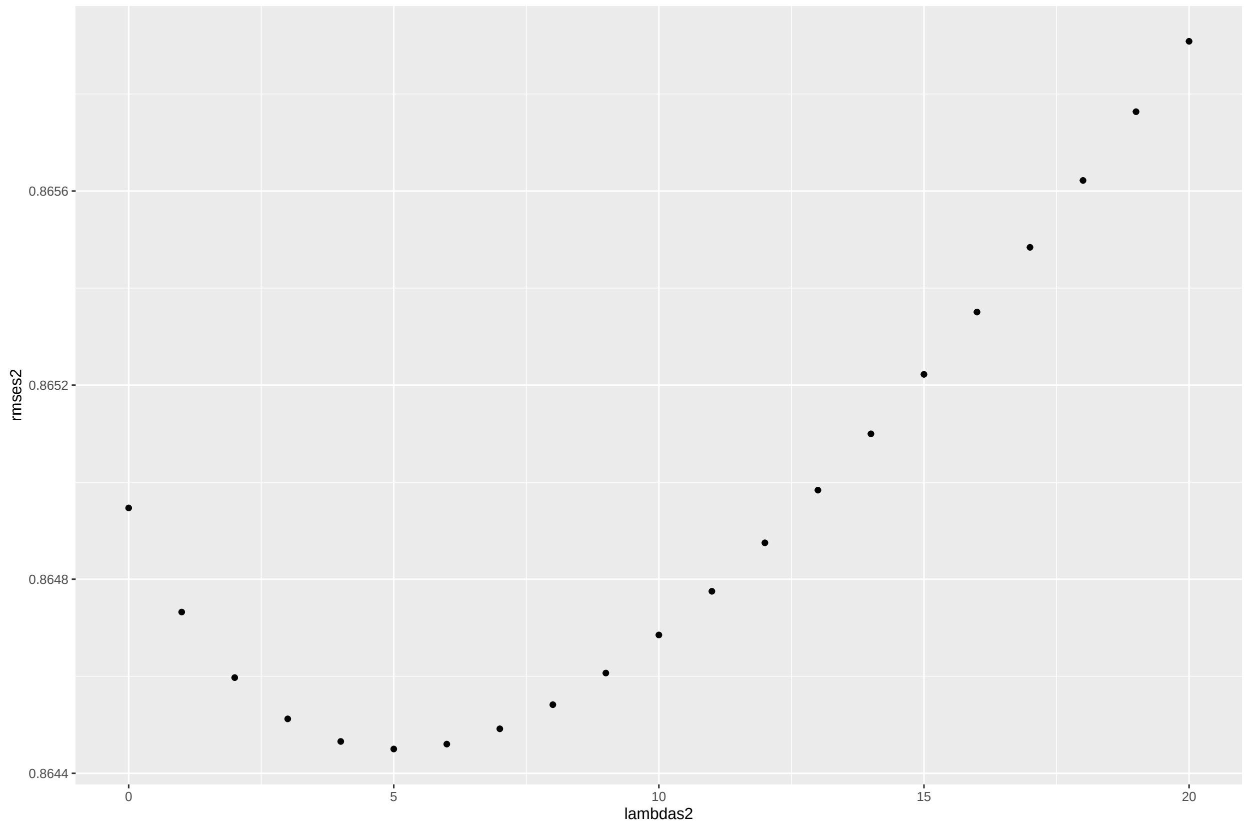
```r
# Movie effect regularized b_i using lambda
movie_avgs_reg <- edx %>%
  group_by(movieId) %>%
  summarize(b_i = sum(rating - mu)/(n()+lambda), n_i = n())
# User effect regularized b_u using lambda
user_avgs_reg <- edx %>%
  left_join(movie_avgs_reg, by='movieId') %>%
  group_by(userId) %>%
  summarize(b_u = sum(rating - mu - b_i)/(n()+lambda), n_u = n())
# Predict ratings
predicted_ratings_reg <- validation %>%
  left_join(movie_avgs_reg, by='movieId') %>%
  left_join(user_avgs_reg, by='userId') %>%
  mutate(pred = mu + b_i + b_u) %>%
  .$pred
```

```r
model_reg_rmse <- RMSE(validation$rating,predicted_ratings_reg)
rmse_results <- bind_rows(rmse_results,
                          data_frame(method="Regularized Movie and User Model",
                                     RMSE = model_reg_rmse ))
rmse_results %>% knitr::kable()
```

method	RMSE
Naive model	1.0612018
Movie Effect Model	0.9439087
Movie and User Effect Model	0.8653488
Regularized Movie and User Model	0.8652226

Genre model

```r
#genre model - optimize tuning parameter lambda2
#b_g...genre effects
lambdas2 <- seq(0, 20, 1)
rmses2 <- sapply(lambdas2, function(lambda2){

  b_i <- edx %>%
    group_by(movieId) %>%
    summarize(b_i = sum(rating - mu)/(n()+lambda2))

  b_u <- edx %>%
    left_join(b_i, by="movieId") %>%
    group_by(userId) %>%
    summarize(b_u = sum(rating - b_i - mu)/(n()+lambda2))

 b_g <- edx %>%
    left_join(b_i, by='movieId') %>%
    left_join(b_u, by='userId') %>%
    group_by(genres) %>%
    summarize(b_g = sum(rating - mu - b_i - b_u)/(n()+lambda2), n_g = n())

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

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

qplot(lambdas2, rmses2)
```



```
lambda2<-lambdas2[which.min(rmses2)]

# Build model using lambda2
movie_reg_avgs_2 <- edx %>%
  group_by(movieId) %>%
  summarize(b_i = sum(rating - mu)/(n()+lambda2), n_i = n())
user_reg_avgs_2 <- edx %>%
  left_join(movie_reg_avgs_2, by='movieId') %>%
  group_by(userId) %>%
  summarize(b_u = sum(rating - mu - b_i)/(n()+lambda2), n_u = n())
genre_reg_avgs <- edx %>%
  left_join(movie_reg_avgs_2, by='movieId') %>%
  left_join(user_reg_avgs_2, by='userId') %>%
  group_by(genres) %>%
  summarize(b_g = sum(rating - mu - b_i - b_u)/(n()+lambda2), n_g = n())

predicted_ratings <- validation %>%
  left_join(movie_reg_avgs_2, by='movieId') %>%
  left_join(user_reg_avgs_2, by='userId') %>%
  left_join(genre_reg_avgs, by = 'genres') %>%
  mutate(pred = mu + b_i + b_u + b_g) %>%
  .$pred
model_UserMovieGenreReg_rmse <- RMSE(validation$rating,predicted_ratings)
rmse_results <- bind_rows(rmse_results,
                          data_frame(method="Reg Movie, User and Genre Effect Model",
                                     RMSE = model_UserMovieGenreReg_rmse ))
```

Results overview

```r
rmse_results %>% knitr::kable()
```

method	RMSE
Naive model	1.0612018
Movie Effect Model	0.9439087
Movie and User Effect Model	0.8653488
Regularized Movie and User Model	0.8652226
Reg Movie, User and Genre Effect Model	0.8644501

Conclusion

In comparison to the naive approach - just predicting the overall rating average of all movies a big improvment of over 10% of RMSE could be achived by including the movie effect in the model. The RMSE decreased another 8% by including the user effect. Regularization did improve the model but not as much as was expected. This report only covers possible user, movie and genre effects for prediction. By applying the same techniques as above further models could be build taking for example the release year of the movie into account.