Harvard Capstone project movielens

Elin Brännström

26 May 2020

Introduction

One important feature of Machine Learning is the ability to predict outcomes based on attributes, which is the main purpose of this project and report. The data set examined is a subset of the MovieLens data set included in the dslabs packages replicating a Netflix contest improving movie rating predictions. The project is done for the Capstone module of the Harvard Data Science certification.

The report will first attend to examining and cleaning of data, then proceed to splitting data for training, testing & validation and finally to modelling and results. As a last step, conclusions and lessons learned will be discussed.

Modelling techniques explored consist of regularization and cross-validation, motivated by Rafael A.Irizarry's work [1].

Observe that TinyTex is a necessary library for knitting to PDF.

Method & analysis

Data import and partition

As a first step, the necessary libraries and movielens data will be imported into R. Next, the following steps are executed for preparing the data. 1. The imported data is split into training data named "edx" (90 %) and validation data named "validation" (10%) 2. The validation set is adjusted to assure all users and movies in validation set also are represented in the training data

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

# Validation set will be 10% of MovieLens data
set.seed(1, sample.kind="Rounding")
# if using R 3.5 or earlier, use `set.seed(1)` instead
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)
```

Data preparation

The validation set will be left untouched until the final validation of the model. The model will be trained using a train set and tested on a test set.

1. Explore what the data looks like and what attributes it contains

```
##   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
```

2. Make a check there are no missing values in the dataset.

```
which(is.na(edx))
```

```
## integer(0)
```

3. In order to make the data into a tidy format, we observe the release year must be extracted from the movie title, creating a new column "year_release"

```
# Create new column with release year
edx <- edx %>% mutate(year_release=str_sub(title,-5,-2))
validation <- validation %>% mutate(year_release=str_sub(title,-5,-2))
```

Additional Data partition

1. The "edx" data set is partitioned for training named "train_set" (80%) and test set named "test_set" (20%)
2. The test set is adjusted to assure all users and movies in "test_set" are also represented in the training set

```
#Create new partition for training & testing of edx data,
# using 80% for training and 20% for testing
set.seed(1, sample.kind="Rounding")
test_index <- createDataPartition(y = edx$rating, times=1, p=0.2, list=FALSE)
training_set <- edx[-test_index,]
test_set <- edx[test_index,]

#Remove users and movies from test set that are not included in training set
test_set <- test_set %>% semi_join(training_set, by="movieId") %>%
  semi_join(training_set, by ="userId")

#Put removed lines back into training set
removed_new <- anti_join(training_set,test_set)
training_set <- rbind(training_set,removed_new)
```

The data is now in a tidy format and prepared for modelling. But before proceeding to the modelling part, we will do some data examination by visualizing the data.

Data visualization

Hypothesis

Firstly, I will form a working hypothesis based on potential biases related to users and ratings.

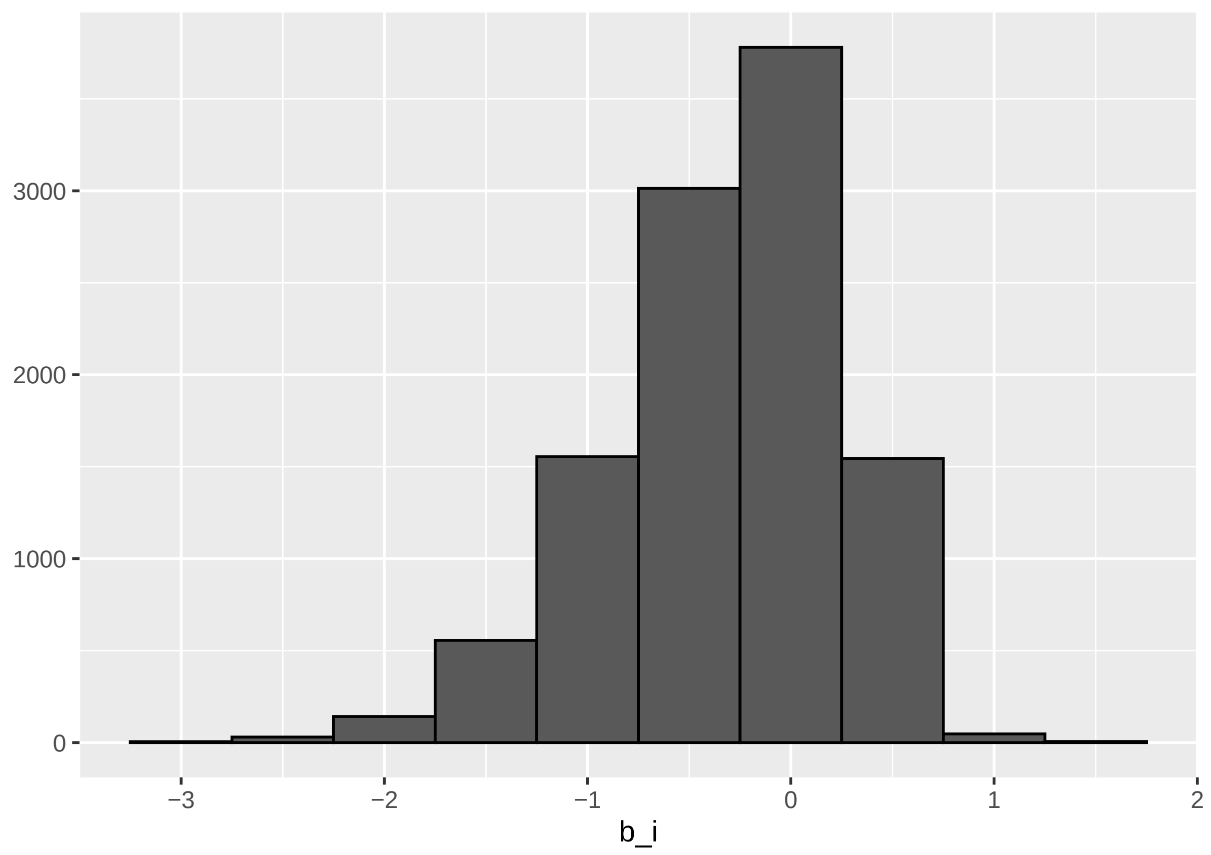
- There is a user bias where some users tend to be more generous in ratings than others
- There is a movie bias where some movies are rated higher than others
- There is a genre bias where some genres have higher ratings than others
- There is a year bias where some release years have higher ratings than others

Let's justify or reject the hypothesis with data.

Firstly, examining the average rating of each movie with the average rating of all movies generates the following plot. We can see that some movies clearly are better than others.

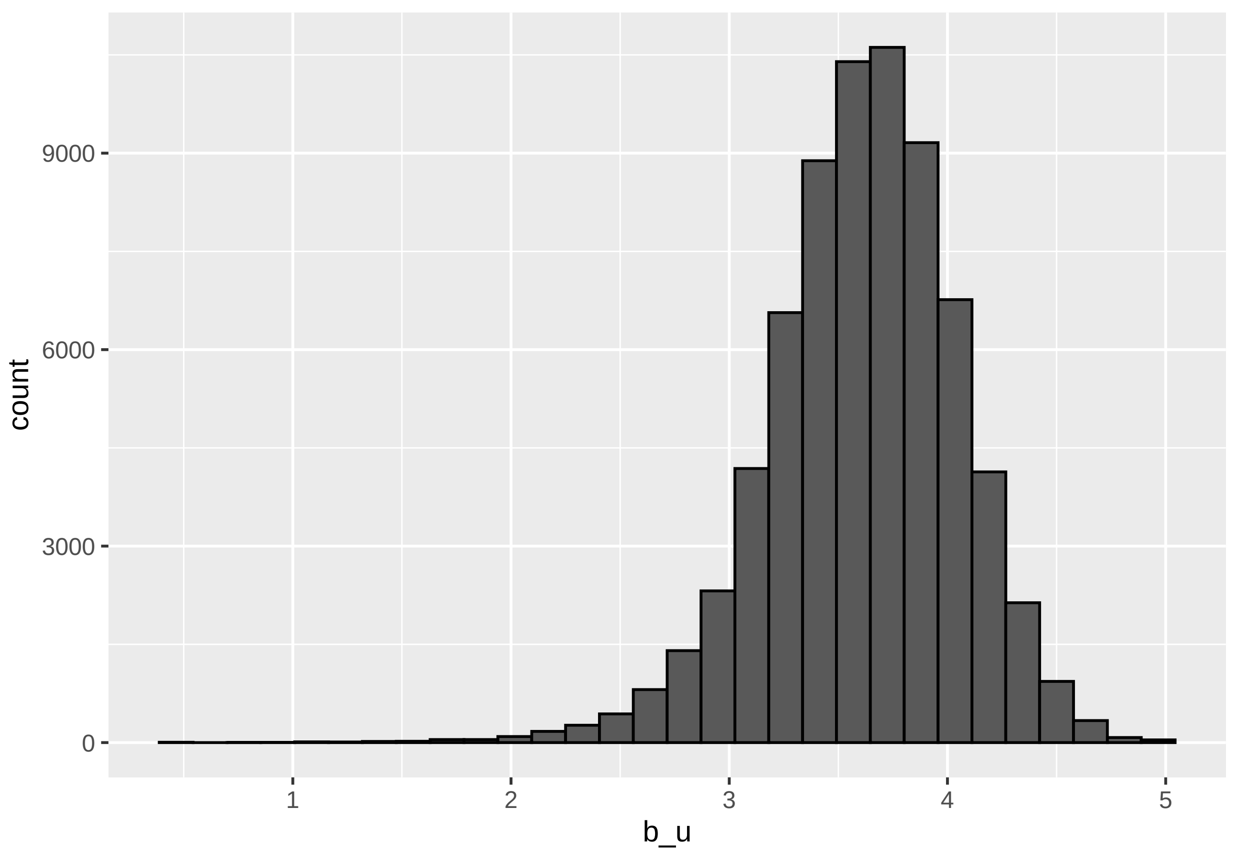
```
mu_hat <- mean(edx$rating)
movie_avgs <- edx %>%
     group_by(movieId) %>%
     summarize(b_i = mean(rating - mu_hat))

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



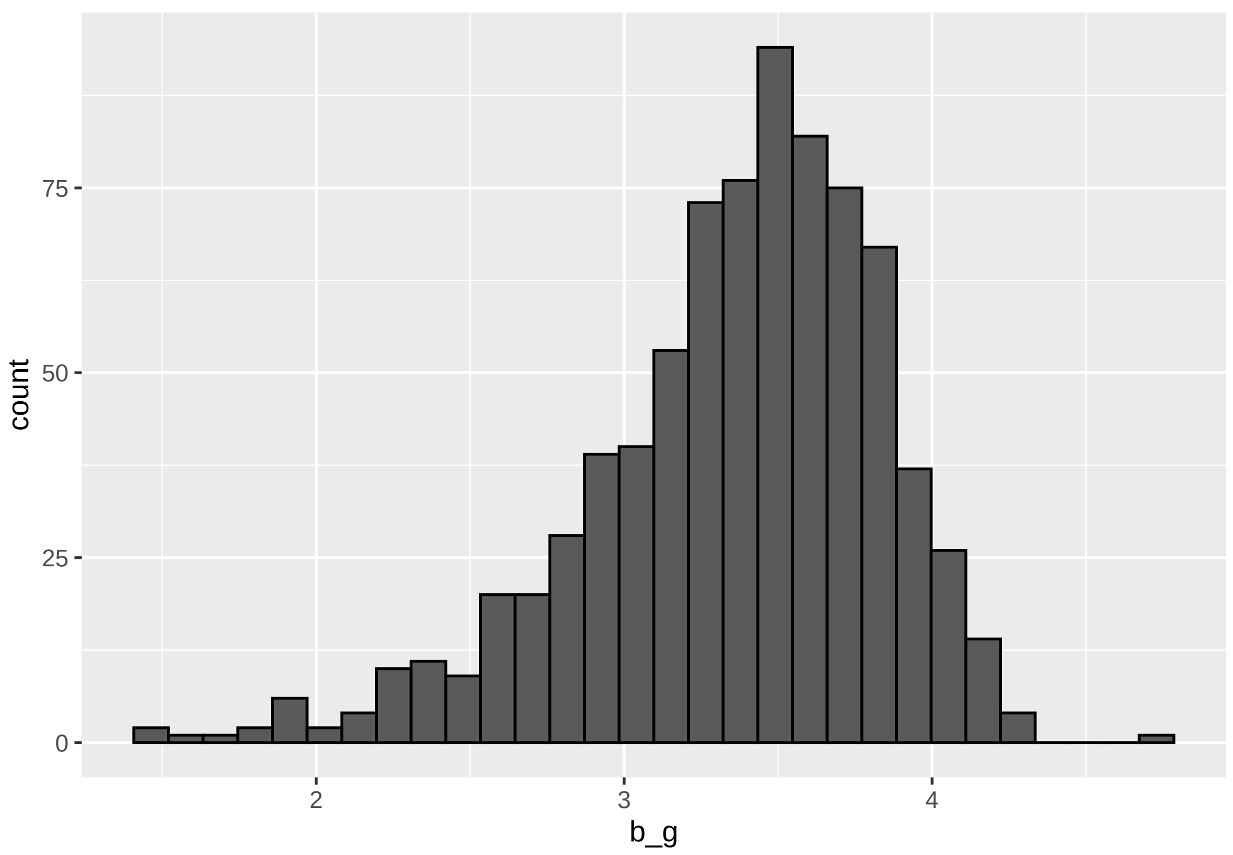
For users that have rated over 100 movies, we can plot their average rating accordingly. Observe how some users give greater ratings in general compared to others.

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



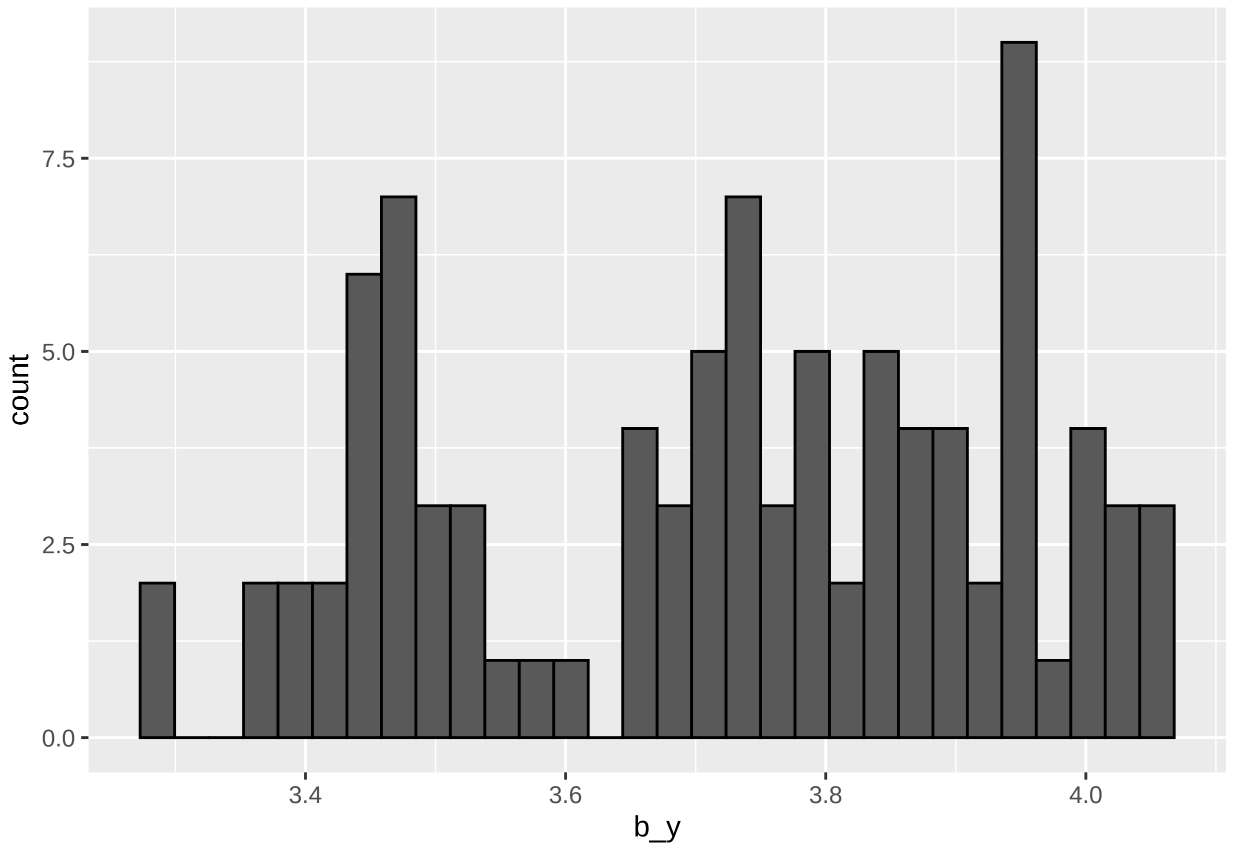
For genres, we compare the mean rating per genre as seen below. Clearly there is a difference in rating for different genre combinations.

```r
edx %>%
  group_by(genres) %>%
  summarize(b_g = mean(rating)) %>%
  ggplot(aes(b_g)) +
  geom_histogram(bins=30, color = "black")
```



Lastly, we look at different release years. We can see there is a difference in average rating compared to release year.

```
edx %>%
  group_by(year_release) %>%
  summarize(b_y = mean(rating)) %>%
  ggplot(aes(b_y)) +
  geom_histogram(bins=30, color = "black")
```



Hypothesis justification

The data visualization does not contradict the hypothesis, and we will therefore model based on the proposed biases.

Modelling

We start with defining a function for calculating the RMSE.

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

Mean value

Starting with the simplies possible method of predicting the mean value of all ratings and displaying the RMSE.

```r
mu_hat <- mean(training_set$rating)
mean_prediction <- RMSE(test_set$rating,mu_hat)
results_rmse <- data_frame(method = "Using the average for prediction" ,
                           RMSE=mean_prediction)
results_rmse %>% knitr::kable()
```

method	RMSE
Using the average for prediction	1.059904

Adding the movie bias term

As a next step, let's add the movie bias term to the prediction.

```
movie_avgs <- training_set %>% group_by(movieId) %>% summarize(bi=mean(rating - mu_hat))
test_set_with_bi <- test_set %>% left_join(movie_avgs, by="movieId")
predictions_bi <- test_set_with_bi$bi+mu_hat
bi_prediction <- RMSE(test_set$rating, predictions_bi)
results_rmse <- bind_rows(results_rmse, tibble(method="Adding movie bias",
                                               RMSE=bi_prediction))
results_rmse %>% knitr::kable()
```

method	RMSE
Using the average for prediction	1.0599043
Adding movie bias	0.9437429

Adding the user bias term

Let's now add the user bias term in order to improve the ratings.

```
user_avgs <- training_set %>% left_join(movie_avgs, by="movieId") %>%
  group_by(userId) %>% summarize(b_u=mean(rating-mu_hat-bi))
test_set_with_bi_bu <- test_set_with_bi %>% left_join(user_avgs, by="userId")
test_set_with_bu <- test_set %>% left_join(user_avgs, by="userId")
predictions_bu <- test_set_with_bi_bu$bi+test_set_with_bu$b_u+mu_hat
bu_predictions <- RMSE(test_set$rating, predictions_bu)
results_rmse <- bind_rows(results_rmse, tibble(method="Adding user bias",
                                               RMSE=bu_predictions))
results_rmse %>% knitr::kable()
```

method	RMSE
Using the average for prediction	1.0599043
Adding movie bias	0.9437429
Adding user bias	0.8659320

Cross-validation

In order to improve the rmse further, we will need to include cross-validation using both movie bias and user bias.

```
lambdas <- seq(3,10,0.5)
sum_movieId <- training_set %>% group_by(movieId) %>%
  summarize(s=sum(rating-mu_hat), n=n())
rmses <- sapply(lambdas, function(lambda){
```

```r
b_i_crossv <- training_set %>% group_by(movieId) %>%
  summarize(b_i_crossv=sum(rating-mu_hat)/(n()+lambda))
b_u_crossv <- training_set %>% left_join(b_i_crossv,by="movieId") %>%
  group_by(userId) %>%
  summarize(b_u_crossv=sum(rating-mu_hat-b_i_crossv)/(n()+lambda))
predictions <- test_set %>% left_join(b_i_crossv, by="movieId") %>%
  left_join(b_u_crossv, by="userId") %>%
  mutate(preds = b_i_crossv+b_u_crossv+mu_hat)

  return(RMSE(test_set$rating,predictions$preds))
})
results_rmse <- bind_rows(results_rmse,
                          tibble(method="Cross-valid. movie & user bias",
                                 RMSE=min(rmses)))
results_rmse %>% knitr::kable()
```

method	RMSE
Using the average for prediction	1.0599043
Adding movie bias	0.9437429
Adding user bias	0.8659320
Cross-valid. movie & user bias	0.8652421

To improve the rmse further, we will include the genre bias into the cross-validation.

```r
lambdas <- seq(3,10,0.5)
sum_movieId <- training_set %>% group_by(movieId) %>%
  summarize(s=sum(rating-mu_hat), n=n())
rmses <- sapply(lambdas, function(lambda){
  b_i_crossv <- training_set %>% group_by(movieId)%>%
    summarize(b_i_crossv=sum(rating-mu_hat)/(n()+lambda))
  b_u_crossv <- training_set %>% left_join(b_i_crossv,by="movieId") %>%
    group_by(userId) %>%
    summarize(b_u_crossv=sum(rating-mu_hat-b_i_crossv)/(n()+lambda))
  b_g_crossv <- training_set %>% left_join(b_i_crossv,by="movieId") %>%
    left_join(b_u_crossv, by="userId") %>% group_by(genres) %>%
    summarize(b_g_crossv=sum(rating-mu_hat-b_i_crossv-b_u_crossv)/(n()+lambda))
  predictions <- test_set %>% left_join(b_i_crossv, by="movieId") %>%
    left_join(b_u_crossv, by="userId") %>%
    left_join(b_g_crossv, by="genres") %>%
    mutate(preds = b_i_crossv+b_u_crossv+b_g_crossv+mu_hat)

  return(RMSE(test_set$rating,predictions$preds))
})
results_rmse <- bind_rows(results_rmse,
                          tibble(method="Cross-valid. movie, user & genre bias", RMSE=min(rmses)))
results_rmse %>% knitr::kable()
```

method	RMSE
Using the average for prediction	1.0599043
Adding movie bias	0.9437429

method	RMSE
Adding user bias	0.8659320
Cross-valid. movie & user bias	0.8652421
Cross-valid. movie, user & genre bias	0.8649406

Now let's include the release year bias into the cross-validation to see if we can improve it even further.

```
lambdas <- seq(3,10,0.5)
sum_movieId <- training_set %>% group_by(movieId) %>%
  summarize(s=sum(rating-mu_hat), n=n())
rmses <- sapply(lambdas, function(lambda){
  b_i_crossv <- training_set %>% group_by(movieId)%>%
    summarize(b_i_crossv=sum(rating-mu_hat)/(n()+lambda))
  b_u_crossv <- training_set %>% left_join(b_i_crossv,by="movieId") %>%
    group_by(userId) %>%
    summarize(b_u_crossv=sum(rating-mu_hat-b_i_crossv)/(n()+lambda))
  b_g_crossv <- training_set %>% left_join(b_i_crossv,by="movieId") %>%
    left_join(b_u_crossv, by="userId") %>% group_by(genres) %>%
    summarize(b_g_crossv=sum(rating-mu_hat-b_i_crossv-b_u_crossv)/(n()+lambda))
  b_y_crossv <- training_set %>% left_join(b_i_crossv,by="movieId") %>%
    left_join(b_u_crossv, by="userId") %>% left_join(b_g_crossv,
                                                     by="genres") %>%
    group_by(year_release) %>%
    summarize(b_y_crossv=sum(rating-mu_hat-b_i_crossv-b_u_crossv-b_g_crossv)/
                (n()+lambda))
  predictions <- test_set %>% left_join(b_i_crossv, by="movieId") %>%
    left_join(b_u_crossv, by="userId") %>%
    left_join(b_g_crossv, by="genres") %>%
    left_join(b_y_crossv, by="year_release") %>%
    mutate(preds = b_i_crossv+b_u_crossv+b_g_crossv+b_y_crossv+mu_hat)

  return(RMSE(test_set$rating,predictions$preds))
})
results_rmse <- bind_rows(results_rmse,
                          tibble(method="Cross-valid. all biases", RMSE=min(rmses)))
results_rmse %>% knitr::kable()
```

method	RMSE
Using the average for prediction	1.0599043
Adding movie bias	0.9437429
Adding user bias	0.8659320
Cross-valid. movie & user bias	0.8652421
Cross-valid. movie, user & genre bias	0.8649406
Cross-valid. all biases	0.8647970

Results

To validate our trained model, we will use the so far untouched validation set for the final rmse.

```r
lambdas <- lambdas[which.min(rmses)] #Selects the optimal lambda
rmses <- sapply(lambdas, function(lambda){
  mu_hat <- mean(edx$rating)
  b_i_crossv <- edx %>% group_by(movieId) %>%
    summarize(b_i_crossv=sum(rating-mu_hat)/(n()+lambda))
  b_u_crossv <- edx %>% left_join(b_i_crossv,by="movieId") %>%
    group_by(userId) %>%
    summarize(b_u_crossv=sum(rating-mu_hat-b_i_crossv)/(n()+lambda))
  b_g_crossv <- edx %>% left_join(b_i_crossv,by="movieId") %>%
    left_join(b_u_crossv, by="userId") %>% group_by(genres) %>%
    summarize(b_g_crossv=sum(rating-mu_hat-b_i_crossv-b_u_crossv)/(n()+lambda))
  b_y_crossv <- edx %>% left_join(b_i_crossv,by="movieId") %>%
    left_join(b_u_crossv, by="userId") %>%
    left_join(b_g_crossv, by="genres") %>%
    group_by(year_release) %>%
    summarize(b_y_crossv=sum(rating-mu_hat-b_i_crossv-b_u_crossv-b_g_crossv)
              /(n()+lambda))
  predictions <- validation %>% left_join(b_i_crossv, by="movieId") %>%
    left_join(b_u_crossv, by="userId") %>%
    left_join(b_g_crossv, by="genres") %>%
    left_join(b_y_crossv, by="year_release") %>%
    mutate(preds = b_i_crossv+b_u_crossv+b_g_crossv+b_y_crossv+mu_hat)

  return(RMSE(predictions$rating,predictions$preds))
})
results_rmse <- bind_rows(results_rmse, tibble(method="Validation set",
                                               RMSE=rmses))
results_rmse %>% knitr::kable()
```

method	RMSE
Using the average for prediction	1.0599043
Adding movie bias	0.9437429
Adding user bias	0.8659320
Cross-valid. movie & user bias	0.8652421
Cross-valid. movie, user & genre bias	0.8649406
Cross-valid. all biases	0.8647970
Validation set	0.8644662

Conclusion

The final model using cross-validation for movie, user, genre and release year bias was found to be minimizing the RMSE, yielding a RMSE of:

```r
results_rmse[dim(results_rmse)[1],dim(results_rmse)[2]]
```

```
## # A tibble: 1 x 1
##    RMSE
##   <dbl>
## 1 0.864
```

Starting with the baseline approach of just predicting the mean rating resulted in an RMSE greater than 1, meaning ratings are missed with over one star. Including biases and regularization in order to penalize large effects coming from small sample sizes, we were able to lower the RMSE substantially.

A further development of the model could include an investigation of matrix factorization on the final result, accounting for the fact that groups of movies might have similar rating patters, and the same for groups of users [2].

References

1. Irizarry, R. Introduction to Data Science. (2020). Section 33.9 Regularization. Available from: https://rafalab.github.io/dsbook/large-datasets.html#user-effects
2. Irizarry, R. Introduction to Data Science. (2020). Section 33.11 Matrix factorization. Available from: https://rafalab.github.io/dsbook/large-datasets.html#user-effects