MovieLens Project

Dennis Sim

March 1, 2019

1 Introduction

1.1 Data set

This project uses the MovieLens data set that contains 10,000,054 ratings and 95,580 tags applied to 10,681 movies by 71,567 users of the online movie recommender service. Each randomly selected user is associated with an id and have rated at least 20 movies. More information about this data set is described in MovieLens 10M/100k Data Set README.

1.2 Project goal summary

The project aims to help users look for movie recommendations by making use of this data set to predict the rating of a movie by a given user based on ratings of similar movies and user groups.

1.3 Key steps performed

The data set is first partitioned into edx and validation data sets using codes provided in the project instructions with some modifications to extract movie release year from the movie title and install/load additional libraries.

The *edx* data set is then further splitted into a training set (90% *edx_train*) and a test set (10% *edx_test*). This *edx_test* set is used for experimenting with different models by comparing the residual mean squared error (RMSE) of each model that are build in this project.

```
# split edx into train/test sets - to experiment with different model
edx_test_ratio <- 0.1
edx_test_index <- createDataPartition(y = edx$rating, times = 1,
                                      p = edx_test_ratio, list = FALSE)
edx_train <- edx[-edx_test_index,]
temp_test <- edx[edx_test_index,]

# make sure userId and movieId in edx_test set are also in edx_train set
edx_test <- temp_test %>%
  semi_join(edx_train, by = "movieId") %>%
  semi_join(edx_train, by = "userId")

# add rows removed from edx_test set back into edx_train set
edx_removed <- anti_join(temp_test, edx_test)
edx_train <- rbind(edx_train, edx_removed)

# remove temp data
rm(temp_test, edx_test_index, edx_removed)
```

Before embarking on model building, we explore the data to see if the data set requires further clean up and whether there is variability in the movie ratings across all movies and also ratings given by each user in the user population. The next section will discuss this in detail.

2 Methods/Analysis

This section describes the process and techniques used for data cleaning, data exploration and visualisation, insights gained, and discusses different modelling approaches.

2.1 Data cleaning

The *movielens* data frame is a combination of users and movie ratings. A quick look at the *movielens* data set shows each row presents the movie rating given by a particular user for the movie.

```
##   userId movieId rating timestamp                         title
## 1      1     122      5 838985046              Boomerang (1992)
## 2      1     185      5 838983525               Net, The (1995)
## 3      1     231      5 838983392          Dumb & Dumber (1994)
## 4      1     292      5 838983421               Outbreak (1995)
## 5      1     316      5 838983392               Stargate (1994)
## 6      1     329      5 838983392 Star Trek: Generations (1994)
##                         genres movie_year
## 1                Comedy|Romance       1992
## 2          Action|Crime|Thriller       1995
## 3                        Comedy       1994
## 4  Action|Drama|Sci-Fi|Thriller       1995
## 5       Action|Adventure|Sci-Fi       1994
## 6 Action|Adventure|Drama|Sci-Fi       1994
```

The users in the data set should contain only those who have rated at least 20 movies - as shown in the output below with user rating counts sorted in ascending order.

```
## # A tibble: 6 x 2
##   userId user_rating_count
##    <int>             <int>
## 1      2                20
## 2     59                20
## 3     71                20
## 4     74                20
## 5    132                20
## 6    134                20
```

Although there are 4 movies that are not rated, they are not included in the data set to be used for building and testing the models. It is also a small number compared the size of the data set and should not affect the findings.

```
##   movieId                                                          title
## 1   25942                                          Louisiana Story (1948)
## 2   60566 Just Another Love Story (K<U+00E6>rlighed p<U+00E5> film) (2007)
```

```
## 3   62669                                  Black River (Kuroi kawa) (1957)
## 4   64959                      Divide and Conquer (Why We Fight, 3) (1943)
##                  genres
## 1                 Drama
## 2 Crime|Drama|Thriller
## 3                 Drama
## 4      Documentary|War
```

There is no empty or unknown ratings in the data set that is to be used for training, testing and validation.

```
sum(is.na(movielens$rating))
```

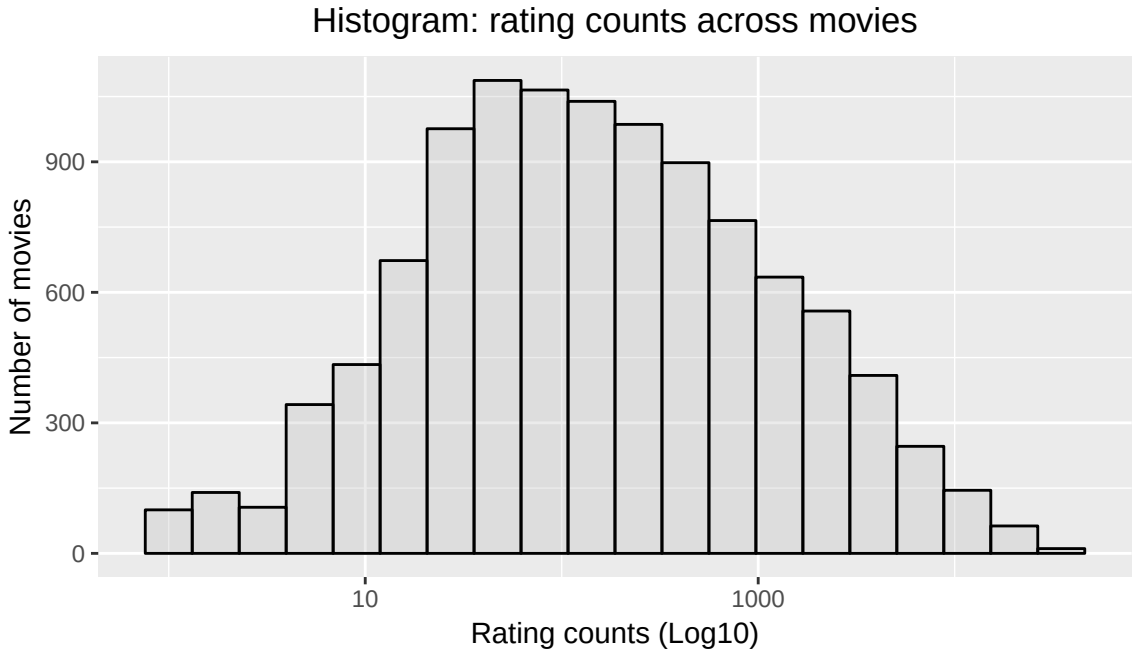
```
## [1] 0
```

The data set appears to be in good shape and does not require further cleaning for the purpose of this project.

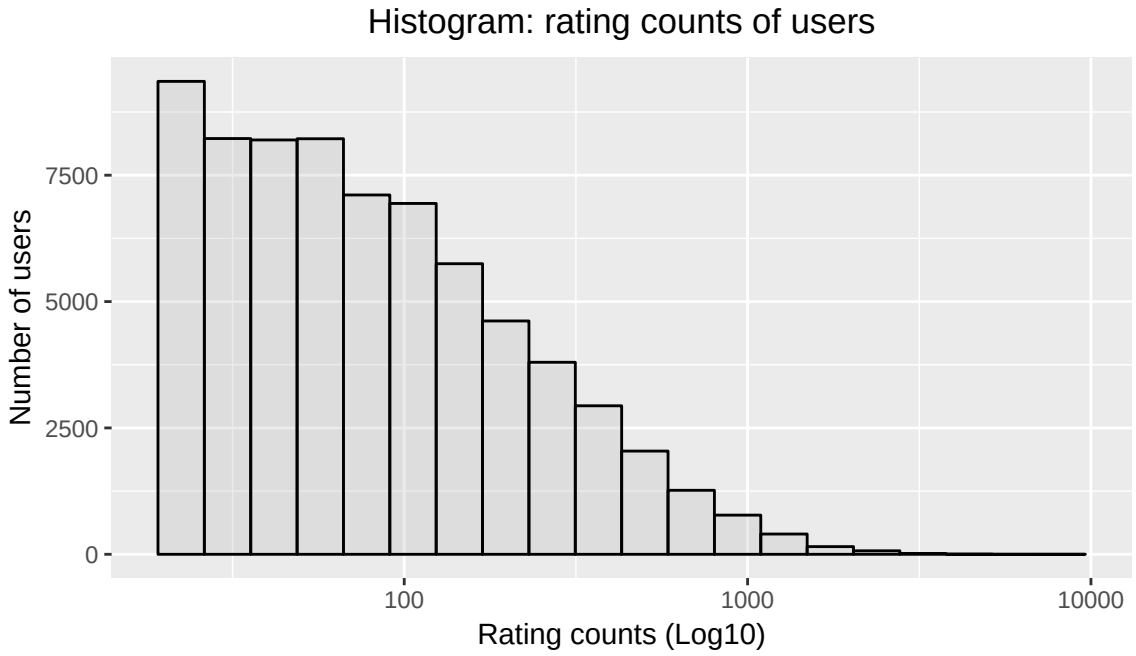
2.2 Data exploration and visualisation

We move on to explore the data set and hope to gain some insights of the data before building the models.

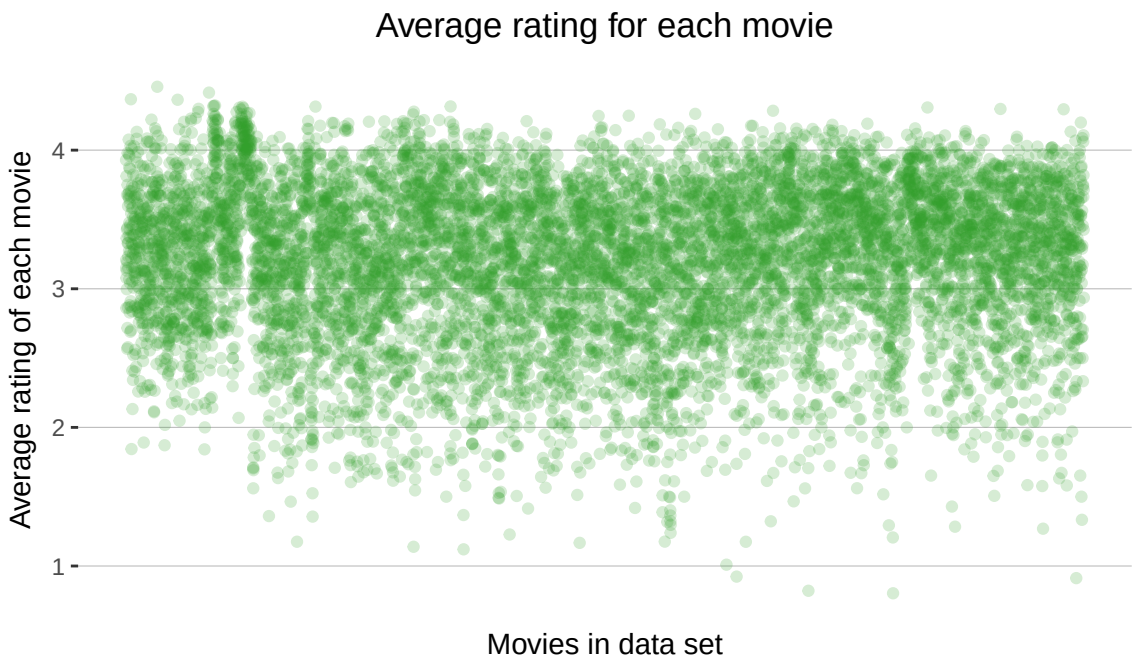
The following histogram shows some movies get rated more than others. This is likely the case for blockbusters that are watched by many users and independent films that are only watched by a few users.



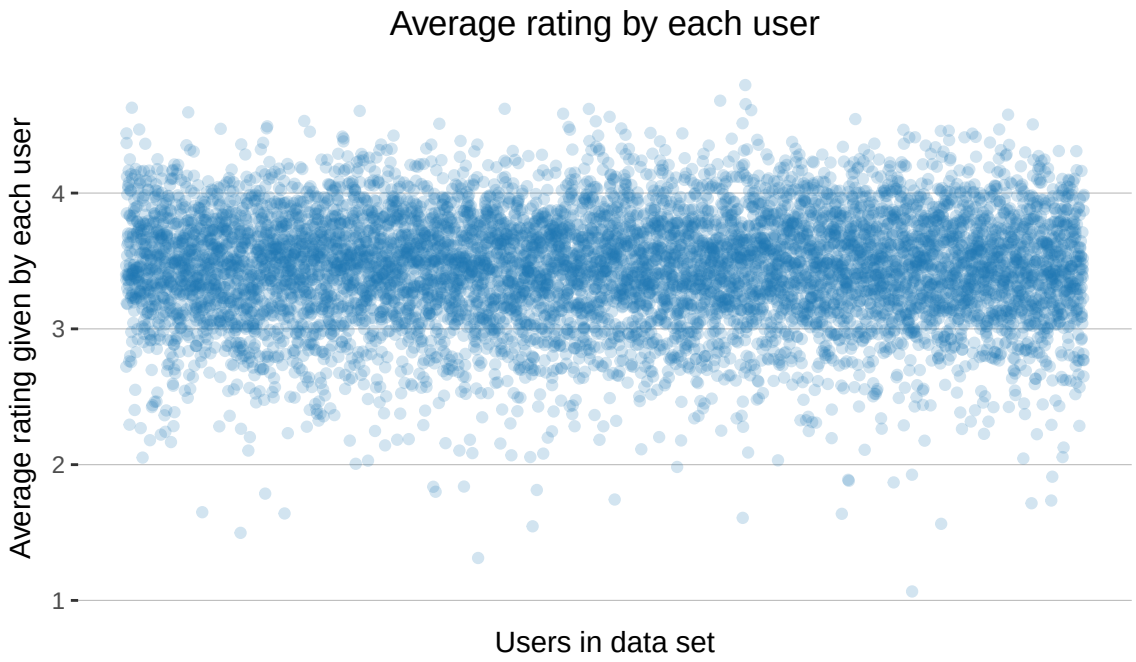
The following histogram shows user activity in rating the movies. Some users rated over a thousand movies; with most users rated in hundreds or less; and some users only rated a few movies.



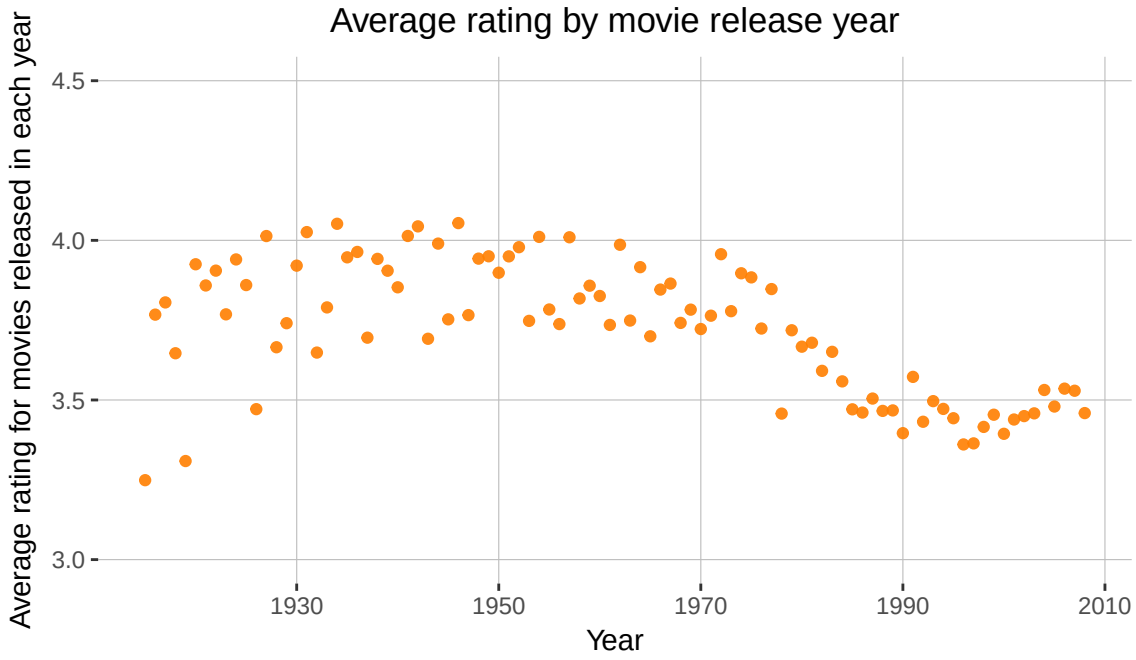
Each movie appear to have a different average rating. The plot of average rating for each movie shows each movie in the data set has a different average rating.



The user population also displays different rating behaviours. The plot below shows different users rate movies differently - some with high average ratings, others with low average ratings and most users somewhere in between.



We also see that the average rating differs across the years in which the movie is released, as shown by the following plot.



2.3 Insights gained

The plots in the previous section show that there is variability in ratings across movies; different user rating behaviours in the user population and varying average rating over the years which the movies are released. These could have effects on the rating of movies. We will investigate whether these variabilities have any effects on movie rating prediction by introducing these effects to the different models we build in this project.

2.4 Modeling approach

We need to quantify what metrics need to do well as a basis to compare between different models and RMSE (residual means squared error) is chosen for this purpose.

A RMSE function is defined as follows:

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

We first build a simple model whereby the same rating is predicted for all movies, regardless of users and movies. In this case, that is the mean of all ratings in the training set. We will use the *edx_train* and *edx_test* data sets for training and test purpose at this stage.

The average rating of all movies across all users is

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

```
## [1] 3.512509
```

To see how well this model perform, we look at how the prediction model predicts the outcome on the test data set. That is predicting all ratings of the test data set with this average.

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

```
## [1] 1.061135
```

The result of predicting the same average rating for all movies shows a pretty large RMSE (1.061135), which does not suggest a very good prediction.

In fact, if we simply predict any other ratings, the RMSE would be even larger as shown below.

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

```
## [1] 1.466388
```

```
predictions <- rep(2, nrow(edx_test))
RMSE(edx_test$rating, predictions)
```

```
## [1] 1.847259
```

A table is used to store this result and also results for the different models that we will continue to build and discuss later.

```
rmse_results <- data_frame(method = "Predicting the average", RMSE = naive_rmse)
kp <- 5
rmse_results %>% knitr::kable(padding=kp)
```

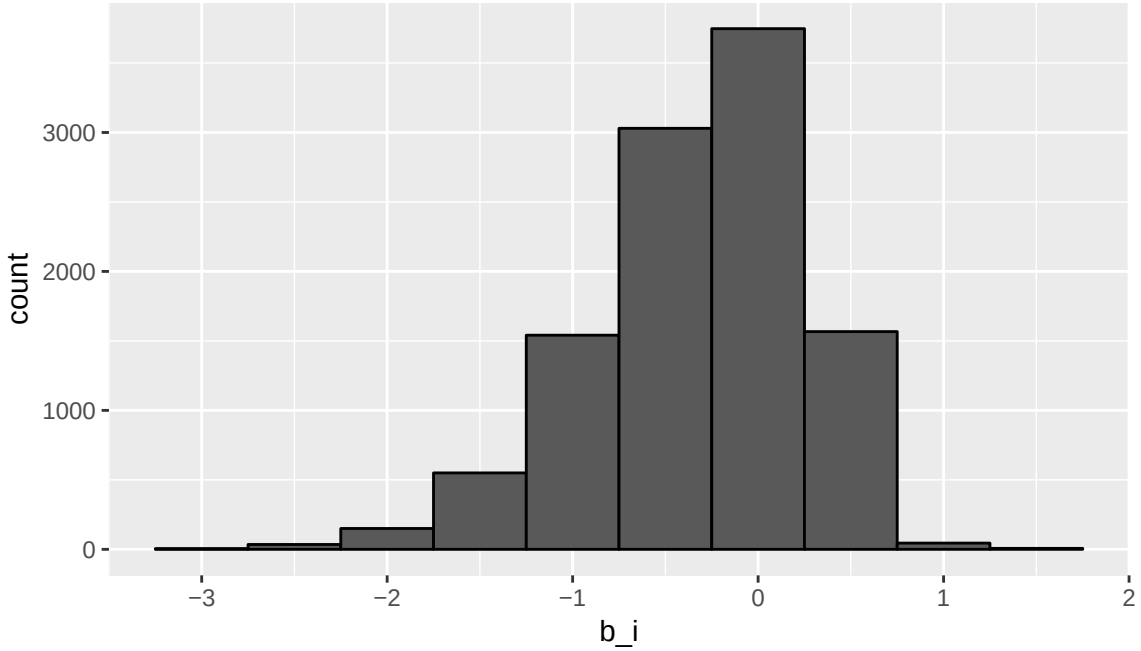
method	RMSE
Predicting the average	1.061135

In the previous section where we explore and visualise the data set, we know that each movie has a different average rating. So, we can augment the first model by adding the average rating of each movie to each movie's rating prediction (with each movie's rating - denoted by b_i that represents average rating for movie i, also known as effects in statistics).

```
# fit <- lm(rating ~ as.factor(userId), data = movielens)
# lm function will take some time
# we use this instead for each movie i: b_i = y(u,i) - overall mean
mu <- mean(edx_train$rating)
movie_avgs <- edx_train %>%
  group_by(movieId) %>%
  summarize(b_i = mean(rating - mu))
```

The following plot shows the estimates varies substantially.

```
# the estimates varies substantially as shown in this plot
movie_avgs %>% qplot(b_i, geom ="histogram", bins = 10, data = .,
                     color = I("black"))
```




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

model_1_rmse <- RMSE(predicted_ratings, edx_test$rating)
rmse_results <- bind_rows(rmse_results,
                          data_frame(method="Movie Effect Model",
                                     RMSE = model_1_rmse ))
rmse_results %>% knitr::kable(padding=kp)
```

method	RMSE
Predicting the average	1.0611350

method	RMSE
Movie Effect Model	0.9441568

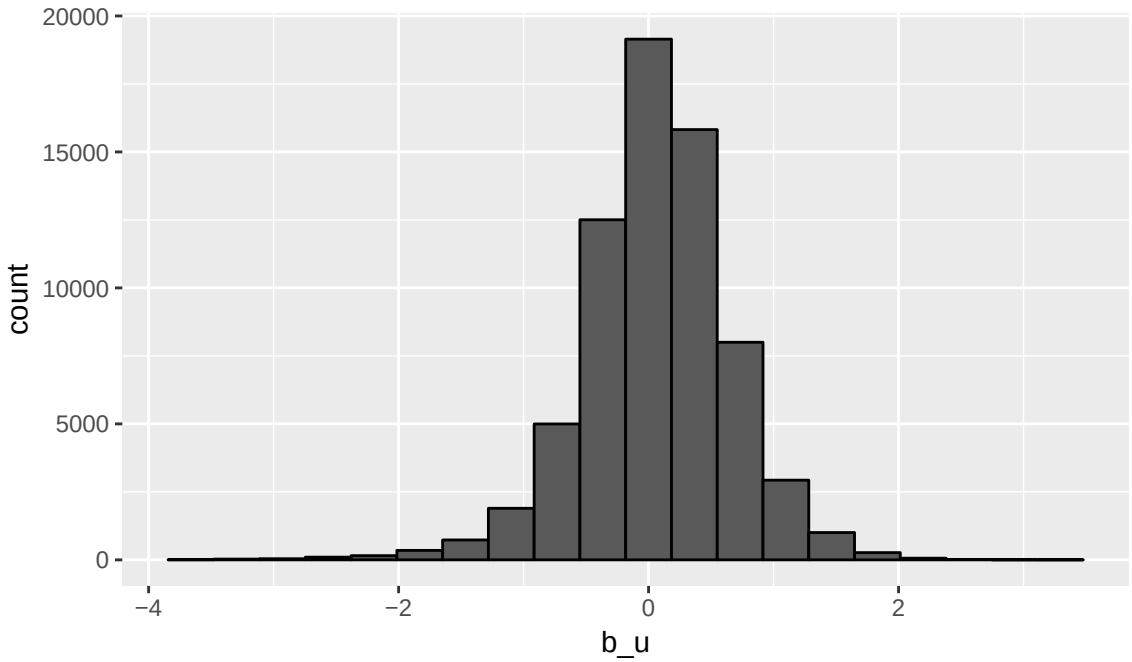
Comparing the RMSE from the first simple model against the model augmented with movie effects clearly shows that the model with movie effect has a slightly better prediction, supported by a smaller RMSE (0.9441568).

We also know that different users display different behaviours in rating the movies. We augment the model again by adding user-specific effects (denoted by b_u in the formula)

```
# try out: lm on movie+user effects
# lm(rating ~ as.factor(movieId) + as.factor(userId))
# lm will take some time to run;
# we use this instead: b_u = y(u,i) - overall mean - b_i
user_avgs <- edx_test %>%
  left_join(movie_avgs, by='movieId') %>%
  group_by(userId) %>%
  summarize(b_u = mean(rating - mu - b_i))
```

The plot below shows the estimates varies substantially.

```
# the estimates varies substantially as shown in this plot
# this suggests variability in average rating across different users
user_avgs %>% qplot(b_u, geom ="histogram", bins = 20, data = .,
                    color = I("black"))
```




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

method	RMSE
Predicting the average	1.0611350
Movie Effect Model	0.9441568
Movie + User Effects Model	0.8262583

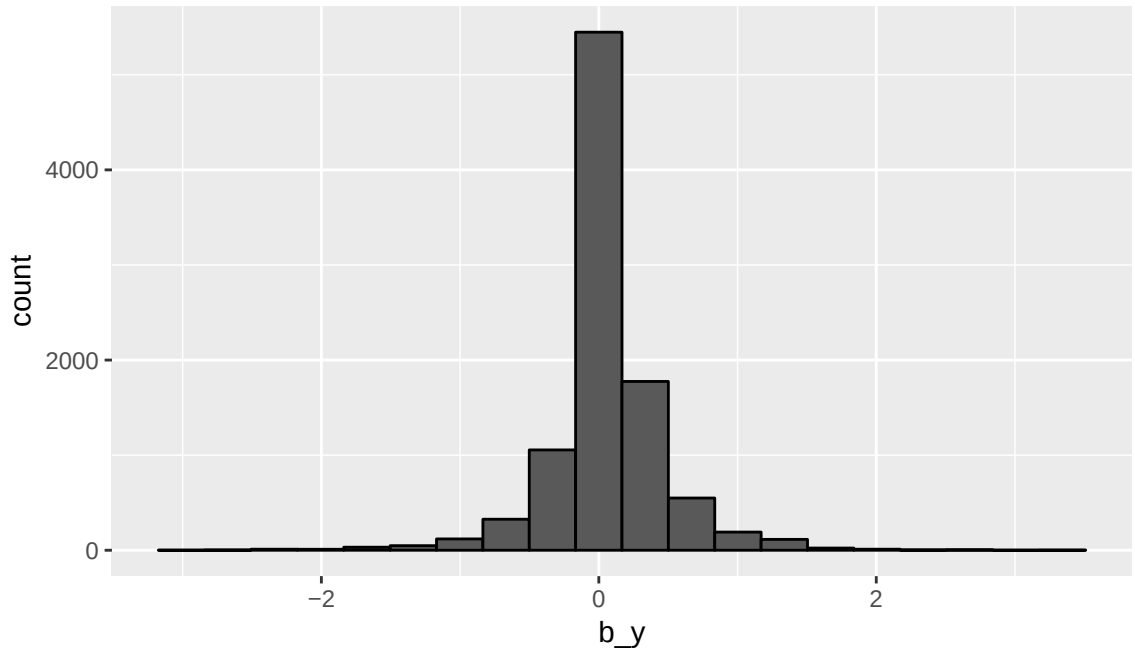
We now see the RMSE of the model with movie and user effects dropped even further to 0.8262583, suggesting improvement in this prediction model.

Findings from data exploration and visualisation suggest that the year the movie is released can have an effect on the movie ratings. We further augment the model to add year effects (denoted by b_y in the formula).

```
# try out: lm on movie+user+year effects
# lm(rating ~ as.factor(movieId) + as.factor(userId) + as.factor(movie_year))
year_avgs <- edx_test %>%
  left_join(movie_avgs, by='movieId') %>%
  left_join(user_avgs, by='userId') %>%
  group_by(movieId) %>%
  summarize(b_y = mean(rating - mu - b_i - b_u))
```

The plot below shows the estimates varies substantially.

```
# the estimates varies substantially as shown in this plot
# this suggests variability in average rating across different
# years in which the movie is released
year_avgs %>% qplot(b_y, geom ="histogram", bins = 20, data = .,
                    color = I("black"))
```




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

model_3_rmse <- RMSE(predicted_ratings, edx_test$rating)
rmse_results <- bind_rows(rmse_results,
                          data_frame(method="Movie + User + Year Effects Mode",
                                     RMSE = model_3_rmse ))
rmse_results %>% knitr::kable(padding=kp)
```

method	RMSE
Predicting the average	1.0611350
Movie Effect Model	0.9441568
Movie + User Effects Model	0.8262583
Movie + User + Year Effects Mode	0.8197169

The RMSE of the model that includes movie, user-specific and year effects (0.8197169) is less compared to the previous models - suggesting improvement to the prediction model.

3 Results

3.1 Test findings

The above experiment uses training and test data sets splitted from the edx data set (as *edx_train* 90% and *edx_test* 10%). The results from training different models with *edx_train* data set, and

testing with *edx_test* data set shows that predicting the just average has a RSME of 1.061135.

method	RMSE
Predicting the average	1.0611350
Movie Effect Model	0.9441568
Movie + User Effects Model	0.8262583
Movie + User + Year Effects Mode	0.8197169

The simple model improved when we introduced the movie effect. This brings the RMSE slightly down to 0.9441568, as each movie has a different average rating across all movies.

We also observe from the plots that different users give different ratings to movies; and we augmented the model with user effect. This brings the prediction further down to 0.8262583, suggesting an even better prediction than the previous two models.

As we continue to augment the model with year effect, it brings the RMSE even lower (0.8197169) that suggests closer prediction by this model.

3.2 Validation

After experimenting with the different models on the *edx* data set, we repeated the predictions and compared the results, this time using all data in *edx* data set for training and validated all models against the data in the *validation* data set.

The RMSE of the different models train using *edx* data set and tested against the *validation* data set is presented below.

method	RMSE
Predicting the average	1.0612018
Movie Effect Model	0.9439087
Movie + User Effects Model	0.8292477
Movie + User + Year Effects Model	0.8232682

Results from both sets of training and test data sets are close (train with *edx_train* and test with *edx_test*; train with *edx* and test with *validation*). This suggests that the models are consistent across these data sets.

The final model that includes movie, user-specific and year effects has RMSE of 0.8232682.

4 Conclusion

The findings suggest that augmenting models with additional predictors with significant variability can help in building more accurate model as demonstrated by adding movie effects, user-specific effects and year effects to the simple model that was initially constructed.