MovieLens Project

Anton Preto

April 2023

Executive Summary

This project is part of the HarvardX PH125.9x Data Science: Capstone online course.
The aim of the project is to build movie recommendation system using machine learning by creating of a model with sufficient quality in predicting movie. Sufficient quality is defined as a Residual Mean Square Error (RMSE) lower than 0.86490.
To create the model we are exploring different approaches and adding biases (movie, user) that has impact on the quality (RMSE) of the model. We are using the 10M version of the MovieLens dataset to make the computation a little easier.
We are training a machine learning algorithm using the inputs in one subset to predict movie ratings in the final holdout test set.
After comparing 4 different models we were able to find the most suitable model that has sufficient RMSE. The model is using regularization and penalized least squares to predict ratings. Model applied to final test dataset has RMSE of 0.86482.
We were able to fulfill the aim of the project.

Dataset and Methods

Dataset used in this project is the 10M version of the MovieLens by GroupLens and can be downloaded here.
For computations we are using R version 4.2.2 (The R Project for Statistical Computing) and RStudio Desktop (Open Source Edition AGPL v3) with appropriate packages. We are using code provided by the course to generate initial datasets. To develop our algorithms we are using the edx set. For a final test of our final algorithm, we predict movie ratings in the final_holdout_test.

In the next step we are splitting the edx dataset into separate training (90%) and test (10%) dataset. Typical choices are to use 10% - 20% of the data for testing. We are training our model and testing RMSEs using edx training and edx test dataset.

To show rating per Genre and Year in edx dataset we need to subtract the Information from "genres" column (separated by "|") and Year from "title" column.

After creating datasets we need to inspect them. We see main categories. We are mostly interested ing category (column) "rating".

```
##   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
```

Together there is 10 000 046 ratng observations in edx (9 000 047) and final_holdout_test (999 999) dataset. Median value of rating in edx dataset used for building model is 4 and average is 3.51.

```
length(edx$rating) + length(final_holdout_test$rating)
```

```
## [1] 10000046
```

```
median(edx$rating)
```

```
## [1] 4
```

```
mean(edx$rating)
```

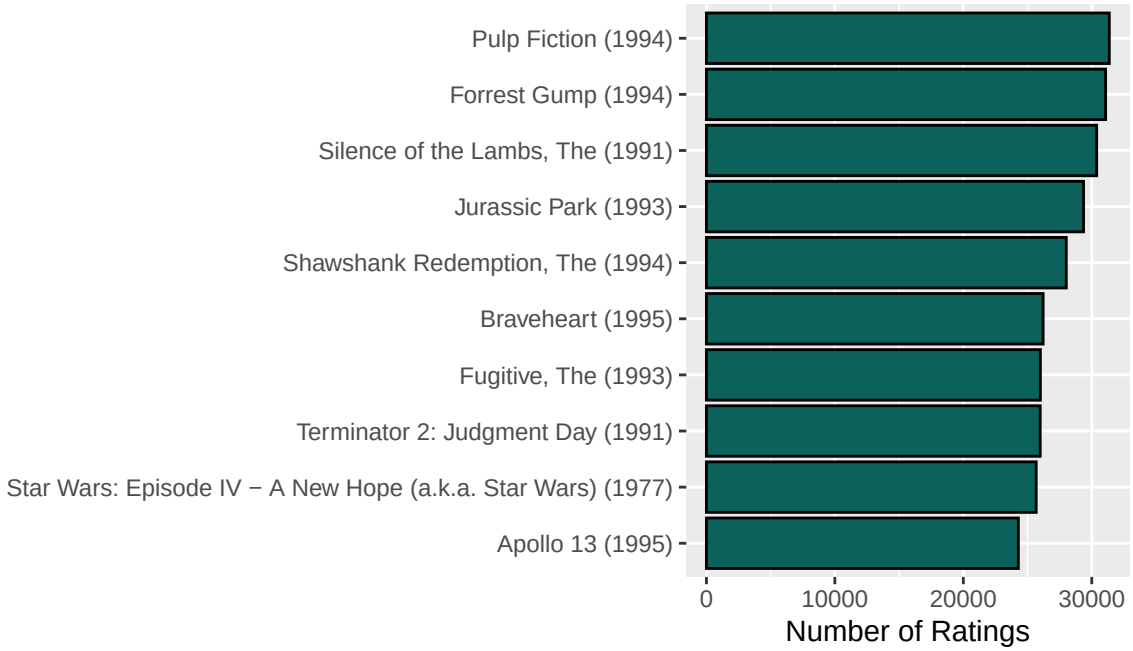
```
## [1] 3.5125
```

In edx dataset there is 10 669 unique movies and 69 878 users.

```
edx %>% summarize(number_movies = n_distinct(movieId), number_users = n_distinct(userId))
```

```
##   number_movies number_users
## 1         10669        69878
```

Most rated movie is Pulp Fiction followed by Forrest Gump, both aired in 1994. Looks like people like to watch and rate movies from 1990s. Pulp Fiction was rated 31 362 times.



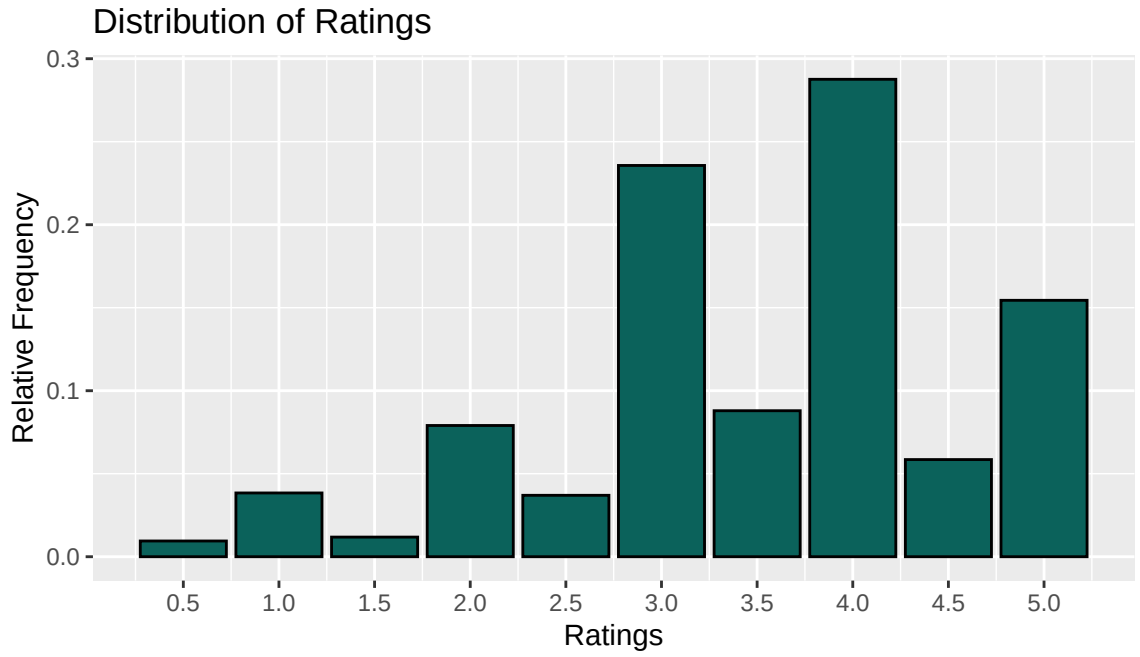
From everyday life we already know that people have distinct taste in movies and they can rate them differently according to their taste, genre liking, mood etc.
Some genres are rated more often. Most rated genre is Drama.

```
## # A tibble: 10 x 2
##    genres      count
##    <chr>       <int>
##  1 Drama     3910124
##  2 Comedy    3540928
##  3 Action    2560544
##  4 Thriller  2325897
##  5 Adventure 1908892
##  6 Romance   1712100
##  7 Sci-Fi    1341183
##  8 Crime     1327715
##  9 Fantasy    925637
## 10 Children   737993
```

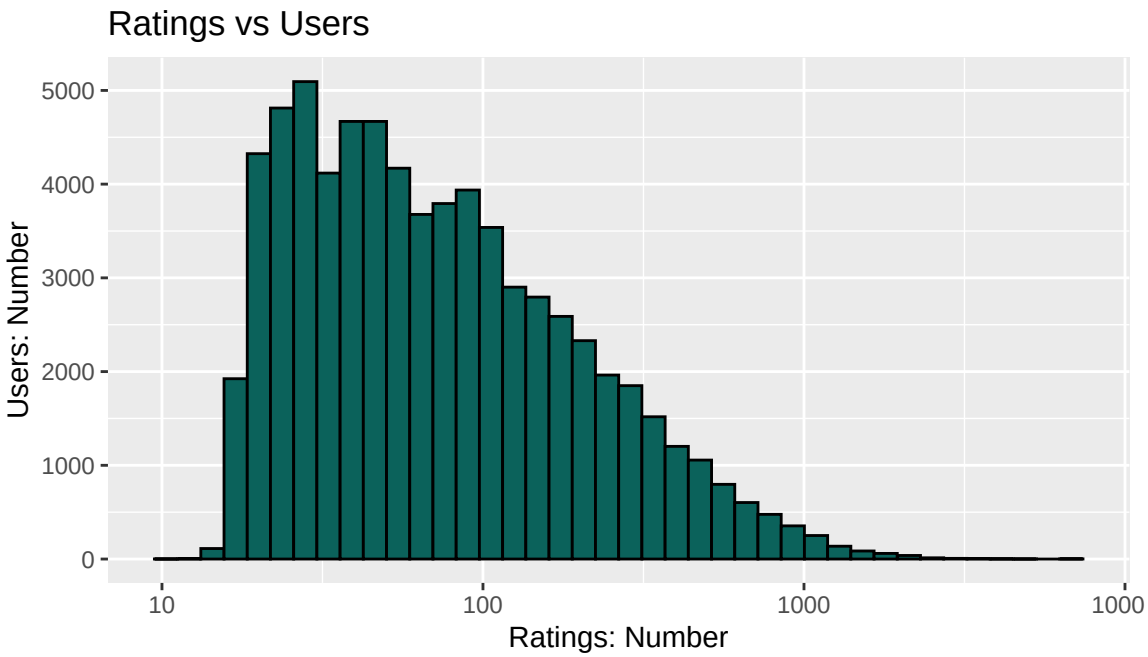
That also means that some people use whole number ratings more often then decimal ratings.
79.5% of ratings are whole numbers. Distribution of whole numbers shows that most common rating is 4 (2 588 429 ratings) followed by 3 (2 121 238 ratings).

```
sum(edx$rating %% 1 == 0) / length(edx$rating)
```

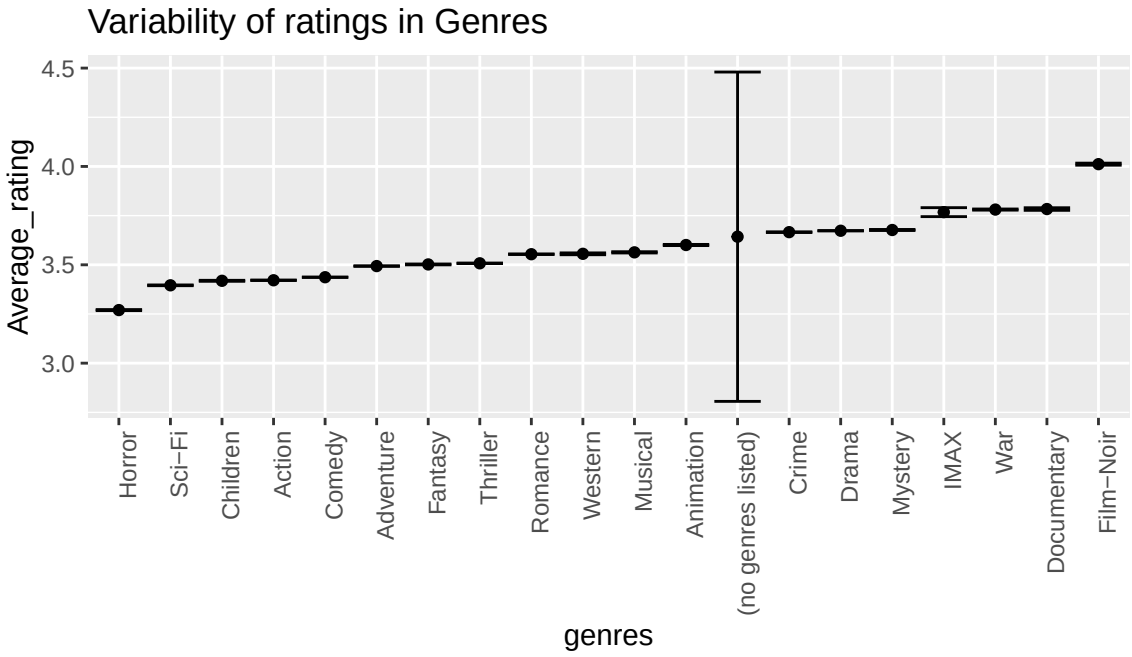
```
## [1] 0.7952
```



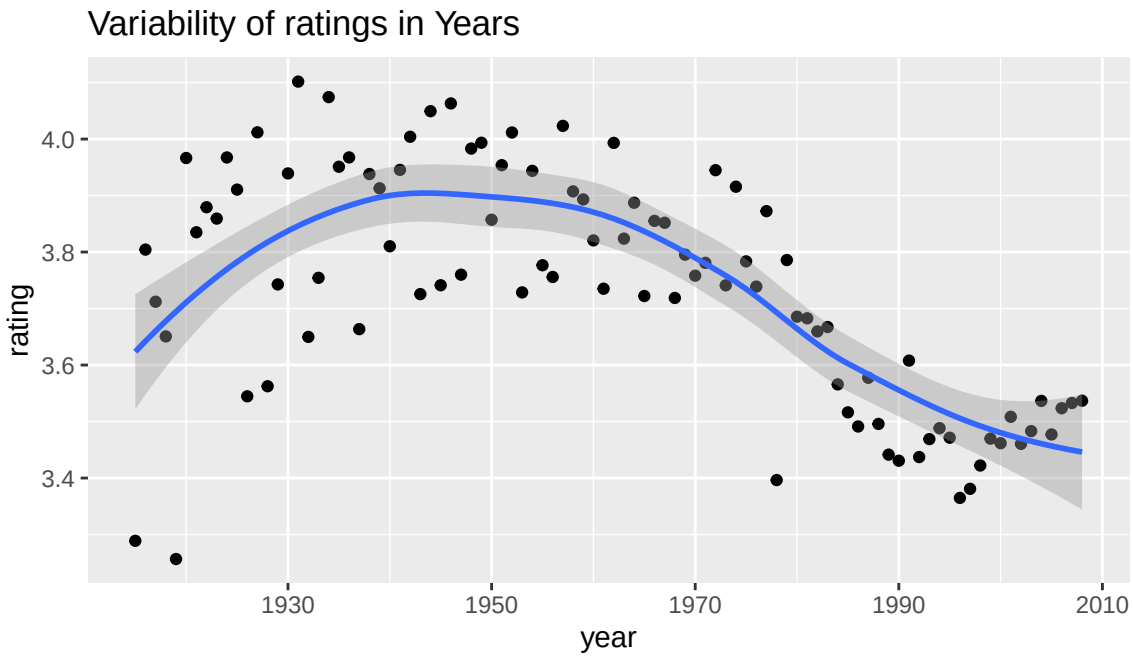
Using distribution graph we can see that the data are skewed, that means that not every person rate same amount of movies. A lot of people rate less than 100 times.



There is variability of ratings in genres. For example, Film-Noir is a very specific genre, has highest average rating among genres, however was rated only 118 541 times (in comparison with Drama that has the highest number of ratings and the average rating is around 3.65).



We can see that the golden age of movies (according to movie ratings by users amd by year first aired) was between 1940 and 1950 with the average rating of 3.9 out of 5. There is also steep decline of average ratings after 1960.



Building movie recommendation system

We are building movie recommendation system from using the simplest to the more advanced techniques and methods. We are comparing systems (models) and its predicting quality with Residual Mean Square Error (RMSE). The RMSE is then defined as:

$$RMSE = \sqrt{\frac{1}{N}\sum_{u,i}(\hat{y}_{u,i} - y_{u,i})^2}$$

It is the typical error we make when predicting a movie rating. If this number is larger than 1, it means our typical error is larger than one star, which is not good.
We are using edx, edx test and edx train dataset created.

Model 1: using only average (mean) of values

Simplest possible Recommendation System. We predict the same rating for all movies regardless of user. From our general knowledge, this method is not sufficient for building of an adequate recommendation system.
Model assumes the same rating for all movies and users with all the differences explained by random variation:

$$Y_{u,i} = \mu + \epsilon_{u,i}$$

The estimate that minimizes the RMSE is the least squares estimate of u and, in this case, is the average of all ratings:

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

```
## [1] 3.5125
```

We can predict all unknown ratings with mu and calculate the RMSE.

```
RMSE_model_1 <- RMSE(edx_train$rating, mu)
RMSE_model_1
```

```
## [1] 1.0604
```

After adding RMSE of "Model 1: Mean" to Data Frame for RMSE comparison and results we can see that the value is higher than 1.

```
## # A tibble: 1 x 2
##   Method          RMSE
##   <chr>          <dbl>
## 1 Model 1: Mean  1.06
```

After previously inspected data, we see there are some specifics (biases) that can affects the prediction and the ability to correctly predict values, for example:
* different type of movies rated differently (Movie Effect: b_i),
* different type of users, taste and their ratings (User Effect: b_u),
* different range of ratings - Regularization - Netflix challenge (Regularized Movie and User Effect).

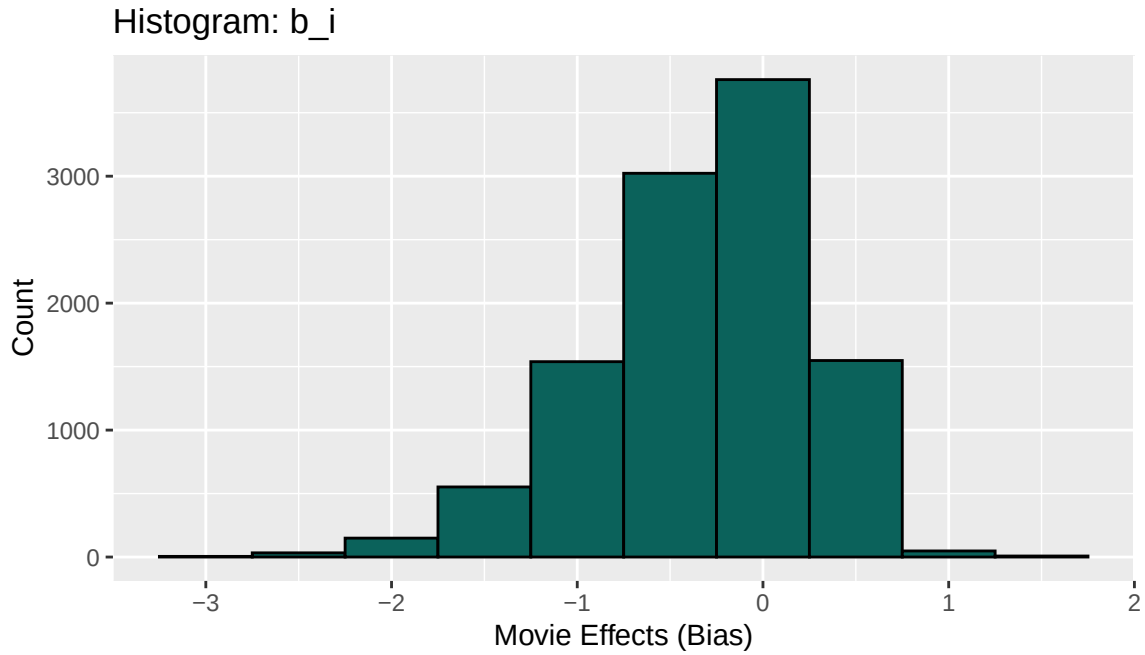
Model 2: Movie Effect: b_i

Logic behind this method is that some movies are just generally rated higher than others. To incorporate this, we can add term b_i to represent average ranking for movie i.

$$Y_{u,i} = \mu + b_i + \epsilon_{u,i}$$

First, we are computing Bias based on Movie (b_i).

Using histogram we are showing variation of estimates b_i. Estimates vary substantially.



Now we can compute predicted ratings for "Model 2: Movie Effect".

```
model_2_pred_ratings <- edx_test %>%
  left_join(b_i, by='movieId') %>%
  mutate(pred = mu + b_i) %>%
  pull(pred)
model_2_pred_ratings
```

We are using RMSE function to compute RMSE for "Model 2: Movie Effect".

```
RMSE_model_2 <- RMSE(model_2_pred_ratings, edx_test$rating)
RMSE_model_2
```

Adding RMSE of "Model 2: Movie Effect" to Data Frame for RMSE comparison and results.

```
rmse_comp <- bind_rows(rmse_comp,
                           data_frame(Method = "Model 2: Movie Effect",
                                      RMSE = RMSE_model_2))
rmse_comp
```

```
## # A tibble: 2 x 2
##   Method                 RMSE
##   <chr>                 <dbl>
## 1 Model 1: Mean         1.06
## 2 Model 2: Movie Effect 0.943
```

There is an improvement in RMSE.

Model 3: User Effect: b_u (with Movie Effects)

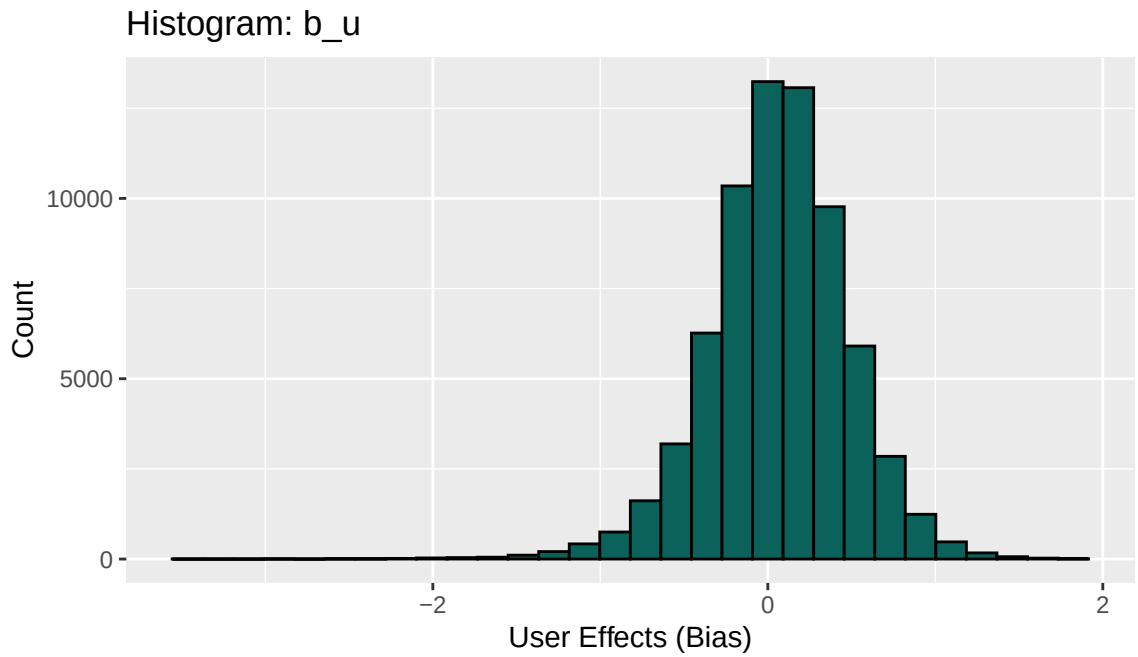
There is substantial variability across users (different taste, different ratings). We are incorporating the variability (user-specific effect) by adding term b_u to the equation.

$$Y_{u,i} = \mu + b_i + b_u + \epsilon_{u,i}$$

Again, we need to compute and add Bias based on User (b_u).

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

Using histogram we are showing variation of estimates b_u.



Now, we are predicted ratings for "Model 3: Movie + User Effect" using formula below.

```
model_3_pred_ratings <- edx_test %>%
  left_join(b_i, by='movieId') %>%
  left_join(b_u, by='userId') %>%
  mutate(pred = mu + b_i + b_u) %>%
  pull(pred)
model_3_pred_ratings
```

We are using RMSE function to compute RMSE for "Model 3: Movie + User Effect".

```
RMSE_model_3 <- RMSE(model_3_pred_ratings, edx_test$rating)
RMSE_model_3
```

Adding RMSE of "Model 2: Movie Effect" to Data Frame for RMSE comparison and results.

```
rmse_comp <- bind_rows(rmse_comp,
                       data_frame(Method = "Model 3: Movie + User Effect",
                                  RMSE = RMSE_model_3))
rmse_comp
```

```
## # A tibble: 3 x 2
##   Method                          RMSE
##   <chr>                          <dbl>
## 1 Model 1: Mean                  1.06
## 2 Model 2: Movie Effect          0.943
## 3 Model 3: Movie + User Effect 0.865
```

There is another improvement in RMSE.

Model 4: Regularization - Movie and User Effect

Best movies can be rated by very few users and small sample sizes lead to uncertainty. There can be noisy estimates that we should not trust. We need to lower the total variability.
Regularization is technique that is used to calibrate machine learning models in order to minimize the adjusted loss function and prevent overfitting or underfitting. Regularization permits us to penalize large estimates that are formed using small sample sizes. The general idea behind regularization is to constrain the total variability of the effect sizes. We are minimizing an equation that adds a penalty instead of minimizing the least squares equation:

$$\sum_{u,i}(y_{u,i} - \mu - b_i)^2 + \lambda \sum_i b_i^2$$

We can show that the values of b_i that minimize this equation are:

$$\hat{b}_i(\lambda) = \frac{1}{\lambda + n_i} \sum_{u=1}^{n_i} (Y_{u,i} - \hat{\mu})$$

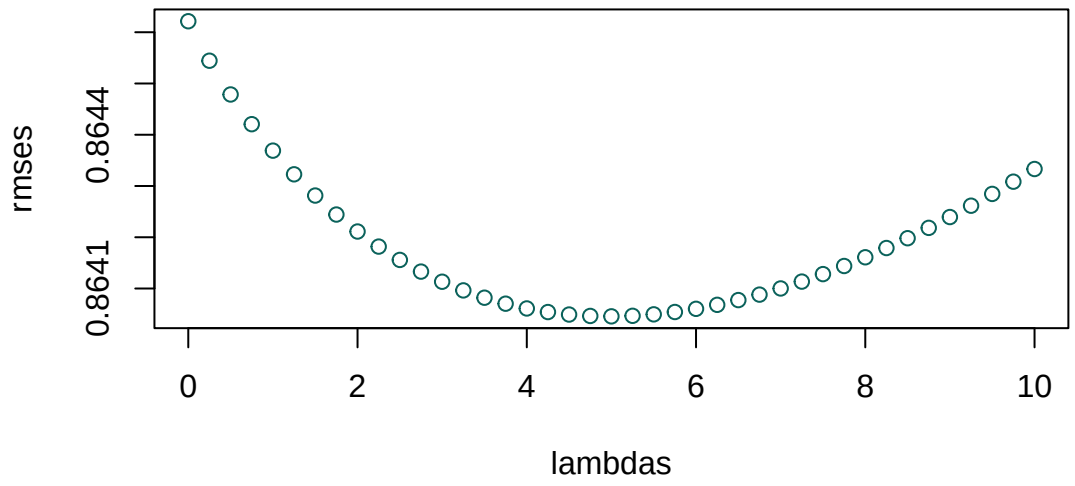
Lambdas (tuning parameter) defined to compute RMSES (from 0 to 10, addition of 0.25):

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

We are constructing and using function to find RMSEs for Model 4 (Regularized Movie and User Effects) based on defined Lambdas.

```
rmses <- sapply(lambdas, function(lambda){

  b_i <- edx_train %>%                    # Movie Effect (Regularized)
    group_by(movieId) %>%
    summarize(b_i = sum(rating - mu)/(n()+lambda))

  b_u <- edx_train %>%                    # Movie and User Effect (Regularized)
    left_join(b_i, by = "movieId") %>%
    group_by(userId) %>%
    summarize(b_u = sum(rating - b_i - mu)/(n()+lambda))

  model_4_pred_ratings <- edx_test %>%    # Predicted Ratings based on Model 4
    left_join(b_i, by = "movieId") %>%
    left_join(b_u, by = "userId") %>%
    mutate(pred = mu + b_i + b_u) %>%
    pull(pred)                           # We want to use predictions based on Model 4

  return(RMSE(model_4_pred_ratings, edx_test$rating))      # We want RMSE
})
```

We continue with selecting the lambda value that minimizes the RMSE (according to graph and data).




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

```
## [1] 5
```

After defining minimal RMSE for Model 4 we are adding it to comparison table.

```
RMSE_model_4 <- min(rmses)
RMSE_model_4
```

```
## [1] 0.86405
```

```
rmse_comp <- bind_rows(rmse_comp,
                       data_frame(Method = "Model 4: Regularization: Movie + User Effect",
                                  RMSE = RMSE_model_4))
rmse_comp
```

```
## # A tibble: 4 x 2
##   Method                                          RMSE
##   <chr>                                          <dbl>
## 1 Model 1: Mean                                  1.06
## 2 Model 2: Movie Effect                          0.943
## 3 Model 3: Movie + User Effect                   0.865
## 4 Model 4: Regularization: Movie + User Effect 0.864
```

There is another slight improvement in RMSE.

Data comparison anc Conclusion: RMSE of models

We want to use most suitable model (lowest RMSE) to test it using "final_holdout_test" dataset. We are looking for the model with lowest RMSE.
Based on previously defined models, we see that lowest RMSE has "Model 4: Regularization: Movie + User Effect".
The RMSE for Model 4 is 0.864.
To lower the RMSE we could continue for example with new model based on Matrix factorization (possible future work).

Final model

We are applying "Model 4: Regularization: Movie + User Effect" on "final_holdout_test" dataset. For final model we choose the minimizing lambda computed in "Model 4: Regularization: Movie + User Effect".

```
lambda_min
```

```
## [1] 5
```

Final model is based on "Model 4: Regularization: Movie + User Effect". For training of our model we use full edx dataset and to validate we use final_holdout_test" dataset.
First, we are computing average of rating from full edx dataset.

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

```
## [1] 3.5125
```

We are computing regularized movie (b_i_FINAL) and and user (b_u_FINAL) effect using full edx dataset with lambda that minimizes RMSE.

```
b_i_FINAL <- edx %>%
  group_by(movieId) %>%
  summarize(b_i_FINAL = sum(rating - mu_FINAL)/(n()+lambda_min))
```

```
b_u_FINAL <- edx %>%
  left_join(b_i_FINAL, by = "movieId") %>%
  group_by(userId) %>%
  summarize(b_u_FINAL = sum(rating - b_i_FINAL - mu_FINAL)/(n()+lambda_min))
```

Predicted ratings based on final model.

```
model_FINAL_pred_ratings <- final_holdout_test %>%
  left_join(b_i_FINAL, by = "movieId") %>%
  left_join(b_u_FINAL, by = "userId") %>%
  mutate(pred = mu_FINAL + b_i_FINAL + b_u_FINAL) %>%
  pull(pred)
```

Now, we are computing and saving RMSE for Final Model: using "final_holdout_test" dataset.

```
RMSE_model_FINAL <- RMSE(model_FINAL_pred_ratings, final_holdout_test$rating)
RMSE_model_FINAL
```

```
## [1] 0.86482
```

Final RMSE is 0.86482. Target RMSE by Course is 0.86490.
Our final RMSE is below target RMSE that means we achieved our aim.

References

- Irizarry RA. Introduction to Data Science : Data Analysis and Prediction Algorithms with R. BC Press. Leanpub; 2019.