# MovieLens recommendation system

Uta Pfennig

3/6/2022

## Introduction

Nowadays recommendation systems are commonly used for different products (e.g. movies, wine, books, cars). Recommendation systems apply ratings that users have given for a particular item to make specific recommendations.

This project intends to build and test the performance of a machine learning algorithm to predict movie ratings by users, as part of the Harvard PH125.9x Data Science Capstone course.

The project follows a 3-step approach: * Step 1: Data exploration: Explore and visualize the data to get an overview and understand how the data is structured * Step 2: Modeling: Build a model using the edx data set to train the model and evaluate its performance using RMSE (residual mean squared error). The RMSE can be interpreted similarly to a standard deviation. It's the typical error when predicting a movie rating. Initially, the algorithm will be built without considering any bias effects. Afterwards the model will be gradually improved by considering movie and user effect. In general, the target is to build a model whose RMSE is below 0.8649. * Step 3: Model evaluation: The performance of the model will be evaluated against the true values contained in the validation set.

### Underlying data set

Movie data provided by grouplens.org will be used within the project. The data is directly downloaded from the website.

The downloaded and transformed data set has been split into: * a train set (edx) and * a test set (validation). 10% of the data will be used for validation. The algorithm will be trained with the "edx" data set.

When creating the validation set, it needs to be ensured that the userId and movieId are both present in the training set (edx) as well as test set (validation).

```
set.seed(1, sample.kind="Rounding")
```

```
## Warning in set.seed(1, sample.kind = "Rounding"): non-uniform 'Rounding' sampler
## used
```

```
test_index <- createDataPartition(y = movielens$rating, times = 1, p = 0.1, list = FALSE)
edx <- movielens[-test_index,]
temp <- movielens[test_index,]

validation <- temp %>%
      semi_join(edx, by = "movieId") %>%
      semi_join(edx, by = "userId")

removed <- anti_join(temp, validation)
```

```
## Joining, by = c("userId", "movieId", "rating", "timestamp", "title", "genres")
```

```
edx <- rbind(edx, removed)

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

# Methods and analysis

## Data exploration

To get familiar with the movieLens data set, some basic data analysis was conducted.

### Evaluation of train data set - edx

The "edx" data set consists of 6 columns and 9,000,055 rows (movie ratings).

```
## [1] 9000055       6
```

Each observation represents a rating from a specific user related to a specific move. The following columns are available: UserId, MovieId, rating, timestamp, title, genres.

However, the timestamp is an integer and not a date. Moreover, movie genres are stored in 1 column separated by '|'. Both columns should be adapted to allow further data insights.

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

All 10,677 movies have a rating. There are no N/A data entries. The movies have been rated by 69,878 users.

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

```
##   movie
## 1 10677
```

```
##    user
## 1 69878
```

### Movie rating

As illustrated above, movies were rated on a scale from 0.5 to 5 in the edx data set. The higher the star, the more the user liked the movie.

With a median of 4 and mean of 3.512, users tend to give high ratings. The standard deviation is 1.06.

| avg | median | sd |
|---|---|---|
| 3.512465 | 4 | 1.060331 |

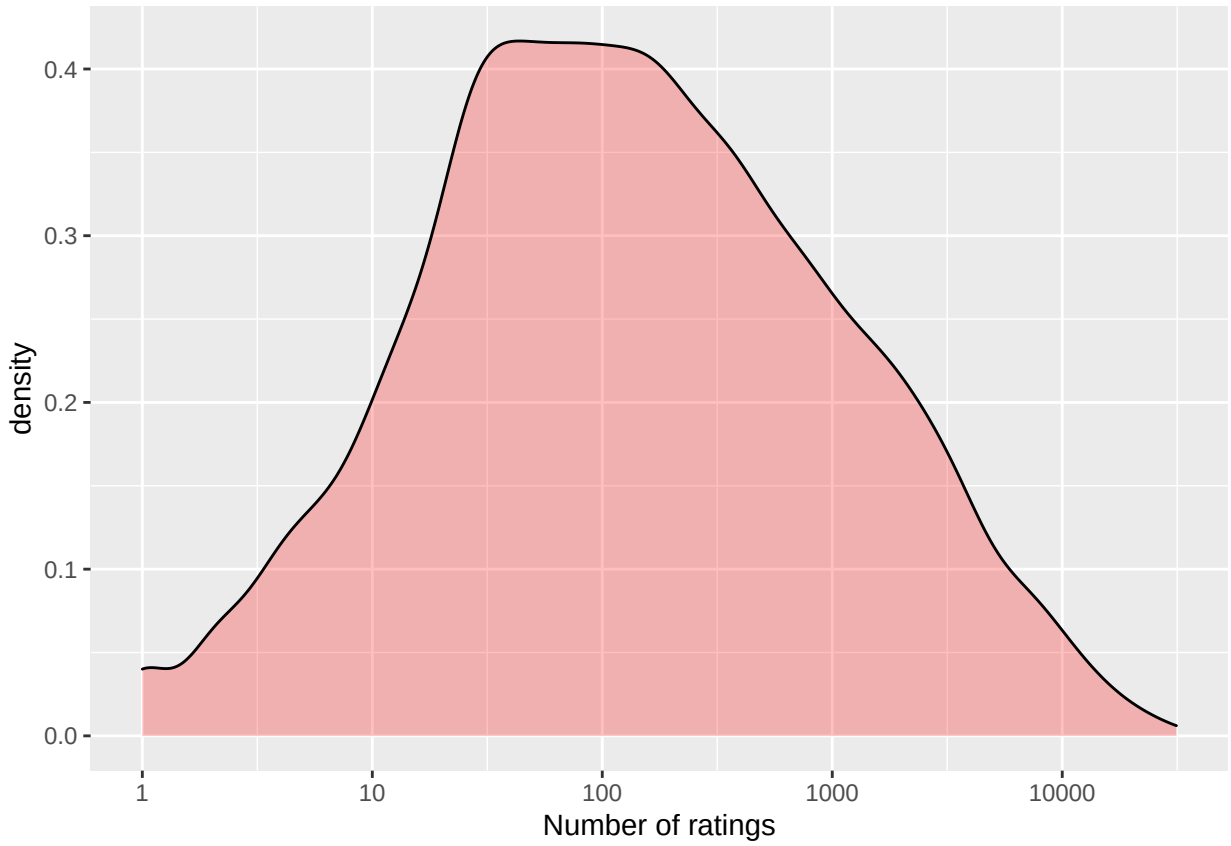
**Which movies were rated most?** The table below lists the top-10 movies rated most often with their average rating.

| title | n | avg |
|---|---|---|
| Pulp Fiction (1994) | 31362 | 4.154789 |
| Forrest Gump (1994) | 31079 | 4.012822 |
| Silence of the Lambs, The (1991) | 30382 | 4.204101 |
| Jurassic Park (1993) | 29360 | 3.663522 |
| Shawshank Redemption, The (1994) | 28015 | 4.455131 |
| Braveheart (1995) | 26212 | 4.081852 |
| Fugitive, The (1993) | 25998 | 4.009155 |
| Terminator 2: Judgment Day (1991) | 25984 | 3.927859 |
| Star Wars: Episode IV - A New Hope (a.k.a. Star Wars) (1977) | 25672 | 4.221311 |
| Apollo 13 (1995) | 24284 | 3.885789 |

On average, movies received 834 ratings. However, the median is only 122. The density graph below confirms that movies most often received between 50 and 100 ratings. There is a wide spread of ratings per movie.

Only a few movies received more than 10,000 ratings. As the top-10 list suggests, movies rated most often tend to be block busters.

| title | n | avg |
|---|---|---|
| Length:10676 | Min. : 1.0 | Min. :0.500 |
| Class :character | 1st Qu.: 30.0 | 1st Qu.:2.844 |
| Mode :character | Median : 122.0 | Median :3.268 |
| NA | Mean : 843.0 | Mean :3.192 |
| NA | 3rd Qu.: 565.2 | 3rd Qu.:3.609 |
| NA | Max. :31362.0 | Max. :5.000 |




The analysis below shows that 8,554 out of 10,677 movies (80%) have received less ratings than on average. Only 2,122 movies received more ratings than average.

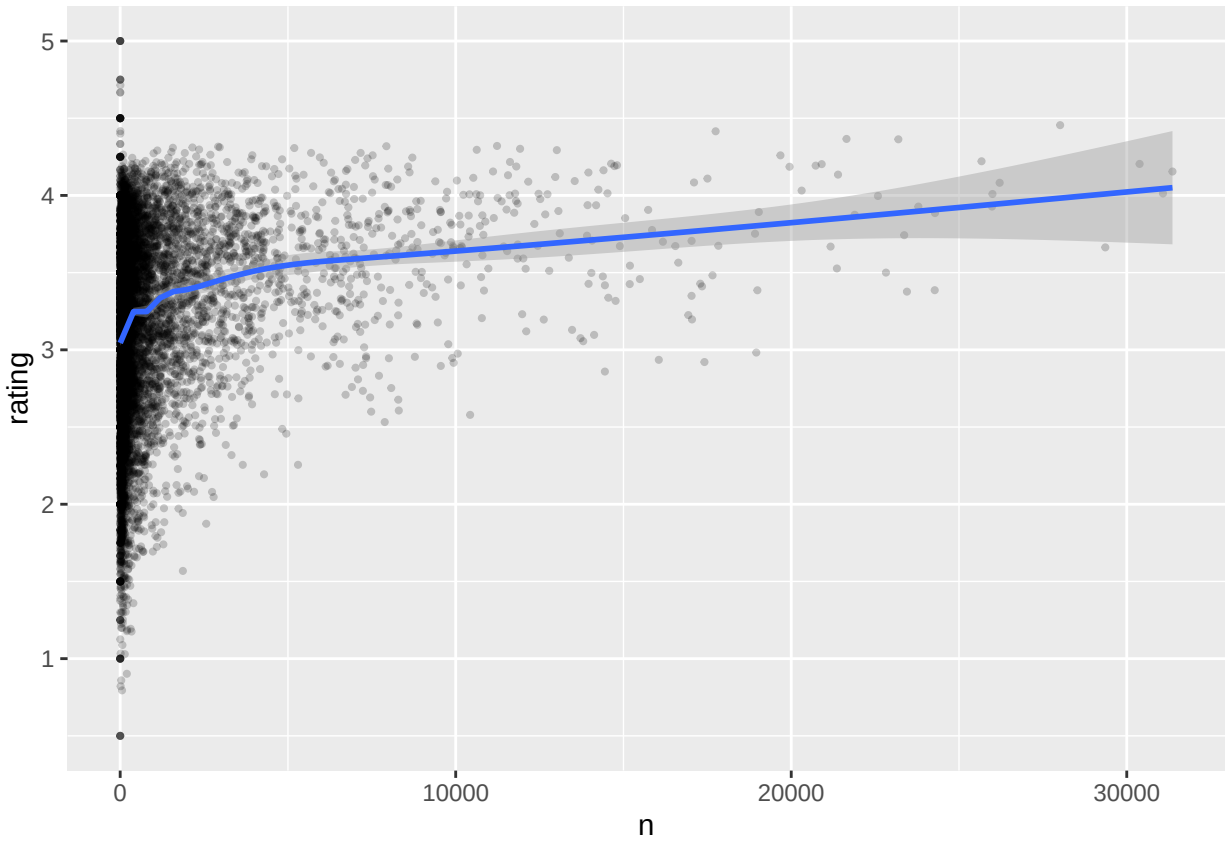
```
## # A tibble: 2 x 2
##   group_id movie
##      <dbl> <int>
## 1        1  2122
## 2        2  8554
```

Interestingly, movies which were less often rated have a lower average rating than movies which were more often rated. The findings are illustrated in the table below. Group 1 represents movies which received more ratings than average and group 2 represents movies less often rated.

```
## # A tibble: 2 x 2
##   group_id   avg
##      <dbl> <dbl>
## 1        1  3.44
## 2        2  3.13
```

The difference in rating depending on the frequency become even more apparent in the scatterplot below. Movies with little nunber of ratings tend to get a lower rating than movies with more ratings. This movie bias should be considered for optimizing the model.

```
## `geom_smooth()` using method = 'gam' and formula 'y ~ s(x, bs = "cs")'
```

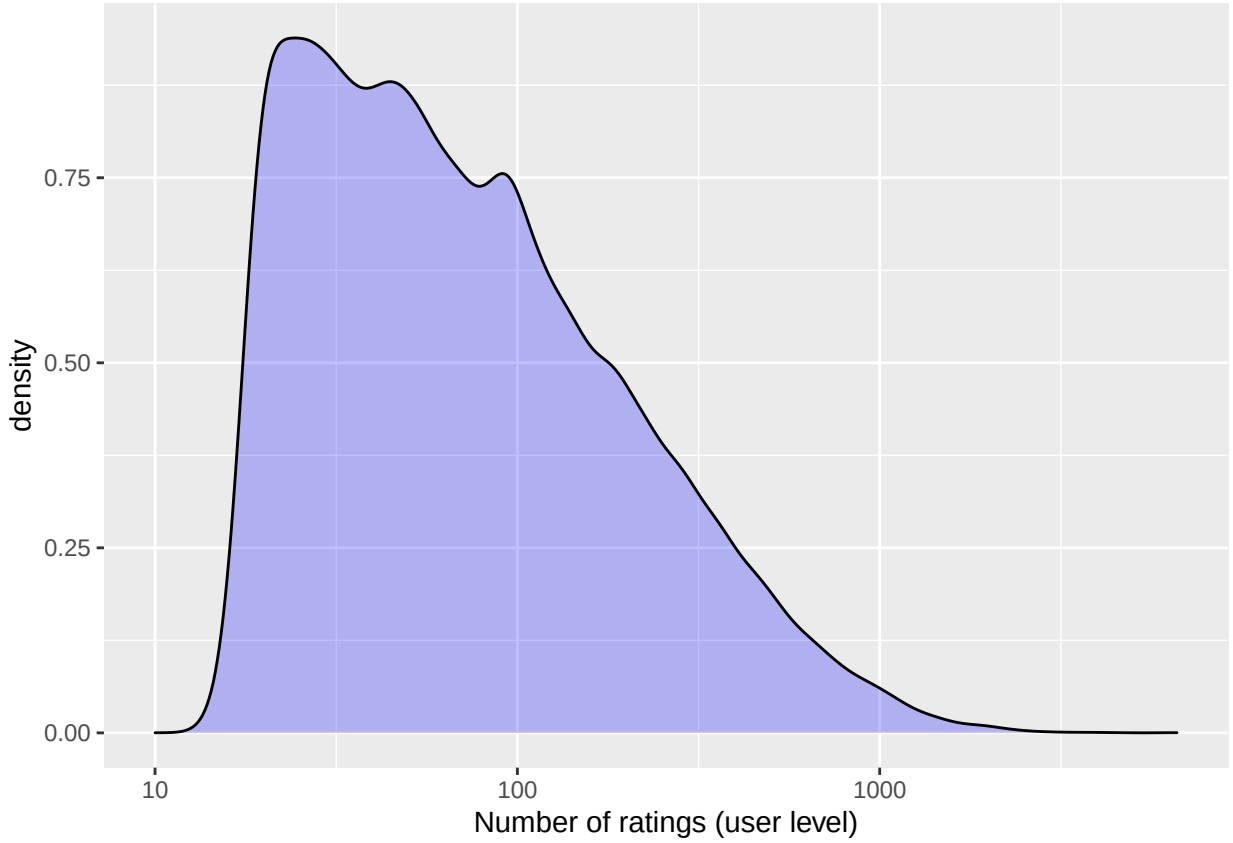



**User rating**

**How often do users rate movies?** 69,878 users have rated various movies. On average, users submit 129 movie ratings.

The data is ranging from 10 to 6616 ratings per user. The majority of users submitted between 30 and 100 ratings as illustrated in the density plot below.

| userId | n | avg |
|---|---|---|
| Min. : 1 | Min. : 10.0 | Min. :0.500 |
| 1st Qu.:17943 | 1st Qu.: 32.0 | 1st Qu.:3.357 |
| Median :35798 | Median : 62.0 | Median :3.635 |
| Mean :35782 | Mean : 128.8 | Mean :3.614 |
| 3rd Qu.:53620 | 3rd Qu.: 141.0 | 3rd Qu.:3.903 |
| Max. :71567 | Max. :6616.0 | Max. :5.000 |

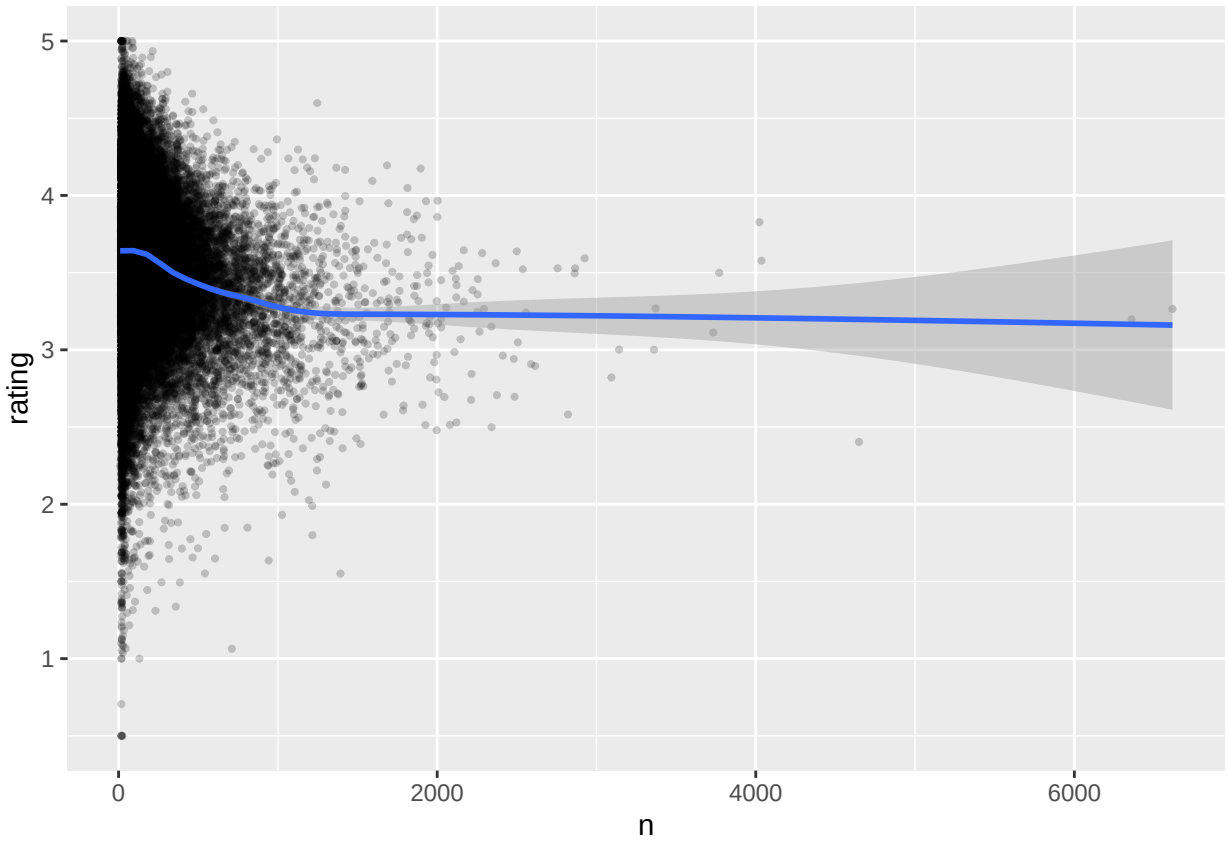



The 73% of the users are less active and rate less than 128 movies as shown in the break-down below.

```
## # A tibble: 2 x 2
##   group_id  user
##      <dbl> <int>
## 1        1 19094
## 2        2 50784
```

Moreover, the plot below illustrates that the average rating varies more among users who have given less ratings compared to users who are very active in submitting ratings. These user rating behavior needs to be considered when building the model.

```
## `geom_smooth()` using method = 'gam' and formula 'y ~ s(x, bs = "cs")'
```




## Modelling

As baseline, the RMSE is calculated which is defined as follows:

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

With the following parameters: * $y_{u,i}$ = rating for movie i by user u * $\hat{y}_{u,i}$ = prediction of rating * N = the number of user/movie combinations * sum occurring over all combinations

As stated in the "introduction" section, RMSE can be interpreted similarly to a standard deviation. It's the typical error when predicting a movie rating.

RMSE can be computed using the function below.

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

### Average movie rating model

As a starting point, a very simple recommendation system is constructed which predicts the rating for all movies across all users. The model assumes that all differences in movie ratings are explained by random variation alone. This means that movie or user effects, which are highlighted in the section "Data exploration", are not considered.

The formula for the "average movie rating model" looks as follows:

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

with $\epsilon_{u,i}$ independent errors sampled from the same distribution centered at 0 and $\mu$ the "true" rating for all movies. We know that the estimate that minimize the RMSE is the least square estimate of $Y_{u,i}$ , in this case, is the average of all ratings. Considering the previous data analysis, the expected rating is between 3 and 4.

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

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

If all unknown ratings with $Y_{u,i}$ are predicted, the RMSE below is obtained.

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

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

RMSE is 1.061 which doesn't meet the objective of RMSE $< 0.865$. The result is stored in a results table. The table is updated with additional results after tuning the model.

```
## # A tibble: 1 x 2
##   method                RMSE
##   <chr>                <dbl>
## 1 Average rating model  1.06
```

**Adjust model considering movie effect**

As highlighted in the section "data exploration", popular movies with more ratings were generally rated higher than less known movies with a lower number of ratings.

To augment the model, the movie bias needs to integrated by adding the variable $b_i$. This variable can be computed by estimating the deviation of each movies' mean rating from the total mean of all movies $\mu$.

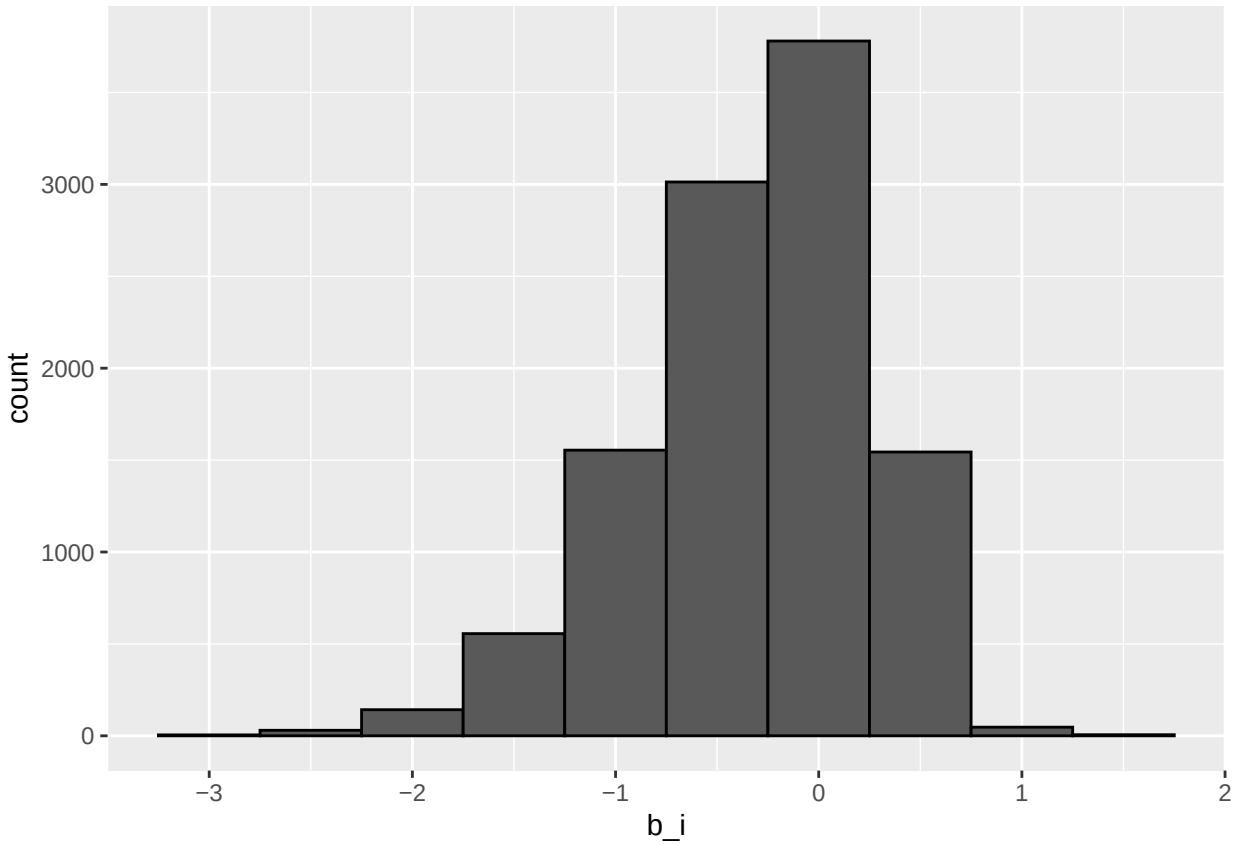
The enhanced formula for the "movie effect model" looks as follows:

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

In an initial step, the least squares estimate per movie $b_i$ is calculated.

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

LSE are displayed in a histogram which is clearly left skewed. The plot confirms that more movies have negative effects.




The augmented formula is used to predict movie ratings considering the fact that some movies are rated differently. RMSE is 0.9439 which is a better value than RMSE for the "average rating model".

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

```
## [1] 0.9439087
```

Overview of results

| method | RMSE |
|---|---|
| Average rating model | 1.0612018 |
| Movie effect model | 0.9439087 |

**Adjust model considering movie & user effects**

As highlighted in the section "data exploration", movie ratings are not only affected by the popularity of the movie but also by individual user rating behavior. A cranky user may give a lower rating to a great movie.

To account for user rating behavior, the prediction model will be further enhanced with an additional attribute $b_u$ as follows.

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

The variable $b_u$ is computed as the average of

$$Y_{u,i} - \mu - b_i$$

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

Based on the computed estimate per user and estimate per movie, the predictors of the model are updated and RMSE is calculated again. The value for RMSE could be even further improved to 0.8653. RMSE has been decreased by around 18% after considering the movie and user effect. But still the project target is not yet achieved.

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

```
## [1] 0.8653488
```

Overview of results

| method | RMSE |
|---|---|
| Average rating model | 1.0612018 |
| Movie effect model | 0.9439087 |
| Movie + user effect model | 0.8653488 |

**Adjust model by regularizing movie & user effects**

As highlighted in the section "data exploration", the data set contains movies with just one or two ratings. Movies with such few ratings have a higher uncertainty causing higher variability. These are "noisy estimates" which shouldn't be considered in the prediction model. The same is applicable to users rating only very few movies.

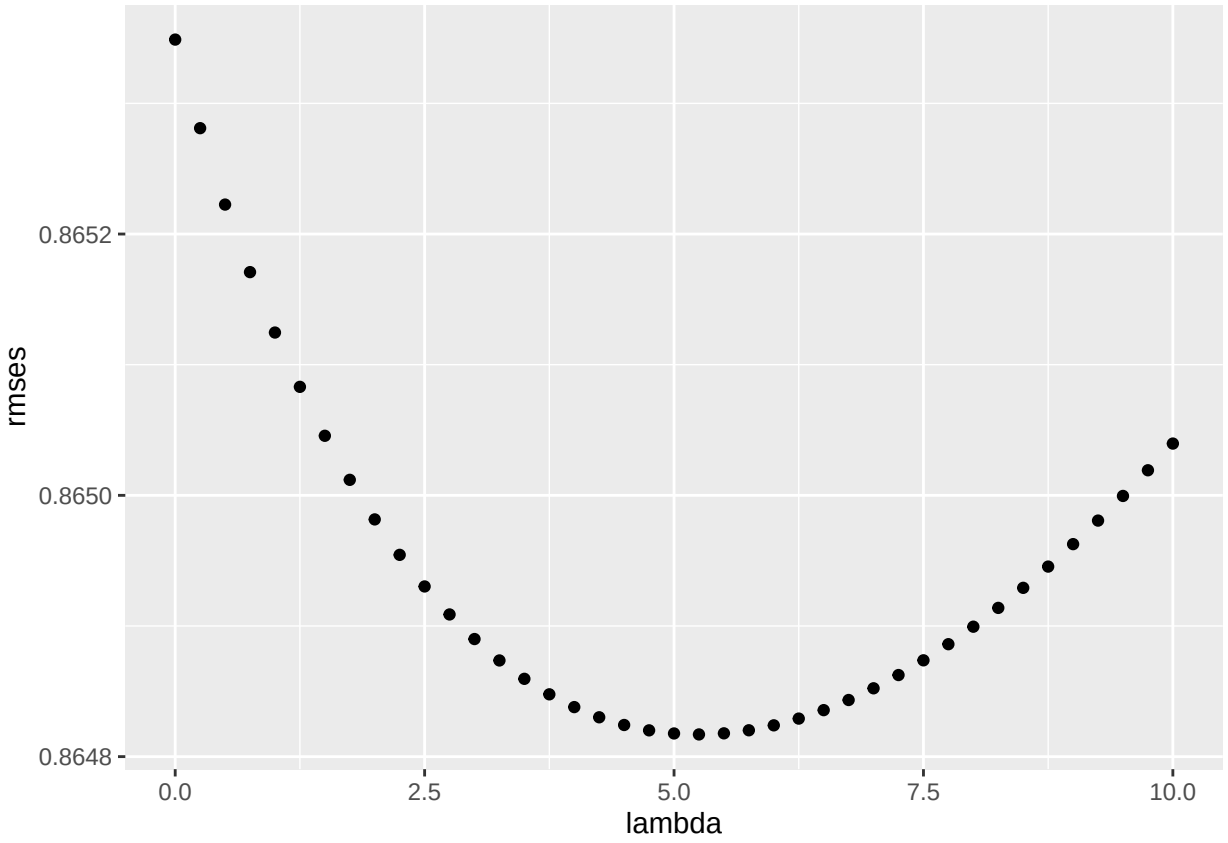
Regularization is a useful technique to constraint the variability caused by large estimates coming from small sample sizes. This is achieved by adding a penalty term to the equation.

Initially, the value of lambda (the tining parameter) needs to be identified which will minimize RMSE.

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

rmses <- sapply(lambda, 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) %>%
    pull(pred)
```

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

The graph below outlines the results for lambda versus RMSE in order to select the optimal lambda. The optimal lambda is 5.25.




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

By penalizing small sample sizes, the algorithm was further optimized. RMSE has been reduced to 0.8648. The value means the target of this project.

```
## [1] 0.864817
```

# Results

The table below lists the RMSE values for the different models. The lowest value of RMSE is 0.864817.

| method | RMSE |
|---|---|
| Average rating model | 1.0612018 |
| Movie effect model | 0.9439087 |
| Movie + user effect model | 0.8653488 |
| Regularized movie + user effect model | 0.8648170 |

# Conclusion

This report outlines the process of constructing a recommendation algorithm using MovieLens data. Initially, a baseline as average rating for all movies was calculated. On top of the average rating, two additional predictors were included: (1) movie effects and (2) user effects.

The model could be further improved by considering other predictors like genre, actor(s) or release year. Furthermore, other machine learning techniques like Penalized Least Squares or Matrix Factorization could further improve results.

# References

Irizzary,R., 2018 “Introduction to Data Science”, https://rafalab.github.io/dsbook/