

# Movie Rating Prediction Project: Movie Lens Dataset

HarvardX - PH125.9x Data Science: Capstone Course

Mauricio Rabelo Soares

25 julho, 2022

## Introduction

A recommendation system, is a subclass of information filtering system that provide suggestions for items that are most pertinent to a particular user. Typically, the suggestions refer to various decision-making processes, such as what product to purchase, what music to listen to, or what movies to watch.<sup>1</sup> The goal of this project is to create a recommendation system, which the  $RMSE < 0.86490$ , using all the tools we have learn throughout the multi-part course in HarvardX's Data Science Professional Certificate series. As presented in the Machine Learning course<sup>2 3</sup>, a recommendation systems for a movie use *ratings* ( $y_{u,i}$ ) that *users* ( $u$ ) have given to some items, or *movies* ( $i$ ), to make specific recommendations to specific user. In this model, movies for which a high rating is predicted for a given user are then recommended to that user. For this challenge we going to use a subset of dataset provided by the GroupLens. In our case the subset of this data is provided by MovieLens 10M movie ratings. This a stable benchmark dataset with **10 million ratings** applied to **10,000 movies** by **72,000 users**.

## Methods

The methods section explains the process and techniques used in this project, in the first part, then explains the data cleaning used to extract and clean the data, in the second part. In the third part of this section we present the data exploration and visualization of the data to gain some insights. The fourth, and last part of this section, we show the modeling approach used in this project.

## The process and techniques used

Recommendation systems are particularly useful when an individual needs to choose an item from a potentially overwhelming number of items that a service may offer. The recommendations system build in this project are made using the same process and techniques used by the winners of the Netflix challenges<sup>4</sup>, which is presented in Chapter 34 Large datasets at Irizarry (2019)<sup>5</sup>, and some new approaches[Qiu et al. (2021)]<sup>6</sup>

The process begin with the download, cleaning and tidy the data. After the data cleaning we explore the data to gain some insight trough visualization and selected table. This insights is the basis of the modelling approach of this project, that starts building the simplest possible recommendation system: predict the same rating for all movies regardless of user, and ends with recommendation system that use parallel matrix

---

<sup>1</sup>[https://en.wikipedia.org/wiki/Recommender\\_system](https://en.wikipedia.org/wiki/Recommender_system)

<sup>2</sup>HarvardX - PH125.8x Data Science: Machine Learning <https://www.edx.org/course/data-science-machine-learning>

<sup>3</sup>34.7 Recommendation systems - <https://rafalab.github.io/dsbook/large-datasets.html#recommendation-systems>

<sup>4</sup>[https://en.wikipedia.org/wiki/Netflix\\_Prize](https://en.wikipedia.org/wiki/Netflix_Prize)

<sup>5</sup><https://rafalab.github.io/dsbook/large-datasets.html#fnref112>

<sup>6</sup>[https://github.com/Airborne737/MovieLens\\_Harvard](https://github.com/Airborne737/MovieLens_Harvard)

factorization, the product of two matrices of lower dimensions,  $P_{n \times k}$  and  $Q_{n \times k}$  where  $P$  is user matrix, and  $Q$  is the movie matrix, the `recoSystem`.

## Data cleaning

For this project we going to use a subset of dataset provided by the GroupLens. In our case the subset of this data is provided by MovieLens 10M movie ratings. This a stable benchmark dataset with **10 million ratings** applied to **10,000 movies** by **72,000 users**. This version of the dataset was released in January of 2009<sup>7</sup>. The code below load the library that we need, then download the file from GroupLens site and read the file. After read the file the code create the data frame `movielens` and remove the temporary file.

```
# Load library
library(caret)
library(data.table)
library(fields)
library(tidyverse)
library(knitr)
library(kableExtra)
library(grid)
library(ggplot2)
library(lattice)
library(gridExtra)
library(recoSystem)

options(timeout=100)
#create tempfile and download
dl <- tempfile()
download.file("https://files.grouplens.org/datasets/movielens/ml-10m.zip", dl)

#read the file and give name to columns
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")

#create data frame movielens
movies <- as.data.frame(movies) %>% mutate(movieId = as.numeric(movieId),
                                           title = as.character(title),
                                           genres = as.character(genres))
movielens <- left_join(ratings, movies, by = "movieId")

#remove temporary files
rm(dl, ratings, movies)
```

## Data exploration and visualization

The exploration and visualization of the data provide insightful information about the users, the movies and ratings. The first 5 rows of the dataset that we use in this project is presented in table 1. The columns `movieId`, `userId` and `rating` are the variable of interest.

---

<sup>7</sup><https://grouplens.org/datasets/movielens/10m/>

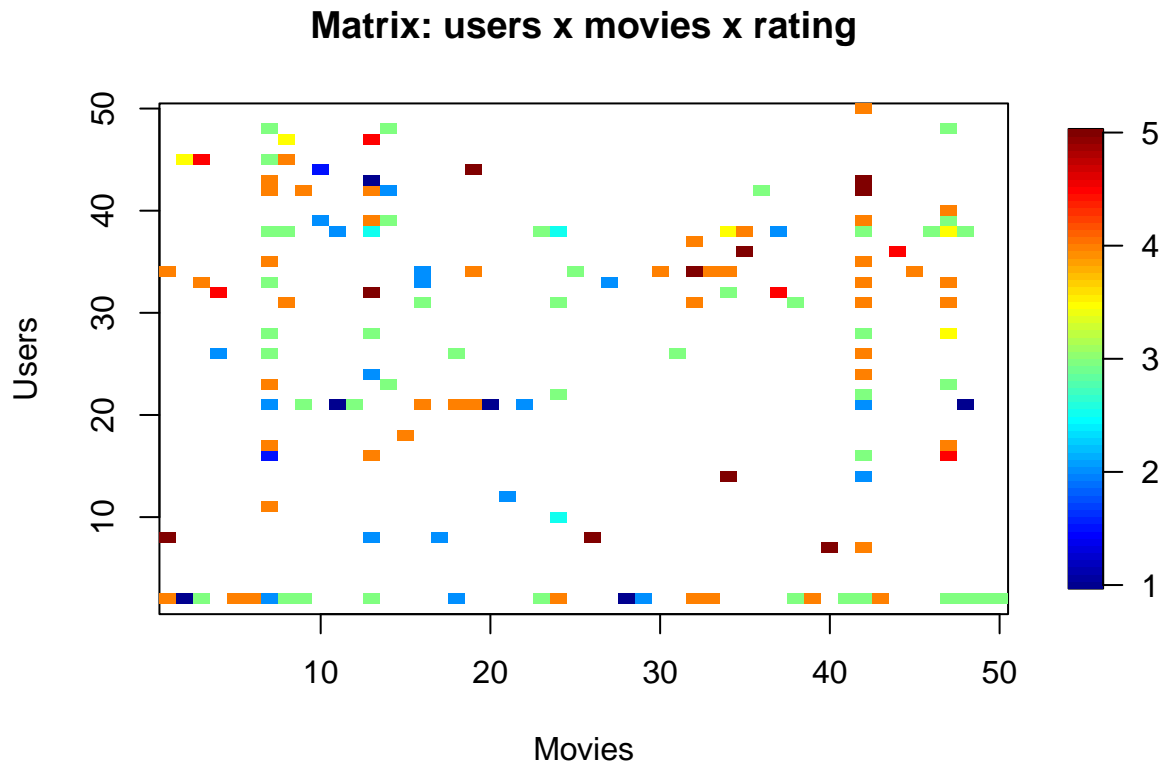
```
knitr::kable(head(movielens %>% as_tibble(),5),
              caption = "The first 5 rows of the dataset, movielens",
              digits = 2,
              align = "ccccc",
              position = "b") %>%
  kable_styling(latex_options = "scale_down")
```

The variables, `movieId`, `userId` and `rating`, that is going to be used to build the model is presented as a matrix in the figure 1. The figure have the movies (`movieId`) in the x axis, the users in the y axis (`userId`), and the respective rating (`rating`). The matrix, extract from a sample of 50 users and 50 movies, provide some insights about the behavior of some users, the preference for some movies, and the sparse of the matrix. The goal of this project is to fill the blank spaces with a rate.

```
#matrix users x movies x rating
users <- sample(unique(movielens$userId), 50)
movielens %>% filter(userId %in% users) %>%
  select(userId, movieId, rating) %>%
  spread(movieId, rating) %>% select(sample(ncol(.), 50)) %>%
  as.matrix() %>% t(.) %>%
  image.plot(1:50, 1:50, . ,
             xlab="Movies",
             ylab="Users",
             main= "Matrix: users x movies x rating")
```

Table 1: The first 5 rows of the dataset, movielens

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	231	5	838983392	Dumb & Dumber (1994)	Comedy
1	292	5	838983421	Outbreak (1995)	Action Drama Sci-Fi Thriller
1	316	5	838983392	Stargate (1994)	Action Adventure Sci-Fi



To see a potential flaw in the data we make a slice of the top 5 most rated movies and users. The unique users that provided ratings, the unique movies that were rated and the unique rating provided by a unique user to a unique movie, are presented to illustrate the possible rating matrix  $users \times movies$   $n\_distinct(userId) \times n\_distinct(movieId) = "n\_distinct(userId) * n\_distinct(movieId)"$  and the realized rating matrix  $length(rating)$ . The extremes values confirm that some users are much more active than others, the most active user rated more than 50% of the total unique movies, and some movies have been rated for more than 1/3 of the total unique users.

```
# Unique users, movies, rating
p0 <- tableGrob(movielens %>% summarize(n_users = n_distinct(userId),
                                     n_movies = n_distinct(movieId),
                                     n_rating = length(rating)))

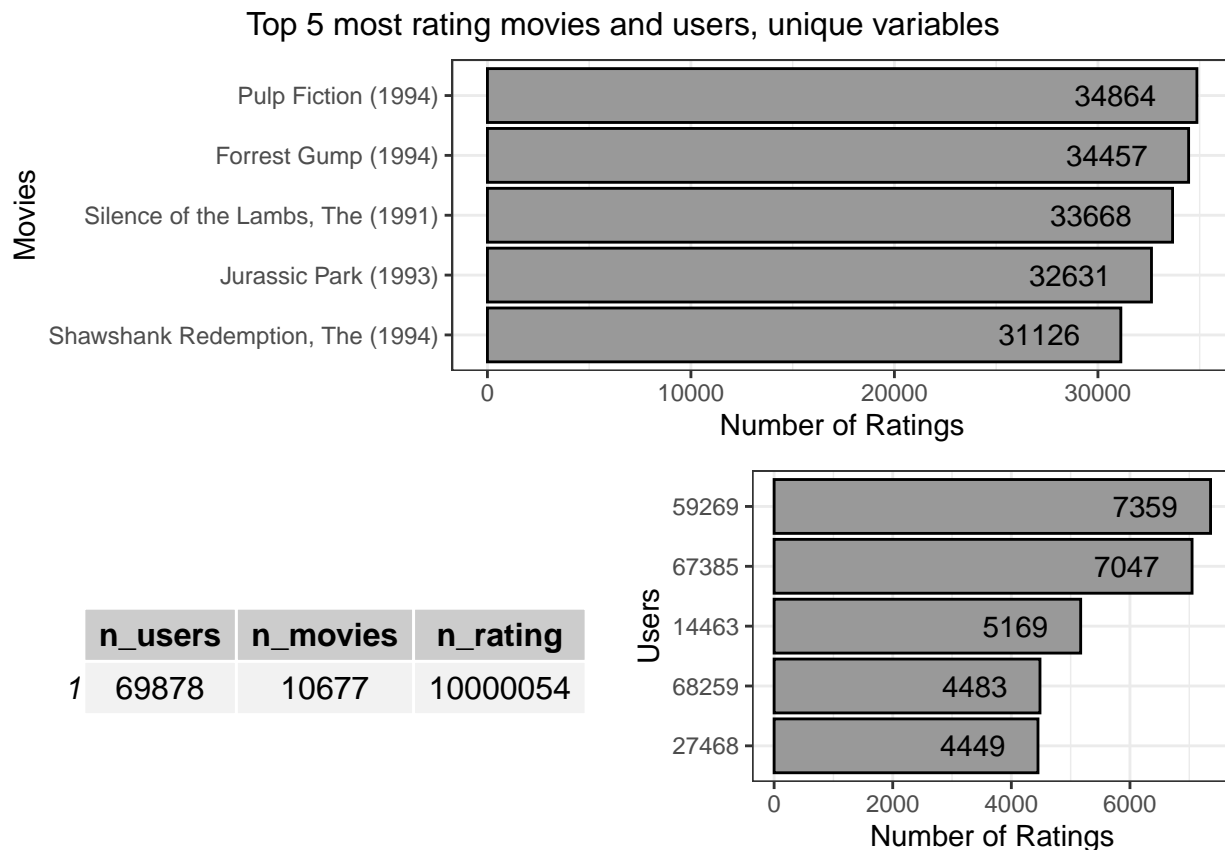
# Top 5 movies
p1 <- movielens %>%
  group_by(title) %>%
  summarize(count = n()) %>%
  arrange(-count) %>%
  top_n(5, count) %>%
  ggplot(aes(count, reorder(title, count))) +
  geom_bar(color = "black", fill = "#999999", stat = "identity") +
  geom_text(aes(label=count), position=position_dodge(width=0.9), hjust=1.5) +
  xlab("Number of Ratings") +
  ylab("Movies") +
  theme_bw()
```

```

# Top 5 users
p2 <- movielens %>%
  group_by(userId) %>%
  summarize(count = n()) %>%
  arrange(-count) %>%
  top_n(5, count) %>%
  ggplot(aes(count, reorder(userId, count))) +
  geom_bar(color = "black", fill = "#999999", stat = "identity") +
  geom_text(aes(label=count), position=position_dodge(width=0.9), hjust=1.5) +
  xlab("Number of Ratings") +
  ylab("Users") +
  theme_bw()

#Top 5 most rating movies and users, unique variables
gridExtra::grid.arrange(p1,
  arrangeGrob (p0, p2, ncol = 2),
  nrow = 2,
  top = "Top 5 most rating movies and users, unique variables")

```



The dataset distribution presented through histograms provide some insights about the general properties of the data. As showed in the slice before some movies get rated more than others, and some user are more active than others.

```

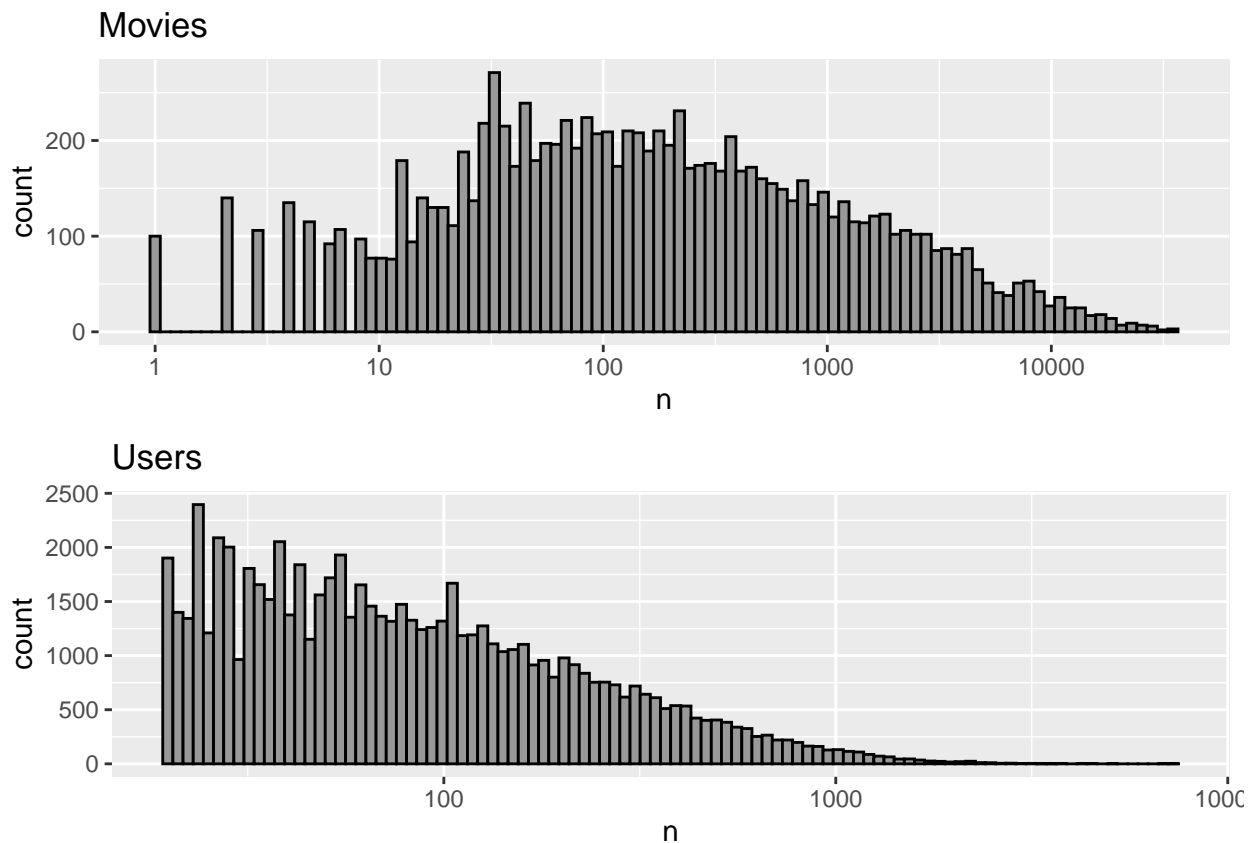
p3 <- movielens %>%
  count(movieId) %>%
  ggplot(aes(n)) +

```

```

geom_histogram(bins = 100, color = "black", fill="#999999") +
scale_x_log10() +
ggtitle("Movies")
p4 <- movielens %>%
count(userId) %>%
ggplot(aes(n)) +
geom_histogram(bins = 100, color = "black", fill="#999999") +
scale_x_log10() +
ggtitle("Users")
gridExtra::grid.arrange(p3, p4, nrow = 2)

```



## Modeling approach

The recommendation system is better as its error has decreased, for this project the error is the typical error we make when predicting a movie rating ( $\hat{y}_{u,i} - y_{u,i}$ ). The loss function used to evaluate the models is based on the residual mean squared error ( $RMSE$ ) on a test set. The definition of  $RMSE$  includes  $N$ , the number of user/movie combination, and the sum occurring over all these combination. In this case the Loss function is:

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

```

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

```

The model which  $RMSE = 0$  is a perfect model prediction, or without errors. For this project the reported  $RMSE < 0.86490$  is the goal. The  $RMSE > 1$  means our error is larger than one star, which means a bad model.

## Machine Learning

The machine learning decisions are based on algorithms build with data, so for this project the dataset MovieLens 10M are going to be used to train and test the model. The train\_set and test\_set are build trough function createDataPartition as presented:

```
set.seed(1, sample.kind="Rounding")
test_index <- createDataPartition(y = movielens$rating, times = 1, p = 0.1, list = FALSE)
train_set <- movielens[-test_index,]
temp <- movielens[test_index,]

# Matching userId and movieId in both train and test sets
test_set <- temp %>%
  semi_join(train_set, by = "movieId") %>%
  semi_join(train_set, by = "userId")

# Adding back rows into train set
removed <- anti_join(temp, test_set)
train_set <- rbind(train_set, removed)

#remove temporary file
rm(test_index, temp, removed)
```

## Naive approach

The simplest model to recommend a movie to any user, is the model that predict the same rate  $\mu$  for all movies regardless of user. In the naive approach the variation of the differences is random, and the independent error  $\epsilon_{u,i}$  centered at 0. The model looks like:

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

The average of all ratings is the estimate that minimize the  $RMSE$ , and if we fill the blank cells in the matrix with the  $\mu$  mu we obtain the naive\_rmse.

```
#Average of all ratings
mu <- mean(train_set$rating)
mu
```

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

```
# Model 1 - Naive_rmse
naive_rmse <- RMSE(test_set$rating, mu)
results <- tibble(Method = "Model 1: Naive RMSE", RMSE = naive_rmse)
results
```

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

## Bias approach

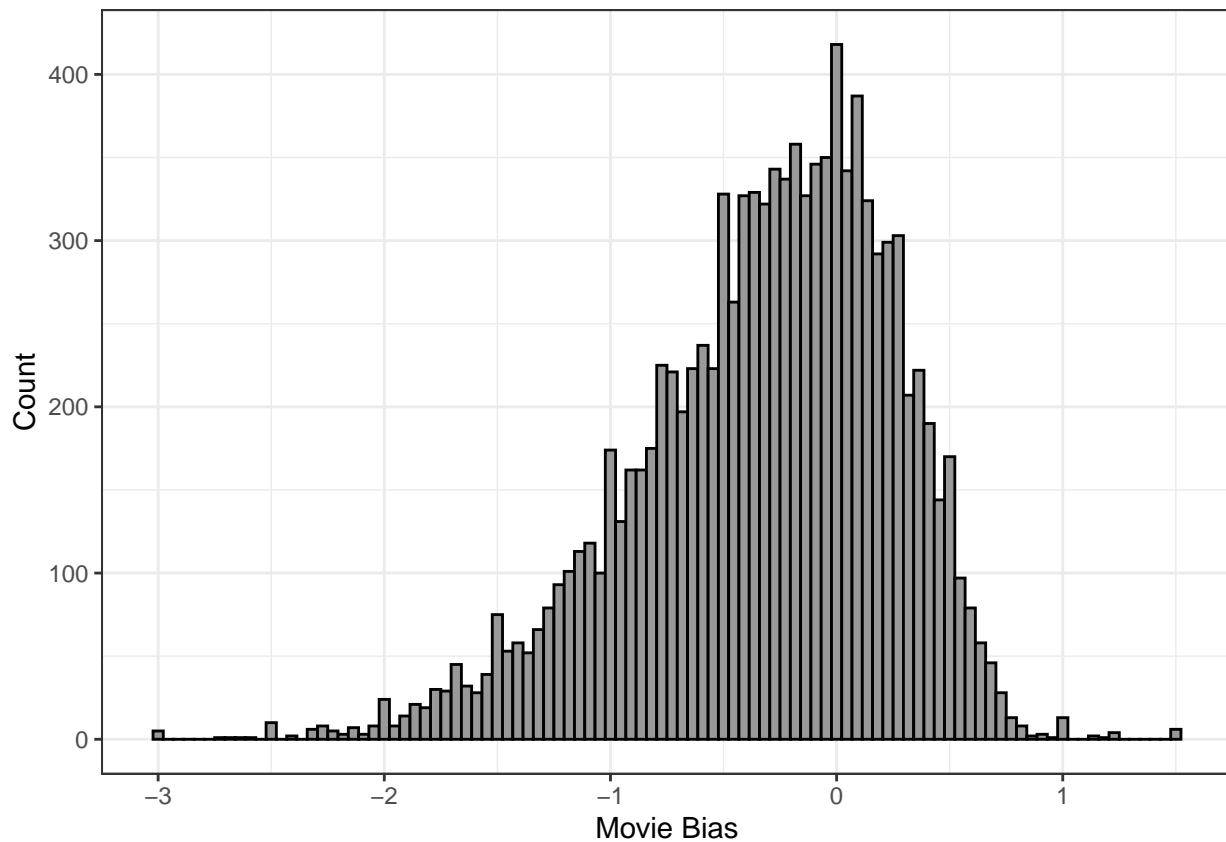
The data exploration and visualization through Matrix: users x movies x rating presented a picture that some users are more active than others, and some movies are better rated than others, so let's model these effects. In the formula below the  $b_i$  and  $b_u$  are respectively the movie and user effect:

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

The movie effects measure the mean of  $Y_{u,i} - \hat{\mu}$  for each movie  $i$ , the code is shown below with the histogram that presents the right-skewed distribution around zero.

```
# Movie bias
movies_bias <- train_set %>%
  group_by(movieId) %>%
  summarize(b_i = mean(rating - mu))

# Movie bias distribution
movies_bias %>% ggplot(aes(b_i)) +
  geom_histogram(color = "black", fill = "#999999", bins = 100) +
  xlab("Movie Bias") +
  ylab("Count") +
  theme_bw()
```



In this approach, using  $y_{u,i} = \mu + b_i$ , our prediction improves by 0.11:



```
# Model 2 - Movie Bias RMSE
predicted_ratings <- mu + test_set %>%
  left_join(movies_bias, by = "movieId") %>%
  pull(b_i)
m_bias_rmse <- RMSE(predicted_ratings, test_set$rating)
results <- bind_rows(results, tibble(Method = "Model 2: Mean + movie bias", RMSE = m_bias_rmse))
results
```

```
## # A tibble: 2 x 2
##   Method      RMSE
##   <chr>      <dbl>
## 1 Model 1: Naive RMSE      1.06
## 2 Model 2: Mean + movie bias 0.944
```

The user effects measure the variability across all users, in special the more actives users since they affect the average rate of more movies. As we saw in the colors of the matrix some users are more demanding than other. To fit the model we add the  $b_u$  term, and the code of the improved model is:

```
# User bias
user_bias <- train_set %>%
  left_join(movies_bias, by = "movieId") %>%
  group_by(userId) %>%
  summarize(b_u = mean(rating - mu - b_i))

# Model 3 - Movies and User Bias RMSE
predicted_ratings <- test_set %>%
  left_join(movies_bias, by = "movieId") %>%
  left_join(user_bias, by = "userId") %>%
  mutate(pred = mu + b_i + b_u) %>%
  pull(pred)
u_bias_rmse <- RMSE(predicted_ratings, test_set$rating)
results <- bind_rows(results, tibble(Method = "Model 3: Mean + movie bias + user effect", RMSE = u_bias_rmse))
results %>% knitr::kable()
```

Method	RMSE
Model 1: Naive RMSE	1.0612018
Model 2: Mean + movie bias	0.9439087
Model 3: Mean + movie bias + user effect	0.8653488

## Regularization approach

Penalized least squares

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

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

Choosing the penalty terms

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

## Matrix Factorization approach

A typical solution for  $P$  and  $Q$  is given by the following optimization problem (Chin, Zhuang, et al. 2015a, 2015b):

$$\min_{P,Q} \sum_{(u,v) \in R} \left[ f(p_u, q_v; r_{u,v}) + \mu_P \|p_u\|_1 + \mu_Q \|q_v\|_1 + \frac{\lambda_P}{2} \|p_u\|_2^2 + \frac{\lambda_Q}{2} \|q_v\|_2^2 \right]$$

where  $(u,v)$  are locations of observed entries in  $R$ ,  $r_{u,v}$  is the observed rating,  $f$  is the loss function, and  $P, Q, \mu_P, \mu_Q$  are penalty parameters to avoid overfitting.

## Results

With the `movielens` data frame the code create the `edx` and `validation` datasets that will be used to train and test our final algorithm.

Section that presents the modeling results and discusses the model performance

## Conclusion

Section that gives a brief summary of the report, its limitations and future work

## References

- bnwicks - <https://github.com/bnwicks/Capstone/blob/master/MovieLens.R>  
Airborne737 - <https://www.rpubs.com/Airborne737/movielens>  
R Markdown: Gerando relatórios usando o R (Parte 1) - <https://www.youtube.com/watch?v=obxa5VH4WvY>  
R Markdown: Gerando relatórios usando o R (Parte 2) - <https://www.youtube.com/watch?v=tcNx0QbDPBo>  
Irizarry, Rafael A. 2019. *Introduction to Data Science*. Chapman; Hall/CRC. <https://doi.org/10.1201/9780429341830>.  
Qiu, Yixuan, David Cortes, Chih-Jen Lin, Yu-Chin Juan, Wei-Sheng Chin, Yong Zhuang, Bo-Wen Yuan, Meng-Yuan Yang, and other contributors. See file AUTHORS for details. 2021. "Recosystem: Recommender System Using Matrix Factorization." <https://CRAN.R-project.org/package=recosystem>.