

# Data Science Capstone - Movielense project

Julia Schröder

07/01/2021

## Introduction

### Description of dataset

The dataset used in this analysis is called “movielens” and contains 10 million ratings of movies by users. The dataset was split into the **train** (90%) and **validation** sets (10%) according to the task. The analyses going forward will only use and describe the **train** dataset. It comprises 23,371,423 ratings of 10,677 movies given by 69,878 users.

### Summary of the project’s goal

Goal of this project was to build a movie recommendation system. It will predict movie ratings by training a machine learning (ML) algorithm using the available ratings from the 10M version of the MovieLens dataset. Herefore, the **train** dataset was divided into a training set comprising 90% of the data and a test dataset with 10% of the data. The different features were added sequentially when building the model using the training data in order to improve the model performance, which was evaluated using the residual mean squared error.

## Methods

The methods applied in this project were taken from the Machine Learning course on EdX.org and the book “Introduction to Data Science” by Prof. Rafael A. Irizarry. Data cleaning, exploratory data analysis, model selection and evaluation were performed in R using the packages from the **tidyverse**.

At first, the data was cleaned using the **stringr** package. The release year and title of the movie was extracted from the title column. The genres were unnested by splitting the genres column by the “|” string and joining the data back using the movieId column.

The data was split according to the task into **train** (90%) and **validation** data (10%).

An exploratory data analysis was performed to find the key features of the dataset necessary to build the model for the recommendation system. For this, it was postulated that rating are influenced by the following effects:

1. movie
2. user
3. genre

Next, the model was build sequentially. The basic model was simply based on the mean rating of the training dataset where every movie is predicted to have the same rating. In the next step, the effect of the rated movie was included by adding the average residual effect of the rating per movie to the model. Afterwards, the effect of the user was included by adding the average residual effect of the rating per user. Since genre also has an effect on the rating, the sum of the residual effect of the genre per movie was added, since some movies are categorized in multiple genres to the model. To avoid skewing the data, a regularized approach was used where large estimates with small sample sizes are penalized when calculating the effects. The parameter  $\lambda$  was tuned using cross-validation using values betwenn 0 and 10 in increments of 0.5.

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

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

## Data cleaning

```
# the title string was separated into title and year by string matching
title_str <- as.data.frame(str_match(movielens$title, pattern = "(.*)\\s\\((\\d{4})\\)"))

# the nested genre information was split into multiple rows per movie with one genre each
genres <- str_split(movielens$genres, fixed("|"), simplify = TRUE) %>%
  as.data.frame() %>%
  bind_cols(select(movielens, movieId)) %>%
  distinct(.keep_all = TRUE) %>%
  gather(col, genre, -movieId) %>%
  filter(genre != "") %>%
  select(-col)

# the movie title, year and genre information was added back to the original dataset and only relevant
movielens <- movielens %>%
  mutate(movie_year = as.character(title_str[,3]),
         movie_title = as.character(title_str[,2])) %>%
  left_join(genres, by = "movieId") %>%
  rename(movie_genre = genre) %>%
  select(userId, rating, movieId, movie_title, movie_genre)
```

2

userId	rating	movieId	movie_title	movie_genre
1	5	122	Boomerang	Comedy
1	5	122	Boomerang	Romance
1	5	185	Net, The	Action
1	5	185	Net, The	Crime
1	5	185	Net, The	Thriller
1	5	231	Dumb & Dumber	Comedy
1	5	292	Outbreak	Action
1	5	292	Outbreak	Drama
1	5	292	Outbreak	Sci-Fi
1	5	292	Outbreak	Thriller

## Split in training and validation data

The data was split into train and validation data using the code provided in the task:

```
# Validation set will be 10% of MovieLens data
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,]

# 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)

## Joining, by = c("userId", "rating", "movieId", "movie_title", "movie_genre")

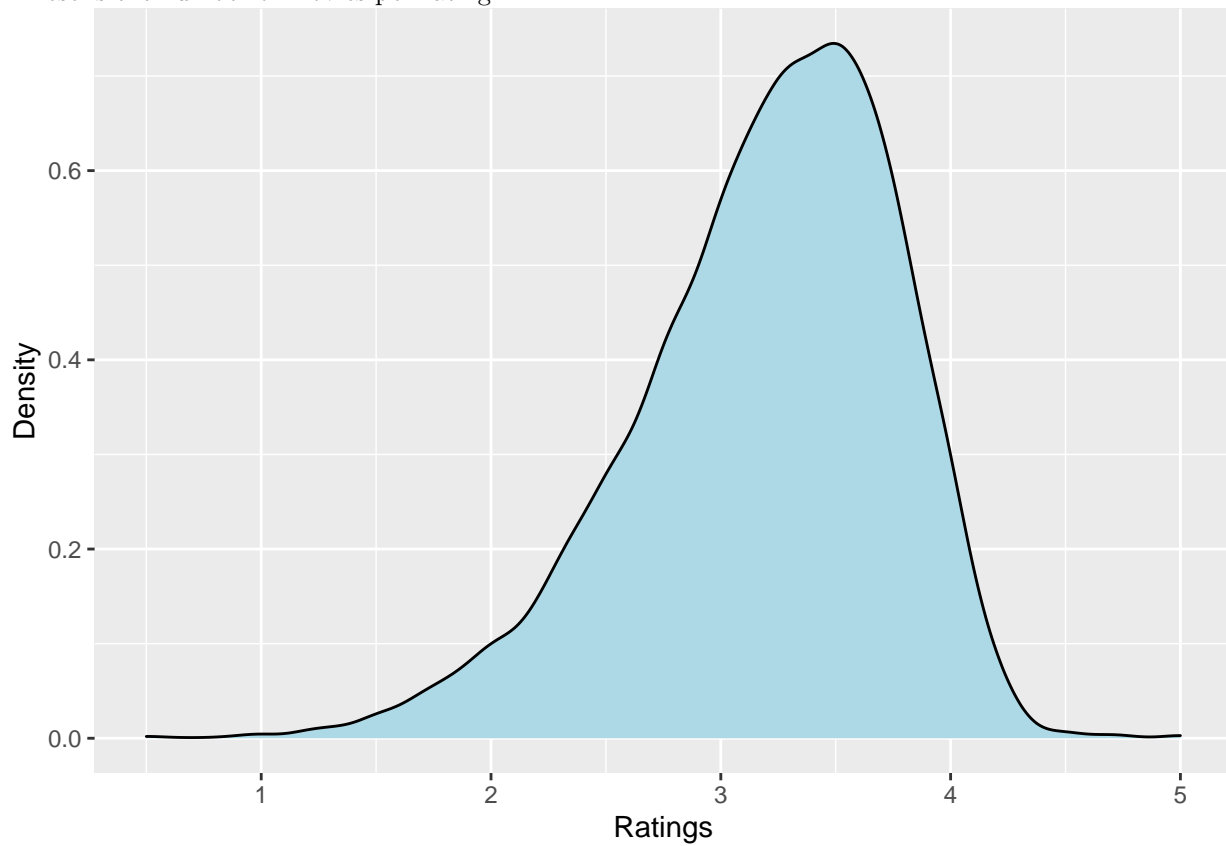
edx <- rbind(edx, removed)

rm(dl, ratings, movies, test_index, temp, movielens, removed)
#
# fwrite(edx, "ML_edx_training_data.tsv", sep = "\t")
# fwrite(validation, "ML_validation_data.tsv", sep = "\t")
```

## Exploratory data analysis

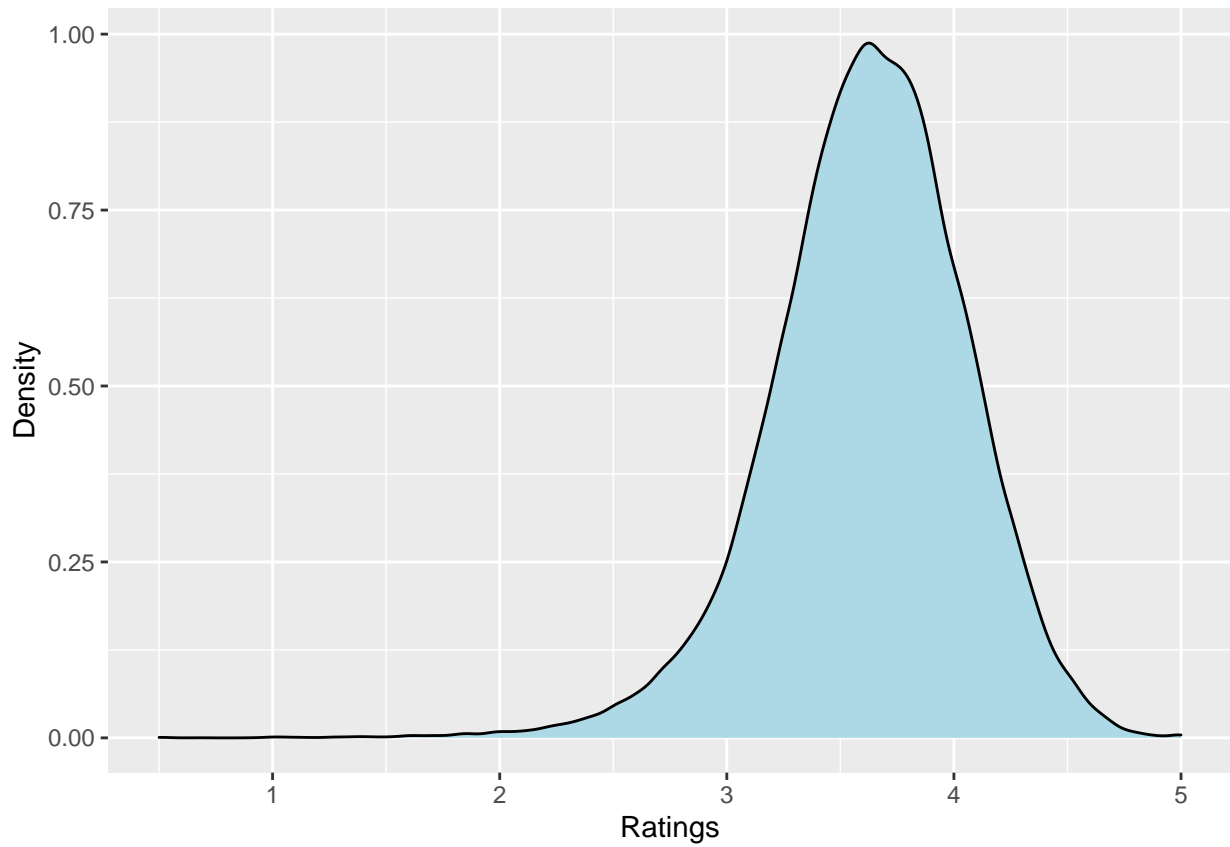
For the exploratory analysis, the effects of the rated movie, the user rating the movie and the genre of the rated movie were analyzed.

These is the number of movies per rating:

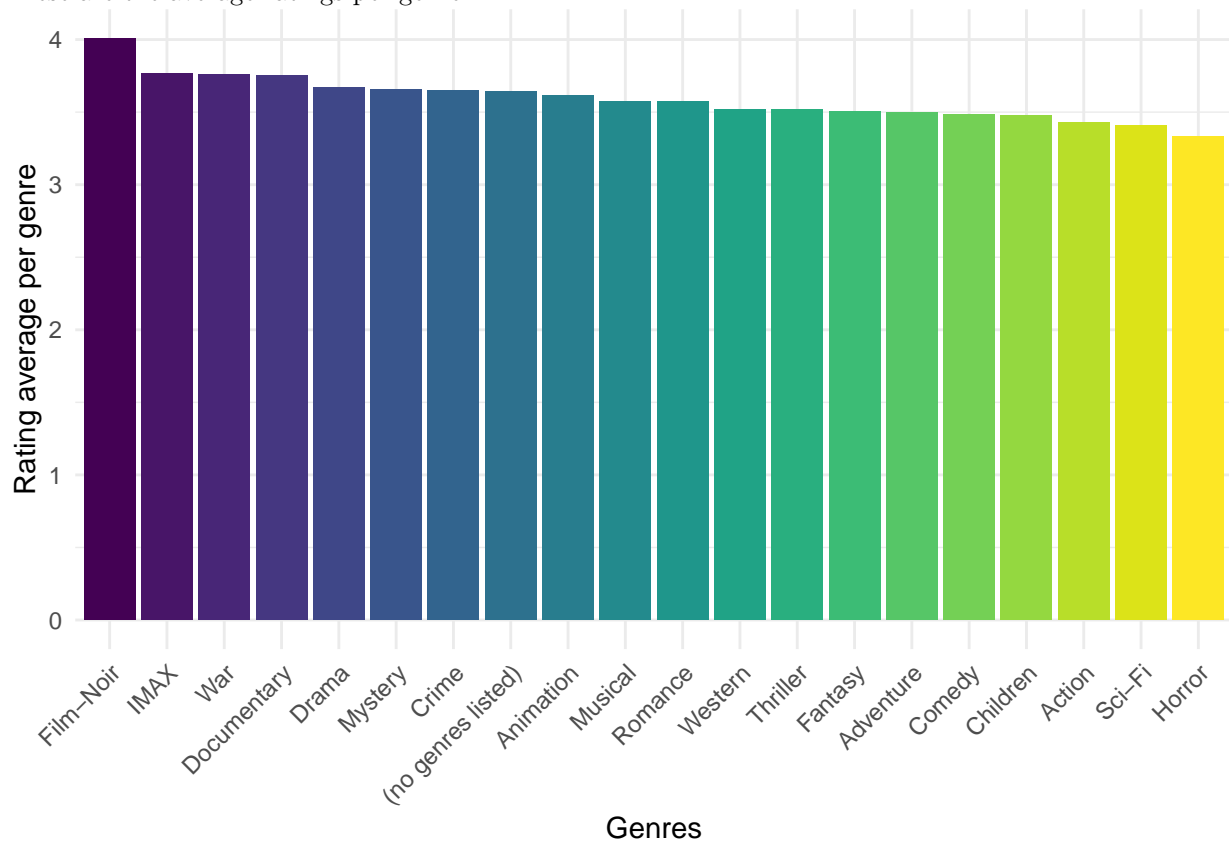


The distribution shows an effect of the movie on the ratings. Often-rated movies have an above-average rating.

These is the distribution of users per rating:



These are the average ratings per genre:



## Data setup

The dataset was split into training and test data by splitting 10% of the dataset for testing.

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

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

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

# exclude users and movies that do not appear in training data
edx_test <- temp %>%
  semi_join(edx_train, by = "movieId") %>%
  semi_join(edx_train, by = "userId")

# add rows removed from test set back into training set
removed <- anti_join(temp, edx_test)

## Joining, by = c("userId", "rating", "movieId", "movie_title", "movie_genre")

edx_train <- rbind(edx_train, removed)

rm(temp, removed)
```

## Modelling

### Basic model

The most basic model for predicting the movie ratings  $Y_{u,i}$  would be the average of all ratings  $\hat{\mu}$  and the independent sampling errors  $\epsilon_{u,i}$ :

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

where  $\hat{\mu}$  is 3.5268879. The RSME is 1.0521194.

```
rmse_all <- tibble(method = "mu_hat", RMSE = rmse_mu)
rmse_all %>%
  head(10) %>%
  kable(format.args = list(digits = 4)) %>%
  kable_styling(full_width = FALSE, position = "center")
```

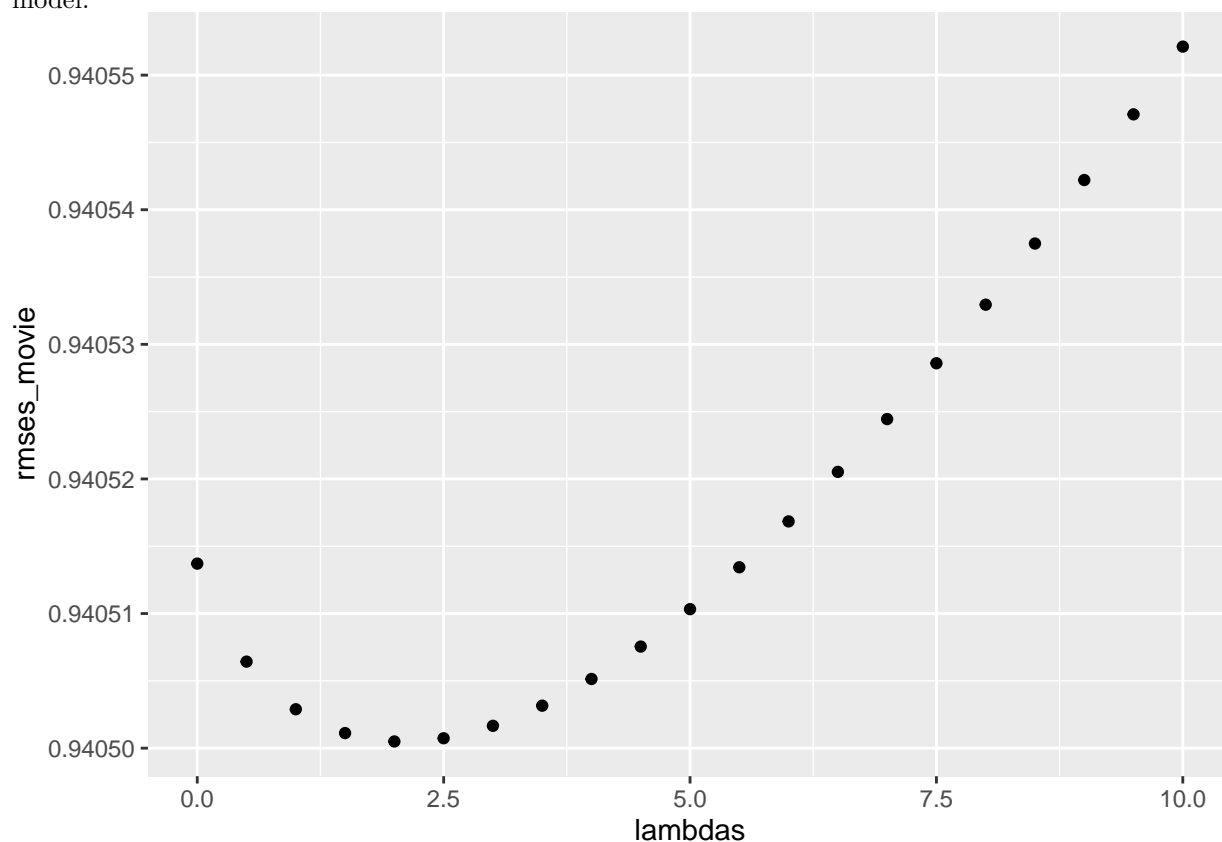
method	RMSE
mu_hat	1.052

## Movie effect

Since certain movies are rated higher than others, the addition of a movie-specific effect  $b_i$  is added to the model:

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

Using cross-validation to tune the parameter  $\lambda$ , the value of  $\lambda$  with the smallest RMSE was chosen for the model.



In this case, the  $\lambda$  is 2.

The least squares estimate for this case is simply the average of the true rating  $Y_{u,i}$  minus the average rating  $\hat{\mu}$  per movie. Adding the movie effect of the model improves the RMSE to 0.940505.

```
rmse_all <- rbind(rmse_all, tibble(method = "+ movie effect (regularized)", RMSE = min(rmses_movie)))
rmse_all %>%
  head(10) %>%
  kable(format.args = list(digits = 4)) %>%
  kable_styling(full_width = FALSE, position = "center")
```

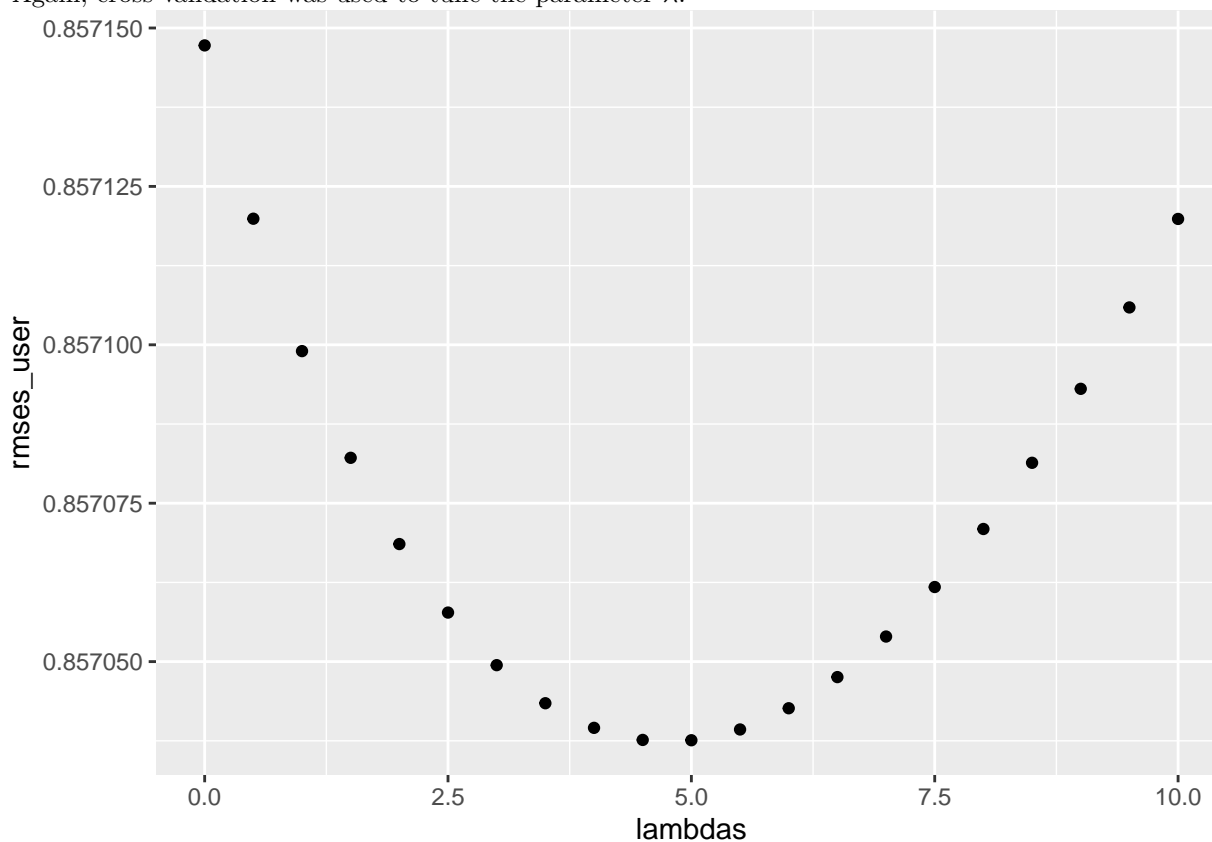
method	RMSE
mu_hat	1.0521
+ movie effect (regularized)	0.9405

## User effect

Similarly, certain users generally rate higher or lower. We can model this as the user effect  $b_u$ :

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

Again, cross-validation was used to tune the parameter  $\lambda$ .



In this case, the  $\lambda$  is 5.

This effect can be estimated by computing the average of each rating  $Y_{u,i}$  minus the average rating  $\hat{\mu}$  and minus the movie effect  $b_i$  per user. This improves the RMSE further to 0.8570376.

```
rmse_all <- rbind(rmse_all, tibble(method = "+ user effect (regularized)", RMSE = min(rmses_user)))
rmse_all %>%
  head(10) %>%
  kable(format.args = list(digits = 4)) %>%
  kable_styling(full_width = FALSE, position = "center")
```

method	RMSE
mu_hat	1.0521
+ movie effect (regularized)	0.9405
+ user effect (regularized)	0.8570

## Final model

The final model that performed best on the training data, as shown by the lowest RMSE in the test data, incorporates the movie-, user- and genre-specific effects. The change in the RMSE when adding the genre-specific effect is low but still improves the RMSE.

Using the final model with the optimized parameter  $\lambda$  of 5, the model can now be trained on the entire **train** dataset before being applied to the final **validation** dataset. This last step will provide the final RMSE.

```

lambda <- lambdas[which.min(rmses_user)]

b_movie <- edx %>%
  group_by(movieId) %>%
  summarize(b_movie = sum(rating - mu)/(n()+lambda), .groups="keep")

b_user <- edx %>%
  left_join(b_movie, by = "movieId") %>%
  group_by(userId) %>%
  summarize(b_user = sum(rating - mu - b_movie)/(n()+lambda), .groups="keep")

preds <- validation %>%
  left_join(b_movie, by = "movieId") %>%
  left_join(b_user, by = "userId") %>%
  group_by(movieId, userId) %>%
  mutate(pred = mu + b_movie + b_user) %>%
  .$pred
rmse_final <- rmse(validation$rating, preds)

```

The RMSE of the final model is 0.8568824.

## Conclusion

In this report, a movie recommendation system was built using the movielens data. The aim was to build a model with an RMSE below 0.86490. The final model incorporates the movie-, user- and genre-specific effects that the exploratory data analysis has shown to be relevant. The RMSE could be further reduced by incorporating more effects, such as the year of the movie or the date and time of the rating.

## Limitations

A few limitations can be noted about the presented project. The effects and predictions could have been obtained using a linear regression model. However, this would be too computationally expensive. Furthermore, the cross-validation could have been performed using more finely incremented values of lambda, which was also too computationally expensive. Finally, other forms of machine learning algorithms could have been used, such as k-nearest neighbors or random forest approaches.