MovieLens Capstone project report

Harald Lie

2020-Feb-09

1. Introduction

This report describes the MovieLens project that is one of two projects required for the last course in the "Professional Certificate in Data Science" programme at Harvard University. MovieLens is a research site run by GroupLens Research at the University of Minnesota. MovieLens users rate movies they have seen. Based on their ratings, the users receive personalized predictions for other movies.
GroupLens Research makes several datasets available for research purposes. The dataset that I have used for this project, the MovieLens 10M Dataset[1], is one of these. It includes more than 10 million movie ratings and 95,580 tags applied to 10,681 movies by 71,567 MovieLens users. The dataset was released in 2009.
The project objective is to develop a model for high-quality movie predictions and to document the results, the process and techniques used in this report. The final model that I submit in this report, the "Movie + user with regularization" model, has an Root-mean-square deviation (RMSE) of 0.8648170 which meets the requirement for the highest score on the RMSE-part of the project.

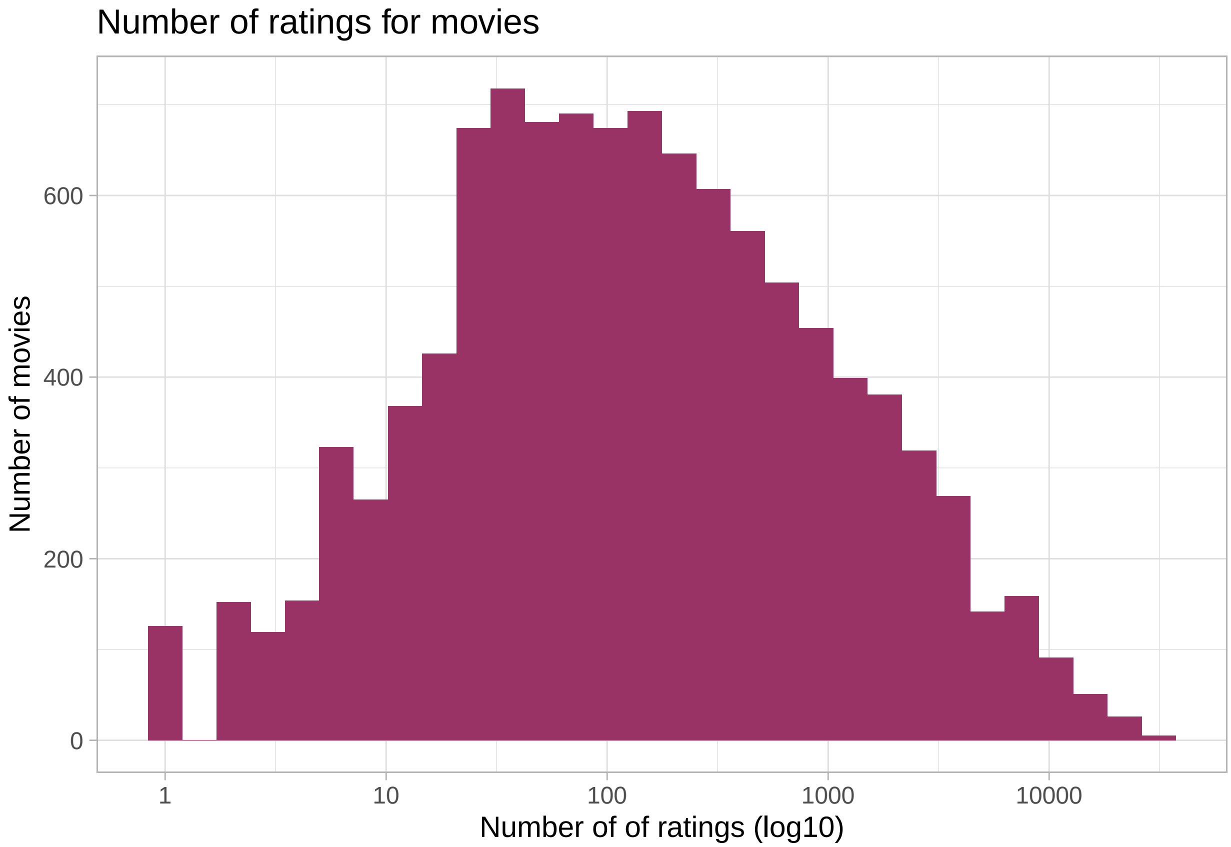
2. Analysis

This chapter describes data wrangling, data exploration, and the modeling approach I decided on.
The course instructors provided the initial code to download the MovieLens dataset, to wrangle the data into a table suitable for analysis, and to split the dataset into two parts: The 'edx' set used for training various models and the validation set used to test the models. The edx set has around 9 million ratings and the validation set has around 1 million ratings. We were under strict instructions to not use the validation set for testing purposes until the final model run. Therefore, I split the edx set into a training set ("edx_test") with ca. 8.1 million rows and a validation set ("edx_train") with ca. 0.9 million rows for the modeling work.

Number of ratings

For data exploration, I started out with looking at the variability of number of movie ratings. It seemed like the most natural place to start. The figure below shows a histogram of the movie ratings in the edx dataset.

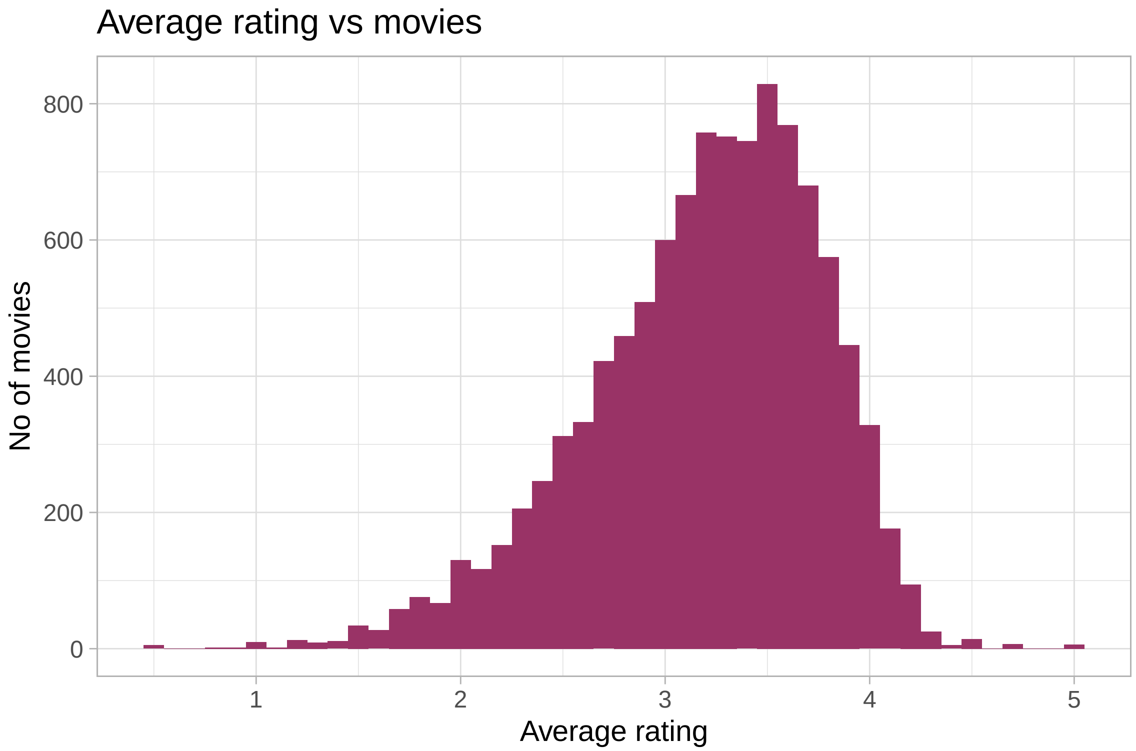
[1] F. Maxwell Harper and Joseph A. Konstan. 2015. The MovieLens Datasets: History and Context. ACM Transactions on Interactive Intelligent Systems (TiiS) 5, 4, Article 19 (December 2015), 19 pages.



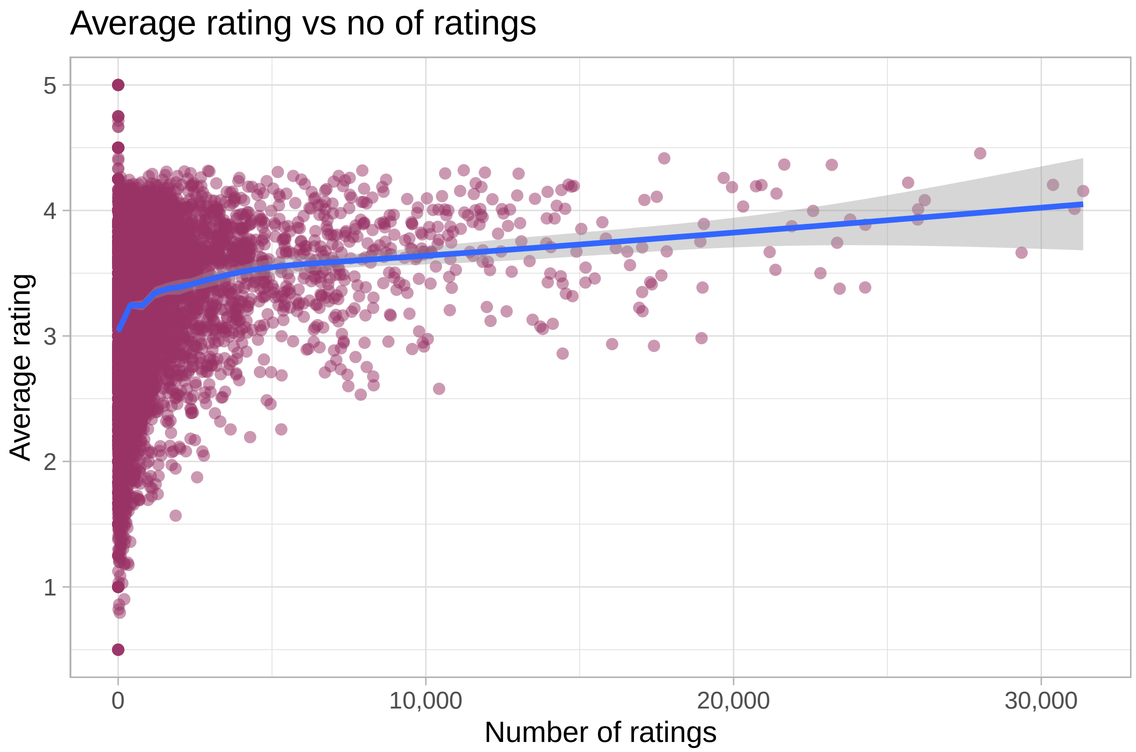
There is a large variability in the number of movie ratings: 126 movies have only one rating, the median number of ratings is 122, while one movie has been rated more than 30 000 times. Please note that the x-axis is shown on a log10-scale.

Movie ratings

What about the ratings themselves? The chart below shows a high variability between movie ratings as well. The average rating is ca. 3.5 stars, and the standard deviation is ca. 1.05 stars.

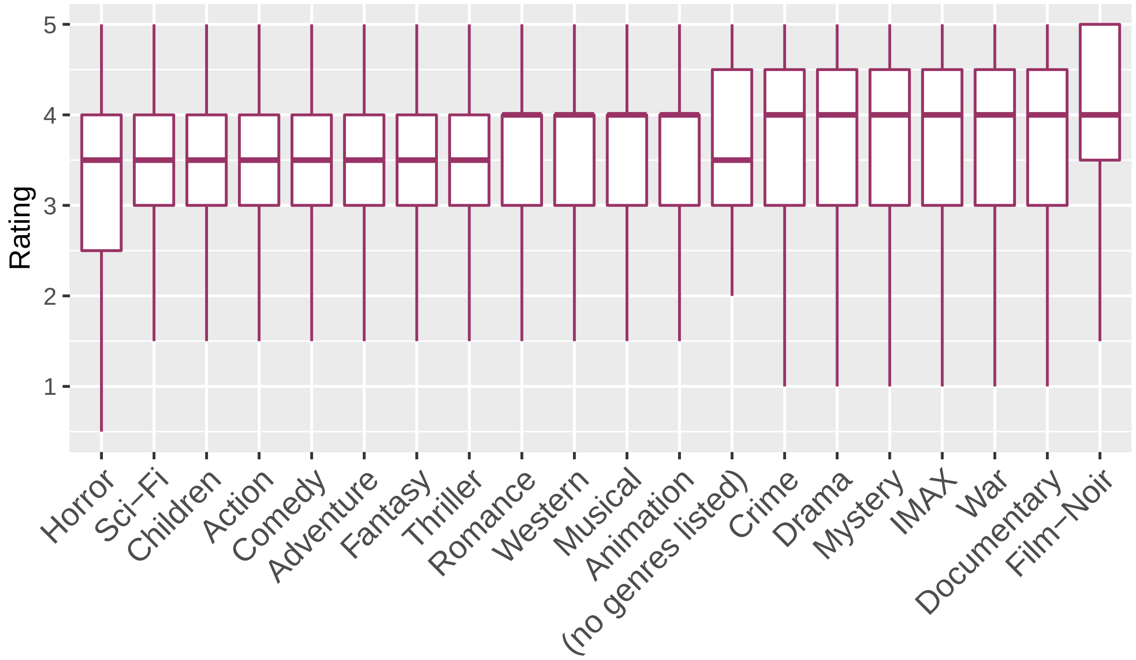


The chart below shows that there is a slight, positive correlation between how often a movie has been rated and the average rating.



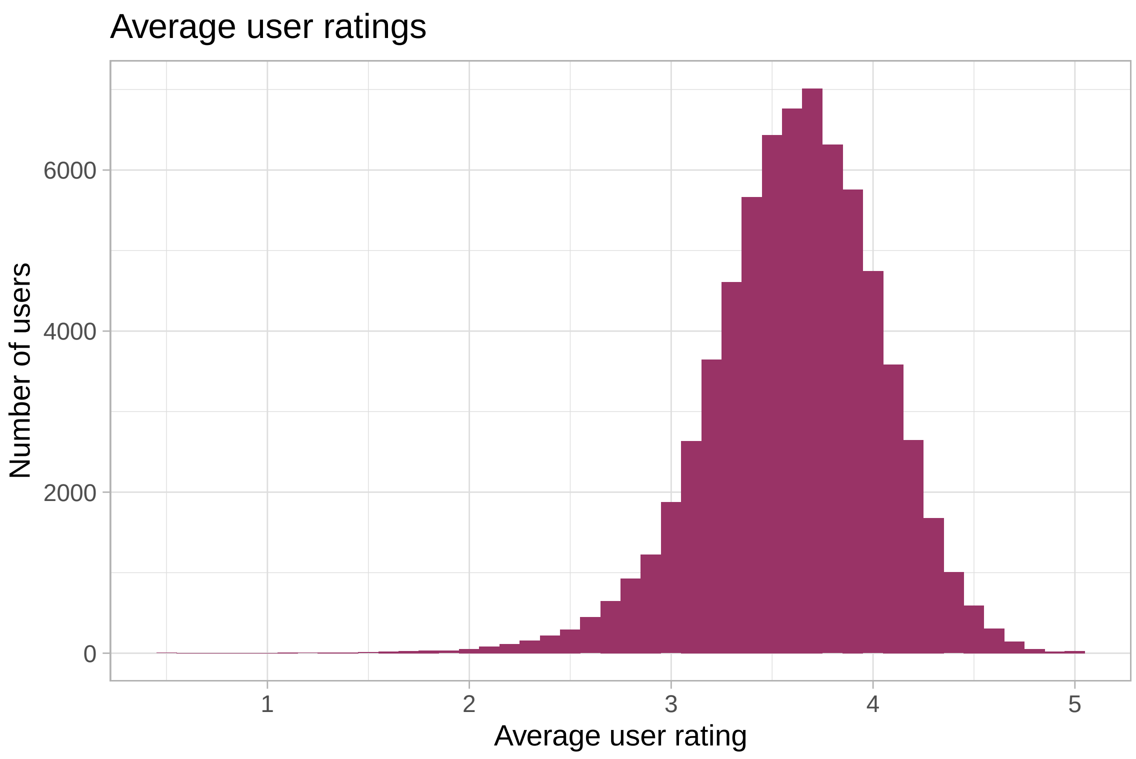
Genres

Are some genres more popular in terms of ratings than others? The chart below shows that the variations between genres is not large.



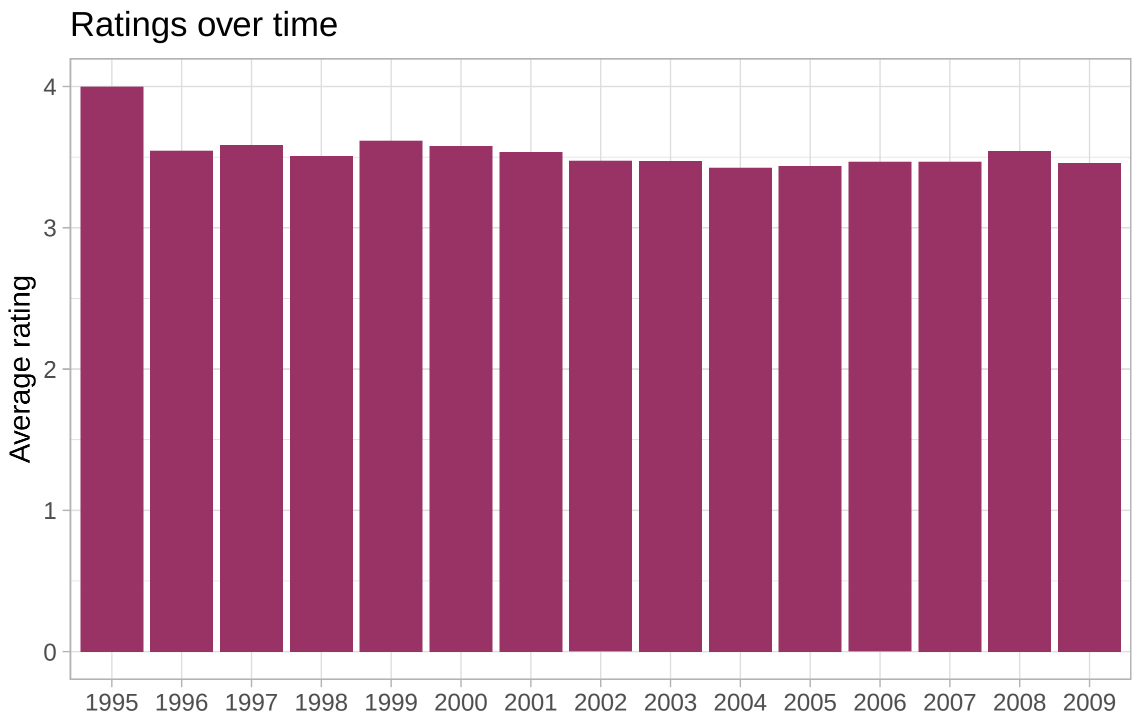
Variability between users

The chart below shows that different users vary quite a bit in how generous they are with their movie ratings. The user ratings appear to be normally distributed with an average user rating of 3.6.



Variability over time

The chart below show stable average ratings over time at ca 3.5 except for the first year where average was ca. 4.0.



Based on this, the most important variables to model is the movie rating itself and variations between users. The genre and the rating year show less variation.

3. Modeling and RMSE - edx data only

This chapter desribes the modeling work I did with the edx data set that I split into a training set and test set. I started out with defining the RMSE function which is similar to standard deviation: The RMSE calculates the typical error when making a prediction and has the following definition:

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

where N is the the number of user/movie combinations and the sum occurring over all these combinations.

```r
### Define the loss function (RSME)

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

A first model - just the average

In the first model I simply took the mean movie rating in the edx training set and used that as the predictor.

```r
### A first model - just the average rating for all movies

mu_edx <- mean(edx_train$rating)
# average movie rating is ca 3.5 - let us use that as the predictor

## Run on edx set
rmse1_edx <- RMSE(edx_test$rating, mu_edx)
```

The average movie rating is 3.5124556 on the training set. The RMSE is 1.0600537 so the typical prediction error is more than one star.

Include movie effects

We know there is high variability between movie ratings, and intuitively this is the most important predictor in our dataset. The model is based on this function:

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

where:
* $\hat{\mu}$ is the mean
* $\varepsilon_{i,u}$ are the independent errors sampled from the same distribution centered at 0
* b_i is a measure for the popularity of movie i

Let us run the following code to see:

```r
avgs_movie_edx <- edx_train %>%
  group_by(movieId) %>%
  summarize(b_i = mean(rating - mu_edx))

```

```
pred_movie <- edx_test %>%
  left_join(avgs_movie_edx, by='movieId') %>%
  mutate(movie_pred = mu_edx + b_i) %>%
  pull(movie_pred)

# calculate RMSE
rmse2_edx <- RMSE(edx_test$rating, pred_movie)
```

The RMSE has been reduced from 1.0600537 to 0.9429615. Not bad!

Include movie and user effects

But can we do even better? We have already established that there is a large variability between users, so let us try and include them in the model using this function:

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

where:
* $\hat{\mu}$ is the mean
* $\varepsilon_{i,u}$ are the independent errors sampled from the same distribution centered at 0
* b_i is a measure for the popularity of movie i
* b_u is a measure for the variation between users

Here's the code:

```
## edx data set
avgs_user_edx <- edx_train %>%
  left_join(avgs_movie_edx, by='movieId') %>%
  group_by(userId) %>%
  summarize(b_u = mean(rating - mu_edx - b_i))

pred_user <- edx_test %>%
  left_join(avgs_movie_edx, by='movieId') %>%
  left_join(avgs_user_edx, by='userId') %>%
  mutate(pred = mu_edx + b_i + b_u) %>%
  pull(pred)

# calculate RMSE
rmse3_edx <- RMSE(edx_test$rating, pred_user)
```

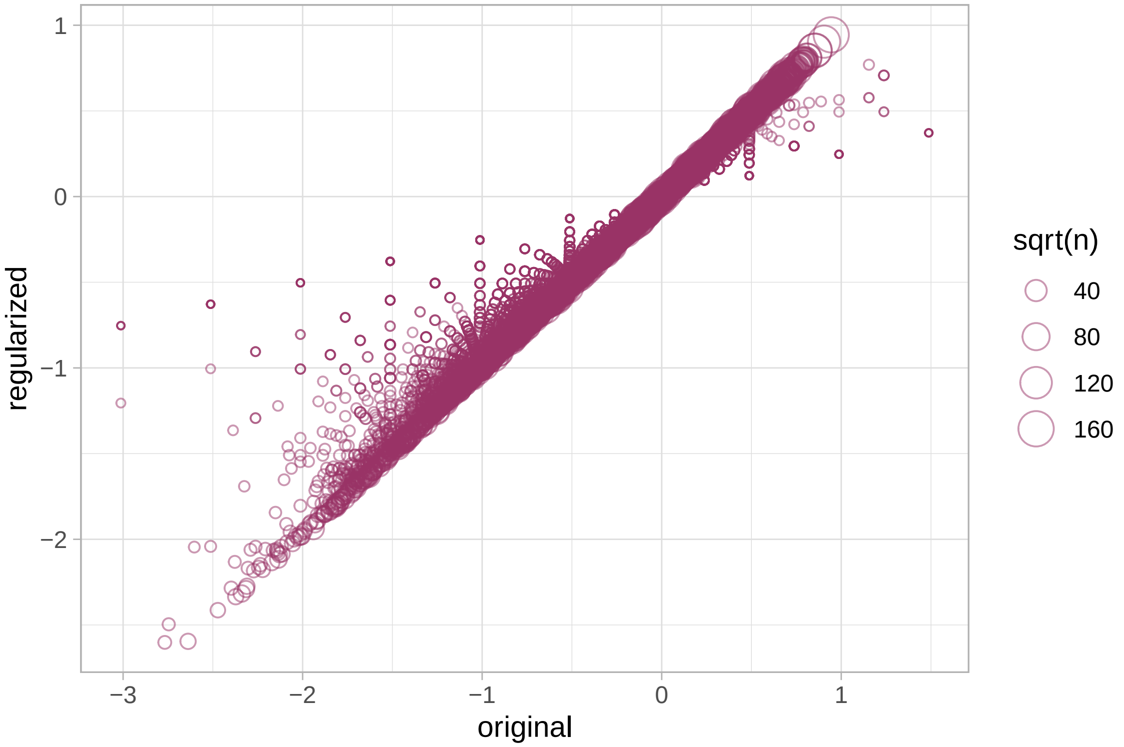
When including user information in the model, the predictions improve from 0.9429615 to an RMSE of 0.8646843.

Include regularization

Regularization is an important concept in mathematics, statistics, and computer science. Regularization reduces the total variability in a dataset by penalizing large estimates that come from small sample sizes. In our dataset, for example, we have more than 100 movies with just one rating. With fewer people rating a movie, we have more uncertainty and higher risk for prediction errors. Therefore we use a regularization formula to penalize movies with a small number of ratings.

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

In the example below, I have used Lambda = 3 and plotted the original and regularized estimates for movie variation. The plot shows that some of the original estimates - the ones with a low number of ratings - have been reduced.

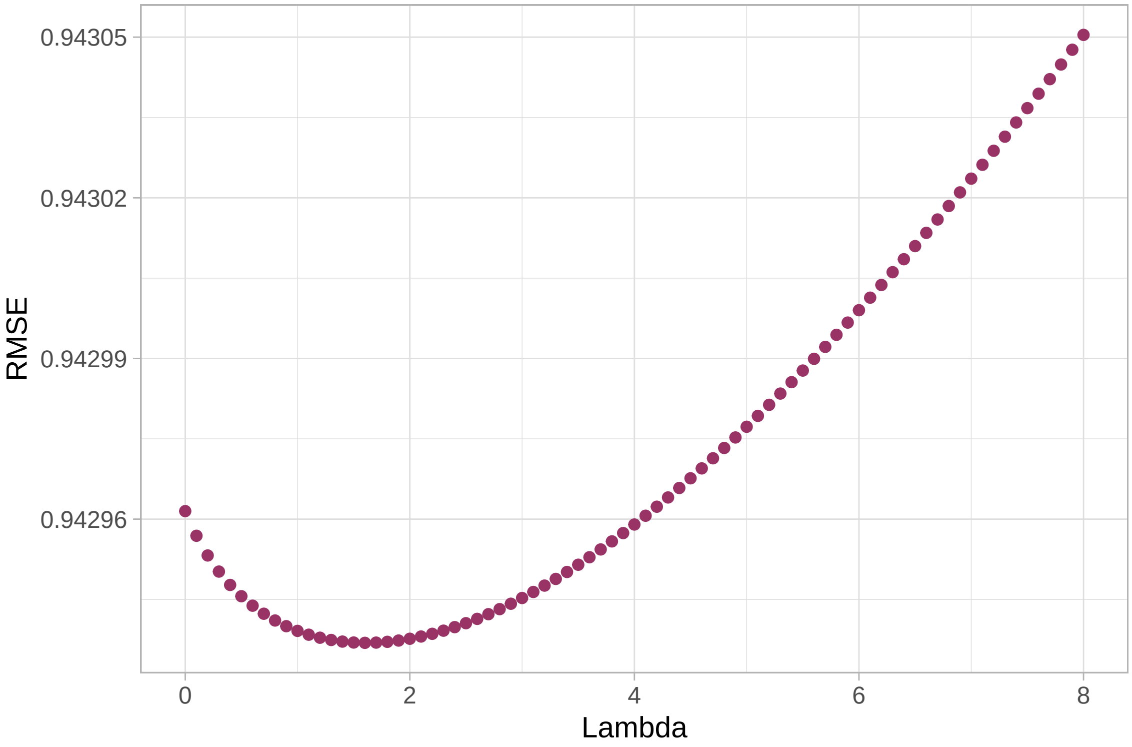


With Lambda = 3, the first attempt at regularization gives an RMSE of 0.9429453.
What if we choose other Lambda values? The code below shows how we can use the `sapply` function to calculate RMSE's for various Lambda values.

```r
### Edx set with lambda optimization - movie effects
# Lambda table
lambdas <- seq(0, 8, 0.1)
# Use sapply to calculate RMSE with various Lambdas
reg1_rmse_edx <- sapply(lambdas, function(lambda) {
  # Calculate average per movie
  b_i <- edx_train %>%
    group_by(movieId) %>%
    summarize(b_i = sum(rating - mu_edx) / (n() + lambda))

  # Predict rating and apply to test set
  pred_ratings <- edx_test %>%
    left_join(b_i, by='movieId') %>%
    mutate(pred = mu_edx + b_i) %>%
    pull(pred)
  # Predict the RMSE on the test set
  return(RMSE(edx_test$rating, pred_ratings))
  })
```

The plot below shows the RMSE for various Lambda values:



The Lambda that minimizes the RMSE is 1.6, and this gives and RMSE of 0.9429369. This is a little lower than the RMSE we achieved when using Lambda = 3.

What happens if we regularize both the movie and the user effects? The code below does just that:

```
#### Regularization with movie and user effects - edx train set
# Use sapply to calculate RMSE with various lambdas
reg2_rmse_edx <- sapply(lambdas, function(lambda) {
  # Calculate movie average
  b_i <- edx_train %>%
    group_by(movieId) %>%
    summarize(b_i = sum(rating - mu_edx) / (n() + lambda))
  # Calculate user average
  b_u <- edx_train %>%
    left_join(b_i, by='movieId') %>%
    group_by(userId) %>%
    summarize(b_u = sum(rating - b_i - mu_edx) / (n() + lambda))

  # Predict ratings on the test set
  predicted_ratings <- edx_test %>%
    left_join(b_i, by='movieId') %>%
    left_join(b_u, by='userId') %>%
    mutate(pred = mu_edx + b_i + b_u) %>%
    pull(pred)
  # Predict the RMSE on the test set
  return(RMSE(edx_test$rating, predicted_ratings))
})

# Find minimum RMSE
reg2_rmse_edx <- min(reg2_rmse_edx)
```

With both user and movie effect regularized, the RMSE is reduced to 0.8641362 on the test set.

4. Modeling and RMSE - final run

This chapter desribes the modeling work I did with the edx and validation datasets that the course instructors provided the code to develop. The metodology and code is simlar to chapter 3, but I use the final datasets in this chapter. Please note that the validation dataset is only used for reporting the final model and final RMSE value.
Here is the code for applying the code for the final model - "Movie + user with regularization" - to the validation data set.

```
# Calculate the average rating for all movies in the train set
mu_val <- mean(edx$rating)

# Lambdas table
lambdas <- seq(0, 8, 0.1)

#### Regularization with movie and user effects - Validation set

# Compute the predicted ratings on validation dataset using different values of lambda
reg2_rmse_val <- sapply(lambdas, function(lambda) {
  # Calculate movie average
  b_i <- edx %>%
    group_by(movieId) %>%
    summarize(b_i = sum(rating - mu_val) / (n() + lambda))
  # Calculate user average
  b_u <- edx %>%
    left_join(b_i, by='movieId') %>%
    group_by(userId) %>%
    summarize(b_u = sum(rating - b_i - mu_val) / (n() + lambda))
  # Compute the predicted ratings on validation dataset
  predicted_ratings <- validation %>%
    left_join(b_i, by='movieId') %>%
    left_join(b_u, by='userId') %>%
    mutate(pred = mu_val + b_i + b_u) %>%
    pull(pred)

  # Predict the RMSE on the validation set
  return(RMSE(validation$rating, predicted_ratings))
})

# Find minimum RMSE
reg2_rmse_val <- min(reg2_rmse_val)
```

And the RMSE is 0.864817.

5. RMSE comparison table and conclusion

RMSE table

The table below summarizes the RMSE results from the various models applied to the edx test set that I made. Also, the table include the RMSE result from running the final model on the validation datatset that was provided by the course instructors.

Table 1: RMSE results

method	RMSE_edx	RMSE_validation
Just the average	1.0600537	
Movie effect	0.9429615	
User effect	0.8646843	
Movie w/ regularization	0.9429369	
Movie + user w/ regularization	0.8641362	0.864817028338387

Conclusion

The report describes my work with the movie prediction model. I have documented how I wrangled the data, explored the data, made different models using the edx dataset, and also how I applied the final model to the validation set.
But is the project objective - to develop a model for high-quality move predictions - met? In my view, no. The final model has an RMSE of less than 0.86490 which is good enough to qualify for best grade according to the MovieLens grading rubric. But I would personally be disappointed with a prediction service that was on average almost 0.9 stars off.
Using other variables in the dataset, such as genre and time of rating, could reduce the RMSE slightly. But in order to make signifiant model improvements I think it is necessary to have a wider dataset with more information about the movies and users. For example, I really like the actor Matthew McConaughey, movies from the interwar period, and movies about France. Oh - and I normally enjoy movies more in the movie theater than at home. This is information that would help improve the prediction for my personal movie ratings, but it is also information that is not available in the MovieLens dataset and that limits the analysis.
For future work, it would be valuable to include more independent variables in the analysis. Also, it would be interesting to see whether other modeling methods - such as those found in R's Caret package, could improve the model quality. In the next project, the "Choose your own" project, I will use the Caret package to compare different modeling techniques.