MovieLens Project

Alonso Quijano

12/17/2019

Executive summary

This project consists of the elaboration of a movie recommendation system on R. We will use the MovieLens dataset, which contains 10 million ratings and 100,000 tag applications applied to 10,000 movies by 72,000 users. We have divided the data into two different sets: edx and validation. The edx set will be used for regularization (training our algorithm), while the validation set will be used for validation (testing our algorithm).

We will design a model that will simulate a regression model. However, due to RAM contraints, we will not apply the lm function as it may not work. We will first analyze each variable to check for patterns. Then, we will run the model stetp by step, testing each variable. Finally, we will regularize the data to eliminate biases produced by high variability among estimates.

Below are the R packages that will be needed for this project:

```
library(tidyverse) # used for data manipulation, exploration, and manipulation
library(lubridate) # used for manipulating date class objects
library(kableExtra) # used for formating tables
```

We can examine the variables that are part of the MovieLens dataset:

Outcome variable

- rating: discrete, rating from 0 to 5 given given by a user to a specific movie

Explanatory variables

- userId: categorical, unique Id for each user
- movieId: categorical, unique Id for each movie
- genres: categorical, genre or genres the movie belongs to
- timestamp: continuous, date and time the rating was given
- title: categorical, name of the movie

```
glimpse(edx)
```

```
## Observations: 9,000,055
## Variables: 6
## $ userId    <int> 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1...
## $ movieId   <dbl> 122, 185, 292, 316, 329, 355, 356, 362, 364, 370, 37...
## $ rating    <dbl> 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5...
## $ timestamp <int> 838985046, 838983525, 838983421, 838983392, 83898339...
## $ title     <chr> "Boomerang (1992)", "Net, The (1995)", "Outbreak (19...
## $ genres    <chr> "Comedy|Romance", "Action|Crime|Thriller", "Action|D...
```

Methods

To create the algorithm that will predict the ratings for each movie by each user, we will consider the following variables: movie effects, user effects, genre effects, and time effects.

The model we will use is the following:

$$Y_{u,i} = \mu + b_i + b_u + b_{genre} + b_{time} + \epsilon$$

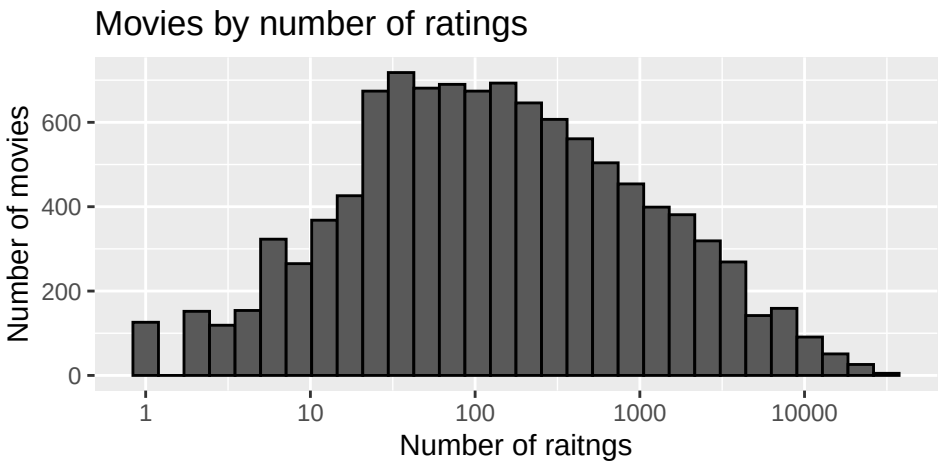
Where μ is the total average of all ratings for all movies by all users; b_i is a movie-specific effect, calculated as the average of $Y - \mu$; b_u is a user-specific effect, calculated as the average of $Y - \mu - b_i$; b_{genre} is a genre-specific effect, calculated as the average of $Y - \mu - b_i - b_u$; b_{time} is a time-specific effect which will consider three variables (year, month, and day of the week), and is calculated as the average of $Y - \mu - b_i - b_u - b_{genre}$; and ϵ is the independent errors sampled from the same distribution centered at 0.

Before we include each of these variables into the model, we need to check on the data whether there seems to be a correlation between each explanatory variable (movieId, userId, genres, timestamp) and the variable we want to predict (ratings).

Movie effects

Some movies get rated more than others. This should no be surprissing as blockbuster movies are watched by millions, while some independent movies are watched just by a few. The following is a distribution of the movies:

```
edx %>% group_by(movieId) %>%
  summarize(n = n()) %>%
  ggplot(aes(n)) +
  geom_histogram(bins = 30, color = "black") +
  scale_x_log10() +
  labs(title = "Movies by number of ratings",
       x = "Number of raitngs",
       y = "Number of movies")
```



Below are the most rated movies:

```
edx %>% group_by(title) %>%
  summarize(n = n(), avg_rating = round(mean(rating),2)) %>%
  arrange(desc(n)) %>% slice(1:10) %>%
  kable(format.args = list(big.mark = ",")) %>% kable_styling()
```

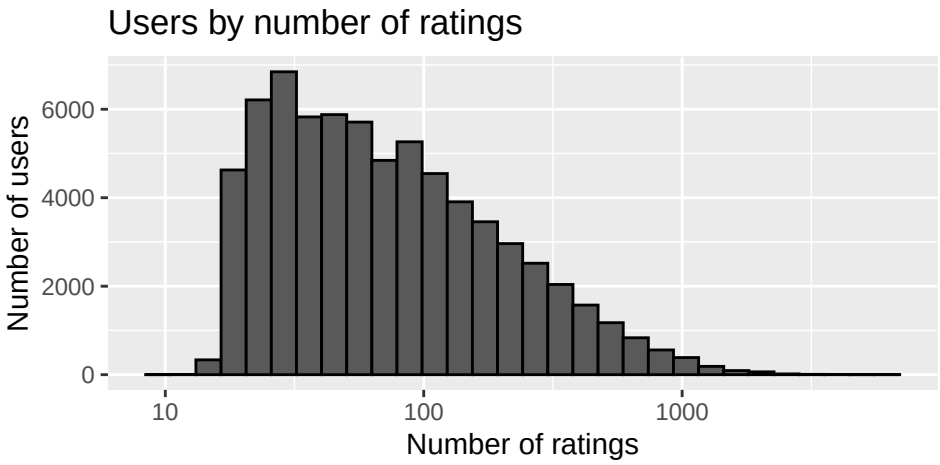
title	n	avg_rating
Pulp Fiction (1994)	31,362	4.15
Forrest Gump (1994)	31,079	4.01
Silence of the Lambs, The (1991)	30,382	4.20
Jurassic Park (1993)	29,360	3.66
Shawshank Redemption, The (1994)	28,015	4.46
Braveheart (1995)	26,212	4.08
Fugitive, The (1993)	25,998	4.01
Terminator 2: Judgment Day (1991)	25,984	3.93
Star Wars: Episode IV - A New Hope (a.k.a. Star Wars) (1977)	25,672	4.22
Apollo 13 (1995)	24,284	3.89

We can see that the most rated movies also have a relatively high rating. This is because blockbuster movies are also famous for being loved by the public and critics.

User effects

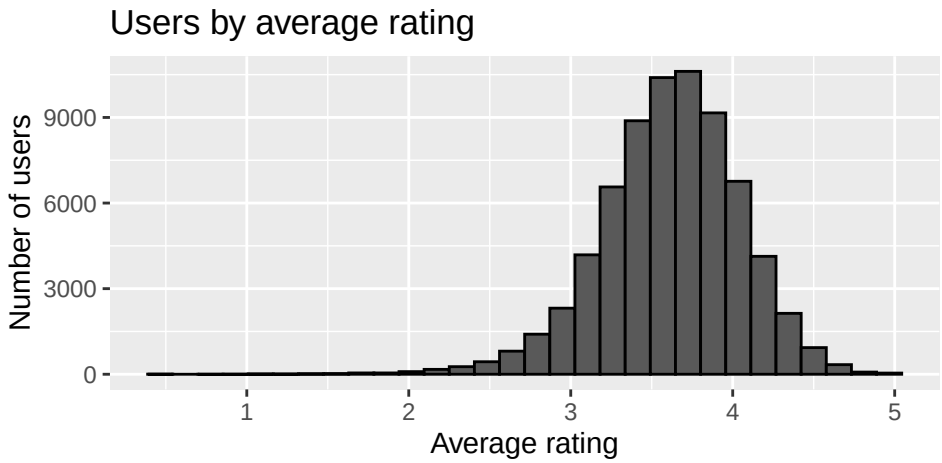
Some users are more active than others. Some are very picky, while some are easy to please. Let's see what the data can show us about the users. Below is the distribution of the users by the number of ratings. We can see some users have watched and rated many movies.

```
edx %>% group_by(userId) %>%
  summarize(n = n()) %>%
  ggplot(aes(n)) +
  geom_histogram(bins = 30, color = "black") +
  scale_x_log10() +
  labs(title = "Users by number of ratings",
       x = "Number of ratings",
       y = "Number of users")
```



We can also illustrate the distribution of the average rating of each user. This distribution, which looks pretty normal, show us that there are a few users who like nearly every movie, and a few who dislike almost every movie. We can see a large concentration of users whose average rating is between 3.5 and 4.

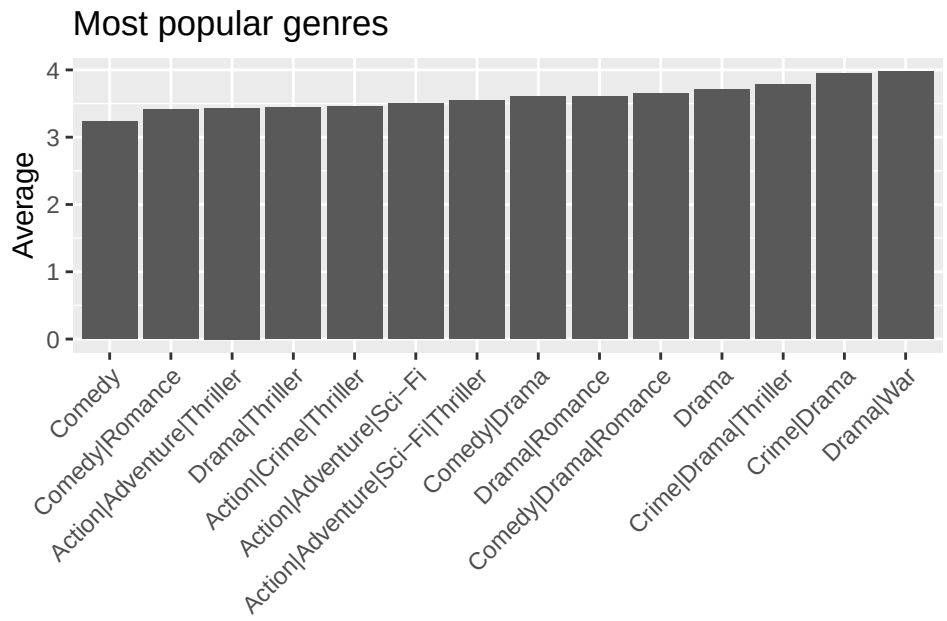
```
edx %>% group_by(userId) %>%
  summarize(avg_rating = mean(rating)) %>%
  ggplot(aes(avg_rating)) +
  geom_histogram(bins = 30, color = "black") +
  labs(title = "Users by average rating", x = "Average rating", y = "Number of users")
```



Genre effects

We can look at the genres with the highest average rating. We only consider those genres with at least 100,000 reviews. We can observe that drama movies are the most popular among the viewers, followed by action movies and comedy movies.

```
edx %>% group_by(genres) %>%
  summarize(n = n(), avg = mean(rating)) %>%
  filter(n >= 100000) %>%
  mutate(genres = reorder(genres, avg)) %>%
  slice(1:20) %>%
  ggplot(aes(x = genres, y = avg)) +
  geom_bar(stat = "identity") +
  theme(axis.text.x = element_text(angle = 45, hjust = 1)) +
  labs(title = "Most popular genres", x = "", y = "Average")
```



Time effects

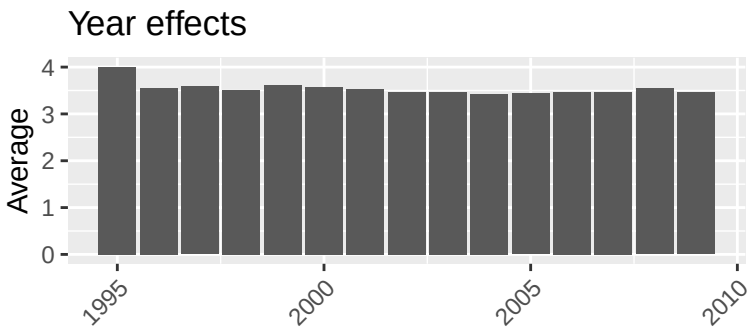
Whether the time someone watches a movie has an effect on whether he or she likes a movie is a little bit more obscure. The average rating does not seem to vary among the days of the week, the months of the year, and the years. However, we will still add those variables to our model to see if we can improve its accuracy.

We first add extra date variables to our data: year, month, and day of the week.

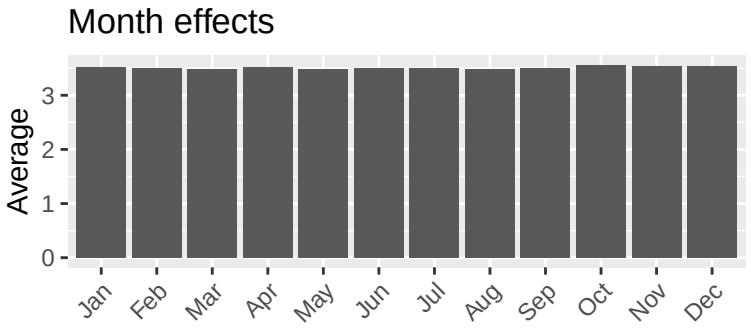
```
edx <- edx %>% mutate(timestamp = as_datetime(timestamp),
                      year = year(timestamp),
                      month = month(timestamp, label = TRUE),
                      wday = wday(timestamp, label = TRUE))

validation <- validation %>%  mutate(timestamp = as_datetime(timestamp),
                                     year = year(timestamp),
                                     month = month(timestamp, label = TRUE),
                                     wday = wday(timestamp, label = TRUE))
```

Now we can illustrate the average rating of each year, month, and day of the week.

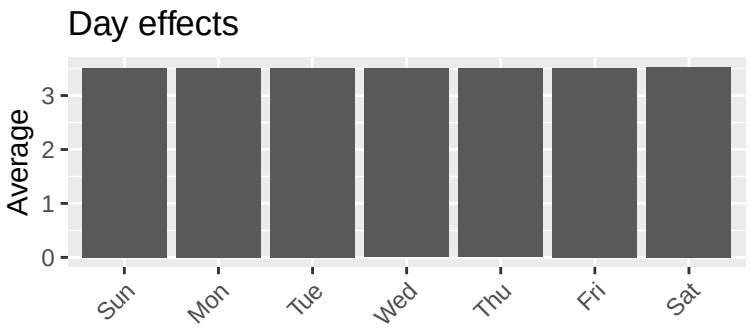
```
edx %>% group_by(year) %>%
  summarize(avg = mean(rating)) %>%
  ggplot(aes(x = year, y = avg)) +
  geom_bar(stat = "identity") +
  theme(axis.text.x = element_text(angle = 45, hjust = 1)) +
  labs(title = "Year effects", x = "", y = "Average")
```




```
edx %>% group_by(month) %>%
  summarize(avg = mean(rating)) %>%
  ggplot(aes(x = month, y = avg)) +
  geom_bar(stat = "identity") +
  theme(axis.text.x = element_text(angle = 45, hjust = 1)) +
  labs(title = "Month effects", x = "", y = "Average")
```




```
edx %>% group_by(wday) %>%
  summarize(avg = mean(rating)) %>%
  ggplot(aes(x = wday, y = avg)) +
  geom_bar(stat = "identity") +
  theme(axis.text.x = element_text(angle = 45, hjust = 1)) +
  labs(title = "Day effects", x = "", y = "Average")
```



The results

The loss function

For measuring the accuracy of our model, we will us the residual mean squared error (RMSE) on the test set. We will define the RMSE as:

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

```
# RMSE

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

Now, we can proceed to run the model step by step, adding each effect to observe how much the RMSE decreases with each additional variable.

Model 1: only the average

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

```
model_1_hat <- mean(edx$rating)

model_1_rmse <- RMSE(validation$rating, model_1_hat)
rmse_results <- data.frame(method = "Just the average", result= model_1_rmse)
```

Model 2: Average + movie effects

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

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

movie_avg <- edx %>% group_by(movieId) %>%
  summarize(b_i = mean(rating - mu))

model_2_hat <- mu + validation %>% left_join(movie_avg, by = 'movieId') %>% .$b_i

model_2_rmse <- RMSE(validation$rating, model_2_hat)

rmse_results <- rbind(rmse_results, data.frame(method = "Movie Effect Model",
                                               result = model_2_rmse))
```

Model 3: Average + movie effects + user effects

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

```
user_avg <-  edx %>% left_join(movie_avg, by= "movieId") %>%
  group_by(userId) %>% summarize(b_u = mean(rating - mu - b_i))

model_3_hat <- validation %>% left_join(movie_avg, by= "movieId") %>%
  left_join(user_avg, by= "userId") %>%
  mutate(pred = mu + b_i + b_u) %>% .$pred

model_3_rmse <- RMSE(validation$rating, model_3_hat)

rmse_results <- rbind(rmse_results,
                      data.frame(method = "Movie Effect + User Effect Model",
                                 result = model_3_rmse))
```

Model 4: Average + movie effects + user effects + genre effect

$$Y_{u,i} = \mu + b_i + b_u + b_{genre} + \epsilon$$

```
genre_avg <- edx %>% left_join(movie_avg, by= "movieId") %>%
  left_join(user_avg, by = "userId") %>%
  group_by(genres) %>% summarize(b_genre = mean(rating - mu - b_i - b_u))

model_4_hat <- validation %>% left_join(movie_avg, by= "movieId") %>%
  left_join(user_avg, by= "userId") %>%
  left_join(genre_avg, by = "genres") %>%
  mutate(pred = mu + b_i + b_u + b_genre) %>% .$pred

model_4_rmse <- RMSE(validation$rating, model_4_hat)

rmse_results <- rbind(rmse_results,
                      data.frame(method = "Movie Effect + User Effect +
                                 Genre Effect Model", result = min(model_4_rmse)))
```

Model 5: Average + movie effects + user effects + genre effect + time effect

$$Y_{u,i} = \mu + b_i + b_u + b_{genre} + b_{time} + \epsilon$$

```
year_avg <- edx %>% left_join(movie_avg, by= "movieId") %>%
  left_join(user_avg, by = "userId") %>% left_join(genre_avg, by = "genres") %>%
  group_by(year) %>%
  summarize(b_year = mean(rating - mu - b_i - b_u - b_genre))

month_avg <- edx %>% left_join(movie_avg, by= "movieId") %>%
  left_join(user_avg, by = "userId") %>% left_join(genre_avg, by = "genres") %>%
  left_join(year_avg, by = "year") %>% group_by(month) %>%
  summarize(b_month = mean(rating - mu - b_i - b_u - b_genre - b_year))

wday_avg <- edx %>% left_join(movie_avg, by= "movieId") %>%
  left_join(user_avg, by = "userId") %>% left_join(genre_avg, by = "genres") %>%
  left_join(year_avg, by = "year") %>% left_join(month_avg, by = "month") %>%
  group_by(wday) %>%
  summarize(b_wday = mean(rating - mu - b_i - b_u - b_genre - b_year - b_month))
```

```
model_5_hat <- validation %>% left_join(movie_avg, by= "movieId") %>%
  left_join(user_avg, by= "userId") %>% left_join(genre_avg, by = "genres") %>%
  left_join(year_avg, by = "year") %>% left_join(month_avg, by = "month") %>%
  left_join(wday_avg, by = "wday") %>%
  mutate(pred = mu + b_i + b_u + b_genre + b_year + b_month + b_wday) %>% .$pred

model_5_rmse <- RMSE(validation$rating, model_5_hat)

rmse_results <- rbind(rmse_results,
                      data.frame(method = "Movie Effect + User Effect +
                                 Genre Effect Model + Time Effect Model",
                                 result = min(model_5_rmse)))

rmse_results %>% kable(format.args = list(big.mark = ",")) %>% kable_styling()
```

method	result
Just the average	1.0612018
Movie Effect Model	0.9439087
Movie Effect + User Effect Model	0.8653488
Movie Effect + User Effect + Genre Effect Model	0.8649469
Movie Effect + User Effect + Genre Effect Model + Time Effect Model	0.8649248

We have managed to lower the RMSE to 0.8649248 in the model, which includes all variables. However, there some variables with very small sample sizes. For example, some movies are rated less, some users are less active, some movies have a rare combination of genres, and a year (1995) has only two observations. This can increase the total variability of the effect sizes, which can largely bias our estimates. We will solve this problem in the following and last model.

Model 6: Average + movie effects (penalized least square)

Regularization permits us to penalize large estimates that are calculated using small sample sizes. We will use a tunning parameter λ that will be used when calculating the average of each estimate. For example, if we previously calculated the specific-movie effect as $\sum_{i=1}^{n_i} \frac{1}{n}(Y_{u,i} - \mu)$, we will now calculate it as $\sum_{i=1}^{n_i} \frac{1}{n+\lambda}(Y_{u,i} - \mu)$.

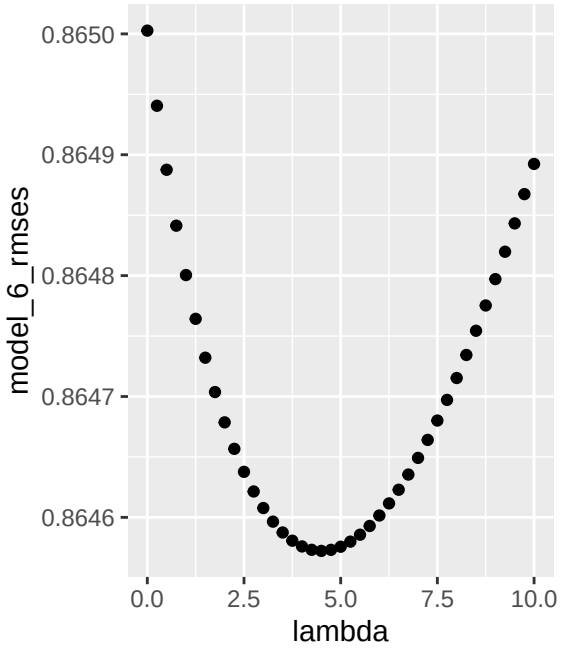
Matematically, if the sample size ni is very large, then the penalty λ is insignificant as $n_i + \lambda \approx \lambda$. However, when the n_i is small, then the estimate is shrunken towards 0.

We will set different λ to see which one improves our estimates the most. We will apply this parameter to all the variables except for the time-specific effects of the months and the days as all of them contain a very large sample size.

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

model_6_rmses <- sapply(lambda, function(l){

  mu <- mean(edx$rating)

  movie_avg_pen <- edx %>% group_by(movieId) %>%
    summarize(b_i = sum(rating - mu)/(n()+l))

  user_avg_pen <- edx %>% left_join(movie_avg_pen, by="movieId") %>%
```

```
    group_by(userId) %>%
    summarize(b_u = sum(rating - mu - b_i)/(n()+l))

  genre_avg_pen <- edx %>% left_join(movie_avg_pen, by="movieId") %>%
    left_join(user_avg_pen, by = "userId") %>% group_by(genres) %>%
    summarize(b_genre = sum(rating - mu - b_i - b_u)/(n()+l))

  year_avg_pen <- edx %>% left_join(movie_avg_pen, by= "movieId") %>%
    left_join(user_avg_pen, by = "userId") %>%
    left_join(genre_avg_pen, by = "genres") %>% group_by(year) %>%
    summarize(b_year = sum(rating - mu - b_i - b_u - b_genre)/(n()+l))

  month_avg_pen <- edx %>% left_join(movie_avg_pen, by= "movieId") %>%
    left_join(user_avg_pen, by = "userId") %>%
    left_join(genre_avg_pen, by = "genres") %>%
    left_join(year_avg_pen, by = "year") %>% group_by(month) %>%
    summarize(b_month = mean(rating - mu - b_i - b_u - b_genre - b_year))

  wday_avg_pen <- edx %>% left_join(movie_avg_pen, by= "movieId") %>%
    left_join(user_avg_pen, by = "userId") %>% left_join(genre_avg_pen, by = "genres") %>%
    left_join(year_avg_pen, by = "year") %>% left_join(month_avg_pen, by = "month") %>%
    group_by(wday) %>%
    summarize(b_wday = mean(rating - mu - b_i - b_u - b_genre - b_year - b_month))

  predicted_rating <- validation %>% left_join(movie_avg_pen, by= "movieId") %>%
    left_join(user_avg_pen, by= "userId") %>% left_join(genre_avg_pen, by = "genres") %>%
    left_join(year_avg_pen, by = "year") %>% left_join(month_avg_pen, by = "month") %>%
    left_join(wday_avg_pen, by = "wday") %>%
    mutate(pred = mu + b_i + b_u + b_genre - b_year + b_month + b_wday) %>% .$pred

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

```
qplot(lambda, model_6_rmses)
```




```
lambda[which.min(model_6_rmses)]
```

```
## [1] 4.5
```

```
rmse_results <- rbind(rmse_results,
                      data.frame(method = "Penalized Movie Effect + User Effect Model +
                                 Genre Effect + Time Effect",
                                 result = min(model_6_rmses)))

rmse_results %>% kable(format.args = list(big.mark = ",")) %>% kable_styling()
```

method	result
Just the average	1.0612018
Movie Effect Model	0.9439087
Movie Effect + User Effect Model	0.8653488
Movie Effect + User Effect + Genre Effect Model	0.8649469
Movie Effect + User Effect + Genre Effect Model + Time Effect Model	0.8649248
Penalized Movie Effect + User Effect Model + Genre Effect + Time Effect	0.8645723

We chose a λ of 4.5. Applying this final model, we have managed to lower the RMSE to 0.8645723, penalizing the estimates with large variability and lowering the bias.

Conclusions

To create this movie recommendation algorithm, we applied a model that simulated a regression model and included different variables (movie effects, user effects, genre effects, and time effects). We first analyzed the data for patterns to check for effects before including each variable into the model. We also provided with various illustrations that helped us find interesting patterns. We used the edx set for training our algorithm

and the validation set for testing. After obtaining the first results of the models, we decided to regularize the data, penalizing large estimates that were calculated using small sample sizes. This approach led us to the optimal model, which gave us a RMSE of 0.8645723.

Limitations of this project and future work

Due to the large size of this dataset, we were not able to apply the lm function or any other method that requires large RAM capacity. We tried to implement a model based on SVD (singular value decomposition), but even a computer with 16Gb of RAM was not able to manage all the data. For future projects, we can work with smaller datasets that allow us to test different models without this constraint.