# MovieLens Report

Eduardo Almada

09/03/2021

## Introduction

In this report I will be reviewing a machine learning challenge set by Netflix back in 2006, the goal is to obtain the minimum root mean square error (RMSE) which is calculated with the given formula:

$$RMSE = \sqrt{\frac{(\sum_{i=1}^{n}(Y_{predicted} - Y)^2)}{N}}$$

For this report the requirements is to obtain a RMSE smaller than 0.86490, five different approaches were made in order to achieve this, which will be discussed in the Methods section. I began inspecting the data already provided by **Edx**.

```
c(Rows = dim(edx)[1], Predictors = dim(edx)[2])
```

```
##       Rows Predictors
##    9000061          6
```

We see 6 predictors and 9000061 rows, after further inspection we can begin our machine learning algorithm to predict the rating given to a movie using the user, genre and movie a predictors.

```
as_tibble(head(sample(edx)))
```

```
## # A tibble: 6 x 6
##   movieId rating userId timestamp title                   genres
##     <dbl>  <dbl>  <int>     <int> <chr>                   <chr>
## 1     122      5      1 838985046 Boomerang (1992)        Comedy|Romance
## 2     185      5      1 838983525 Net, The (1995)         Action|Crime|Thriller
## 3     231      5      1 838983392 Dumb & Dumber (1994)    Comedy
## 4     292      5      1 838983421 Outbreak (1995)         Action|Drama|Sci-Fi|T~
## 5     316      5      1 838983392 Stargate (1994)         Action|Adventure|Sci-~
## 6     329      5      1 838983392 Star Trek: Generations~ Action|Adventure|Dram~
```

## Methods

There were five different approaches to get the minimum RMSE, the simplest model is to use the average rating across every movie and user. The model follows this equation,

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

Where $Yu, i$ is the predicted rating of user $u$ and movie $i$, and $\mu$ is the average rating across the entries

```
mu <- mean(edx$rating)
m1 <- RMSE(validation$rating,mu)
m1
```

```
## [1] 1.060651
```

This was a rough approach, so in order to improve the prediction a new term is introduce to de model $b_i$, this term represents the bias that each movie introduces when predicting, the model would now look lije this:

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

```
# By grouping the movies and scaling the ratings, we obtain b_i as the movie bias
b_i <- edx %>%
  group_by(movieId) %>%
  summarize(b_i = mean(rating - mu))

# Estimates all unknown ratings based on the training set rating average and b_i
bi_ratings <- validation %>%
  left_join(b_i, by='movieId') %>%
  mutate(pred = mu + b_i) %>%
  pull(pred)
# Calculate RMSE
m2 <- RMSE(validation$rating, bi_ratings)
m2
```

```
## [1] 0.9437046
```

This RMSE values looks much better, however it is not what we are looking for, the pass criteria for a great prediction is to have a $RMSE < 0.86490$. The next step to improve the algorithm would be to take into consideration the bias introduced by the User $b_u$, which when applied to our model looks like this:

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

```
# Calculates user bias, b_u
b_u <- edx %>%
  left_join(b_i, by='movieId') %>%
  group_by(userId) %>%
  summarize(b_u = mean(rating - mu - b_i))

# Estimates all unknown ratings based on the training set rating average, b_i and b_u
bu_ratings <- validation %>%
  left_join(b_i, by='movieId') %>%
  left_join(b_u, by='userId') %>%
  mutate(pred = mu + b_i + b_u) %>%
  pull(pred)

# Calculate RMSE
m3 <- RMSE(bu_ratings, validation$rating)
m3
```

```
## [1] 0.8655329
```

*Almost there!*, the next step would be to add the term ,$b_e$, which represents the movie genre bias. For this I take multiple genre movies as one genre, with this considered the model would now look like this:

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

```
# Calculates genres bias, b_e
b_e <- edx %>%
  left_join(b_i, by='movieId') %>%
  left_join(b_u, by = "userId") %>%
  group_by(genres) %>%
  summarize(b_e = mean(rating - mu - b_i - b_u))

# Estimates all unknown ratings based on the training set rating average, b_i, b_u & b_e
be_ratings <- validation %>% left_join(b_i, by='movieId') %>%
  left_join(b_u, by='userId') %>% left_join(b_e, by='genres') %>%
  mutate(pred = mu + b_i + b_u + b_e) %>%
  pull(pred)

# Calculate RMSE
m4 <- RMSE(be_ratings, validation$rating)
m4
```

```
## [1] 0.8651946
```

The improvement was not that much, so as a final step regularization will be applied to the model, Regularization constrains the total variability of the effects sizes by penalizing large estimates from small samples.

The model looks like this:

$$\frac{1}{N}\sum_{u,i}(Y_{u,i} - \mu - b_i - b_u - b_e)^2 + \lambda(\sum_i b_i^2 + \sum_u b_u^2 + \sum_u b_e^2)$$

where $\lambda$ is a tuning parameter, the larger $\lambda$ is, the more the *sum* shrinks. So, cross-validation should be applied inm order to get the best $\lambda$ to reduce the RMSE.

## Results

When doing cross-validation a $\lambda$ vector is created in order to find the best tune.

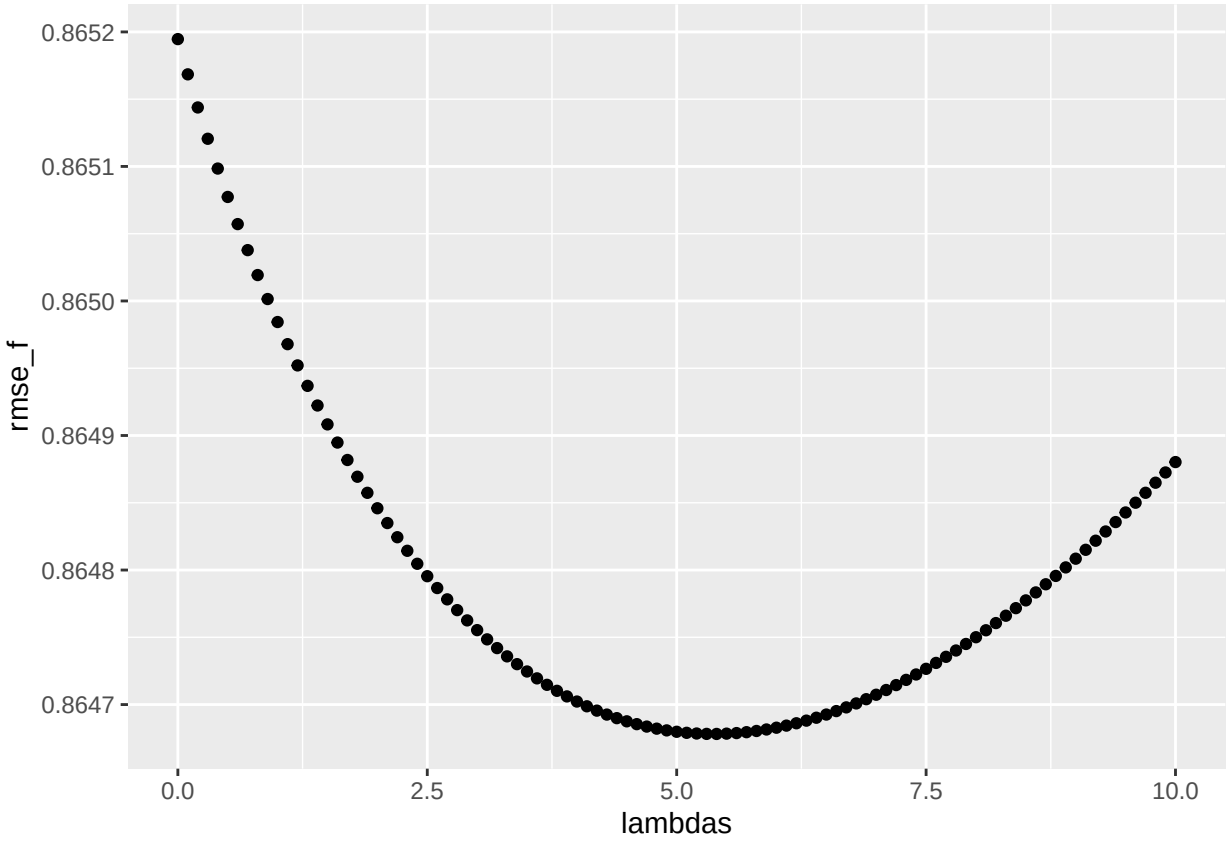
```
index <- which.min(rmse_f)
c(lambda = lambdas[index], RSME = min(rmse_f))
```

```
##    lambda      RSME
## 5.4000000 0.8646782
```

With this we accomplish our goal, building a machine learning algorithm caplable of predicting movie ratings with a $RMSE < 0.86490$.

Where we plotted the lambdas vs the RMSE for visual exploration.

```
qplot(lambdas, rmse_f)
```




## Conclusion

After applying five different methods by including different terms to the average expression, it is concluded that regularization was needed in order to compensate for the large variation within the predictors, the RMSE was reduced from 1.061 to 0.8647. This model has the advantage of using linear models, compared to the ones available in R packages, so the computational time is *better* using this model.

```
Methods <- c(Average_method = m1, Movie_effect = m2, User_effect = m3,
  Genre_effect = m4,Regularized_method = m5)

Results <- data.table(Method = c('Average_method', 'Movie_effect', 'User_effect','Genre_effect','Regula
                Pass_Criteria = ifelse(Methods < 0.86490,'pass','fail'))
Results
```

```
##                Method      RMSE Pass_Criteria
## 1:     Average_method 1.0606506          fail
## 2:       Movie_effect 0.9437046          fail
## 3:        User_effect 0.8655329          fail
## 4:       Genre_effect 0.8651946          fail
## 5: Regularized_method 0.8646782          pass
```