# MovieLens

*JP Pugliese*

*10/29/2019*

## Introduction

This report presents the steps to model the rating of a "movielens" dataset in function of the movie's characteristics such as genres, rating and users. The model uses the root mean squared error to calculate the accuracy of the model. Along the typical linear model some penalty and regularisation methods have been applied. This allowed to obtain an overall accuracy of **0.8648177** using the validation dataset. The model is refine using a train and test subset datasets which belong to the "movielens" dataset.

Datasets preparation, provided in the assignment:

```
if(!require(tidyverse)) install.packages("tidyverse", repos = "http://cran.us.r-project.org")
```

```
## Loading required package: tidyverse

## -- Attaching packages -------------------------------------------------------------------

## v ggplot2 3.2.1     v purrr   0.3.2
## v tibble  2.1.3     v dplyr   0.8.3
## v tidyr   0.8.3     v stringr 1.4.0
## v readr   1.3.1     v forcats 0.4.0

## -- Conflicts -----------------------------------------------------------------------------
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```
if(!require(caret)) install.packages("caret", repos = "http://cran.us.r-project.org")
```

```
## Loading required package: caret

## Loading required package: lattice

##
## Attaching package: 'caret'

## The following object is masked from 'package:purrr':
##
##     lift
```

```
if(!require(data.table)) install.packages("data.table", repos = "http://cran.us.r-project.org")
```

```
## Loading required package: data.table

##
## Attaching package: 'data.table'

## The following objects are masked from 'package:dplyr':
##
##     between, first, last

## The following object is masked from 'package:purrr':
##
##     transpose
```

```r
# MovieLens 10M dataset:
# https://grouplens.org/datasets/movielens/10m/
# http://files.grouplens.org/datasets/movielens/ml-10m.zip

dl <- tempfile()
download.file("http://files.grouplens.org/datasets/movielens/ml-10m.zip", dl)

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")
movies <- as.data.frame(movies) %>% mutate(movieId = as.numeric(levels(movieId))[movieId],
                                           title = as.character(title),
                                           genres = as.character(genres))

movielens <- left_join(ratings, movies, by = "movieId")

# 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
```

```r
# if using R 3.5 or earlier, use `set.seed(1)` instead
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", "movieId", "rating", "timestamp", "title", "genres")
```

```r
edx <- rbind(edx, removed)

rm(dl, ratings, movies, test_index, temp, movielens, removed)
```

## Data exploration

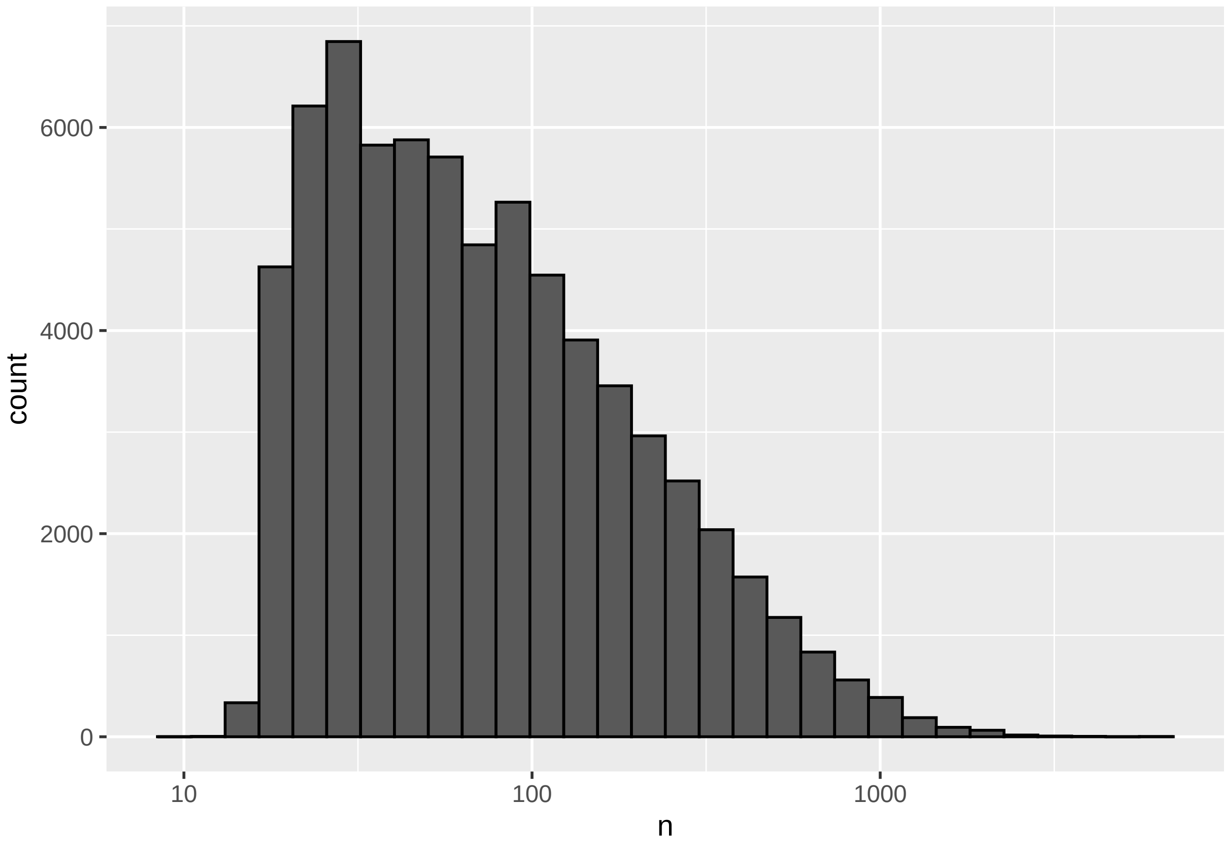
We can see from here that the most common ratings are 4 and 3, with 4 being the highest and then 3:

```r
edx %>% group_by(rating) %>% summarize(count = n()) %>% arrange(desc(count))
```

```
## # A tibble: 10 x 2
##    rating   count
##     <dbl>   <int>
##  1    4   2588430
##  2    3   2121240
##  3    5   1390114
##  4    3.5  791624
##  5    2    711422
##  6    4.5  526736
##  7    1    345679
##  8    2.5  333010
##  9    1.5  106426
## 10    0.5   85374
```

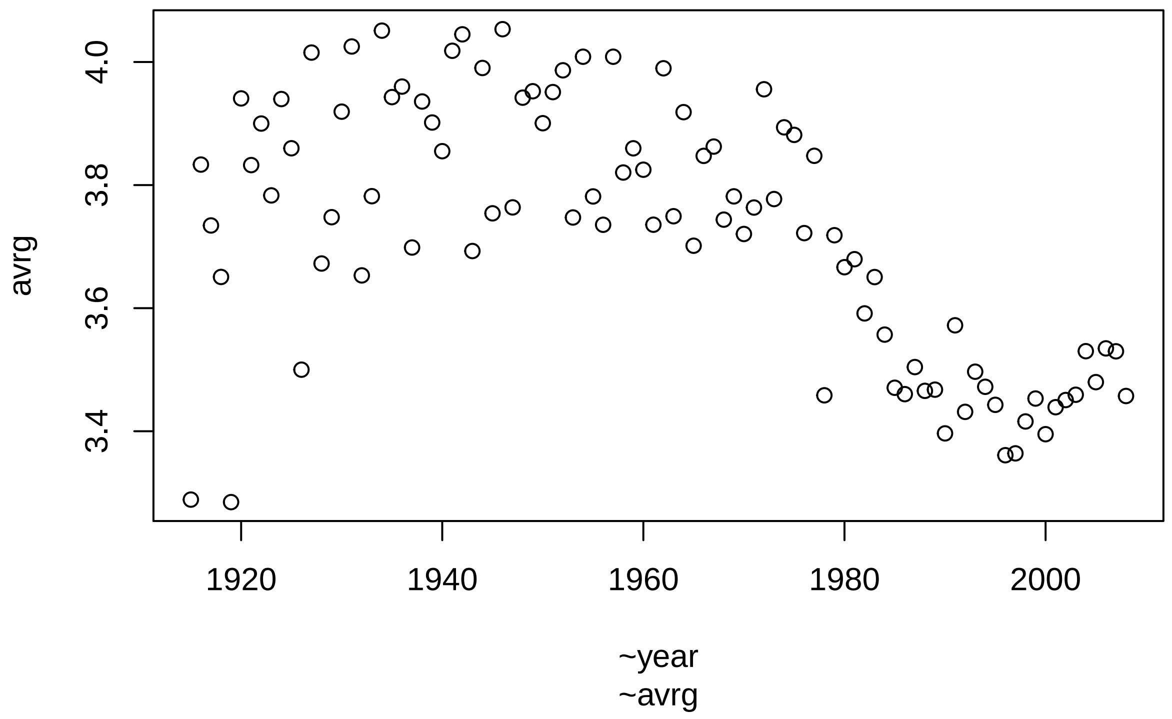
Looking at how user rate movies. This shows that not every users are rating as often as others:

```
edx %>% count(userId) %>% ggplot(aes(n)) + geom_histogram(bins = 30, color = "black") + scale_x_log10()
```



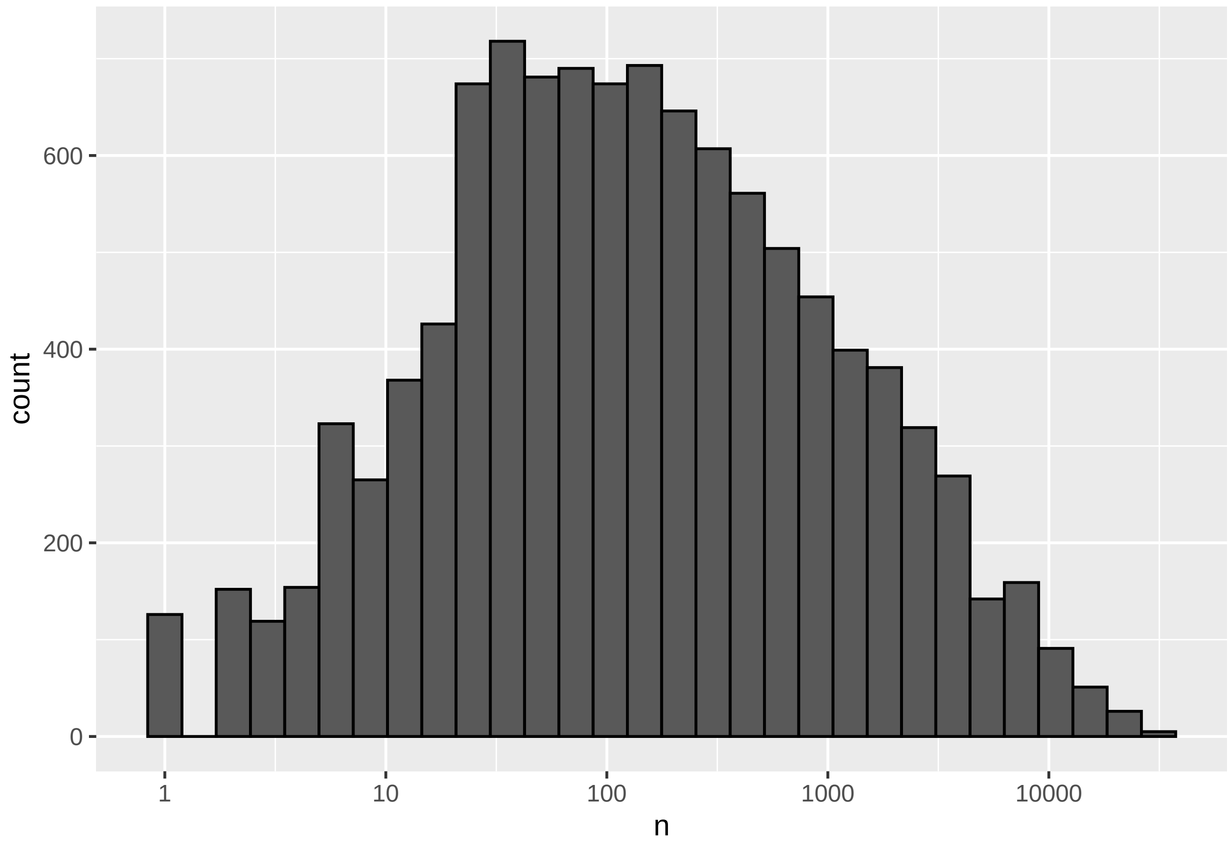
Looking at the rating per movie, showing that movies are not rated in the same way. Some are rated more times than others:

```
edx_new <- edx %>% separate_rows(title, sep = "\\|") %>% mutate(year = as.numeric(str_sub(title,-5,-2)))
edx_new %>% group_by(year) %>% summarize(avrg = mean(rating)) %>% plot(aes(year,avrg))
```



Looking at the average number of ratings in fonction of the year

```
edx %>% count(movieId) %>% ggplot(aes(n)) + geom_histogram(bins = 30, color = "black") + scale_x_log10(
```



## Creating models to calculate the Root Mean Square Error

### Creating a subset to train and test the model. A value of 20% has been chosen

```
library(caret)
set.seed(7)
test_index <- createDataPartition(y = edx$rating, times = 1, p = 0.2, list = FALSE)
train_set <- edx[-test_index,]
test_set <- edx[test_index,]
```

### Making sure that we have the same movieID and userID in the train_set as in the test_set

```
test_set <- test_set %>%
  semi_join(train_set, by = "movieId") %>%
  semi_join(train_set, by = "userId")
```

### Creating the RMSE function

The function used for the model will be in the form of:

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

Where $\hat{\mu}$ is the mean and $\varepsilon_{i,u}$ is the independent errors sampled from the same distribution centered at 0.

The model will then be compared with some dataset to see the accuracy of the model. To quantify the accuracy of the model we will use the Root Mean Squared Error:

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

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

### Simplest model using just the average

```
mu_hat <- mean(train_set$rating)
mu_hat
```

```
## [1] 3.51247
```

### Implementing the simplest average to calculate the RMSE:

```
naive_rmse <- RMSE(test_set$rating, mu_hat)
naive_rmse
```

```
## [1] 1.060591
```

```
rmse_results <- data_frame(method = "Just the average", RMSE = naive_rmse)
```

```
## Warning: `data_frame()` is deprecated, use `tibble()`.
## This warning is displayed once per session.
```

## Considering the movie effect

Because of the large volume of data in the dataset it will take too long to use the lm() function. We saw from the data exploration that movies affect the model. We will consider the movie effect here by applying the following model:

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

```
mu <- mean(train_set$rating)
movie_avgs <- train_set %>%
  group_by(movieId) %>%
  summarize(b_i = mean(rating - mu))
```

We can now calculate the RMSE:

```
predicted_ratings <- mu + test_set %>%
  left_join(movie_avgs, by='movieId') %>%
  pull(b_i)

model_1_rmse <- RMSE(predicted_ratings, test_set$rating)
rmse_results <- bind_rows(rmse_results,
                          data_frame(method="Movie Effect Model",
                                     RMSE = model_1_rmse))
rmse_results
```

```
## # A tibble: 2 x 2
##   method              RMSE
##   <chr>              <dbl>
## 1 Just the average   1.06
## 2 Movie Effect Model 0.944
```

## Considering the user effect and adding it to the model

The users effect is now added into the model since we saw that users affect the rating in different ways from our data exploration. The model that is now used is:

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

```
user_avgs <- train_set %>%
  left_join(movie_avgs, by='movieId') %>%
  group_by(userId) %>%
  summarize(b_u = mean(rating - mu - b_i))
```

Calculating the RMSE provides:

```
predicted_ratings <- test_set %>%
  left_join(movie_avgs, by='movieId') %>%
  left_join(user_avgs, by='userId') %>%
  mutate(pred = mu + b_i + b_u) %>%
  pull(pred)


model_2_rmse <- RMSE(predicted_ratings, test_set$rating)
rmse_results <- bind_rows(rmse_results,
                          data_frame(method="Movie + User Effects Model",
                                     RMSE = model_2_rmse))
rmse_results
```

```
## # A tibble: 3 x 2
##   method                      RMSE
##   <chr>                      <dbl>
## 1 Just the average           1.06 
## 2 Movie Effect Model         0.944
## 3 Movie + User Effects Model 0.866
```

Although we have considered 2 factors, the improvement brought is still minimal and we need to look at outliers/information that are skewing our model. By applying regularization we can penalize movies that have been rated highly or poorly only by a few people and that are affecting our model. Indeed we can see that some movies rated as either very good or terrible have a very few ratings only:

```
movie_titles <- edx %>%
  select(movieId, title) %>%
  distinct()

movie_avgs %>% left_join(movie_titles, by="movieId") %>%
  arrange(desc(b_i)) %>%
  slice(1:10)  %>%
  pull(title)
```

```
##  [1] "Hellhounds on My Trail (1999)"
##  [2] "I'm Starting From Three (Ricomincio da Tre) (1981)"
##  [3] "Shanghai Express (1932)"
##  [4] "Satan's Tango (Sátántangó) (1994)"
##  [5] "Shadows of Forgotten Ancestors (1964)"
##  [6] "Fighting Elegy (Kenka erejii) (1966)"
##  [7] "Sun Alley (Sonnenallee) (1999)"
##  [8] "Aerial, The (La Antena) (2007)"
##  [9] "Blue Light, The (Das Blaue Licht) (1932)"
## [10] "Who's Singin' Over There? (a.k.a. Who Sings Over There) (Ko to tamo peva) (1980)"
```

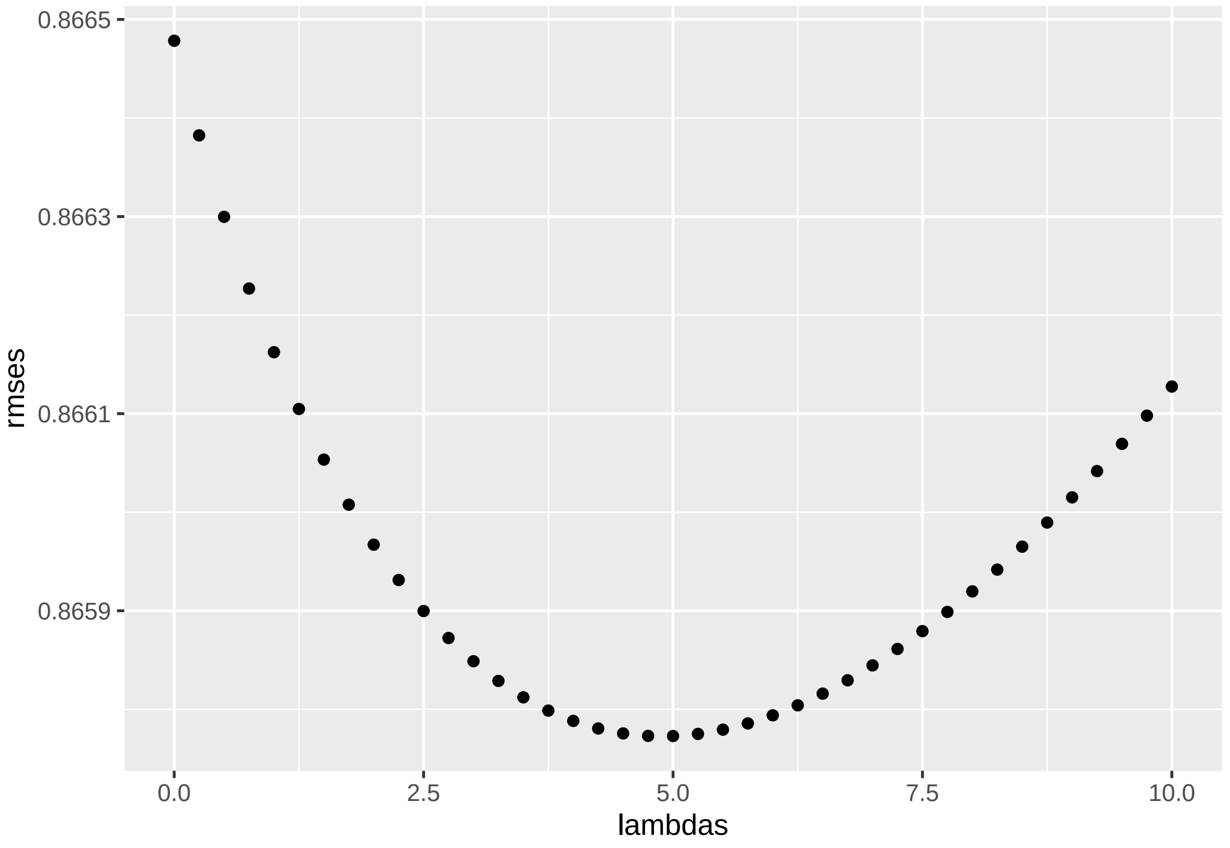
```
train_set %>% count(movieId) %>%
  left_join(movie_avgs, by="movieId") %>%
  left_join(movie_titles, by="movieId") %>%
  arrange(desc(b_i)) %>%
  slice(1:10) %>%
  pull(n)
```

```
##  [1] 1 1 1 2 1 1 1 1 1 4
```

Looking at the best value to set the penality so that the RMSE value is minimised

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

rmses <- sapply(lambdas, function(l){

  mu <- mean(train_set$rating)

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

  b_u <- train_set %>%
    left_join(b_i, by="movieId") %>%
    group_by(userId) %>%
    summarize(b_u = sum(rating - b_i - mu)/(n()+l))
```

```r
  predicted_ratings <-
    test_set %>%
    left_join(b_i, by = "movieId") %>%
    left_join(b_u, by = "userId") %>%
    mutate(pred = mu + b_i + b_u) %>%
    pull(pred)

  return(RMSE(predicted_ratings, test_set$rating))
})
qplot(lambdas, rmses)
```



We can then find lambda that provide the least RMSE:

```r
lambda <- lambdas[which.min(rmses)]
lambda
```

```
## [1] 5
```

```r
min(rmses)
```

```
## [1] 0.865773
```

```r
rmse_results <- bind_rows(rmse_results,
                          data_frame(method="Regularized Movie + User Effect Model",
                                     RMSE = min(rmses)))
rmse_results %>% knitr::kable()
```

| method | RMSE |
|---|---|

| method | RMSE |
|---|---|
| Just the average | 1.0605906 |
| Movie Effect Model | 0.9435514 |
| Movie + User Effects Model | 0.8664784 |
| Regularized Movie + User Effect Model | 0.8657730 |

## Bringing everything together

Now let's use the Validation dataset to check the overall accuracy of the model:

```r
movie_avgs_reg <- edx %>%
  group_by(movieId) %>%
  summarize(b_i = sum(rating - mu)/(n()+lambda))
# Compute regularized estimates of b_u using lambda
user_avgs_reg <- edx %>%
  left_join(movie_avgs_reg, by='movieId') %>%
  group_by(userId) %>%
  summarize(b_u = sum(rating - mu - b_i)/(n()+lambda))
# Predict ratings
predicted_ratings_reg <- validation %>%
  left_join(movie_avgs_reg, by='movieId') %>%
  left_join(user_avgs_reg, by='userId') %>%
  mutate(pred = mu + b_i + b_u) %>%
  pull(pred)
# Test and save results
RMSE(validation$rating,predicted_ratings_reg)
```

```
## [1] 0.8648177
```

We obtain an overall accuracy of ***0.8648177***