HarvardX: PH125.9x Capstone MovieLens Project

Andrea Blasio

March 8th, 2020

Contents

1 Introduction

The project described in this document is aimed at solving the challenges posed by the *HarvardX PH125.9x Capstone MovieLens* exam; its purpose is to build an effective model suitable for predicting users' movie ratings and therefore making movie recommendations based on the *MovieLens 10M Dataset*, available in the public domain.

The R script provided by *HarvardX* as a starting point for the assessment downloads the dataset and splits it in two subsets suitable for respectively training and testing the model:

```
################################
# Create edx set, validation set
################################

# Note: this process could take a couple of minutes

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

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

# Project specific packages

if(!require(ggplot2)) {
  install.packages("ggplot2", repos = "http://cran.us.r-project.org")
}

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

# Libraries required by the project
library(tidyverse)
library(caret)
library(data.table)
library(dplyr)
library(ggplot2)
library(kableExtra)

# 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")
# 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 %>%
```

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

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

```r
# Utility function suitable for converting number formats
# on axis labels to scientific 10^x format
# Credit: Brian Diggs (https://groups.google.com/forum/#!topic/ggplot2/a_xhMoQyxZ4)
fancy_scientific <- function(l) {
  # turn in to character string in scientific notation
  l <- format(l, scientific = TRUE)
  # quote the part before the exponent to keep all the digits
  l <- gsub("^(.*)e", "'\\1'e", l)
  # turn the 'e+' into plotmath format
  l <- gsub("e", "%*%10^", l)
  # return this as an expression
  parse(text=l)
}
```

User movie ratings are predicted using the *edx* subset as input, while testing is performed against the *validation* subset; the subsets are respectively equivalent to 90% and 10% of total data.

Root Mean Square Error (RMSE) is employed as measurement of the model accuracy:

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

```r
RMSE <- function(observed_values, forecasted_values){
  sqrt(mean((observed_values - forecasted_values)^2))
}
```

RMSE outcome can be intended as the standard deviation of prediction errors, also mentioned in statistics literature as residuals; a residual is a measure of how far from the regression line data points are. *Root Mean Square Error* in turn is a measure of how spread out residuals are and is sensitive to such potential outliers; our goal is to achieve an *RMSE* lesser than **0.86490** as per assignment requirements.

Increasingly accurate prediction models are experimented and evaluated via *RMSE* throughout the project; a final decision on the best solution to adopt is based on the *RMSE* outcome.

2 Analysis

The *edx* dataset is structured and characterized as follows:

```r
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, 1, 2,…
## $ movieId   <dbl> 122, 185, 292, 316, 329, 355, 356, 362, 364, 370, 377, 420,…
## $ rating    <dbl> 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5,…
## $ timestamp <int> 838985046, 838983525, 838983421, 838983392, 838983392, 8389…
## $ title     <chr> "Boomerang (1992)", "Net, The (1995)", "Outbreak (1995)", "…
## $ genres    <chr> "Comedy|Romance", "Action|Crime|Thriller", "Action|Drama|Sc…
```

```r
summary(edx)
```

```
##      userId         movieId          rating        timestamp        
##  Min.   :    1   Min.   :    1   Min.   :0.500   Min.   :7.897e+08  
##  1st Qu.:18124   1st Qu.:  648   1st Qu.:3.000   1st Qu.:9.468e+08  
##  Median :35738   Median : 1834   Median :4.000   Median :1.035e+09  
##  Mean   :35870   Mean   : 4122   Mean   :3.512   Mean   :1.033e+09  
##  3rd Qu.:53607   3rd Qu.: 3626   3rd Qu.:4.000   3rd Qu.:1.127e+09  
##  Max.   :71567   Max.   :65133   Max.   :5.000   Max.   :1.231e+09  
##     title              genres         
##  Length:9000055     Length:9000055    
##  Class :character   Class :character  
##  Mode  :character   Mode  :character  
##                                       
##                                       
## 
```

The *MovieLens 10M Dataset* contains **10000054** ratings applied to **10677** movies by **69878** users of the online movie recommender service *MovieLens*:

```r
# Count of unique users and movies in the dataset
edx %>% summarize(users = n_distinct(edx$userId), movies = n_distinct(edx$movieId))
```

```
##   users movies
## 1 69878  10677
```

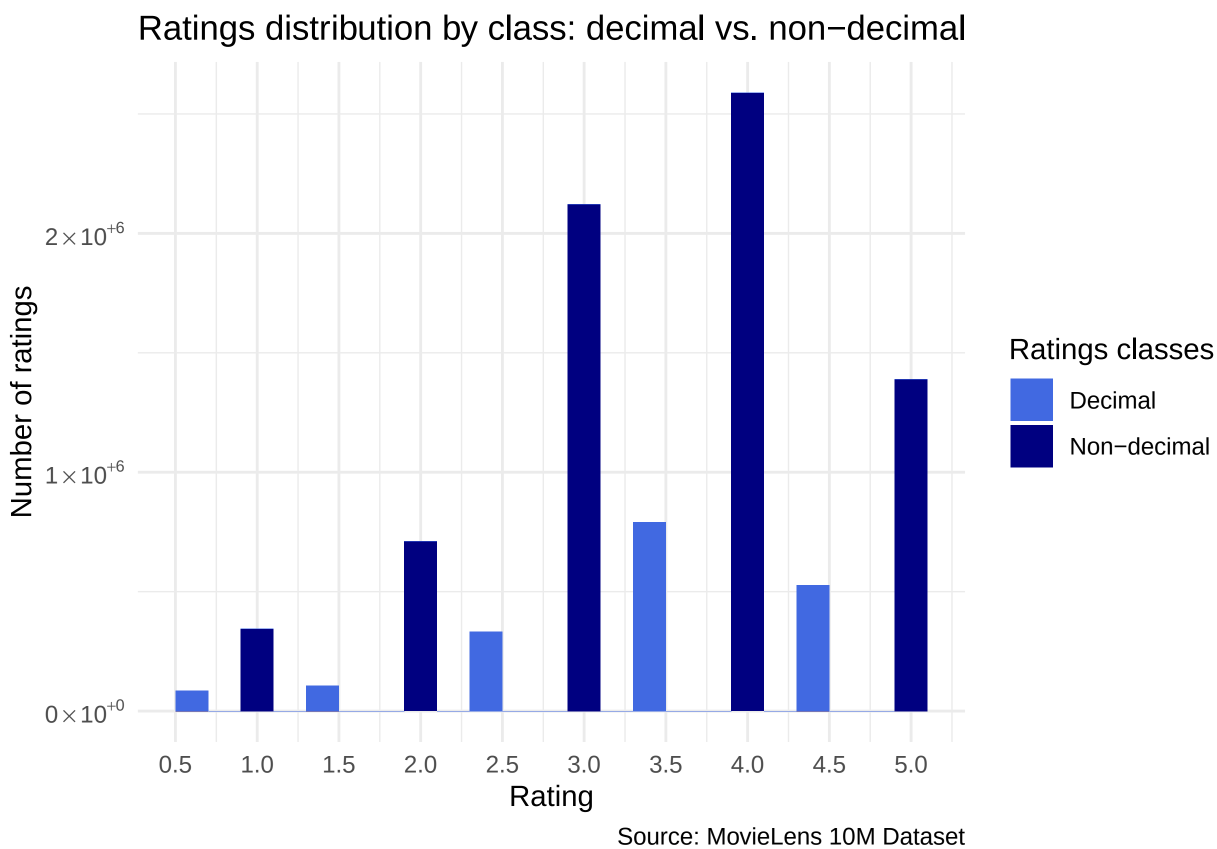
```r
# Total number of ratings available in the dataset
length(edx$rating) + length(validation$rating)
```

```
## [1] 10000054
```

The vast majority of users preferred to express a rating via a non-decimal score:

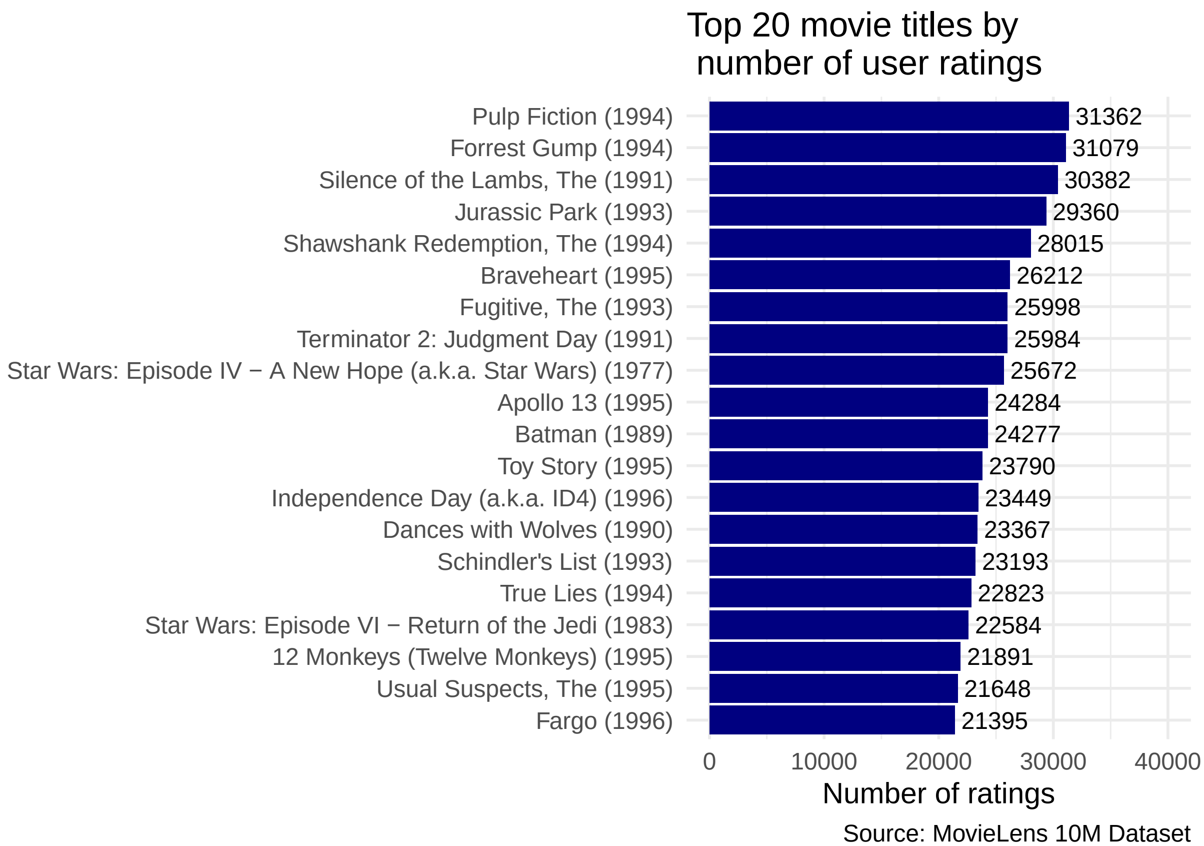
```r
# Discern rating into two classes: decimal and non-decimal
ratings_decimal_vs_nondecimal <- ifelse(edx$rating%%1 == 0, "non_decimal", "decimal")

# Build a new dataframe suitable for inspecting decimal and non-decimal ratings ratio
explore_ratings <- data.frame(edx$rating, ratings_decimal_vs_nondecimal)

# Draw histogram
ggplot(explore_ratings, aes(x= edx.rating, fill = ratings_decimal_vs_nondecimal)) +
  geom_histogram( binwidth = 0.2) +
  scale_x_continuous(breaks=seq(0, 5, by= 0.5)) +
  scale_y_continuous(labels = fancy_scientific) +
```

```
  scale_fill_manual(values = c("decimal"="royalblue", "non_decimal"="navy"),
    name="Ratings classes",
    breaks=c("decimal", "non_decimal"),
    labels=c("Decimal", "Non-decimal")) +
  labs(x="Rating", y="Number of ratings",
      caption = "Source: MovieLens 10M Dataset") +
  ggtitle("Ratings distribution by class: decimal vs. non-decimal") +
  theme_minimal()
```

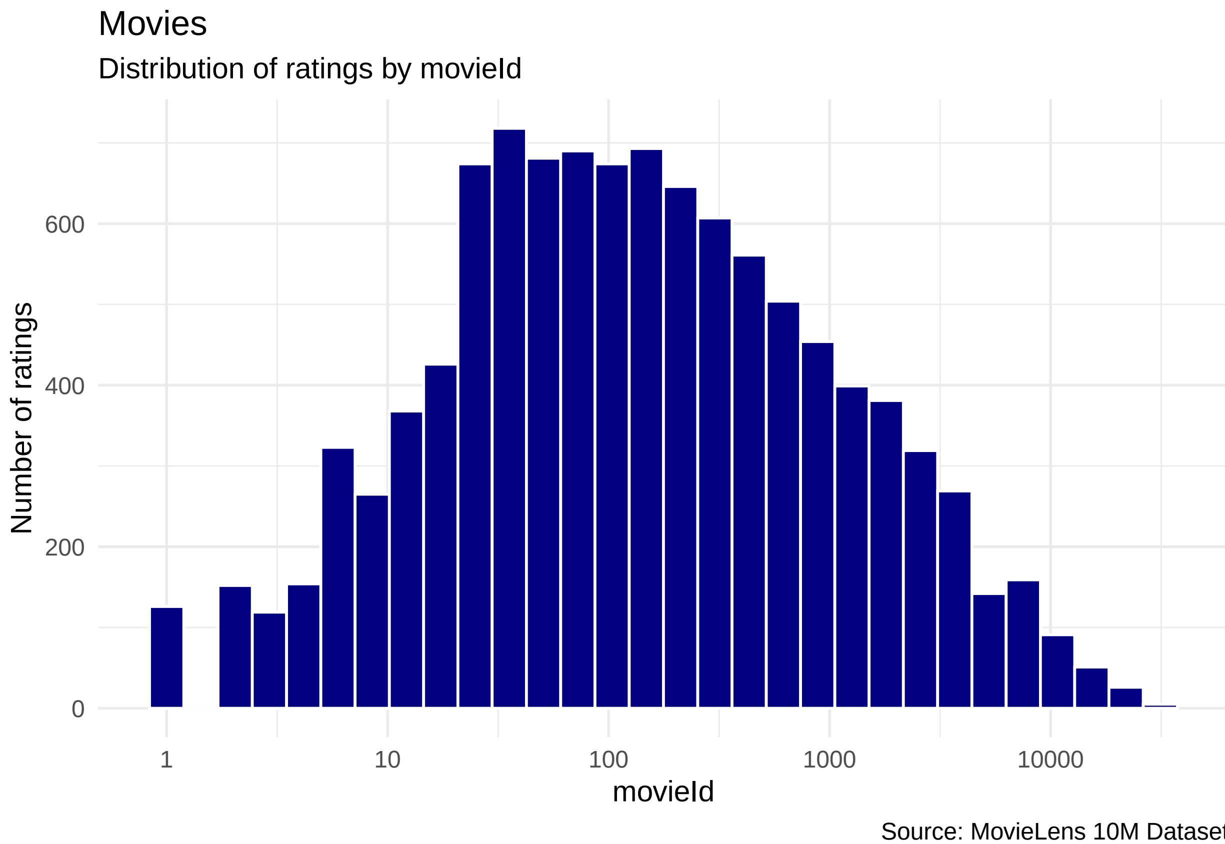



```
# Build a new dataframe suitable for inspecting
# the top 20 movie titles by number of ratings
top_titles <- edx %>%
  group_by(title) %>%
  summarize(count=n()) %>%
  top_n(20,count) %>%
  arrange(desc(count))

# Draw bar chart: top titles
top_titles %>%
  ggplot(aes(x=reorder(title, count), y=count)) +
  ggtitle("Top 20 movie titles by \n number of user ratings") +
  geom_bar(stat='identity', fill="navy") +
  coord_flip(y=c(0, 40000)) +
  labs(x="", y="Number of ratings",
      caption = "Source: MovieLens 10M Dataset") +
```

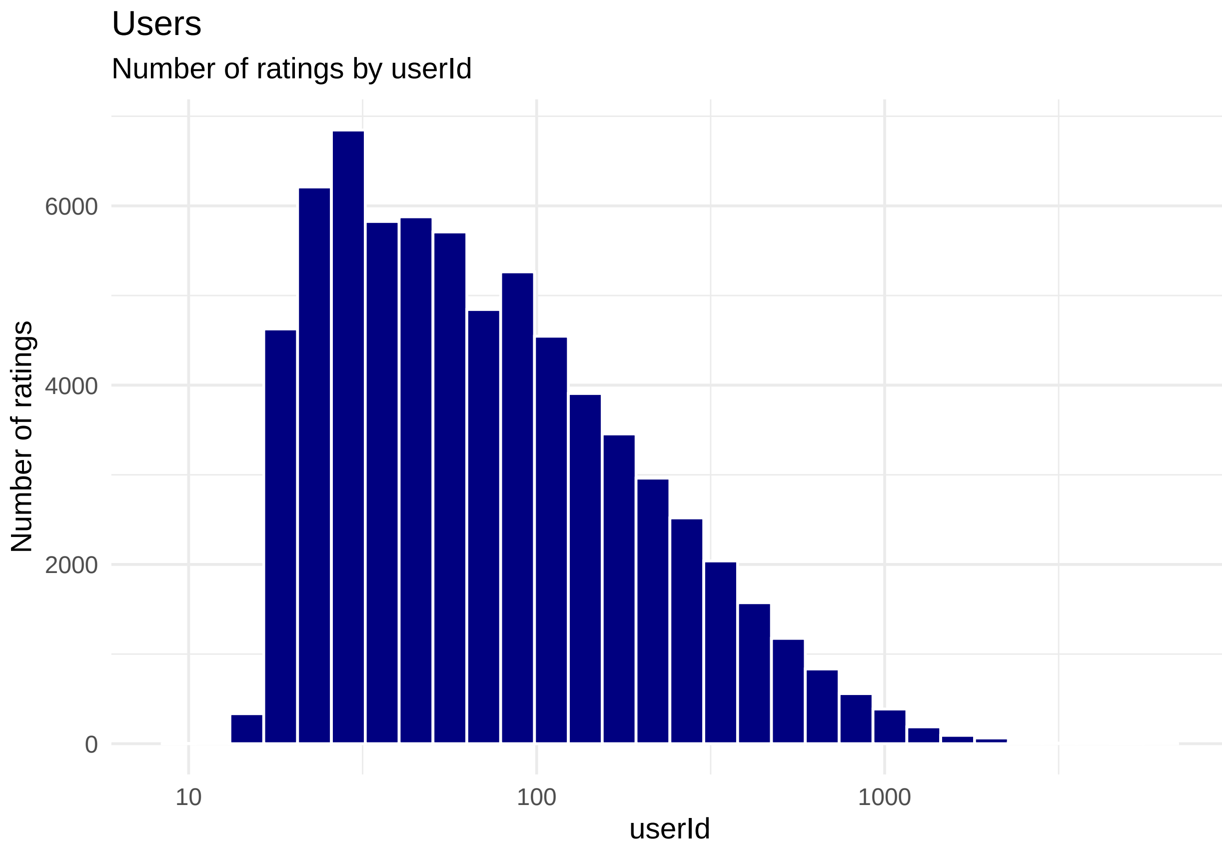
```r
  geom_text(aes(label= count), hjust=-0.1, size=3) +
  theme_minimal()
```




```r
# Draw histogram: distribution of ratings by movieId
edx %>%
  count(movieId) %>%
  ggplot(aes(n)) +
  ggtitle("Movies") +
  labs(subtitle  ="Distribution of ratings by movieId",
       x="movieId" ,
       y="Number of ratings",
       caption ="Source: MovieLens 10M Dataset") +
  geom_histogram(bins = 30, fill="navy", color = "white") +
  scale_x_log10() +
  theme_minimal()
```




```r
# histogram of number of ratings by userId

edx %>%
  count(userId) %>%
  ggplot(aes(n)) +
  geom_histogram( bins=30, fill="navy",color = "white") +
  scale_x_log10() +
  ggtitle("Users") +
  labs(subtitle ="Number of ratings by userId",
       x="userId" ,
       y="Number of ratings") +
  theme_minimal()
```



Examination of number of ratings by *movieId* and *userId* clearly shows how blockbuster movies get more ratings than others and how a subset of users is by far more keen to submit ratings. These peculiar traits are likely to produce a *bias* in the prediction model and are addressed in the next chapter.

3 Results

As in Irizarry (2020), a linear regression model is initially built in its simplest form via mean rating.

3.1 Basic prediction via mean rating

The following baseline prediction model employs the mean of ratings contained in the training dataset, assuming the same rating for all movies and users with all the differences explained by random variation:

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

With:

- $Y_{u,i}$ being the prediction;
- $\epsilon_{u,i}$ being the error;
- μ being the mean rating for all movies.

```r
# Basic prediction via mean rating
mu <- mean(edx$rating)

rmse_naive <- RMSE(validation$rating, mu)

rmse_results = tibble(Method = "Basic prediction via mean rating", RMSE = rmse_naive)

rmse_results %>% knitr::kable() %>% kable_styling()
```

Method	RMSE
Basic prediction via mean rating	1.061202

Basic prediction via mean rating yields a fairly high *RMSE*, which translates to ratings predictions potentially almost an entire star off.

3.2 Movie effects

Data analysis performed in the previous chapter shed light on a possible bias related to the tendency of some movies to get higher ratings than others; the following model includes movie effects to attempt to overcome such phenomenon: the term b_i is added to the formula to represent average ranking for movie i:

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

With:

- $Y_{u,i}$ being the prediction;
- $\epsilon_{u,i}$ being the error;
- μ being the mean rating for all movies;
- b_i being the bias for each movie i.

```r
## Simple model taking into account the movie effects, b_i
mu <- mean(edx$rating)

movie_averages <- edx %>%
  group_by(movieId) %>%
  summarise(b_i = mean(rating - mu))

predicted_ratings <- mu + validation %>%
  left_join(movie_averages, by='movieId') %>%
  pull(b_i)

rmse_model_movie_effects <- RMSE(validation$rating, predicted_ratings)

rmse_results <- bind_rows(rmse_results, tibble(Method="Movie effect model",
                                               RMSE = rmse_model_movie_effects))

rmse_results %>% knitr::kable() %>% kable_styling()
```

Method	RMSE
Basic prediction via mean rating	1.0612018
Movie effect model	0.9439087

Predicting ratings taking into account movie effects b_i generates a lower *RMSE* value.

3.3 Movie and user effects

As a further step towards a more efficient prediction, user effects b_u outlined in the analysis chapter are also included into the model:

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

With:

- $Y_{u,i}$ being the predicted value;
- $\epsilon_{u,i}$ being the error;
- μ being the mean rating for all movies;
- b_i being the bias for each movie i;
- b_u being the bias for each user u.

```r
# Movie and user effects model
user_averages <- edx %>%
  left_join(movie_averages, by="movieId") %>%
  group_by(userId) %>%
  summarise(b_u = mean(rating - mu - b_i))

predicted_ratings <- validation %>%
  left_join(movie_averages, by='movieId') %>%
  left_join(user_averages, by='userId') %>%
  mutate(pred = mu + b_i + b_u) %>%
  pull(pred)

rmse_model_user_effects <- RMSE(validation$rating, predicted_ratings)

rmse_results <- bind_rows(rmse_results,
                          tibble(Method="Movie and user effect model",
                          RMSE = rmse_model_user_effects))

rmse_results %>% knitr::kable() %>% kable_styling()
```

Method	RMSE
Basic prediction via mean rating	1.0612018
Movie effect model	0.9439087
Movie and user effect model	0.8653488

As a result, RMSE is further reduced.

3.4 Movie and user effects with regularization

As seen earlier, movies with few ratings can possibly influence the prediction and skew the error metric; regularization allows to introduce a tuning parameter, λ, to take into account such aspect in the computation: b_i and b_u are subsequently adjusted for movies with limited ratings:

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

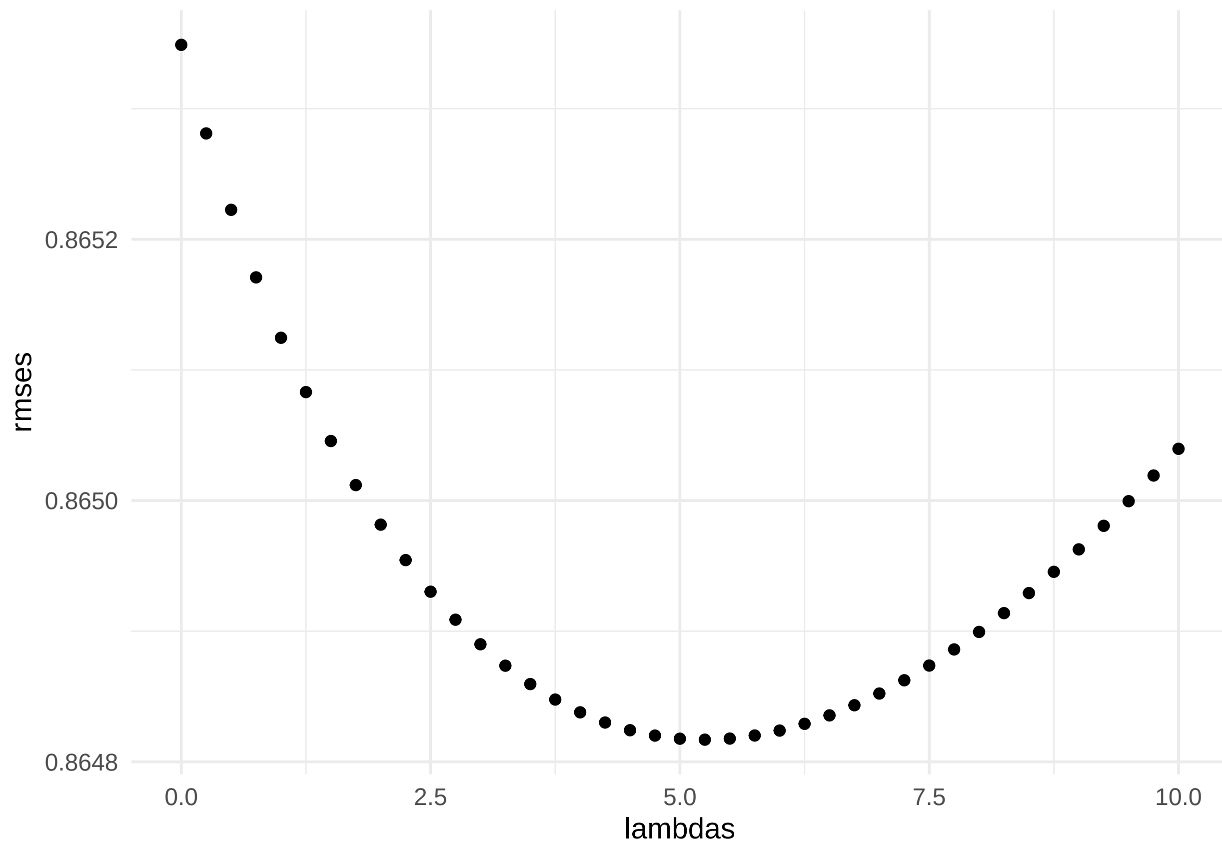
```r
# Prediction via movie and user effects model with regularization
lambdas <- seq(0, 10, 0.25)
rmses <- sapply(lambdas, function(l){

  mu <- mean(edx$rating)

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

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

  predicted_ratings <- validation %>%
    left_join(b_i, by = "movieId") %>%
    left_join(b_u, by = "userId") %>%
    mutate(predicted = mu + b_i + b_u) %>%
    pull(predicted)

  RMSE(validation$rating, predicted_ratings)

})

# Minimum RMSE value
rmse_regularization <- min(rmses)
rmse_regularization
```

```
## [1] 0.864817
```

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

```
## [1] 5.25
```

```r
# Plot RMSE against lambdas to visualize the optimal lambda
qplot(lambdas, rmses) + theme_minimal()
```




```
# Summary of prediction models outcomes
rmse_results <- bind_rows(rmse_results, tibble(
  Method="Movie and user effects model with regularization",
  RMSE = rmse_regularization))
rmse_results %>% knitr::kable() %>% kable_styling()
```

Method	RMSE
Basic prediction via mean rating	1.0612018
Movie effect model	0.9439087
Movie and user effect model	0.8653488
Movie and user effects model with regularization	0.8648170

Incorporating regularization into the model resulted in the lowest *RMSE* value.

4 Conclusion

The experiment contemplated increasingly enriched models to fulfill the goal of an *RMSE* lesser than **0.86490** as per assignment requirements.

In a possible refactoring and evolution of the model, further effects such as time and movie genre could be likely leveraged as well to further decrease *RMSE*.

5 Bibliography

Anthony G. Barnston, Correspondence among the Correlation, RMSE, and Heidke Forecast Verification Measures; Refinement of the Heidke Score, in Weather and Forecasting, december 1992, pp. 699-709.

Yehuda Koren, The BellKor Solution to the Netflix Grand Prize, 2009.

Irizarry Raphael A., Large Datasets in Introduction to Data Science, Data Analysis and Prediction Algorithms with R, 2020.

6 Appendix: system configuration and R version

```
version
```

```
##                _
## platform       x86_64-w64-mingw32
## arch           x86_64
## os             mingw32
## system         x86_64, mingw32
## status
## major          3
## minor          6.1
## year           2019
## month          07
## day            05
## svn rev        76782
## language       R
## version.string R version 3.6.1 (2019-07-05)
## nickname       Action of the Toes
```