Movie Lens edx project

Andree Flores

2022-06-19

Intoduction

This is a project for the course HarvardX PH125.9x Data Science: Capstone. The project consist in use the MovieLens database with the objective of predict movie ratings. The database consist of 10 million ratings from more than 70,000 users and 10,000 movies.

The goal of the project is to predict the rating a user give to some movie with the intention of create a movie recommendation algorithm. And this algorithm needs to have a RMSE lower of 0.8649 in order to get all the points of this part.

The residual mean squared error (RMSE) is used as the error loss function of this project. Where $y_{u,i}$ is the rating for movie i by user u and $\hat{y}_{u,i}$ is the prediction made by the model.

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

Initially the data set has the following columns:

- userId, which is a integer number unique for each user
- movieId, which is a integer number unique for each movie
- rating, which is a number between 0.5 and 5 with 0.5 increments, like stars
- timestamp, which is the time when the rating was made
- title, which is the title of the movie and the year of release
- genres, which a string with | as separators of the genres of the movie

And the columns that I added after are:

- releaseyear, which is a integer number that represents the year release of the movie (unique for each movieId), extracted from the title column.
- unique_ans, which is a integer number that represents how many different ratings that user has made, I added it during the downloading of the data to avoid having different values in the edx and validation edx for an specific userId, but in practice could be different.

The variables I decided to use for my model were: userId, movieId, genres, releaseyear and unique_ans, the reason I didn't use title is because it would be the same thing as using the movieId and for timestamp is because I found some ratings that were made before the movie was release.

I decided to make a linear model, whose justification will be found in the Model approach section.

Methods/Analysis

Data download and data cleaning

Following the instructions provided, we can download the data set and process it:

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

dl <- tempfile()
download.file("https://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(movieId), title = as.character(title),
        genres = as.character(genres))


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

### Added after analysis but it was put here to avoid modify the edx
### and validation separated and having different values in the
### variable unique ratings release year
movielens <- movielens %>%
    mutate(releaseyear = as.numeric(str_extract(str_extract(title, "[/(]\\d{4}[/)]$"),
        regex("\\d{4}"))), title = str_remove(title, "[/(]\\d{4}[/)]$"))
# unique ratings
movielens <- movielens %>%
    group_by(userId) %>%
    mutate(unique_ans = n_distinct(rating)) %>%
    ungroup()

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

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

```
## # A tibble: 6 x 6
##   userId movieId rating timestamp title                  genres
```

```
##    <int>   <dbl>  <dbl>     <int> <chr>                      <chr>
## 1      1     122      5 838985046 "Boomerang "               Comedy|Romance
## 2      1     185      5 838983525 "Net, The "                Action|Crime|Thrill~
## 3      1     292      5 838983421 "Outbreak "                Action|Drama|Sci-Fi~
## 4      1     316      5 838983392 "Stargate "                Action|Adventure|Sc~
## 5      1     329      5 838983392 "Star Trek: Generations " Action|Adventure|Dr~
## 6      1     355      5 838984474 "Flintstones, The "        Children|Comedy|Fan~
```

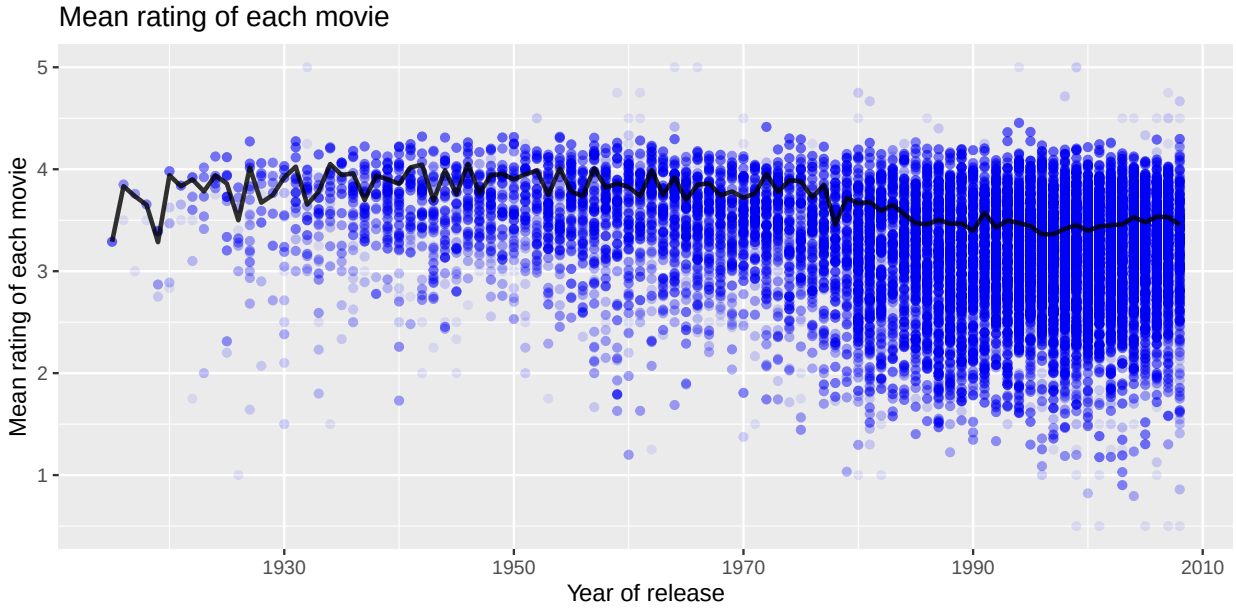
Data Analysis

All the following analysis were made using edx data set.

Year of release

As we can see the year of release is in the title column, we can move the year to a new column and therefore use it in the analysis.

```
### Extract the year of release from the title variable
edx <- edx %>%
    mutate(releaseyear = as.numeric(str_extract(str_extract(title, "[/(]\\d{4}[/)]$"),
        regex("\\d{4}"))), title = str_remove(title, "[/(]\\d{4}[/)]$"))
```



In this plot each point is a movie, and the black line follows the average rating of all movies released in each year. We can see that:

- There are more movies in recent years.
- Older movies tend to have a greater mean rating.
- In general newer movies have a greater variance in their rating, but the average remains almost the same, with a fall of around 0.25.

Unique ratings

We can group the users by how many different ratings has. For example: the user with userId = 1 has all ratings with a 5, but the userId = 2 has 4 different ratings:

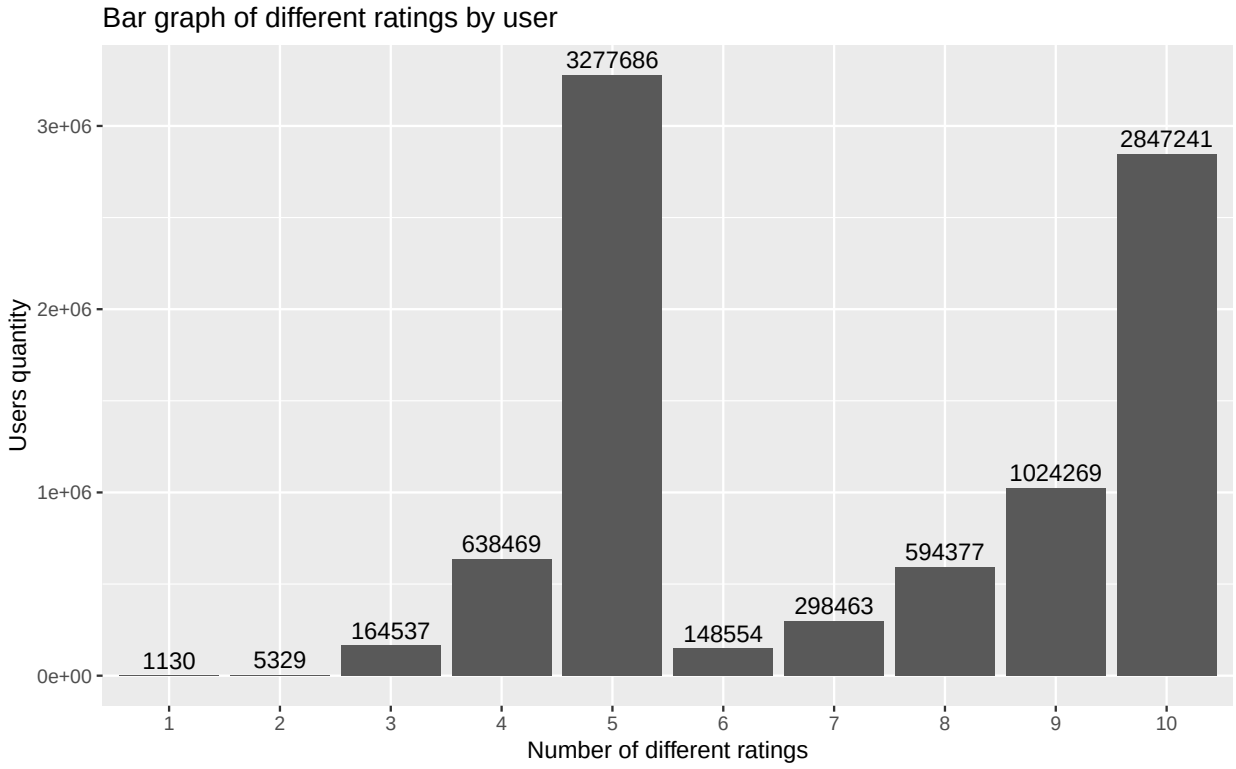
```
unique(edx[edx$userId == 1,]$rating)
```

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

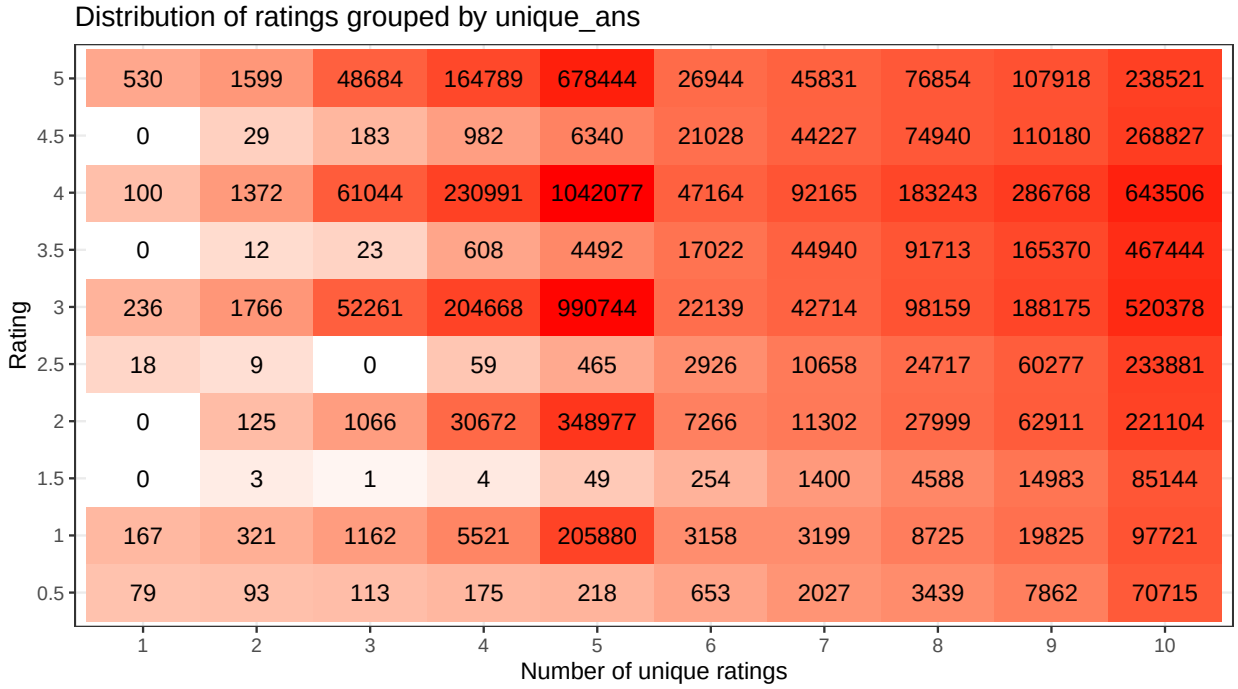
```
unique(edx[edx$userId == 2,]$rating)
```

```
## [1] 5 3 2 4
```

```
###How many different ratings did each user has?
edx <- edx %>% group_by(userId) %>%
  mutate(unique_ans = n_distinct(rating)) %>% ungroup()
```



As we can see the biggest group are the users with 5 different ratings, followed by the group of 10. Is obvious that the group of 10 uses all the ratings available, but do the group of 5 has a tendency? Do the group of 5 different ratings uses only integer ratings more often? Does the group of 4 and 3 do the same?



As we can see for most ratings and for each group, is more common to rate a movie with a whole number than with a fraction. This can be useful especially with groups with a low number of unique ratings. But this approach could over fit the model.

For example, we could use the fact that the user with Id 1 only rates movies with a 5, causing to create a "perfect prediction" of only 5 ratings. And with the objective of the project been a movie recommendation model, it would recommend every movie equally to this user.

Genres

Other important data for the model is the genres of the movie, especially with the personal preferences of each user. Something to highlight is that the sum of the count of all genres is greater than all the the ratings, this is because movies can have more than one genre. In total there are 23371423 genres listed among all movies and 797 combinations of genres in the data set.

```
genres <- edx %>% separate_rows(genres, sep = "\\|") %>%
  select(genres)
```

```
## [1] "Table Combo of Genres"
```

```
##                                Genres Combo Frequency
## 1                                    Drama    733296
## 2                                   Comedy    700889
## 3                           Comedy|Romance    365468
## 4                            Comedy|Drama    323637
## 5                     Comedy|Drama|Romance    261425
## 793                     Adventure|Mystery         2
## 794            Crime|Drama|Horror|Sci-Fi         2
## 795                  Documentary|Romance         2
## 796 Drama|Horror|Mystery|Sci-Fi|Thriller         2
## 797           Fantasy|Mystery|Sci-Fi|War         2
```

```
## [1] "Table Genres"
```

```
##                Genres Frequency
## 1               Drama   3910127
## 2              Comedy   3540930
## 3              Action   2560545
## 4            Thriller   2325899
## 5           Adventure   1908892
## 16            Western    189394
## 17          Film-Noir    118541
## 18        Documentary     93066
## 19               IMAX      8181
## 20 (no genres listed)         7
```

As we can see the most popular genre is Drama which appear in 3910127 in total and 733296 as combo of genres.

Also there are some movies listed as not having genres as we can see in the last row in the Table Genres: (no genres listed) with 7 ratings, further research shows that is only one movie with this issue:

```
edx %>% filter(genres == "(no genres listed)") %>%
  pull(title) %>% unique()
```

```
## [1] "Pull My Daisy "
```

In this project I will use the combination of genres to do the prediction model. Because in this way I can use the relation between different genres more easily.

Time of rating (Timestamp)

As previously said in the introduction of this project, I decided to not use this variable. The reason is that I found some irregularities that makes me doubt of this data.

For example, in total 175 ratings are reported to be made before the movie was released, this could be explained with the release been close to the start or end of a specific year and the rounding of the variables causes this issue. But it doesn't explain the difference of 2 years in 3 cases. I know this are few cases, but creates doubt on the veracity of this variable or in the approach I'm using, that is I'm probably writing code that is wrong but after researching I couldn't find any errors.

Therefore I decided to not use this variable, unlike with the genres variable and the movie with no listed genres, as it's only 1 movie, the effect should be minimal because the movieId is also used in the model.

Methods

Firstly we need to define an R function for calculating the RMSE:

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

Next I'm going to divide the edx data set, in order to avoid using the validation data set during making the model.

```
###Split edx, in train and test
#Test set will be 10% of edx data
set.seed(1, sample.kind="Rounding") # if using R 3.5 or earlier, use `set.seed(1)`
```

```
## Warning in set.seed(1, sample.kind = "Rounding"): non-uniform 'Rounding' sampler
## used
```

```
test_index <- createDataPartition(y = edx$rating, times = 1, p = 0.1, list = FALSE)
train_edx <- edx[-test_index,]
```

```
temp <- edx[test_index,]

#Make sure userId and movieId in validation set are also in train_edx set
test_edx <- temp %>%
  semi_join(train_edx, by = "movieId") %>%
  semi_join(train_edx, by = "userId")

#Add rows removed from test_edx set back into train_edx set
removed <- anti_join(temp, test_edx)
```

```
## Joining, by = c("userId", "movieId", "rating", "timestamp", "title", "genres",
## "releaseyear", "unique_ans")
```

```
train_edx <- rbind(train_edx, removed)

#Remove unnecessary variables
rm(test_index, temp, removed)
```

Model approach

Due to various limitations such as memory size and not having enough RAM, I can't use models that come in the caret package in their full potential. Therefore, I choose a liner model for this project. Where:

- $Y_{u,i}$ is the rating for the movie i from the user u.
- μ is the mean rating across all movies and users.
- b_i is the effect for movie i.
- b_u is the effect for user u.
- $b_{g,i}$ is the effect for the unique combination of genres that the movie i has. For simplicity I will label as b_g
- $b_{r,i}$ is the effect for the year of release of the movie i. For simplicity I will label as b_r
- $b_{ru,u}$ is the effect for the group of unique ratings that the user u belongs. For simplicity I will label as b_{ru}
- $\epsilon_{u,i}$ is the error assuming that has a mean of zero.

Also we need to consider $\hat{Y}_{u,i}$ as the predicted rating for the movie i from the user u.

For a model that assumes the same rating for all ratings with the differences explained by random variation would look like this:

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

With this model, the least squares estimate for μ is the average of all the ratings.

```
  mu <- mean(train_edx$rating)
  naive_rmse <- RMSE(test_edx$rating,mu)

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

Currently our prediction model looks like this:

$$\hat{Y}_{u,i} = \hat{\mu}$$

With a value of $\hat{\mu}$ equal to 3.5124556 and a RMSE of 1.0600537.

Movie effect

We can add the effect that each movie has to its own rating like this:

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

We can use the fact that for this linear model the least squares estimate of b_i is just the average of $Y_{u,i} - \hat{\mu}$ for each movie i.

```
b_i <- train_edx %>%
  group_by(movieId) %>%
  summarize(b_i = mean(rating - mu), .groups = "keep")

predicted_ratings <- test_edx %>%
  left_join(b_i, by="movieId") %>%
  mutate(pred = mu+b_i) %>%
  pull(pred)

rmse_movies <- RMSE(test_edx$rating, predicted_ratings)
rmse_results <- rmse_results %>%
  add_row(method="+ Movie effect", RMSE=rmse_movies)
```

User effect

We can add the effect that each user has to its rating like this:

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

The least squares estimate of b_u is just the the average of $Y_{u,i} - \hat{\mu} - \hat{b}_i$ for each user u.

```
b_u <- train_edx %>%
  left_join(b_i, by='movieId') %>%
  group_by(userId) %>%
  summarize(b_u = mean(rating - mu - b_i), .groups = "keep")

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

rmse_user <- RMSE(test_edx$rating, predicted_ratings)
rmse_results <- rmse_results %>%
  add_row(method="+ User effect", RMSE=rmse_user)
```

Genre effect

We can add the effect that each genre combination has on the rating like this:

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

The least squares estimate of b_g is the the average of $Y_{u,i} - \hat{\mu} - \hat{b}_i - \hat{b}_u$ for each genre combination.

```
b_g <- train_edx %>%
  left_join(b_u, by="userId") %>%
  left_join(b_i, by="movieId") %>%
  group_by(genres) %>%
  summarize(b_g = mean(rating - mu - b_i - b_u), .groups = "keep")

predicted_ratings <- test_edx %>%
  left_join(b_u, by="userId") %>%
  left_join(b_i, by="movieId") %>%
  left_join(b_g, by="genres") %>%
  mutate(pred = mu+b_i+b_u+b_g) %>%
```

```
  pull(pred)

rmse_genres <- RMSE(test_edx$rating, predicted_ratings)
rmse_results <- rmse_results %>%
  add_row(method="+ Genre combo effect", RMSE=rmse_genres)
```

Release year effect

We can add the effect that the year of release has on the rating like this:

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

The least squares estimate of b_r is the the average of $Y_{u,i} - \hat{\mu} - \hat{b}_i - \hat{b}_u - \hat{b}_g$ for each year.

```
b_r <- train_edx %>%
  left_join(b_u, by="userId") %>%
  left_join(b_i, by="movieId") %>%
  left_join(b_g, by="genres") %>%
  group_by(releaseyear) %>%
  summarize(b_r = mean(rating - mu - b_i - b_u - b_g), .groups = "keep")

predicted_ratings <- test_edx %>%
  left_join(b_u, by="userId") %>%
  left_join(b_i, by="movieId") %>%
  left_join(b_g, by="genres") %>%
  left_join(b_r, by="releaseyear") %>%
  mutate(pred = mu+b_i+b_u+b_g+b_r) %>%
  pull(pred)

rmse_releaseyear <- RMSE(test_edx$rating, predicted_ratings)
rmse_results <- rmse_results %>%
  add_row(method="+ Release year effect", RMSE=rmse_releaseyear)
```

Unique ratings effect

We can add the effect that each group of unique ratings has on the rating like this:

$$Y_{u,i} = \mu + b_i + b_u + b_g + b_r + b_{ru} + \epsilon_{u,i}$$

The least squares estimate of b_{ru} is the the average of $Y_{u,i} - \hat{\mu} - \hat{b}_i - \hat{b}_u - \hat{b}_g - \hat{b}_r$ for each group of unique ratings.

```
b_ru <- train_edx %>%
  left_join(b_u, by="userId") %>%
  left_join(b_i, by="movieId") %>%
  left_join(b_g, by="genres") %>%
  left_join(b_r, by="releaseyear") %>%
  group_by(unique_ans) %>%
  summarize(b_ru = mean(rating - mu - b_i - b_u - b_g - b_r), .groups = "keep")

predicted_ratings <- test_edx %>%
  left_join(b_u, by="userId") %>%
  left_join(b_i, by="movieId") %>%
  left_join(b_g, by="genres") %>%
  left_join(b_r, by="releaseyear") %>%
  left_join(b_ru, by="unique_ans") %>%
  mutate(pred = mu+b_i+b_u+b_g+b_r+b_ru) %>%
```

```
    pull(pred)

  rmse_uniqueans <- RMSE(test_edx$rating, predicted_ratings)
  rmse_results <- rmse_results %>%
    add_row(method="+ Unique answers effect", RMSE=rmse_uniqueans)
```

Model

So far our model for predicting the rating is:

$$\hat{Y}_{u,i} = \mu + b_i + b_u + b_g + b_r + b_{ru}$$

And our RMSE for the models are:

method	RMSE
Just the average	1.0600537
+ Movie effect	0.9429615
+ User effect	0.8646843
+ Genre combo effect	0.8643241
+ Release year effect	0.8641262
+ Unique answers effect	0.8640903

As we can see I reach a RMSE 0.8640903 which is lower that the RMSE objective of 0.8649. But of course this is not with the validation data, can we improve the RMSE? Yes! with regularization.

```
as.numeric(rmse_results[nrow(rmse_results),2]) < RMSE_target
```

```
## [1] TRUE
```

Regularization

Regularization consist in instead of instead of minimizing the least squares equation, instead we minimize an equation that adds a penalty. For example, for b_i we minimize:

$$\frac{1}{N}\sum_{u,i}(y_{u.i} - \mu - b_i)^2 + \lambda \sum_i b_i^2$$

Which the values of b_i that minimizes the equations, for each movie i, are:

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

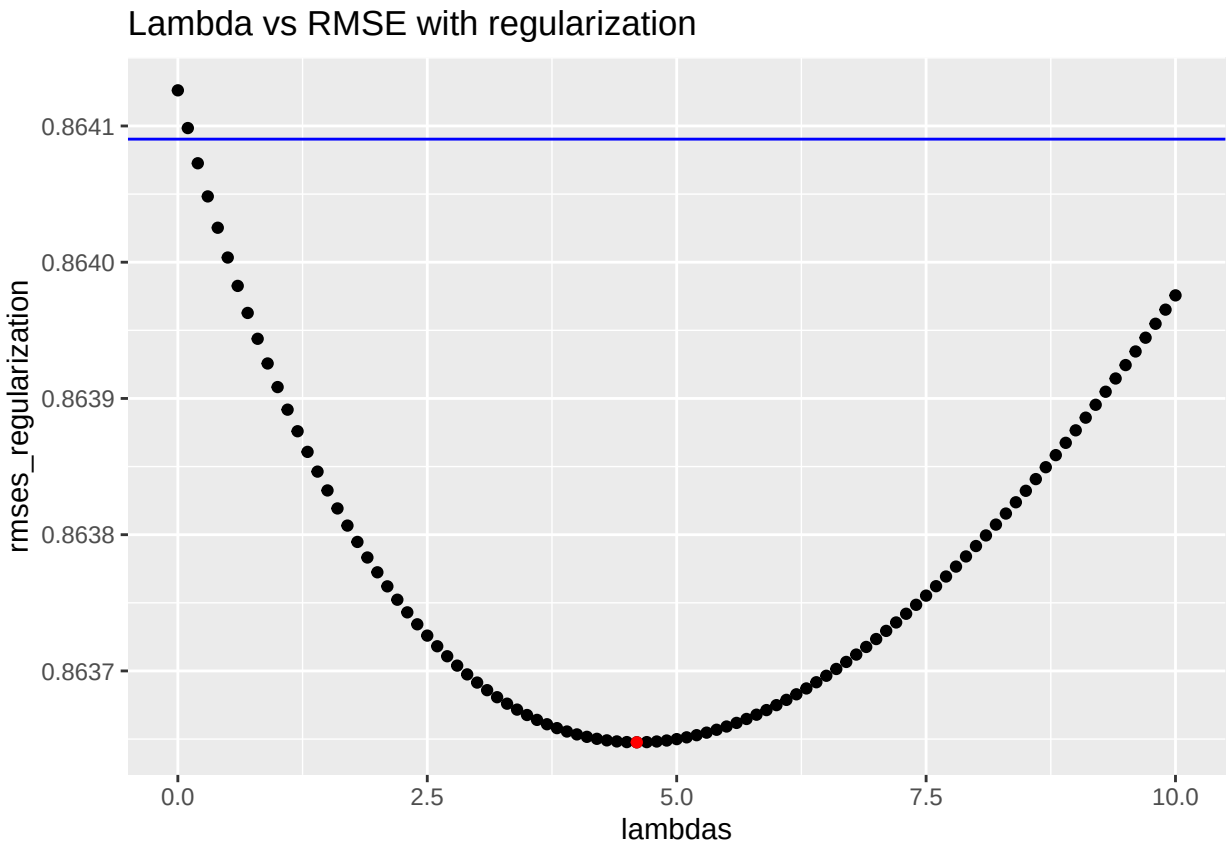
For the next effect, which is the user effect, we can use regularization as well. But we are minimizing the following equation:

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

And similarly for all other effects.

```
lambdas <- seq(0, 10, 0.1)

rmses_regularization <- sapply(lambdas, function(lambda) {

    # Mean
    mu <- mean(train_edx$rating)
```

```
    # Movie effect
    b_i <- train_edx %>%
        group_by(movieId) %>%
        summarize(b_i = sum(rating - mu)/(n() + lambda), .groups = "keep")

    # User effect
    b_u <- train_edx %>%
        left_join(b_i, by = "movieId") %>%
        group_by(userId) %>%
        summarize(b_u = sum(rating - mu - b_i)/(n() + lambda), .groups = "keep")

    # Genre combo effect
    b_g <- train_edx %>%
        left_join(b_u, by = "userId") %>%
        left_join(b_i, by = "movieId") %>%
        group_by(genres) %>%
        summarize(b_g = sum(rating - mu - b_i - b_u)/(n() + lambda), .groups = "keep")

    # Release year effect
    b_r <- train_edx %>%
        left_join(b_u, by = "userId") %>%
        left_join(b_i, by = "movieId") %>%
        left_join(b_g, by = "genres") %>%
        group_by(releaseyear) %>%
        summarize(b_r = sum(rating - mu - b_i - b_u - b_g)/(n() + lambda), .groups = "keep")

    # Unique rating effect
    b_ru <- train_edx %>%
        left_join(b_u, by = "userId") %>%
        left_join(b_i, by = "movieId") %>%
        left_join(b_g, by = "genres") %>%
        left_join(b_r, by = "releaseyear") %>%
        group_by(unique_ans) %>%
        summarize(b_ru = mean(rating - mu - b_i - b_u - b_g - b_r)/(n() + lambda),
            .groups = "keep")

    # Predicted ratings
    predicted_ratings <- test_edx %>%
        left_join(b_u, by = "userId") %>%
        left_join(b_i, by = "movieId") %>%
        left_join(b_g, by = "genres") %>%
        left_join(b_r, by = "releaseyear") %>%
        left_join(b_ru, by = "unique_ans") %>%
        mutate(pred = mu + b_i + b_u + b_g + b_r + b_ru) %>%
        pull(pred)

    return(RMSE(test_edx$rating, predicted_ratings))
})
```



As we can see in the plot, where the blue line is the RMSE that we get before regularization, we get a lot of values of λ which we get a lower RMSE, with $\lambda = 4.6$ the best one (is the one with the red point).

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

rmse_reg <- rmses_regularization[which.min(rmses_regularization)]

rmse_results <- rmse_results %>%
  add_row(method="All effects + Regularization", RMSE=rmse_reg)

knitr::kable(rmse_results)
```

method	RMSE
Just the average	1.0600537
+ Movie effect	0.9429615
+ User effect	0.8646843
+ Genre combo effect	0.8643241
+ Release year effect	0.8641262
+ Unique answers effect	0.8640903
All effects + Regularization	0.8636476

As we can see in the above table, our RMSE with the test data set has a value of 0.8636476. Of course we can find better models but because of my limitation with the hardware of my computer, I don't have the capacity to make a better one.

Results

Now I'm going to train the final model with regularization with the whole edx data set. And test it with the validation data set.

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

# Movie effect
b_i <- edx %>%
    group_by(movieId) %>%
    summarize(b_i = sum(rating - mu)/(n() + lambda), .groups = "keep")

# User effect
b_u <- edx %>%
    left_join(b_i, by = "movieId") %>%
    group_by(userId) %>%
    summarize(b_u = sum(rating - mu - b_i)/(n() + lambda), .groups = "keep")

# Genre combo effect
b_g <- edx %>%
    left_join(b_u, by = "userId") %>%
    left_join(b_i, by = "movieId") %>%
    group_by(genres) %>%
    summarize(b_g = sum(rating - mu - b_i - b_u)/(n() + lambda), .groups = "keep")

# Release year effect
b_r <- edx %>%
    left_join(b_u, by = "userId") %>%
    left_join(b_i, by = "movieId") %>%
    left_join(b_g, by = "genres") %>%
    group_by(releaseyear) %>%
    summarize(b_r = sum(rating - mu - b_i - b_u - b_g)/(n() + lambda), .groups = "keep")

# Unique rating effect
b_ru <- edx %>%
    left_join(b_u, by = "userId") %>%
    left_join(b_i, by = "movieId") %>%
    left_join(b_g, by = "genres") %>%
    left_join(b_r, by = "releaseyear") %>%
    group_by(unique_ans) %>%
    summarize(b_ru = mean(rating - mu - b_i - b_u - b_g - b_r)/(n() + lambda), .groups = "keep")

# Predicted ratings
predicted_ratings <- validation %>%
    left_join(b_u, by = "userId") %>%
    left_join(b_i, by = "movieId") %>%
    left_join(b_g, by = "genres") %>%
    left_join(b_r, by = "releaseyear") %>%
    left_join(b_ru, by = "unique_ans") %>%
    mutate(pred = mu + b_i + b_u + b_g + b_r + b_ru) %>%
    pull(pred)

# Validation RMSE
rmse_final <- RMSE(validation$rating, predicted_ratings)
```

In the end we get a RMSE of 0.8642939. Which is lower than the requested RMSE of 0.8649.

```
rmse_final < RMSE_target
```

```
## [1] TRUE
```

Some issues that I have with this model is that there are some predicted ratings beyond the expected range from 0.5 to 5, with 0.18% of the predictions been the case. With the lowest predicted value was of -0.4374448 and the highest was of 5.9902454.

This can be "fixed" by rounding the predictions, but since the objective of this model is a recommendation algorithm, does a value greater than 5 means the user would really like that movie? I believe there is room for discussion if this is a problem after all.

The errors have a mean of 4.3066722×10^{-4} which is close to 0, which is what should we expect from a linear model.

Conclusion

Although the final RMSE is better than the target set in the course, there is more work that could improve the RMSE.

The first issue is the lack of more variables, for example the variable timestamp that I avoided for the reasons mentioned in the data analysis section. Other important variable for the rating could be the director of the movie which is not in the data set. Other relation can exist is between genres and year of release, i.e. genres have different eras of popularity which greatly affects movies quality and therefore their ratings.

The second issue is with hardware limitations, of which if they didn't exist I could make a better algorithm that could use different models at the same time.

So for a future work there are a lot of possibilities for improvement, but these improvements are restricted primarily by data availability and hardware.

So in conclusion this report shows that I was successful in creating a model for predicting the ratings of movies, with the final objective of being a recommendation algorithm. Not only did I use variables directly given in the course, I modified said data to have more variables for the prediction. Which I think I met the objectives of the rubric.