Movielensproject

Harrie Jonkman

5/13/2021

```r
library(tidyverse)
```

```
## -- Attaching packages --------------------------------------- tidyverse 1.3.0 --
```

```
## v ggplot2 3.3.3     v purrr   0.3.4
## v tibble  3.1.1     v dplyr   1.0.5
## v tidyr   1.1.3     v stringr 1.4.0
## v readr   1.4.0     v forcats 0.5.0
```

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

```r
library(tidymodels)
```

```
## -- Attaching packages -------------------------------------- tidymodels 0.1.2 --
```

```
## v broom     0.7.3      v recipes   0.1.15
## v dials     0.0.9      v rsample   0.0.8
## v infer     0.5.3      v tune      0.1.2
## v modeldata 0.1.0      v workflows 0.2.1
## v parsnip   0.1.4      v yardstick 0.0.7
```

```
## -- Conflicts ----------------------------------------- tidymodels_conflicts() --
## x scales::discard() masks purrr::discard()
## x dplyr::filter()   masks stats::filter()
## x recipes::fixed()  masks stringr::fixed()
## x dplyr::lag()      masks stats::lag()
## x yardstick::spec() masks readr::spec()
## x recipes::step()   masks stats::step()
```

```r
library(caret)
```

```
## Loading required package: lattice
```

```
##
## Attaching package: 'caret'
```

```
## The following objects are masked from 'package:yardstick':
##
##     precision, recall, sensitivity, specificity

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

Introduction

This Capstone Project is a part of the course **HarvardX: PH125.9x Data Science**. In this project a part of the MovieLens data were used collected by GroupLens Research. Machine learning techniques are used to predict movie ratings based on different predictors, like user preferance and age of a movie. Using the MovieLens data set and different models, the following R script calculates the RMSE based on user ratings, movieId and the age of the movie. Different steps of machine learning analysis of the data of ratings and predictors are performed in order to find a pattern or insight to the behavior of the data. In this project we alculate RMSE based on movieId, userId, and age of the movie. In this project we alculate RMSE based on movieId, userId, and age of the movie.

This report contains subsequently and is been written in this order: - Problem definition, - Data Ingestion, - Exploratory Analysis, - Modeling and Data Analysis, - Evaluation - Results summarized - Concluding remarks

Problem Definition

Central aim of this project is: > to develop a machine learning algorithm using the inputs in one subset to predict movie ratings in the validation set. Several machine learning algorithm has been used and results have been compared to get maximum possible accuracy in prediction.

For this project, I created a movie recommendation system using the MovieLens dataset. The version of movielens included in the `dslabs` package (which was used for some of the exercises in **PH125.8x: Data Science: Machine Learning**) is just a small subset of a much larger dataset with millions of ratings. I created my own recommendation system using different tools I used throughout the Harvard courses. I used the 10M version of the MovieLens dataset to make the computation a little easier. I downloaded the MovieLens data and ran the code provided to generate your datasets. I trained a machine learning algorithm using the inputs to predict movie ratings in the validation set.

Data Ingestion

First I downloaded the data and followed the instructions

```
# Note: this process could take a couple of minutes

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

library(tidyverse)
library(caret)
library(data.table)

# 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")


# if using R 4.0 or later:
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")

# 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)
```

Because this took a while every time, for myself I used the downloaded dataset in my projectmap.

```r
load("edx.Rdata")
load("validation.Rdata")
```

Validation dataset can be further modified by removing rating column

```r
validation_CM <- validation
validation <- validation %>% select(-rating)
```

Let us first look at the dataset.

```r
head(edx)
```

```
##    userId movieId rating timestamp                         title
## 1:      1     122      5 838985046              Boomerang (1992)
## 2:      1     185      5 838983525               Net, The (1995)
## 3:      1     292      5 838983421               Outbreak (1995)
## 4:      1     316      5 838983392               Stargate (1994)
## 5:      1     329      5 838983392 Star Trek: Generations (1994)
## 6:      1     355      5 838984474       Flintstones, The (1994)
##                           genres
## 1:                Comedy|Romance
## 2:         Action|Crime|Thriller
## 3:  Action|Drama|Sci-Fi|Thriller
## 4:       Action|Adventure|Sci-Fi
## 5: Action|Adventure|Drama|Sci-Fi
## 6:       Children|Comedy|Fantasy
```

Give a summary of the variables but let us look also at the data from another perspective.

```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
##
##
##
```

```r
glimpse(edx)
```

```
## Rows: 9,000,055
## Columns: 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, 83898~
## $ title     <chr> "Boomerang (1992)", "Net, The (1995)", "Outbreak (1995)", "S~
## $ genres    <chr> "Comedy|Romance", "Action|Crime|Thriller", "Action|Drama|Sci~
```

Let us look at the lengthe of the dataset, the number of observations and the colums (variables).

```
length(edx$rating)
```

```
## [1] 9000055
```

```
length(validation$rating)
```

```
## [1] 0
```

```
total_obs<-length(edx$rating)+length(validation$rating)
total_obs
```

```
## [1] 9000055
```

```
ncol(edx)
```

```
## [1] 6
```

```
ncol(validation)
```

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

Because RMSE(Root Mean Square Error) $RMSE = sqrt(mean((true_ratings - predicted_ratings)^2)$ has to be compared in this study, it is important to define its function at the sart of the study.

```
RMSE <- function(true_ratings, predicted_ratings){
  sqrt(mean((true_ratings-predicted_ratings)^2,na.rm=T))
}
```

Ok, after we know something of the dataset, it is time for some preprocess-steps. Year could be an additional variable for the analysis. But, let us first extract year from the title-variable. We do this on similiar way for the edx, validation and validation_CM datasets.

```
edx <- edx %>% mutate(year = as.numeric(str_sub(title,-5,-2)))
validation <- validation %>% mutate(year = as.numeric(str_sub(title,-5,-2)))
validation_CM <- validation_CM %>% mutate(year = as.numeric(str_sub(title,-5,-2)))
```

We also do some preprocessing on the genre-variable to make the analysis more clear and do this for the edx and both validation datasets.

```r
split_edx  <- edx  %>% separate_rows(genres, sep = "\\|")
split_valid <- validation   %>% separate_rows(genres, sep = "\\|")
split_valid_CM <- validation_CM  %>% separate_rows(genres, sep = "\\|")
```

Let us compare what happened and see what the first rows of edx-dataset show us now.

```r
head(edx)
```

```
##    userId movieId rating timestamp                         title
## 1:      1     122      5 838985046              Boomerang (1992)
## 2:      1     185      5 838983525               Net, The (1995)
## 3:      1     292      5 838983421               Outbreak (1995)
## 4:      1     316      5 838983392               Stargate (1994)
## 5:      1     329      5 838983392 Star Trek: Generations (1994)
## 6:      1     355      5 838984474       Flintstones, The (1994)
##                           genres year
## 1:                Comedy|Romance 1992
## 2:         Action|Crime|Thriller 1995
## 3:  Action|Drama|Sci-Fi|Thriller 1995
## 4:       Action|Adventure|Sci-Fi 1994
## 5: Action|Adventure|Drama|Sci-Fi 1994
## 6:       Children|Comedy|Fantasy 1994
```

```r
head(split_edx)
```

```
## # A tibble: 6 x 7
##   userId movieId rating timestamp title            genres    year
##    <int>   <dbl>  <dbl>     <int> <chr>            <chr>    <dbl>
## 1      1     122      5 838985046 Boomerang (1992) Comedy    1992
## 2      1     122      5 838985046 Boomerang (1992) Romance   1992
## 3      1     185      5 838983525 Net, The (1995)  Action    1995
## 4      1     185      5 838983525 Net, The (1995)  Crime     1995
## 5      1     185      5 838983525 Net, The (1995)  Thriller  1995
## 6      1     292      5 838983421 Outbreak (1995)  Action    1995
```

Let us summarize the dataset also.

```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               year
##  Length:9000055     Length:9000055     Min.   :1915
##  Class :character   Class :character   1st Qu.:1987
##  Mode  :character   Mode  :character   Median :1994
##                                        Mean   :1990
##                                        3rd Qu.:1998
##                                        Max.   :2008
```

And summarize it also for the splitted set also.

```r
summary(split_edx)
```

```
##      userId         movieId          rating        timestamp
##  Min.   :    1   Min.   :    1   Min.   :0.500   Min.   :7.897e+08
##  1st Qu.:18140   1st Qu.:  616   1st Qu.:3.000   1st Qu.:9.472e+08
##  Median :35784   Median : 1748   Median :4.000   Median :1.042e+09
##  Mean   :35886   Mean   : 4277   Mean   :3.527   Mean   :1.035e+09
##  3rd Qu.:53638   3rd Qu.: 3635   3rd Qu.:4.000   3rd Qu.:1.131e+09
##  Max.   :71567   Max.   :65133   Max.   :5.000   Max.   :1.231e+09
##     title              genres               year
##  Length:23371423    Length:23371423    Min.   :1915
##  Class :character   Class :character   1st Qu.:1987
##  Mode  :character   Mode  :character   Median :1995
##                                        Mean   :1990
##                                        3rd Qu.:1998
##                                        Max.   :2008
```

Explorative Analysis

After we prepared our dataset (preprocessing and splitting), it is important to understand the dataset very well. For this we do some explorative analysis which is very important before we start modeling. Understanding the dataset precedes modeling and training the dataset.

Unique numbers

Let us first research the number of unique movies and users in the edx dataset.

```
edx %>% summarize(n_users = n_distinct(userId), n_movies = n_distinct(movieId))
```

```
##   n_users n_movies
## 1   69878    10677
```

Total movie ratings per genre

Let us also count the total movie ratings per genre, counts in descent order.

```
genre_rating <- split_edx%>%
  group_by(genres) %>%
  summarize(count = n()) %>%
  arrange(desc(count))

genre_rating
```

```
## # A tibble: 20 x 2
##    genres               count
##    <chr>                <int>
##  1 Drama              3910127
##  2 Comedy             3540930
##  3 Action             2560545
##  4 Thriller           2325899
##  5 Adventure          1908892
##  6 Romance            1712100
##  7 Sci-Fi             1341183
##  8 Crime              1327715
##  9 Fantasy             925637
## 10 Children            737994
## 11 Horror              691485
## 12 Mystery             568332
## 13 War                 511147
## 14 Animation           467168
## 15 Musical             433080
## 16 Western             189394
## 17 Film-Noir           118541
## 18 Documentary          93066
## 19 IMAX                  8181
## 20 (no genres listed)       7
```

Ratings distribution

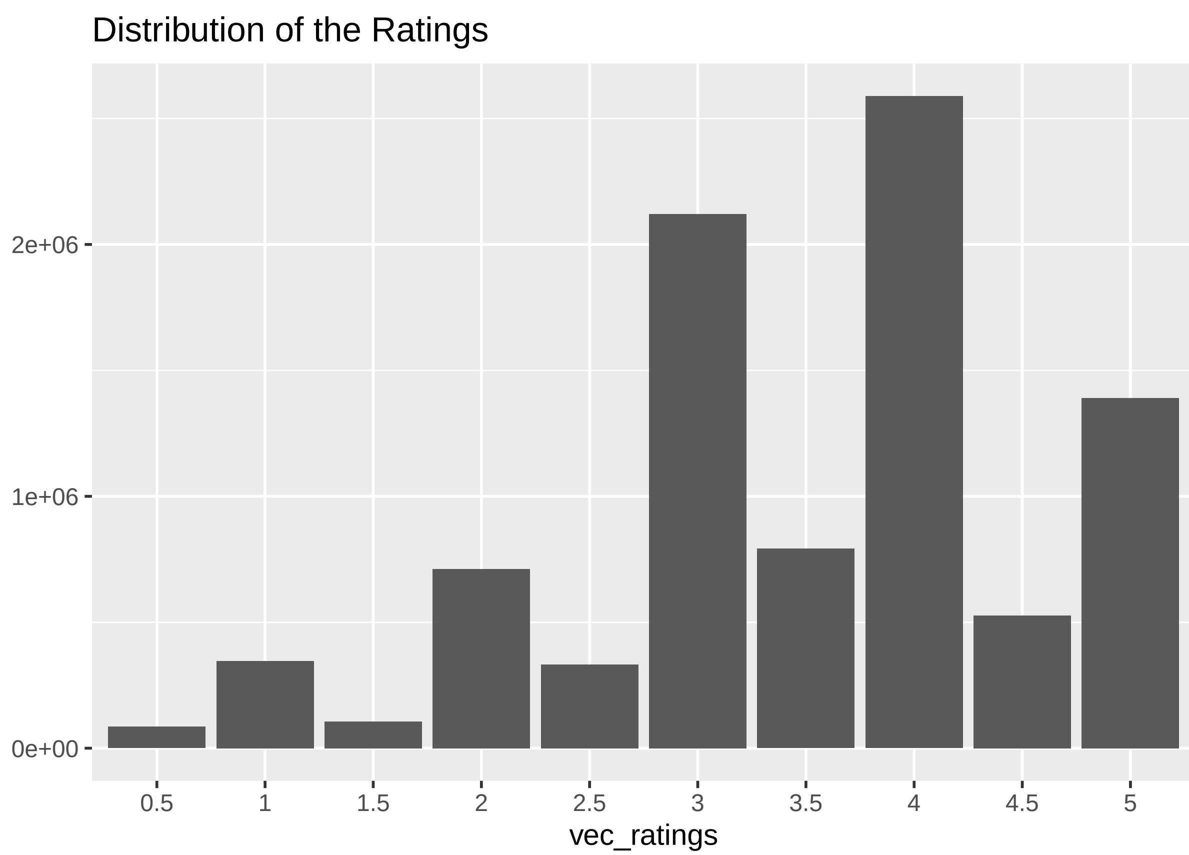
How are the movies rated?

```
vec_ratings <- as.vector(edx$rating)
unique(vec_ratings)
```

```
##  [1] 5.0 3.0 2.0 4.0 4.5 3.5 1.0 1.5 2.5 0.5
```

And how are these ratings distributed?

```
vec_ratings <- vec_ratings[vec_ratings != 0]
vec_ratings <- factor(vec_ratings)
qplot(vec_ratings) +
  ggtitle("Distribution of the Ratings")
```

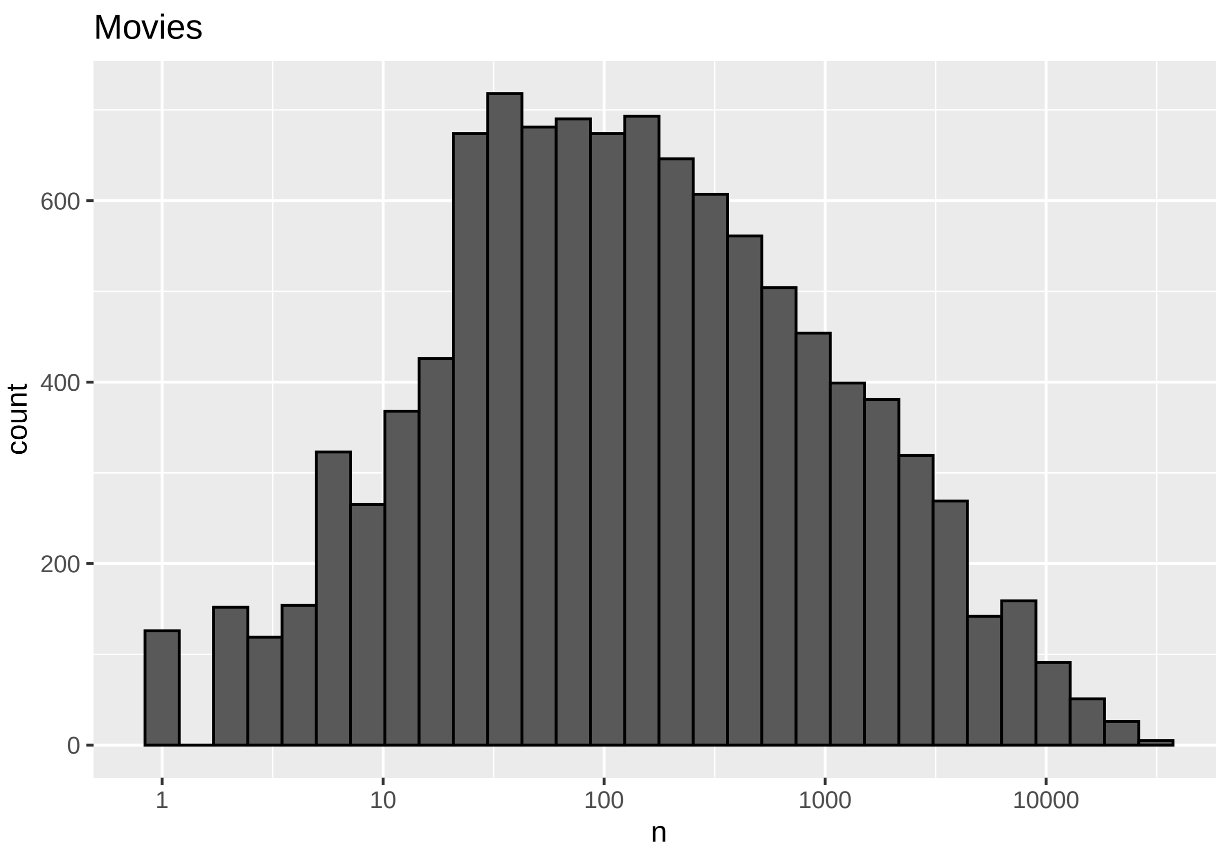


The above histogram and rating distribution show us that the users have a general tendency to rate movies between 3 and 4 (more exact mainly 3 and 4). This is a very general conclusion. We should further explore the effect of different features to make a good predictive model. Let us look at the movies, the users, the genres and the years subsequently.

Data Analysis Strategies

Let us do four analyses to see what kind of models we can use later on. ### 1. Movie biases Some movies are rated more often than other movies. Below you see this distribution. This explores *movie biases*. We have to incorporate this knowledge later.

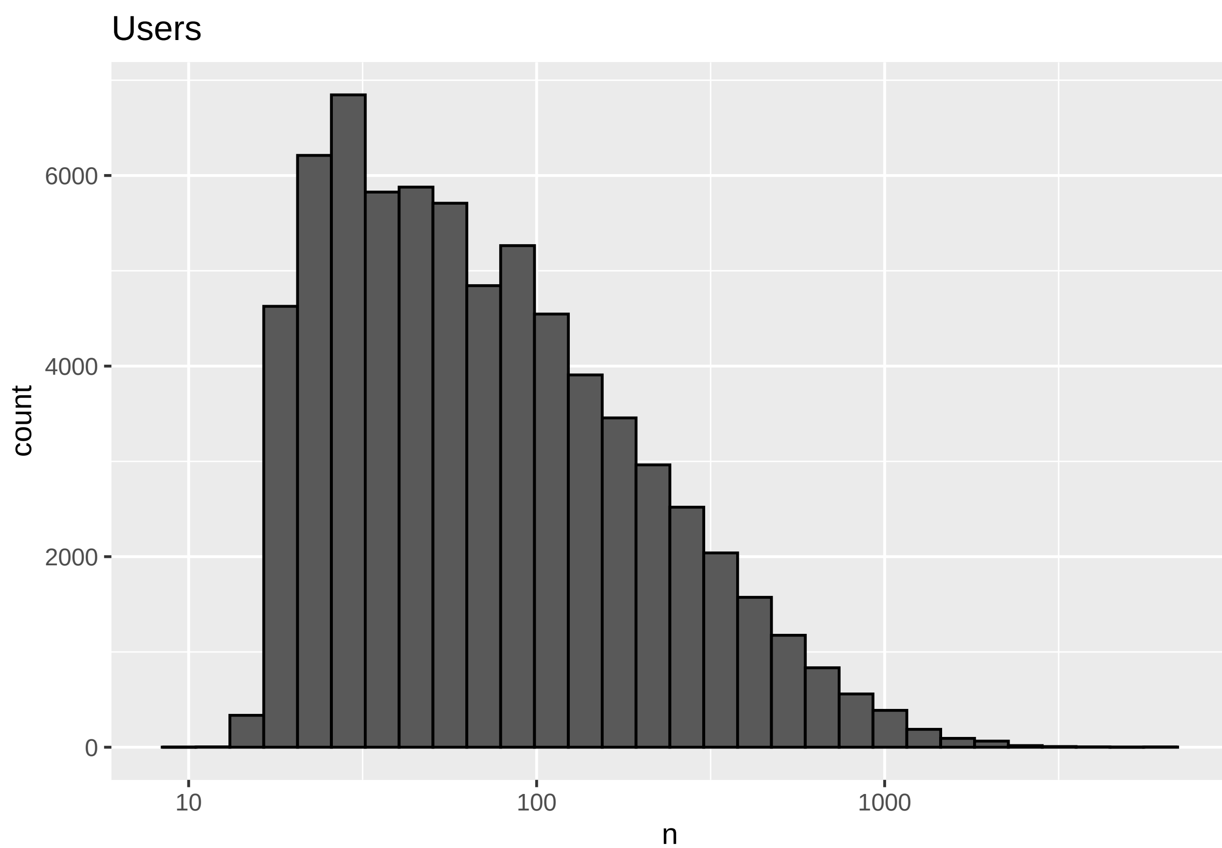
```
edx %>%
  count(movieId) %>%
  ggplot(aes(n)) +
  geom_histogram(bins = 30, color = "black") +
  scale_x_log10() +
  ggtitle("Movies")
```



2. Ueser bias

Some users are positive and some have negative reviews because of their own personal liking/disliking regardless of movie. The distribution of each user's ratings for movies. This shows the *users bias*. How to address this later in our model?

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



Above plot shows that not every user is equally active. Some users have rated very few movie and their opinion may bias the prediction results.

3. Genres popularity per year.

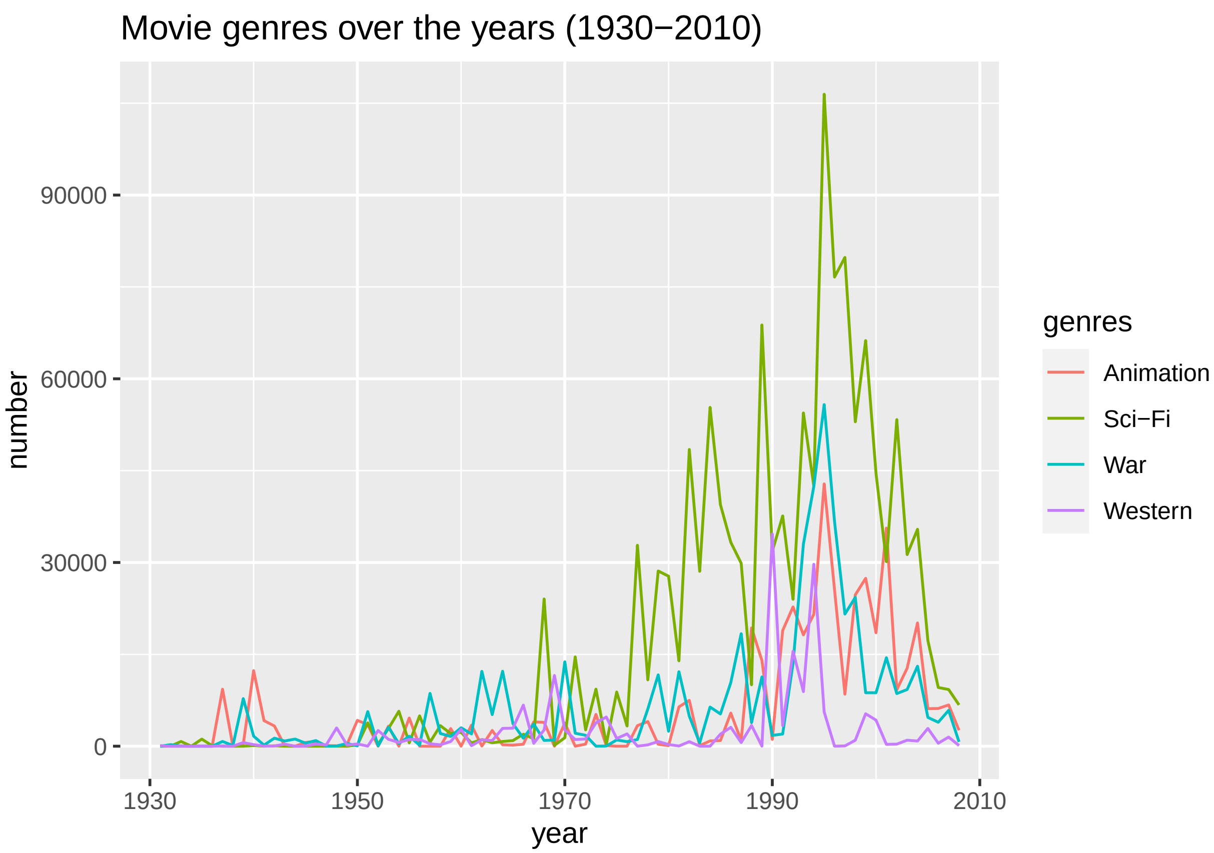
The popularity of the movie genre depends strongly on the contemporary issues. So we should also explore the time dependent analysis. Here we tackle the issue of temporal evolution of users taste over different popular genre (*genres popularity per year*). So define an object, omit the missing values, select three colums we are interested in, define genre as factor, count them add missings yeears/genres

```
genres_popularity <- split_edx %>%
  na.omit() %>%
  select(movieId, year, genres) %>%
  mutate(genres = as.factor(genres)) %>%
  group_by(year, genres) %>%
  summarise(number = n(), .groups = 'drop') %>%
  complete(year = full_seq(year, 1), genres, fill = list(number = 0))
```

Let us plot this object Genres vs year; 4 genres are chosen for readability: animation, science fiction, war and western movies. This plots depicts some genre become more popular over others for different period of time.

```
genres_popularity %>%
  filter(year > 1930) %>%
  filter(genres %in% c("War", "Sci-Fi", "Animation", "Western")) %>%
```

```
  ggplot(aes(x = year, y = number)) +
  geom_line(aes(color=genres)) +
  scale_fill_brewer(palette = "Paired") +
 ggtitle("Movie genres over the years (1930-2010)")
```

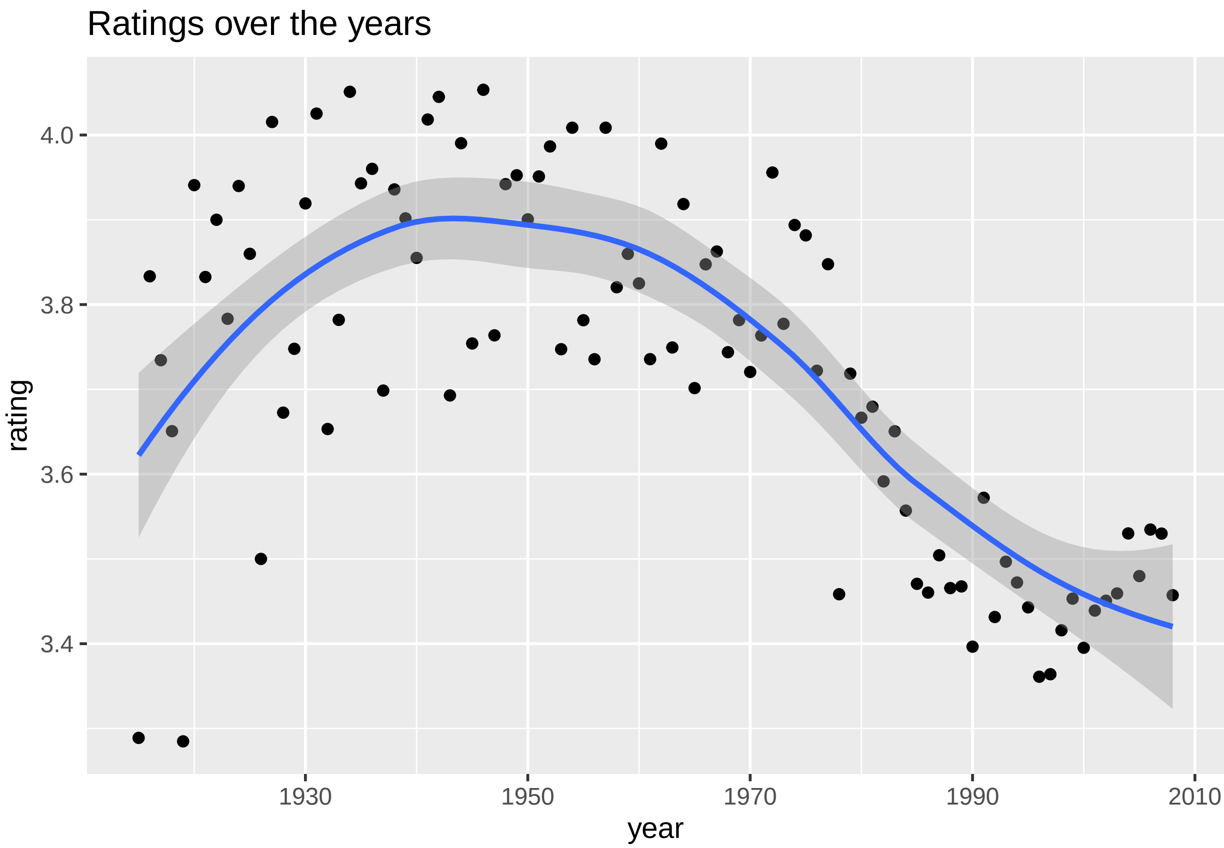


4. Average rating of movies over the years

Do the users mindset also evolve over time? This can also effect the *average rating of movies over the years*. How do visualize such effect: plot rating vs release year

```
edx %>% group_by(year) %>%
  summarize(rating = mean(rating)) %>%
  ggplot(aes(year, rating)) +
  geom_point() +
  geom_smooth()+
   ggtitle("Ratings over the years")
```

```
## `geom_smooth()` using method = 'loess' and formula 'y ~ x'
```



Conclusions of data analysis strategies: - There are users biases;
- There are movie biases;
- There are differences between genres of movies;
- There are rating differences over the years.

Modeling and Data Analysis

Using the knowledge of de explorative data analysis and the four data analysis strategies we set up different models and compare their results.

Let us first initiate RMSE results to compare various models

```
rmse_results <- data_frame()
```

```
## Warning: `data_frame()` was deprecated in tibble 1.1.0.
## Please use `tibble()` instead.
```

1. Simplest possible model

Then we start with the most simple model. Dataset's mean rating is used to predict the same rating for all movies, regardless of the user and movie.

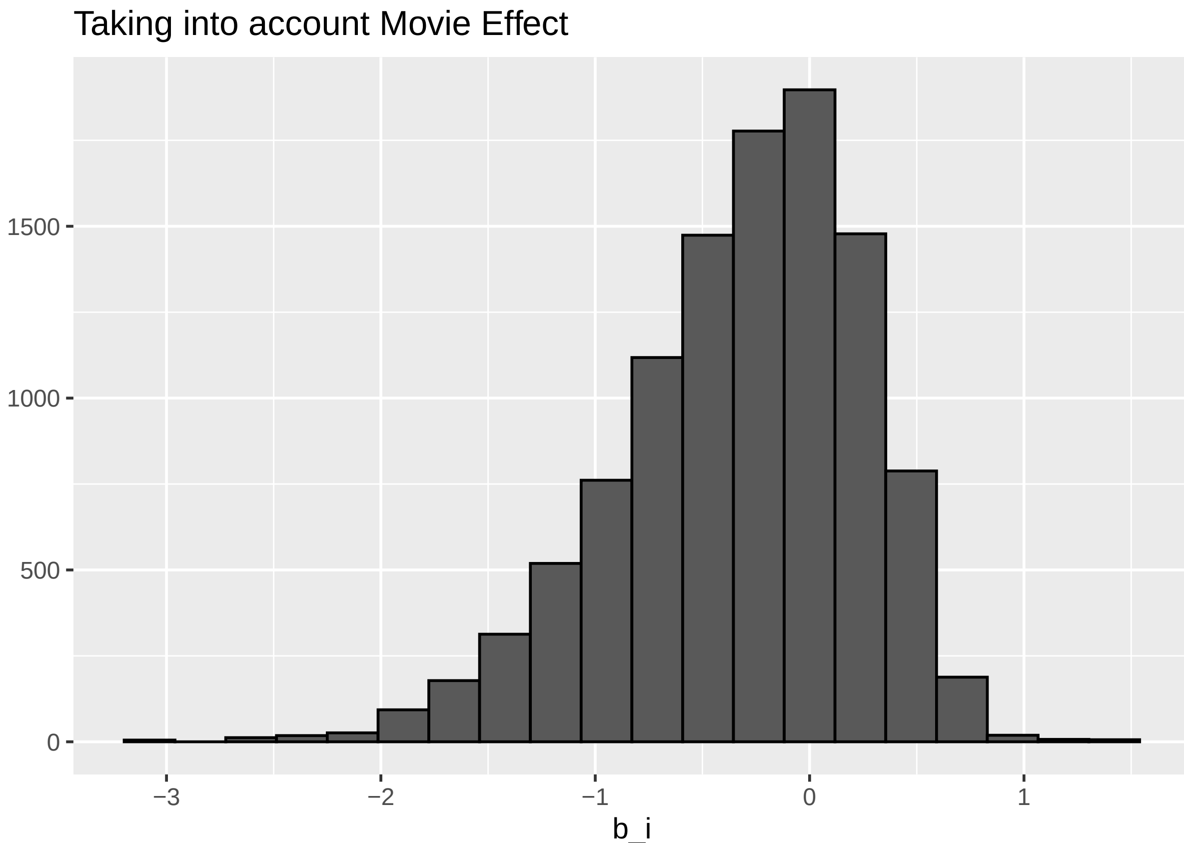
```
mu <- mean(edx$rating)
mu
```

```
## [1] 3.512465
```

2. Penalty Term (b_i)- Movie Effect

Different movies are rated differently. As shown in the exploration, the histogram is not symmetric and is skewed towards negative rating effect. The movie effect can be taken into account by taking the difference from mean rating as shown in the following chunk of code.

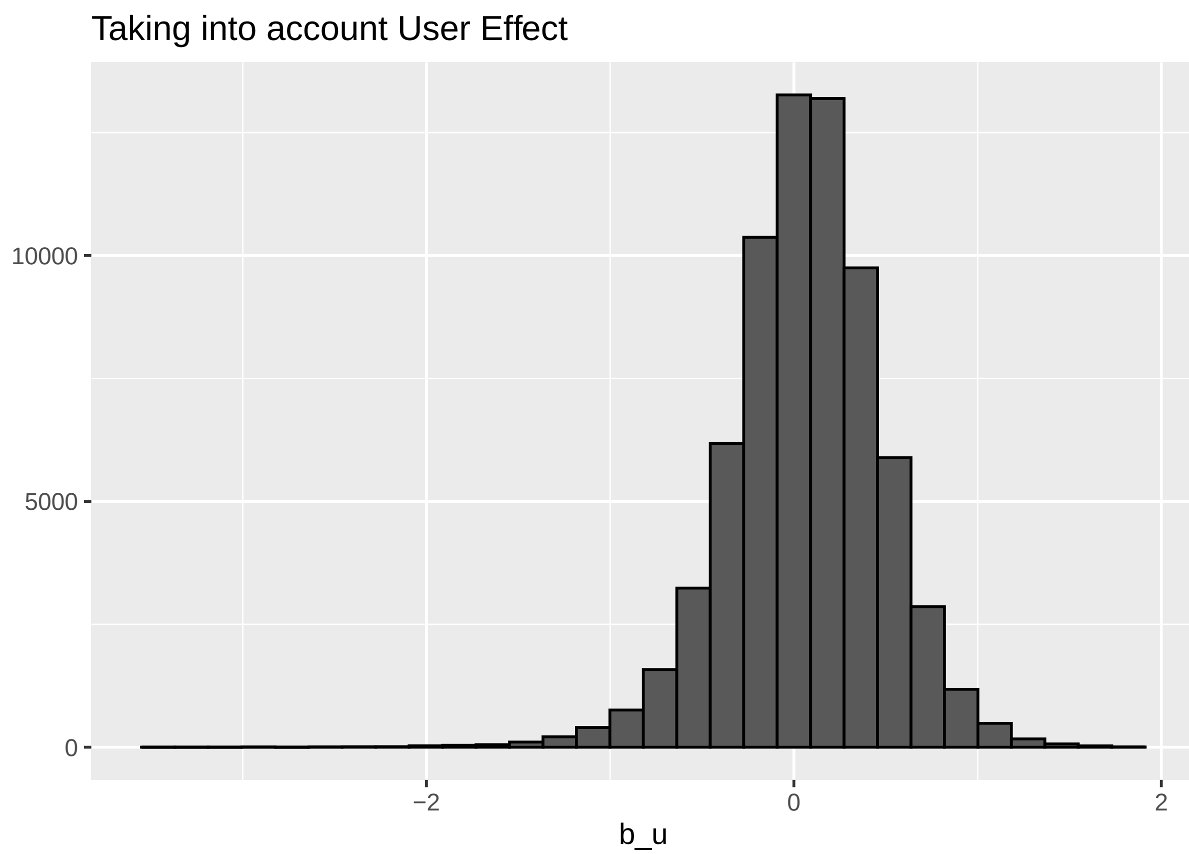
```
movie_avgs_norm <- edx %>%
  group_by(movieId) %>%
  summarize(b_i = mean(rating - mu))
movie_avgs_norm %>% qplot(b_i, geom ="histogram", bins = 20, data = ., color = I("black")) +
  ggtitle("Taking into account Movie Effect")
```



3. Penalty Term (b_u)- User Effect

Different users are different in terms of how they rate movies. Some cranky users may rate a good movie lower or some very generous users just don't care for the assessment. We have already seen this pattern in our data exploration plot (**user bias**). We can show this by using this code.

```
user_avgs_norm <- edx %>%
  left_join(movie_avgs_norm, by='movieId') %>%
  group_by(userId) %>%
  summarize(b_u = mean(rating - mu - b_i))
user_avgs_norm %>% qplot(b_u, geom ="histogram", bins = 30, data = ., color = I("black")) +
  ggtitle("Taking into account User Effect")
```



Evaluation

The quality of different models will be assessed by the RMSE (the lower the score on this is, the better). For this we use the `validation_CM` dataset.

1. Baseline Model

It's simply a model which ignores all the feathers and simply calculates mean rating. This model acts as a baseline model and we will try to improve RMSE relative to this baseline standard model. We test the results based on simple prediction.

```
baseline_rmse <- RMSE(validation_CM$rating,mu)

baseline_rmse
```

```
## [1] 1.061202
```

Let us show the results on this way.

```
rmse_results <- data_frame(method = "Using mean only", RMSE = baseline_rmse)
rmse_results
```

```
## # A tibble: 1 x 2
##   method           RMSE
##   <chr>           <dbl>
## 1 Using mean only  1.06
```

2. Movie Effect Model

An improvement in the RMSE is achieved by adding the movie effects only.

```
predicted_ratings_movie_norm <- validation %>%
  left_join(movie_avgs_norm, by='movieId') %>%
  mutate(pred = mu + b_i)
model_1_rmse <- RMSE(validation_CM$rating,predicted_ratings_movie_norm$pred)
rmse_results <- bind_rows(rmse_results,
                          data_frame(method="Movie Effect Model",
                                     RMSE = model_1_rmse ))
rmse_results %>% knitr::kable()
```

method	RMSE
Using mean only	1.0612018
Movie Effect Model	0.9439087

We see an improvement from 1.0612018 to 0.9439087. Or show it on this way.

```
rmse_results
```

```
## # A tibble: 2 x 2
##   method              RMSE
##   <chr>              <dbl>
## 1 Using mean only    1.06
## 2 Movie Effect Model 0.944
```

The error has dropped and this motivates us to move on this path further.

3. Movie and User Effect Model

Given that movie and users biases both obscure the prediction of movie rating, a further improvement in the RMSE is achieved by adding the user effect.We test and save the rmse results.

```
predicted_ratings_user_norm <- validation %>%
  left_join(movie_avgs_norm, by='movieId') %>%
  left_join(user_avgs_norm, by='userId') %>%
  mutate(pred = mu + b_i + b_u)

model_2_rmse <- RMSE(validation_CM$rating,predicted_ratings_user_norm$pred)
rmse_results <- bind_rows(rmse_results,
                          data_frame(method="Movie and User Effect Model",
                                     RMSE = model_2_rmse ))
rmse_results %>% knitr::kable()
```

method	RMSE
Using mean only	1.0612018
Movie Effect Model	0.9439087
Movie and User Effect Model	0.8653488

It drops furhter to 0.8653488. Or on the similar way as above and see a good improvement from our last model.

```
rmse_results
```

```
## # A tibble: 3 x 2
##   method                       RMSE
##   <chr>                       <dbl>
## 1 Using mean only             1.06
## 2 Movie Effect Model          0.944
## 3 Movie and User Effect Model 0.865
```

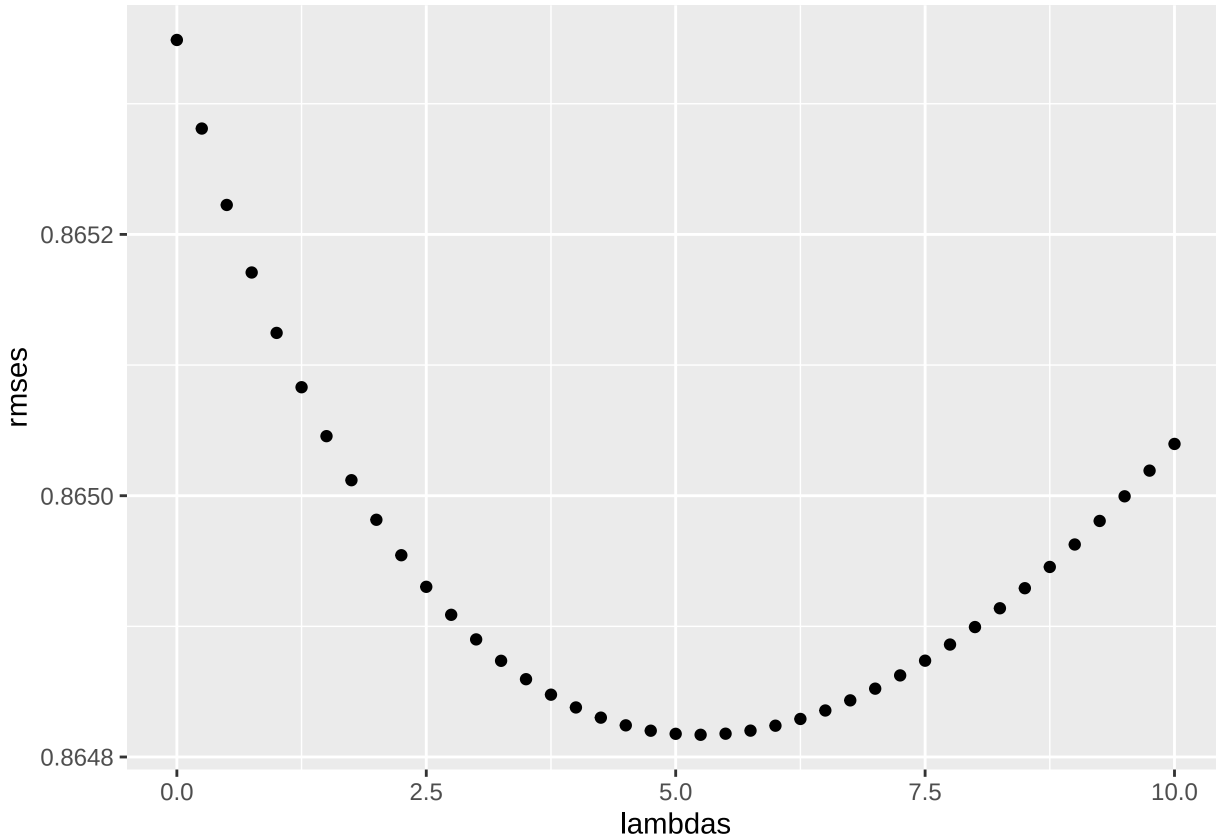
Model 4. Regularized movie and user effect model

So estimates of b_i and b_u are caused by movies with very few ratings and of some users that only rated a very small number of movies. Hence this can strongly influence the prediction. The use of the regularization permits to penalize these aspects. We should find the value of lambda (that is a tuning parameter) that will minimize the RMSE. This shrinks the b_i and b_u in case of small number of ratings. Let us use the cross-validation for this tuning part. We research different lambda's for b_i and b_u, and rate prediction and test.

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

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

  mu <- mean(edx$rating)

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

  b_u <- edx %>%
    left_join(b_i, by="movieId") %>%
    group_by(userId) %>%
    summarize(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(pred = mu + b_i + b_u) %>%
    .$pred

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

Let us plot RMSE vs lambdas to select the optimal lambda

```r
qplot(lambdas, rmses)
```



For the full model, the optimal lambda is the lowest:

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

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

Results summarized

For the full model, the optimal lambda is: 5.25. Let us regularized the estimates of b_i and b_u using the chosen lambda, predict the rating, test, save and show the results.

The new results will be:

```
movie_avgs_reg <- edx %>%
  group_by(movieId) %>%
  summarize(b_i = sum(rating - mu)/(n()+lambda), n_i = n())

user_avgs_reg <- edx %>%
  left_join(movie_avgs_reg, by='movieId') %>%
  group_by(userId) %>%
  summarize(b_u = sum(rating - mu - b_i)/(n()+lambda), n_u = n())

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) %>%
  .$pred

model_3_rmse <- RMSE(validation_CM$rating,predicted_ratings_reg)
rmse_results <- bind_rows(rmse_results,
                          data_frame(method="Regularized Movie and User Effect Model",
                                     RMSE = model_3_rmse ))
rmse_results %>% knitr::kable()
```

method	RMSE
Using mean only	1.0612018
Movie Effect Model	0.9439087
Movie and User Effect Model	0.8653488
Regularized Movie and User Effect Model	0.8648170

Concluding Remarks

The RMSE values of all the represented models are the following:

method	RMSE
Using mean only	1.0612018
Movie Effect Model	0.9439087
Movie and User Effect Model	0.8653488
Regularized Movie and User Effect Model	0.8648170

We therefore found the lowest value of RMSE that is 0.8648170.

So we can confirm that the final model for our project is the following:

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

- This model work well if the average user doesn't rate a particularly good/popular movie with a large positive b_i, by disliking a particular movie.

- We can affirm to have built a machine learning algorithm to predict movie ratings with MovieLens dataset.

- The regularized model including the effect of user is characterized by the lower RMSE value and is hence the optimal model to use for the present project.

- The optimal model characterised by the lowest RMSE value (0.8648170).

- We could also affirm that improvements in the RMSE could be achieved by adding other effect (genre, year, age,..).

- Other different machine learning models could also improve the results further, but my brain and hardware have limitations, as well as the RAM. They are a constraint.

Appendix - Enviroment

```r
print("Operating System:")
```

```
## [1] "Operating System:"
```

```r
version
```

```
##                _
## platform       x86_64-apple-darwin17.0
## arch           x86_64
## os             darwin17.0
## system         x86_64, darwin17.0
## status
## major          4
## minor          0.3
## year           2020
## month          10
## day            10
## svn rev        79318
## language       R
## version.string R version 4.0.3 (2020-10-10)
## nickname       Bunny-Wunnies Freak Out
```

Literature