Movielens Project

Harvardx Data Science Capstone Project

by

Nedal Shami

November 2021

Introduction

In this project we will create a recommendation system for movies. The data that we will use is MovieLens 10M data set, which provided by GroupLens, a research group from University of Minnesota. It contains over ten million ratings for over ten thousand movies from various users. This dataset is a small subset of a much larger (and famous) dataset with several millions of ratings with approximately 70,000 users and 11,000 different movies divided in 20 genres such as Action, Adventure, Horror, Drama, Thriller and more. MovieLens 10M dataset contains 10 million ratings for over 10,000 movies by more than 72,000 users. It's includes the id of the user and the movie, as well as the rating, genre, and the timestamp. The goal of this project is to predict movie ratings. To do that, the dataset was subsetted into two: the train and validation set. The validation set is 10% of the original data and is not used in the construction of the model.

Loading libraries and data

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

## Loading required package: tidyverse

## Warning in as.POSIXlt.POSIXct(Sys.time()): unable to identify
current timezone 'W':
## please set environment variable 'TZ'

## -- Attaching packages ---------------------------------------
tidyverse 1.3.1 --

## v ggplot2 3.3.5     v purrr   0.3.4
## v tibble  3.1.3     v dplyr   1.0.7
## v tidyr   1.1.3     v stringr 1.4.0
## v readr   2.0.0     v forcats 0.5.1
## -- Conflicts ------------------------------------------
tidyverse_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

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

## Warning: package 'ggpubr' was built under R version 4.1.1

# 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("ml-
10M100K/ratings.dat")),
                 col.names = c("userId", "movieId", "rating",
"timestamp"))

movies <- str_split_fixed(readLines( "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))
```

Checking data and data preprocessing

```
head(movies)

##   movieId                              title
## 1       1                   Toy Story (1995)
## 2       2                     Jumanji (1995)
## 3       3            Grumpier Old Men (1995)
## 4       4           Waiting to Exhale (1995)
## 5       5 Father of the Bride Part II (1995)
## 6       6                        Heat (1995)
##                                         genres
## 1 Adventure|Animation|Children|Comedy|Fantasy
## 2                  Adventure|Children|Fantasy
## 3                              Comedy|Romance
## 4                        Comedy|Drama|Romance
## 5                                      Comedy
## 6                       Action|Crime|Thriller

head(ratings)

##    userId movieId rating timestamp
## 1:      1     122      5 838985046
## 2:      1     185      5 838983525
## 3:      1     231      5 838983392
## 4:      1     292      5 838983421
## 5:      1     316      5 838983392
## 6:      1     329      5 838983392

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

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

extract release year from title field, and split it to *title* that contains movie's name and *year* that contains release year.

```
movies_df <- movielens%>%
  extract(title, c("title", "year"), regex = "^(.*) \\(([0-9 \\-
]*)\\)$", remove = T)
head(movies_df)
```

```
##    userId movieId rating timestamp                  title year
## 1:      1     122      5 838985046              Boomerang 1992
## 2:      1     185      5 838983525               Net, The 1995
## 3:      1     231      5 838983392          Dumb & Dumber 1994
## 4:      1     292      5 838983421               Outbreak 1995
## 5:      1     316      5 838983392               Stargate 1994
## 6:      1     329      5 838983392 Star Trek: Generations 1994
##                          genres
## 1:               Comedy|Romance
## 2:        Action|Crime|Thriller
## 3:                       Comedy
## 4:  Action|Drama|Sci-Fi|Thriller
## 5:      Action|Adventure|Sci-Fi
## 6: Action|Adventure|Drama|Sci-Fi
```

Split data to training set and validation set, Validation set will be 10% of MovieLens data.

```
# 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)`
```

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

```
test_index <- createDataPartition(y = movielens$rating, times = 1, p =
0.1, list = FALSE)
edx <- movies_df[-test_index,]
temp <- movies_df[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",
"year", "genres")

nrow(removed)

## [1] 8

edx <- rbind(edx, removed)

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

Checking edx data

```
head(edx)

##    userId movieId rating timestamp                  title year
## 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
```

Exploratory Data Analysis (EDA)

The 10 Millions dataset is divided into two dataset: edx for training purpose and validation for the validation phase. The edx dataset contains approximately 9 Millions of rows with 70.000 different users and 11.000 There is no missing values (0 or NA).

to see how many unique user and movie we have, we use the below code:

```r
number_of_users <- edx %>% distinct(userId) %>% summarise(n())
number_of_movies <- edx %>% distinct(movieId)%>% summarise(n())

kable(tibble(
    Users = number_of_users,
    Movies = number_of_movies
  ))
```

Users	Movies
69878	10677

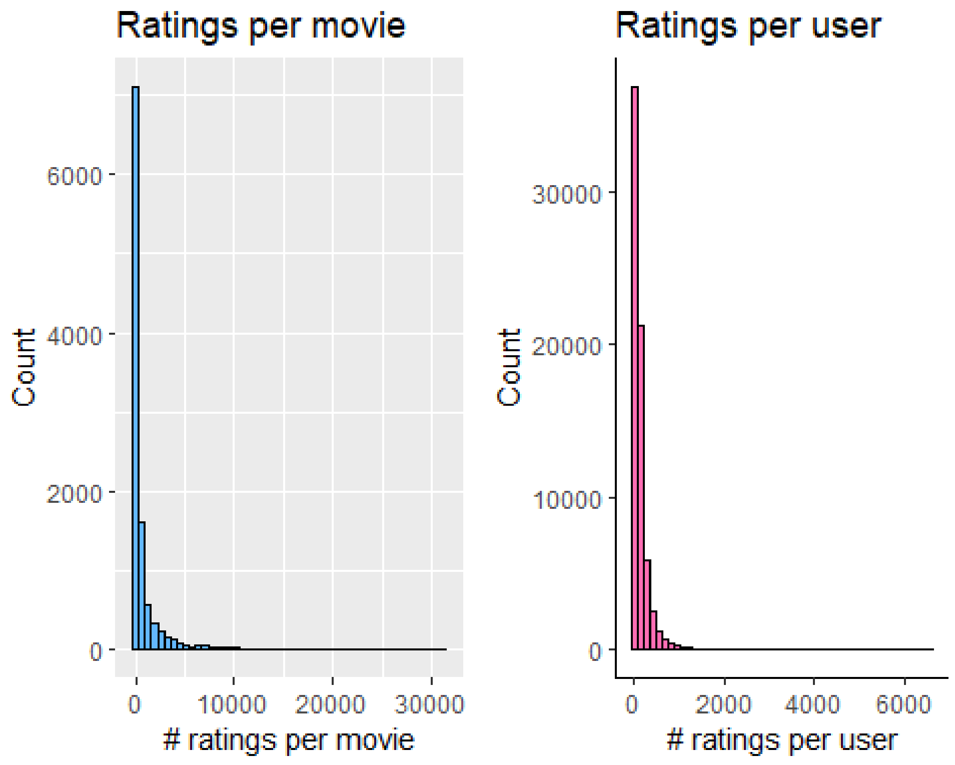
To check if there are any missing values, we use the following code. and sure enough, there is no missing values.

```r
#sum(is.na(edx))

NA_count <- t(data.frame(lapply(edx, function(x) sum(is.na(x)))))
colnames(NA_count) <- c("NA_count")
kable(NA_count)
```

	NA_count
userId	0
movieId	0
rating	0
timestamp	0
title	0
year	0
genres	0

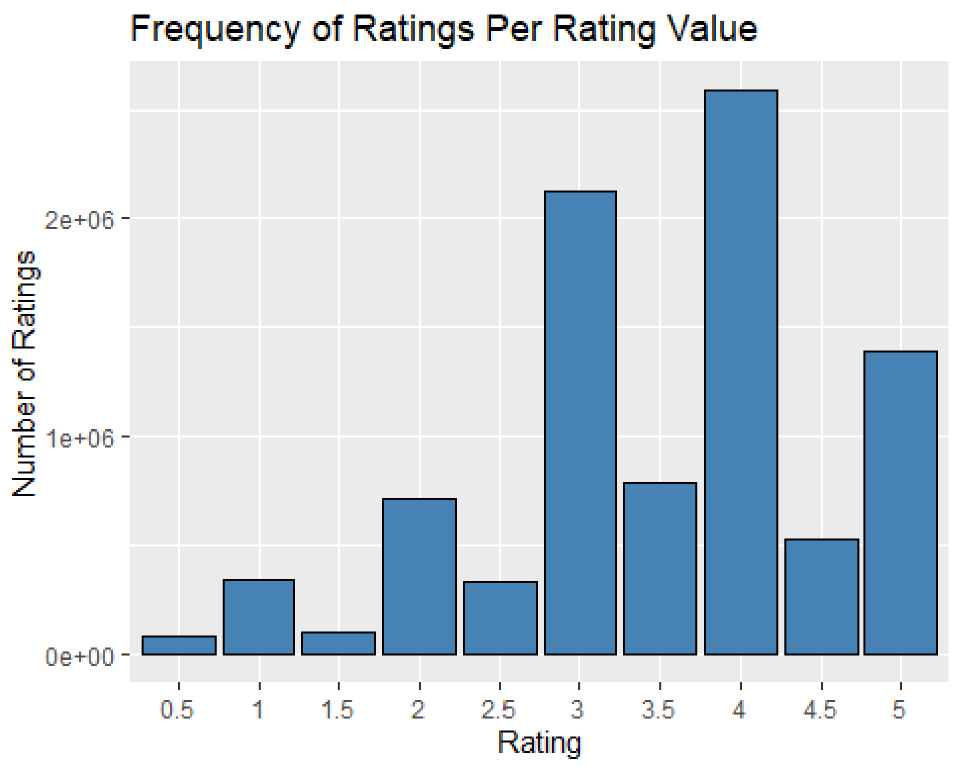
As we saw before, 69878 unique users provided ratings and 10677 unique movies were rated. Our test set has about 9 million rows, which implies that not every user has rated every movie. In addition to not having every movie rated by every user, some movies were rated more than others and some users have rated more than others, as shown in the two histograms below

```r
movies_hist <- edx %>%
  count(movieId) %>%
  ggplot(aes(n)) +
  geom_histogram(bins = 50, color = "black", fill = "steelblue1") +
  labs(title = "Ratings per movie",
       x = "# ratings per movie", y = "Count")

users_hist<-edx %>%
  count(userId) %>%
  ggplot(aes(n)) +
  geom_histogram(bins = 50, color = "black",fill = "hotpink1") +
  labs(title = "Ratings per user",
       x = "# ratings per user", y = "Count") +
  theme_classic()

ggarrange(movies_hist, users_hist,
          ncol = 2, nrow = 1)
```



the below figure shows the distribution of the ratings. The plot indicates that, although ratings are on a scale of 0.5-5, users tent to rate movies more positively than negatively. It is also apparent that integer ratings are more common than half star ratings.

```r
ratings_count <- edx %>%
  mutate(rating = as.factor(rating)) %>%
  group_by(rating) %>%
  summarise(number_of_ratings = n()) %>%
  mutate(prop = (number_of_ratings / sum(number_of_ratings)) * 100) %>%
  select(rating, number_of_ratings, prop)

ggplot(ratings_count, aes(rating, number_of_ratings)) +
  geom_bar(stat = "identity", fill="steelblue", color="black") +
  xlab("Rating") + ylab("Number of Ratings") +
  ggtitle("Frequency of Ratings Per Rating Value")
```



Data preprocessig

From EDA analysis, we notice that the genres are pipe-separated values. It's necessary to extract them for more consistent estimate. We need also to convert timestamp filed to date format and extract year and month of rating as a separate fields. The pre-processing phase is composed by this steps: 1. Separate each genre from the pipe-separated value. It increases the size of both datasets (edx and validation). 2. Convert timestamp to a human readable date format 3. Extract the month and the year from the date

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

head(edx_seperated_genres)

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

```
validation_seperated_genres <-validation %>% separate_rows(genres, sep
= "\\|")

head(validation_seperated_genres)
```

```
## # A tibble: 6 x 7
##   userId movieId rating timestamp title         year  genres
##    <int>   <dbl>  <dbl>     <int> <chr>         <chr> <chr>
## 1      1     231      5 838983392 Dumb & Dumber 1994  Comedy
## 2      1     480      5 838983653 Jurassic Park 1993  Action
## 3      1     480      5 838983653 Jurassic Park 1993  Adventure
## 4      1     480      5 838983653 Jurassic Park 1993  Sci-Fi
## 5      1     480      5 838983653 Jurassic Park 1993  Thriller
## 6      1     586      5 838984068 Home Alone    1990  Children
```

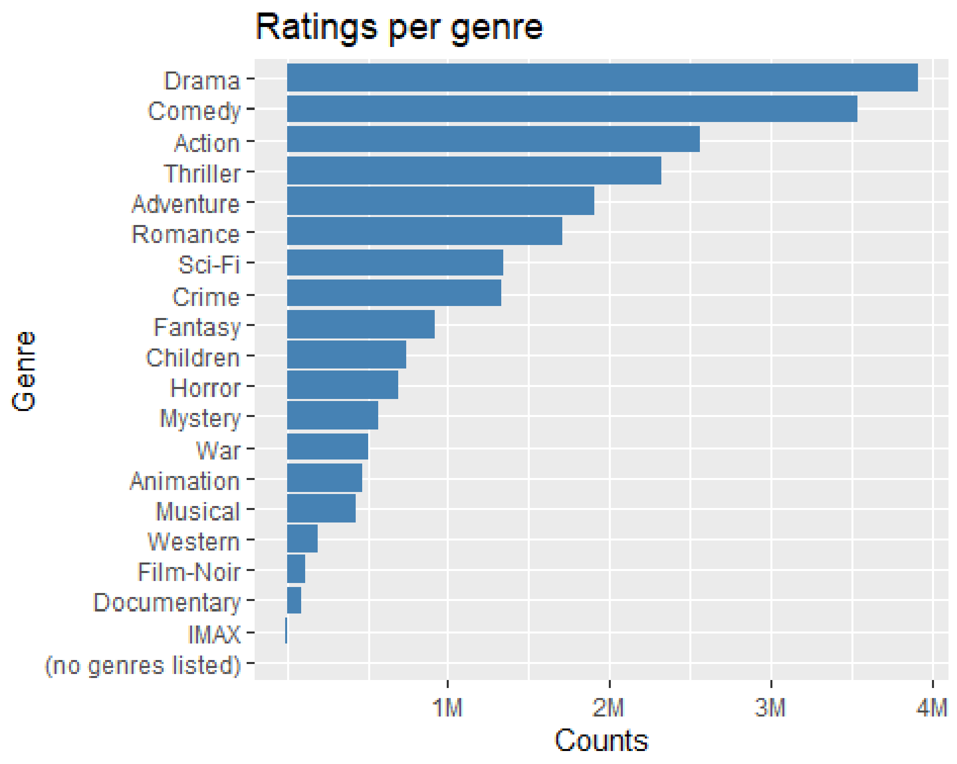
after separating genres, we can now analyze data based on genres. we can find that some genre has been rated much than others. The most rated genres are Drama and Comedy.

```
genres_count <- edx_seperated_genres %>%
  group_by(genres) %>%
  summarize(count = n())%>%
  arrange(desc(count))
kable(genres_count)
```

genres	count
Drama	3910127
Comedy	3540930
Action	2560545
Thriller	2325899
Adventure	1908892
Romance	1712100
Sci-Fi	1341183
Crime	1327715
Fantasy	925637
Children	737994
Horror	691485
Mystery	568332
War	511147

Animation	467168
Musical	433080
Western	189394
Film-Noir	118541
Documentary	93066
IMAX	8181
(no genres listed)	7

```r
genres_count %>% ggplot(aes(x= reorder(genres, count),y=count))+
  geom_bar(stat="identity" , fill ="steelblue")+
  labs(title = "Ratings per genre",x = "Genre", y = "Counts")+
  scale_y_continuous(labels = paste0(1:4, "M"), breaks = 10^6 * 1:4)+
  coord_flip()
```



convert time stamp to date format and check the result.

```r
edx_seperated_genres$date <- as.POSIXct(edx_seperated_genres$timestamp,
origin = "1970-01-01")  # as.POSIXct function
head(edx_seperated_genres)
```

```
## # A tibble: 6 x 8
##   userId movieId rating timestamp title     year  genres   date
##    <int>   <dbl>  <dbl>     <int> <chr>     <chr> <chr>    <dttm>
## 1      1     122      5 838985046 Boomerang 1992  Comedy   1996-08-
```

```
02 11:24:06
## 2       1      122       5 838985046 Boomerang 1992  Romance   1996-08-
02 11:24:06
## 3       1      185       5 838983525 Net, The  1995  Action    1996-08-
02 10:58:45
## 4       1      185       5 838983525 Net, The  1995  Crime     1996-08-
02 10:58:45
## 5       1      185       5 838983525 Net, The  1995  Thriller 1996-08-
02 10:58:45
## 6       1      292       5 838983421 Outbreak  1995  Action    1996-08-
02 10:57:01
```

```
validation_seperated_genres$date <-
as.POSIXct(validation_seperated_genres$timestamp, origin = "1970-01-
01")  # as.POSIXct function
head(validation_seperated_genres)
```

```
## # A tibble: 6 x 8
##    userId movieId rating timestamp title         year  genres    date
##     <int>   <dbl>  <dbl>     <int> <chr>         <chr> <chr>
<dttm>
## 1       1     231       5 838983392 Dumb & Dumber 1994  Comedy
1996-08-02 10:56:32
## 2       1     480       5 838983653 Jurassic Park 1993  Action
1996-08-02 11:00:53
## 3       1     480       5 838983653 Jurassic Park 1993  Adventure
1996-08-02 11:00:53
## 4       1     480       5 838983653 Jurassic Park 1993  Sci-Fi
1996-08-02 11:00:53
## 5       1     480       5 838983653 Jurassic Park 1993  Thriller
1996-08-02 11:00:53
## 6       1     586       5 838984068 Home Alone    1990  Children
1996-08-02 11:07:48
```

Create features for rate year and rate month for both test and validation sets.

```
edx_seperated_genres$rate_year <-
format(edx_seperated_genres$date,"%Y")
edx_seperated_genres$rate_month <-
format(edx_seperated_genres$date,"%m")

validation_seperated_genres$rate_year <-
format(validation_seperated_genres$date,"%Y")
validation_seperated_genres$rate_month <-
format(validation_seperated_genres$date,"%m")
```

Checking data

```
head(edx_seperated_genres)

## # A tibble: 6 x 10
##   userId movieId rating timestamp title     year  genres   date
##    <int>   <dbl>  <dbl>     <int> <chr>     <chr> <chr>    <dttm>
## 1      1     122      5 838985046 Boomerang 1992  Comedy   1996-08-
02 11:24:06
## 2      1     122      5 838985046 Boomerang 1992  Romance  1996-08-
02 11:24:06
## 3      1     185      5 838983525 Net, The  1995  Action   1996-08-
02 10:58:45
## 4      1     185      5 838983525 Net, The  1995  Crime    1996-08-
02 10:58:45
## 5      1     185      5 838983525 Net, The  1995  Thriller 1996-08-
02 10:58:45
## 6      1     292      5 838983421 Outbreak  1995  Action   1996-08-
02 10:57:01
## # ... with 2 more variables: rate_year <chr>, rate_month <chr>
```

Model Building and Evaluation

Methodology

Because dataset is very large, usual data wrangling (for example, the *lm* model) was not possible because of memory allocation. To solve this problem, we computed the least square estimates (RMSE) manually. Exploratory data analysis show that 69878 unique users provided ratings and 10677 unique movies were rated. If that's about 746 million combinations between users and movies, because of that; dataset is very sparse, so we included regularization in the model to add penalty for high ratings that come from small group of users

Naive Model

This is the simplest model, we will predict the rating for all movies to be the mean. in our case mean is 3.51 as shown below. we can use the following formula:

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

Where u is the index for users, and i for movies.

calculating the mean of rating for test set

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

## [1] 3.527019
```

Just average model (Naive model)

```
just_avg = RMSE(validation_seperated_genres$rating, mu)
rmse_results <- tibble(Method = "Just the average", RMSE = just_avg)
kable(rmse_results)
```

Method	RMSE
Just the average	1.052558

Average + movie bias model

if we add movie bias to our model, our formula become:

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

```
b_i <- edx_seperated_genres %>%
  group_by(movieId) %>%
  summarize(b_i = mean(rating - mu))
```

predict all unknown ratings with mu and b_i

```
predicted_ratings_movie <- validation_seperated_genres %>%
  left_join(b_i, by='movieId') %>%
  mutate(pred = mu + b_i) %>%
  pull(pred)
```

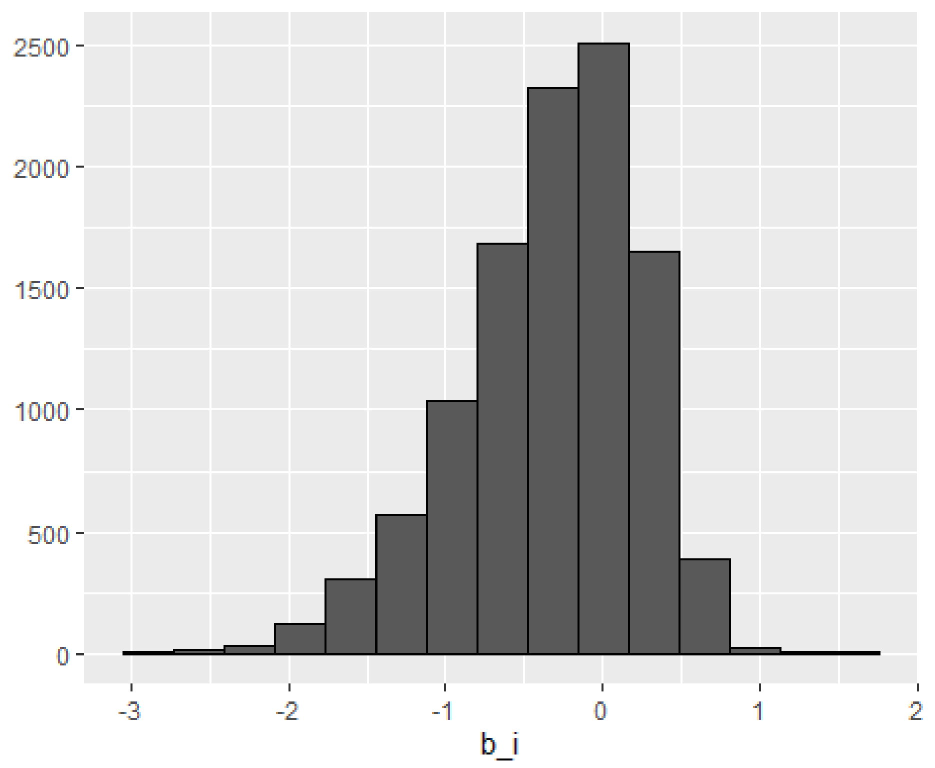
calculate RMSE of movie ranking effect

```
avg_movie = RMSE(validation_seperated_genres$rating,
predicted_ratings_movie)
rmse_results <- bind_rows(rmse_results,tibble(Method = "avg + movie
bias ", RMSE = avg_movie))
kable(rmse_results)
```

Method	RMSE
Just the average	1.052558
avg + movie bias	0.941070

plot the distribution of b_i's

```
qplot(b_i, data = b_i, bins = 15, color = I("black"))
```



Average + Movie and user effect method

In this method we will use the formula:

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

where b_u is user bias.

compute user bias term, b_u

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

predict new ratings with movie and user bias

```r
predicted_ratings_movie_user <- validation_seperated_genres %>%
  left_join(b_i, by='movieId') %>%
  left_join(b_u, by='userId') %>%
  mutate(pred = mu + b_i + b_u) %>%
  pull(pred)
```

calculate RMSE of movie ranking effect

```r
# calculate RMSE of movie ranking effect
avg_movie_user <- RMSE(predicted_ratings_movie_user,
validation_seperated_genres$rating)
```

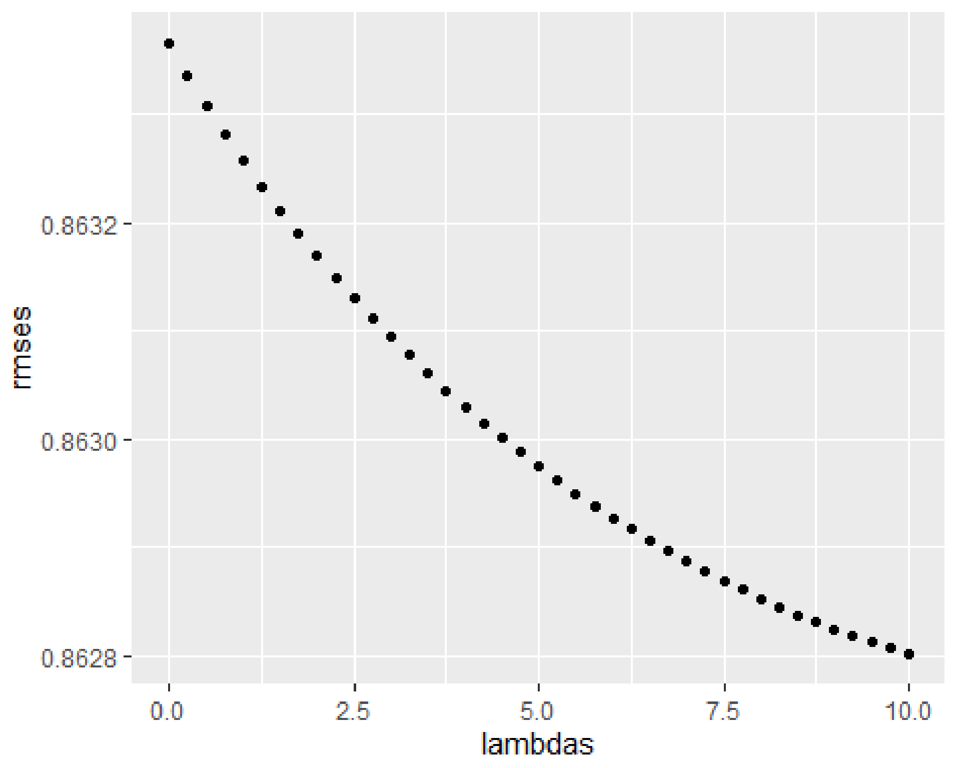
```
rmse_results <- bind_rows(rmse_results,tibble(Method = "avg + movie
bias + user bias", RMSE = avg_movie_user))

print(avg_movie_user)

## [1] 0.863366
```

Regularization

The idea behind regularization method is to add a penalty trem (lambda) to penalizes movies with large or low estimates from a small sample size. as we see in the course, we can select the bias values using a regularization factor as follows:

This regularization is tuned by running over a sequence of λ values and selecting the best RMSE result.

```
# determine best lambda from a sequence
lambdas <- seq(from=0, to=10, by=0.25)

# output RMSE of each lambda, repeat earlier steps (with
regularization)
rmses <- sapply(lambdas, function(l){
  # calculate average rating across training data
  mu <- mean(edx_seperated_genres$rating)
  # compute regularized movie bias term
  b_i <- edx_seperated_genres %>%
    group_by(movieId) %>%
    summarize(b_i = sum(rating - mu)/(n()+l))
  # compute regularize user bias term
  b_u <- edx_seperated_genres %>%
    left_join(b_i, by="movieId") %>%
    group_by(userId) %>%
    summarize(b_u = sum(rating - b_i - mu)/(n()+l))
  # compute predictions on validation set based on these above terms
  predicted_ratings <- validation_seperated_genres %>%
    left_join(b_i, by = "movieId") %>%
    left_join(b_u, by = "userId") %>%
    mutate(pred = mu + b_i + b_u) %>%
    pull(pred)
  # output RMSE of these predictions
  return(RMSE(predicted_ratings, validation_seperated_genres$rating))
})

# quick plot of RMSE vs lambdas
qplot(lambdas, rmses)
```



```
# print minimum RMSE
min(rmses)

## [1] 0.8628015

rmse_results <- bind_rows(rmse_results,tibble(Method = "Regularized avg
+ movie bias + user bias", RMSE = min(rmses)))
#kable(rmse_results)

######################################################
# Final model with regularized movie and user effects
######################################################

# The final linear model with the minimizing lambda
lam <- lambdas[which.min(rmses)]

b_i <- edx_seperated_genres %>%
  group_by(movieId) %>%
  summarize(b_i = sum(rating - mu)/(n()+lam))
# compute regularize user bias term
b_u <- edx_seperated_genres %>%
  left_join(b_i, by="movieId") %>%
  group_by(userId) %>%
  summarize(b_u = sum(rating - b_i - mu)/(n()+lam))
# compute predictions on validation set based on these above terms
predicted_ratings <- validation_seperated_genres %>%
  left_join(b_i, by = "movieId") %>%
  left_join(b_u, by = "userId") %>%
  mutate(pred = mu + b_i + b_u) %>%
  pull(pred)

# output RMSE of these predictions
RMSE(predicted_ratings, validation_seperated_genres$rating)

## [1] 0.8628015
```

Final model with regularized movie and user and genres effects

```
######################################################
# Final model with regularized movie and user and genres effects
######################################################

# The final linear model with the minimizing lambda
lam <- lambdas[which.min(rmses)]

b_i <- edx_seperated_genres %>%
  group_by(movieId) %>%
  summarize(b_i = sum(rating - mu)/(n()+lam))
# compute regularize user bias term
b_u <- edx_seperated_genres %>%
  left_join(b_i, by="movieId") %>%
  group_by(userId) %>%
  summarize(b_u = sum(rating - b_i - mu)/(n()+lam))
# compute regularize genres bias term
b_g <- edx_seperated_genres %>%
    left_join(b_i, by="movieId") %>%
    left_join(b_u, by="userId") %>%
    group_by(genres) %>%
    summarize(b_g = sum(rating - mu - b_i - b_u)/(n()+lam))
# compute predictions on validation set based on these above terms
predicted_ratings <- validation_seperated_genres %>%
  left_join(b_i, by = "movieId") %>%
  left_join(b_u, by = "userId") %>%
  left_join(b_g, by = "genres") %>%
  mutate(pred = mu + b_i + b_u + b_g) %>%
  pull(pred)

reg_movie_user_genre <- RMSE(predicted_ratings,
validation_seperated_genres$rating)

rmse_results <- bind_rows(rmse_results,tibble(Method = "Regularized avg
+ movie bias + user + genre", RMSE = reg_movie_user_genre))
```

Final Results

```
kable(rmse_results)
```

Method	RMSE
Just the average	1.0525579
avg + movie bias	0.9410700
avg + movie bias + user bias	0.8633660
Regularized avg + movie bias + user bias	0.8628015
Regularized avg + movie bias + user + genre	0.8627121