movielensproj

Jennifer Young

7/30/2021

---

title: "MovielensProject" author: "Jennifer Young" date: "7/28/2021" output: html_document

```
local({r <- getOption("repos")
       r["CRAN"] <- "http://my.local.cran"
      options(repos=r)})
```

Movie Recommendation Project

Recommendation systems are used more and more, as consumers expect suggestions based on their known likes so that they can discover new likes in products, movies, music and other interests. They assist users in finding what they might be interested in based on their preferences and previous interactions. In this report, a movie recommendation system using the MovieLens dataset from HarvardX's Data Science Professional Certificate3 program will be covered. GroupLens Research is the organization that collected the data sets for this project from their site: (https://movielens.org).

```
install.packages("scales")
```

```
## Warning: unable to access index for repository http://my.local.cran/src/contrib:
##   cannot open URL 'http://my.local.cran/src/contrib/PACKAGES'

## Warning: package 'scales' is not available (for R version 4.0.2)

## Warning: unable to access index for repository http://my.local.cran/bin/macosx/contrib/4.0:
##   cannot open URL 'http://my.local.cran/bin/macosx/contrib/4.0/PACKAGES'
```

```
install.packages("tidyverse")
```

```
## Warning: unable to access index for repository http://my.local.cran/src/contrib:
##   cannot open URL 'http://my.local.cran/src/contrib/PACKAGES'

## Warning: package 'tidyverse' is not available (for R version 4.0.2)

## Warning: unable to access index for repository http://my.local.cran/bin/macosx/contrib/4.0:
##   cannot open URL 'http://my.local.cran/bin/macosx/contrib/4.0/PACKAGES'
```

```
install.packages("caret")
```

```
## Warning: unable to access index for repository http://my.local.cran/src/contrib:
##   cannot open URL 'http://my.local.cran/src/contrib/PACKAGES'

## Warning: package 'caret' is not available (for R version 4.0.2)

## Warning: unable to access index for repository http://my.local.cran/bin/macosx/contrib/4.0:
##   cannot open URL 'http://my.local.cran/bin/macosx/contrib/4.0/PACKAGES'
```

```
install.packages("data.table")
```

```
## Warning: unable to access index for repository http://my.local.cran/src/contrib:
##   cannot open URL 'http://my.local.cran/src/contrib/PACKAGES'
## Warning: package 'data.table' is not available (for R version 4.0.2)
## Warning: unable to access index for repository http://my.local.cran/bin/macosx/contrib/4.0:
##   cannot open URL 'http://my.local.cran/bin/macosx/contrib/4.0/PACKAGES'
```

```r
install.packages("lubridate")
```

```
## Warning: unable to access index for repository http://my.local.cran/src/contrib:
##   cannot open URL 'http://my.local.cran/src/contrib/PACKAGES'
## Warning: package 'lubridate' is not available (for R version 4.0.2)
## Warning: unable to access index for repository http://my.local.cran/bin/macosx/contrib/4.0:
##   cannot open URL 'http://my.local.cran/bin/macosx/contrib/4.0/PACKAGES'
```

```r
install.packages("recosystem")
```

```
## Warning: unable to access index for repository http://my.local.cran/src/contrib:
##   cannot open URL 'http://my.local.cran/src/contrib/PACKAGES'
## Warning: package 'recosystem' is not available (for R version 4.0.2)
## Warning: unable to access index for repository http://my.local.cran/bin/macosx/contrib/4.0:
##   cannot open URL 'http://my.local.cran/bin/macosx/contrib/4.0/PACKAGES'
```

```r
install.packages("kableExtra")
```

```
## Warning: unable to access index for repository http://my.local.cran/src/contrib:
##   cannot open URL 'http://my.local.cran/src/contrib/PACKAGES'
## Warning: package 'kableExtra' is not available (for R version 4.0.2)
## Warning: unable to access index for repository http://my.local.cran/bin/macosx/contrib/4.0:
##   cannot open URL 'http://my.local.cran/bin/macosx/contrib/4.0/PACKAGES'
```

```r
install.packages("devtools")
```

```
## Warning: unable to access index for repository http://my.local.cran/src/contrib:
##   cannot open URL 'http://my.local.cran/src/contrib/PACKAGES'
## Warning: package 'devtools' is not available (for R version 4.0.2)
## Warning: unable to access index for repository http://my.local.cran/bin/macosx/contrib/4.0:
##   cannot open URL 'http://my.local.cran/bin/macosx/contrib/4.0/PACKAGES'
```

```r
install.packages("Rcpp")
```

```
## Warning: unable to access index for repository http://my.local.cran/src/contrib:
##   cannot open URL 'http://my.local.cran/src/contrib/PACKAGES'
## Warning: package 'Rcpp' is not available (for R version 4.0.2)
## Warning: unable to access index for repository http://my.local.cran/bin/macosx/contrib/4.0:
##   cannot open URL 'http://my.local.cran/bin/macosx/contrib/4.0/PACKAGES'
```

```r
install.packages("tinytex")
```

```
## Warning: unable to access index for repository http://my.local.cran/src/contrib:
##   cannot open URL 'http://my.local.cran/src/contrib/PACKAGES'
## Warning: package 'tinytex' is not available (for R version 4.0.2)
```

```
## Warning: unable to access index for repository http://my.local.cran/bin/macosx/contrib/4.0:
##   cannot open URL 'http://my.local.cran/bin/macosx/contrib/4.0/PACKAGES'
```

```r
update.packages()
```

```
## Warning: unable to access index for repository http://my.local.cran/src/contrib:
##   cannot open URL 'http://my.local.cran/src/contrib/PACKAGES'
```

```r
library(scales)
library(tidyverse)
```

```
## -- 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 readr::col_factor() masks scales::col_factor()
## x purrr::discard()    masks scales::discard()
## x dplyr::filter()     masks stats::filter()
## x dplyr::lag()        masks stats::lag()
```

```r
library(dplyr)
library(caret)
```

```
## Loading required package: lattice

##
## Attaching package: 'caret'

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

```r
library(data.table)
```

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

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

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

```r
library(lubridate)
```

```
##
## Attaching package: 'lubridate'

## The following objects are masked from 'package:data.table':
##
##     hour, isoweek, mday, minute, month, quarter, second, wday, week,
##     yday, year

## The following objects are masked from 'package:base':
##
##     date, intersect, setdiff, union
```

```r
library(stringr)
library(recosystem)
library(kableExtra)
```

```
##
## Attaching package: 'kableExtra'

## The following object is masked from 'package:dplyr':
##
##     group_rows
```

```r
library(tinytex)
library(ggplot2)
```

```r
library(Rcpp)

dl <- tempfile()
download.file("http://files.grouplens.org/datasets/movielens/ml-10m.zip", dl)
ratings <- read.table(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.integer(movieId),
                                           title = as.character(title),
                                           genres = as.character(genres))
movielens <- left_join(ratings, movies, by = "movieId")
```

Methods and Analysis

There are five steps in the data analysis process that must be completed. In this case, the data must be prepared. The dataset from was downloaded from the MovieLens website and split into two subsets used for training and validation. In this case, we named the training set "edx" and the validation set "validation". For training and testing, the edx set was split again into two subsets. The edx set is trained with the model when it reaches the RMSE goal and the validation set is used for final validation. During data exploration and visualization, charts are crated to understand the data and how it affects the outcome. We observe the mean of observed values, the distribution of ratings,# mean movie ratings, movie effect, user effect and number of ratings per movie. We improve the RMSE by including the user and movie effects and applying the regularization parameter for samples that have few ratings.

The Validation subset will be 10% of the MovieLens data.

```r
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 subset:

```r
validation <- temp %>%
  semi_join(edx, by = "movieId") %>%
  semi_join(edx, by = "userId")
```

Add rows removed from validation set back into edx set

```r
removed <- anti_join(temp, validation)
```

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

```
edx <- rbind(edx, removed)
rm(dl, ratings, movies, test_index, temp, movielens, removed)
```

Lists six variables “userID”, “movieID”, “rating”, “timestamp”, “title”, and “genres” in data frame

```
head(edx) %>%
  print.data.frame()
```

```
##   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
```

```
dim(edx)
```

```
## [1] 9000061       6
```

```
n_distinct(edx$movieId)
```

```
## [1] 10677
```

```
n_distinct(edx$title)
```

```
## [1] 10676
```

```
n_distinct(edx$userId)
```

```
## [1] 69878
```

```
n_distinct(edx$movieId)*n_distinct(edx$userId)
```

```
## [1] 746087406
```

```
n_distinct(edx$movieId)*n_distinct(edx$userId)/dim(edx)[1]
```

```
## [1] 82.89804
```

Looking for missing values

```
summary(edx)
```

```
##      userId         movieId          rating        timestamp
##  Min.   :    1   Min.   :    1   Min.   :0.500   Min.   :7.897e+08
##  1st Qu.:18122   1st Qu.:  648   1st Qu.:3.000   1st Qu.:9.468e+08
##  Median :35743   Median : 1834   Median :4.000   Median :1.035e+09
##  Mean   :35869   Mean   : 4120   Mean   :3.512   Mean   :1.033e+09
##  3rd Qu.:53602   3rd Qu.: 3624   3rd Qu.:4.000   3rd Qu.:1.127e+09
##  Max.   :71567   Max.   :65133   Max.   :5.000   Max.   :1.231e+09
##     title              genres
##  Length:9000061     Length:9000061
```

```
##   Class :character   Class :character
##   Mode  :character   Mode  :character
##
##
##
```

Unique movies and users in the edx subset

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

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

Extracting age of movies at rating

Every movie was released in a certain year, which is provided in the title of the movie. Every user rated a movie in a certain year, which is included in the timestamp information. I define the difference between these two years, i.e., how old the movie was when it was watched/rated by a user, as the age of movies at rating. From the original dataset, I first pulled the rating year (year_rated) from timestamp, and then exacted the release year (year_released) of the movie from the title. age_at_rating was later determined.

Change timestamp to year

```
edx_1 <- edx %>% mutate(year_rated = year(as_datetime(timestamp)))
```

Extract the release year of the movie

```
edx_1 <- edx_1 %>% mutate(title = str_replace(title,"^(.+)\\s\\((\\d{4})\\)$","\\1__\\2" )) %>%
  separate(title,c("title","year_released"),"__") %>%
  select(-timestamp)
edx_1 <- edx_1 %>% mutate(age_at_rating= as.numeric(year_rated)-as.numeric(year_released))
head(edx_1)
```

```
##   userId movieId rating                  title year_released
## 1      1     122      5              Boomerang          1992
## 2      1     185      5               Net, The          1995
## 3      1     231      5           Dumb & Dumber          1994
## 4      1     292      5               Outbreak          1995
## 5      1     316      5               Stargate          1994
## 6      1     329      5 Star Trek: Generations          1994
##                          genres year_rated age_at_rating
## 1                Comedy|Romance       1996             4
## 2         Action|Crime|Thriller       1996             1
## 3                        Comedy       1996             2
## 4  Action|Drama|Sci-Fi|Thriller       1996             1
## 5       Action|Adventure|Sci-Fi       1996             2
## 6 Action|Adventure|Drama|Sci-Fi       1996             2
```

Extracting the genres information

The genres information was provided in the original dataset as a combination of different classifications. We will need to split it into single ones.

The mixture of genres is split into different rows

```
edx_2 <- edx_1 %>% separate_rows(genres,sep = "\\|") %>% mutate(value=1)
n_distinct(edx_2$genres)
```

```
## [1] 20
```

```r
genres_rating <- edx_2 %>% group_by(genres) %>% summarize(n=n())
genres_rating
```

```
## # A tibble: 20 x 2
##    genres                   n
##    <chr>                <int>
##  1 (no genres listed)       6
##  2 Action             2560649
##  3 Adventure          1908692
##  4 Animation           467220
##  5 Children            737851
##  6 Comedy             3541284
##  7 Crime              1326917
##  8 Documentary          93252
##  9 Drama              3909401
## 10 Fantasy             925624
## 11 Film-Noir           118394
## 12 Horror              691407
## 13 IMAX                  8190
## 14 Musical             432960
## 15 Mystery             567865
## 16 Romance            1712232
## 17 Sci-Fi             1341750
## 18 Thriller           2325349
## 19 War                 511330
## 20 Western             189234
```

```r
edx_3 <- edx_2 %>% spread(genres, value, fill=0) %>% select(-"(no genres listed)")
dim(edx_3)
```

```
## [1] 9000061      26
```

```r
head(edx_3)
```

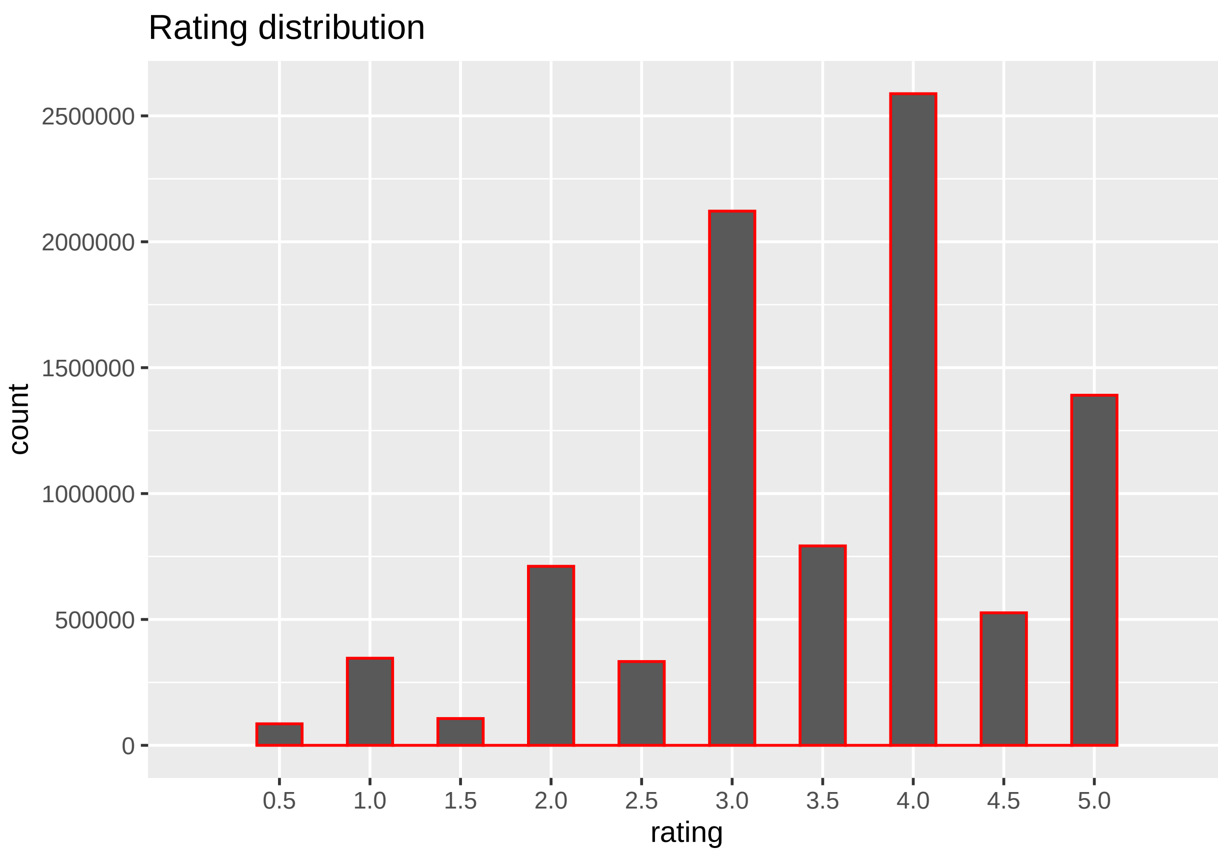
```
## # A tibble: 6 x 26
##   userId movieId rating title      year_released year_rated age_at_rating Action
##    <int>   <int>  <dbl> <chr>      <chr>              <dbl>         <dbl>  <dbl>
## 1      1     122      5 Boomerang  1992                1996             4      0
## 2      1     185      5 Net, The   1995                1996             1      1
## 3      1     231      5 Dumb & Du~ 1994                1996             2      0
## 4      1     292      5 Outbreak   1995                1996             1      1
## 5      1     316      5 Stargate   1994                1996             2      1
## 6      1     329      5 Star Trek~ 1994                1996             2      1
## # ... with 18 more variables: Adventure <dbl>, Animation <dbl>, Children <dbl>,
## #   Comedy <dbl>, Crime <dbl>, Documentary <dbl>, Drama <dbl>, Fantasy <dbl>,
## #   Film-Noir <dbl>, Horror <dbl>, IMAX <dbl>, Musical <dbl>, Mystery <dbl>,
## #   Romance <dbl>, Sci-Fi <dbl>, Thriller <dbl>, War <dbl>, Western <dbl>
```

The dataset actually duplicated each record into multiple ones, depending on the combination of the genres for each movie. We need to split into multiple columns to indicate different combinations of the 19 basic genres by spreading genres to the "wide" format.

Distribution of ratings (histogram)

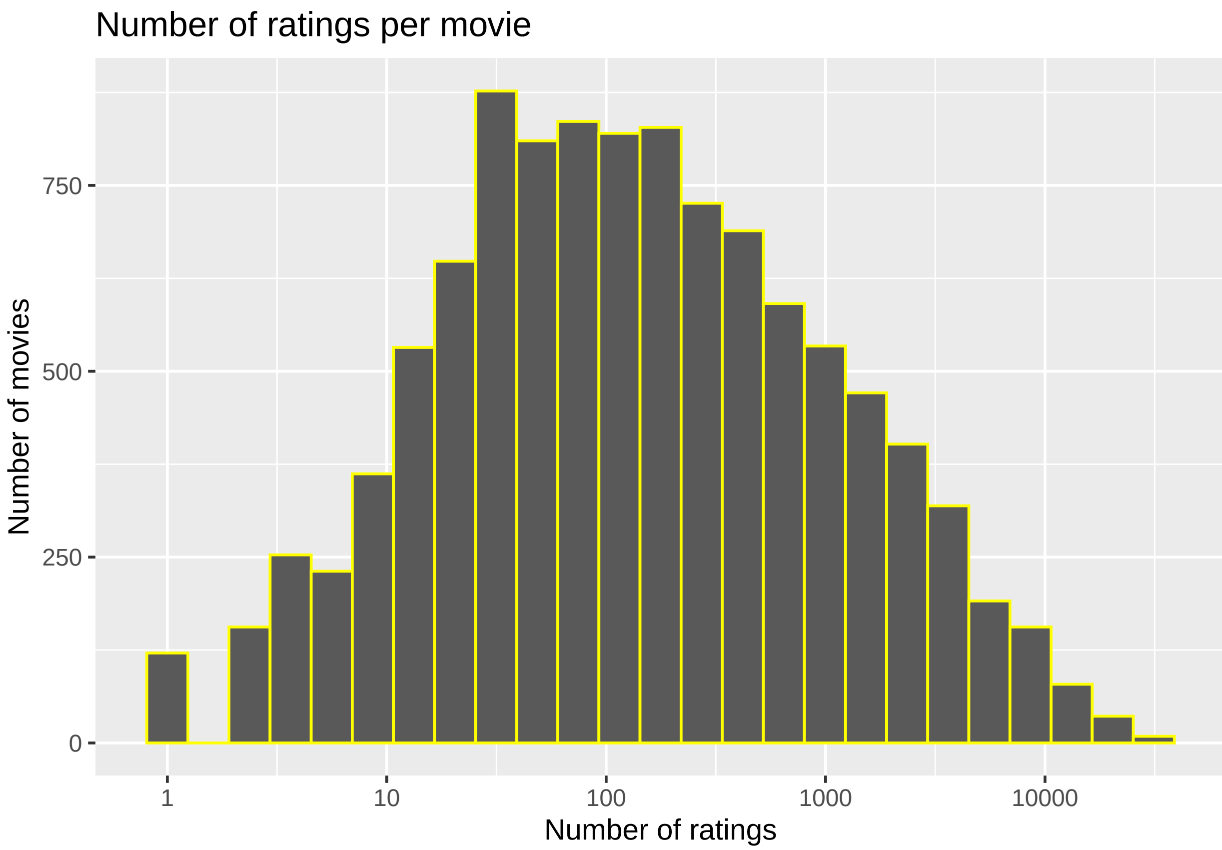
```r
edx %>%
  ggplot(aes(rating)) +
  geom_histogram(binwidth = 0.25, color = "red") +
  scale_x_discrete(limits= c(seq(0.5,5,0.5))) +
  scale_y_continuous(breaks = c(seq(0, 3000000, 500000))) +
  ggtitle("Rating distribution")
```

```
## Warning: Continuous limits supplied to discrete scale.
## Did you mean `limits = factor(...)` or `scale_*_continuous()`?
```



Ratings per movie (Histogram)

```r
edx %>%
  count(movieId) %>%
  ggplot(aes(n)) +
  geom_histogram(bins = 25, color = "yellow") +
  scale_x_log10() +
  xlab("Number of ratings") +
  ylab("Number of movies") +
  ggtitle("Number of ratings per movie")
```



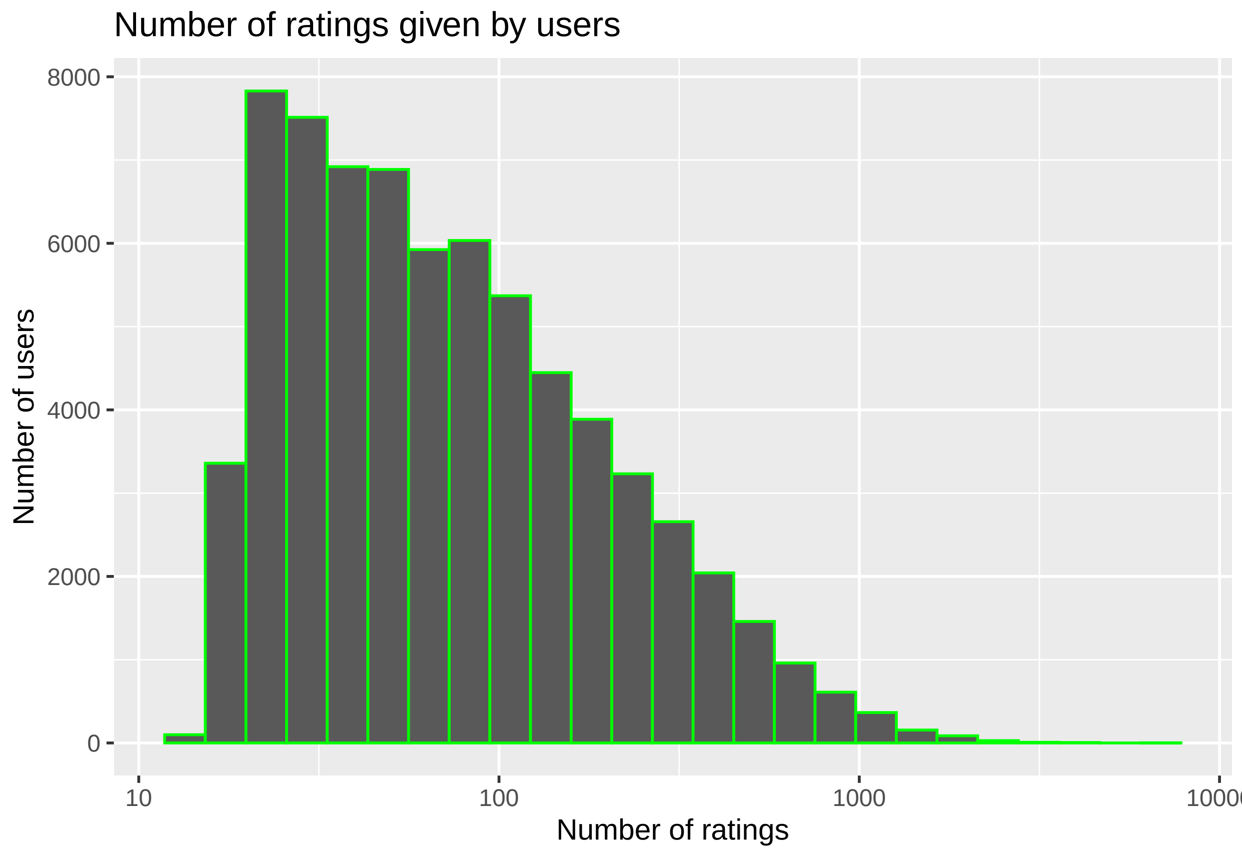
Movies that were rated only once (chart)

```
edx %>%
  group_by(movieId) %>%
  summarize(count = n()) %>%
  filter(count == 1) %>%
  left_join(edx, by = "movieId") %>%
  group_by(title) %>%
  summarize(rating = rating, n_rating = count) %>%
  slice(1:20) %>%
  knitr::kable()
```

title	rating	n_rating
100 Feet (2008)	2.0	1
4 (2005)	2.5	1
5 Centimeters per Second (Byôsoku 5 senchimêtoru) (2007)	3.5	1
Accused (Anklaget) (2005)	0.5	1
Ace of Hearts (2008)	2.0	1
Ace of Hearts, The (1921)	3.5	1
Adios, Sabata (Indio Black, sai che ti dico: Sei un gran figlio di...) (1971)	1.5	1
Africa addio (1966)	3.0	1
Archangel (1990)	2.5	1
Bad Blood (Mauvais sang) (1986)	4.5	1
Battle of Russia, The (Why We Fight, 5) (1943)	3.5	1
Bell Boy, The (1918)	4.0	1
Black Tights (1-2-3-4 ou Les Collants noirs) (1960)	3.0	1
Blind Shaft (Mang jing) (2003)	2.5	1
Blue Light, The (Das Blaue Licht) (1932)	5.0	1
Borderline (1950)	3.0	1
Boys Life 4: Four Play (2003)	3.0	1
Brothers of the Head (2005)	2.5	1
Caótica Ana (2007)	4.5	1
Chapayev (1934)	1.5	1

User ratings (Histogram)

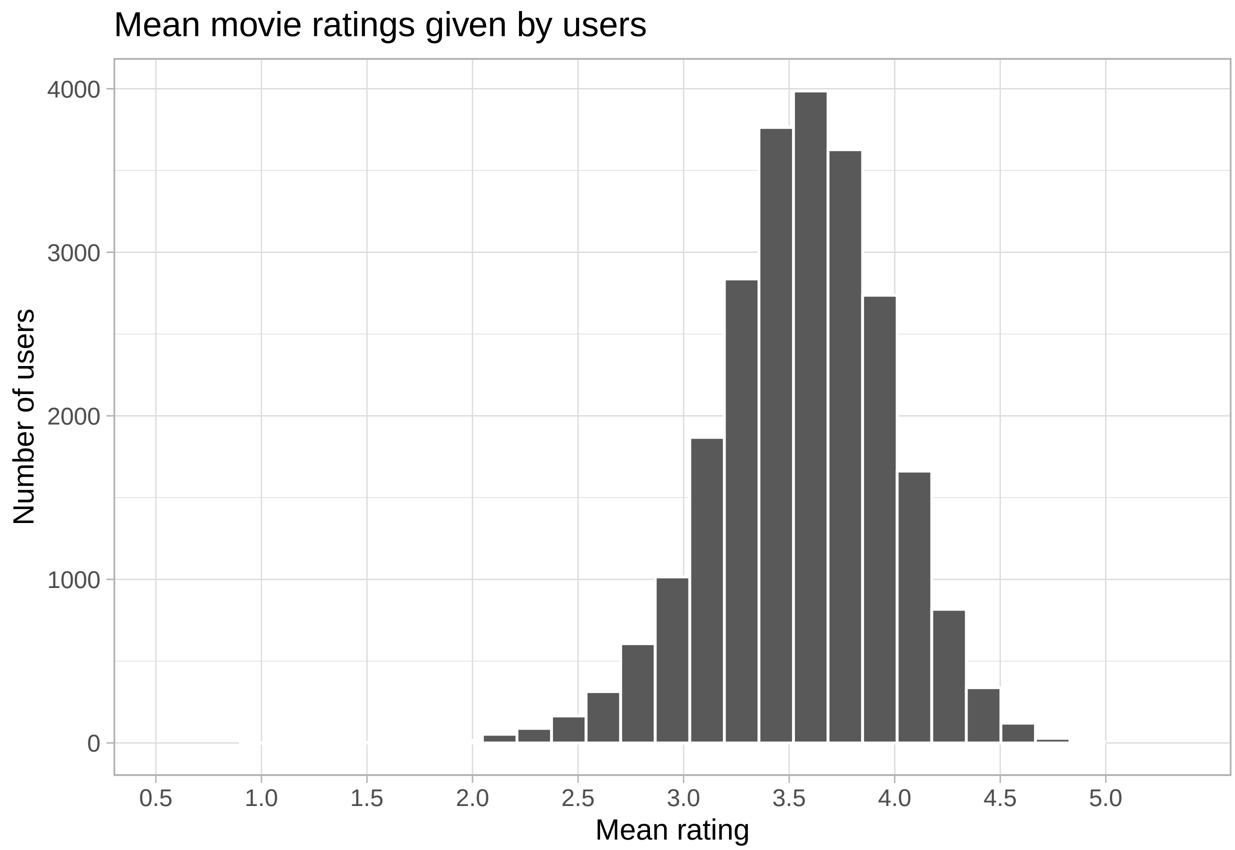
```
edx %>%
  count(userId) %>%
  ggplot(aes(n)) +
  geom_histogram(bins = 25, color = "green") +
  scale_x_log10() +
  xlab("Number of ratings") +
  ylab("Number of users") +
  ggtitle("Number of ratings given by users")
```



Mean user ratings

```
edx %>%
  group_by(userId) %>%
  filter(n() >= 100) %>%
  summarize(b_u = mean(rating)) %>%
  ggplot(aes(b_u)) +
  geom_histogram(bins = 25, color = "white") +
  xlab("Mean rating") +
  ylab("Number of users") +
  ggtitle("Mean movie ratings given by users") +
  scale_x_discrete(limits = c(seq(0.5,5,0.5))) +
  theme_light()
```

```
## Warning: Continuous limits supplied to discrete scale.
## Did you mean `limits = factor(...)` or `scale_*_continuous()`?
```



Compute the RMSE

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

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

```
## [1] 3.512464
```

```
naive_rmse <- RMSE(validation$rating, mu)
naive_rmse
```

```
## [1] 1.060651
```

```
rmse_results <- data_frame(Model = "Basic Average", RMSE = naive_rmse)
```

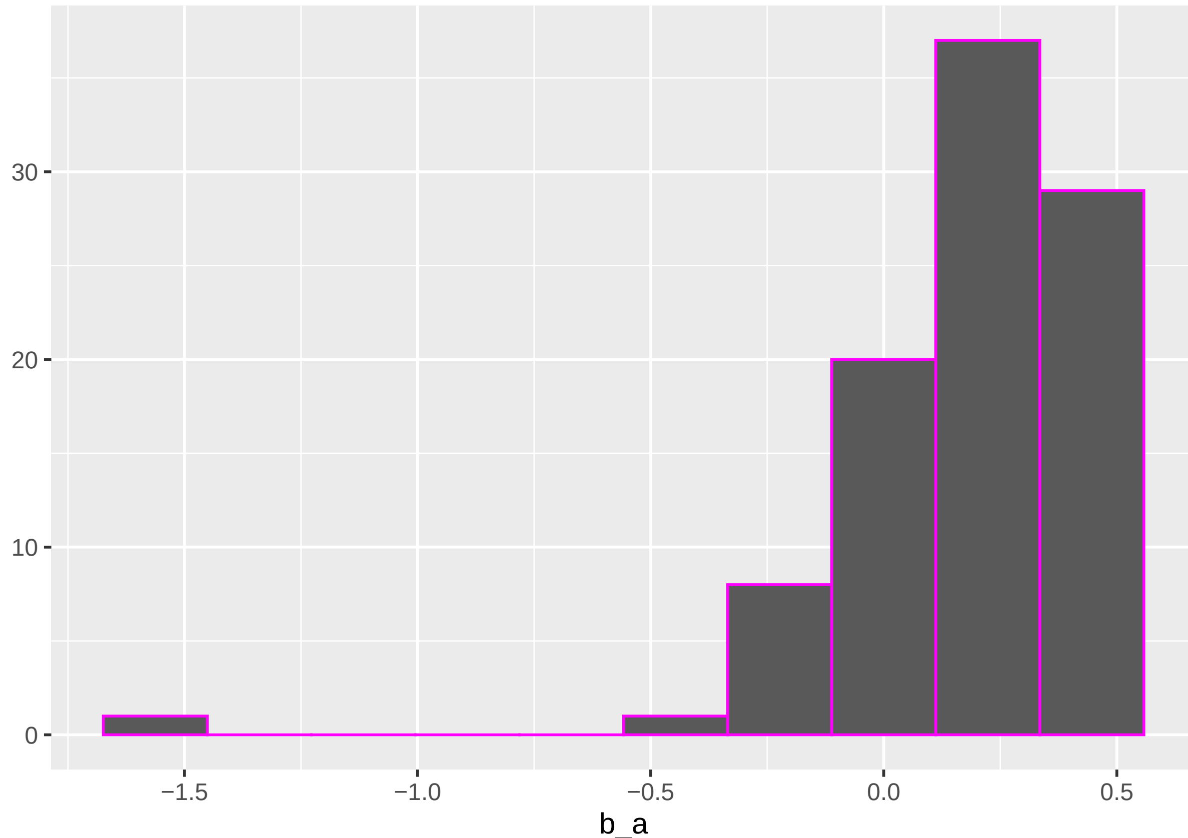
```
## Warning: `data_frame()` was deprecated in tibble 1.1.0.
## Please use `tibble()` instead.
```

```
rmse_results
```

```
## # A tibble: 1 x 2
##   Model          RMSE
##   <chr>         <dbl>
## 1 Basic Average  1.06
```

Age bias distribution

```r
age_effect<- edx_1 %>%
  group_by(age_at_rating) %>%
  summarize(b_a = mean(rating)-mu)
age_effect %>% qplot(b_a, geom ="histogram", bins = 10, data = ., color = I("magenta"))
```



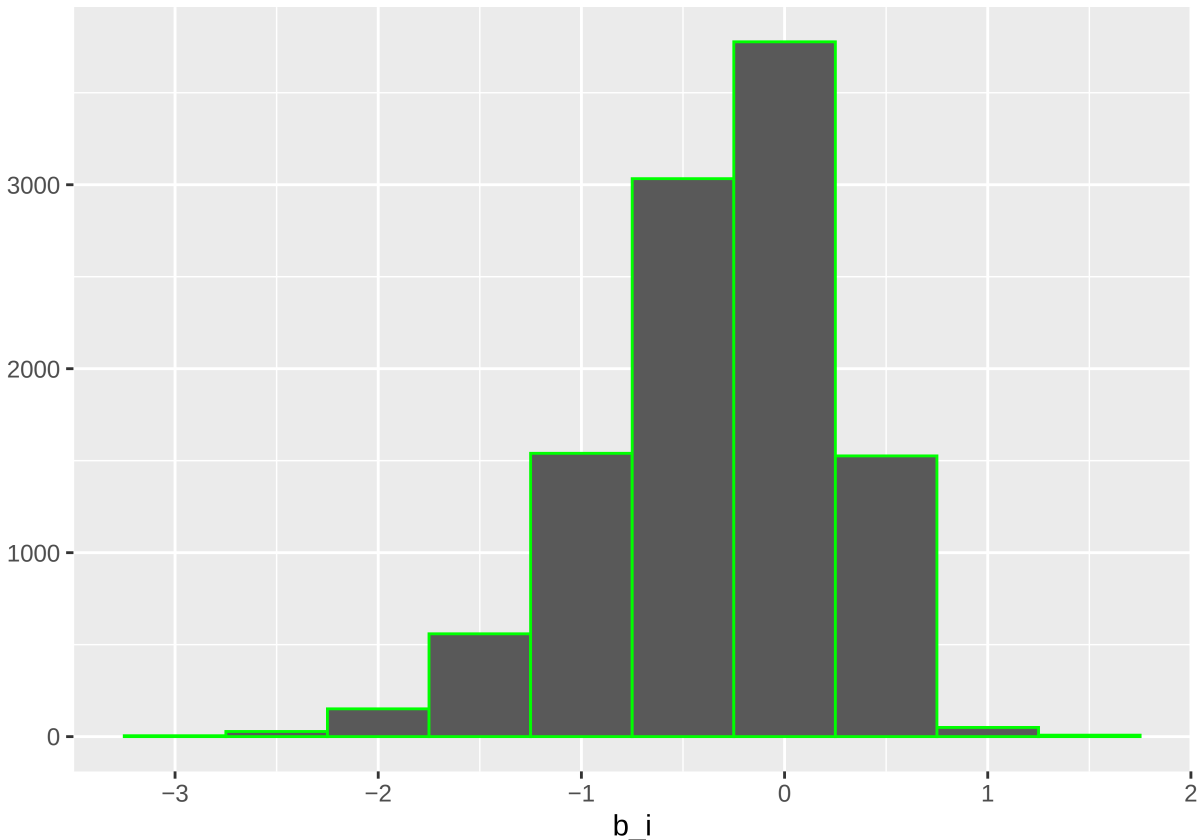

```r
validation_1 <- validation %>%
  mutate(year_rated = year(as_datetime(timestamp)))%>%
  mutate(title = str_replace(title,"^(.+)\\s\\((\\d{4})\\)$","\\1__\\2" )) %>%
  separate(title,c("title","year_released"),"__") %>%
  select(-timestamp) %>%
  mutate(age_at_rating= as.numeric(year_rated)-as.numeric(year_released))

predicted_ratings_2 <- mu + validation_1 %>%
  left_join(age_effect, by='age_at_rating') %>%
  pull(b_a)
model_2_rmse <- RMSE(validation$rating,predicted_ratings_2) # 1.05239
rmse_results <- bind_rows(rmse_results,
                          data_frame(Model="Age Effect Model",
                                     RMSE = model_2_rmse))
rmse_results
```

```
## # A tibble: 2 x 2
##   Model               RMSE
##   <chr>              <dbl>
## 1 Basic Average      1.06
## 2 Age Effect Model   1.05
```

Age Effect Model did not improve the RMSE much so it will not be used as a predictor.

Now we are adding movie effects to the model

```
movie_avgs <- edx %>%
  group_by(movieId) %>%
  summarize(b_i = mean(rating - mu))
movie_avgs %>% qplot(b_i, geom ="histogram", bins = 10, data = ., color = I("green"))
```




```
predicted_ratings_3 <- mu + validation %>%
  left_join(movie_avgs, by='movieId') %>%
  pull(b_i)
model_3_rmse <- RMSE(validation$rating,predicted_ratings_3)
rmse_results <- bind_rows(rmse_results,
                          data_frame(Model="Movie Effect Model",
                                     RMSE = model_3_rmse))
rmse_results
```

```
## # A tibble: 3 x 2
##   Model                RMSE
##   <chr>               <dbl>
## 1 Basic Average       1.06
## 2 Age Effect Model    1.05
## 3 Movie Effect Model 0.944
```

The Movie Effect Model brought the RMSE under 1

Now we add the bias of the user.

```
user_avgs <- edx %>%
  left_join(movie_avgs, by='movieId') %>%
```

```
  group_by(userId) %>%
  summarize(b_u = mean(rating - mu - b_i))
predicted_ratings_4 <- validation %>%
  left_join(movie_avgs, by='movieId') %>%
  left_join(user_avgs, by='userId') %>%
  mutate(pred = mu + b_i + b_u) %>%
  pull(pred)
model_4_rmse <- RMSE(validation$rating,predicted_ratings_4)
rmse_results <- bind_rows(rmse_results,
                          data_frame(Model="User Effects Model+Movie Effect Model",
                                     RMSE = model_4_rmse))
rmse_results
```

```
## # A tibble: 4 x 2
##   Model                                  RMSE
##   <chr>                                 <dbl>
## 1 Basic Average                         1.06
## 2 Age Effect Model                      1.05
## 3 Movie Effect Model                    0.944
## 4 User Effects Model+Movie Effect Model 0.866
```

Adding user bias further improves the RMSE. However, Regularization technique should be used to take into account the number of ratings made for a specific movie, by adding a larger penalty to estimates from smaller samples. Lambda will be used to do this. Cross validation within the test set can be performed to optimize this parameter before being applied to the validation set. In this case, we are doing this for movie effects only.

We will use 10-fold cross validation to pick a lambda for movie effects regularization by splitting the data into 10 parts.

```
set.seed(2019)
cv_splits <- createFolds(edx$rating, k=10, returnTrain =TRUE)
```

Define a matrix to store the results of cross validation

```
rmses <- matrix(nrow=10,ncol=51)
lambdas <- seq(0, 5, 0.1)
```

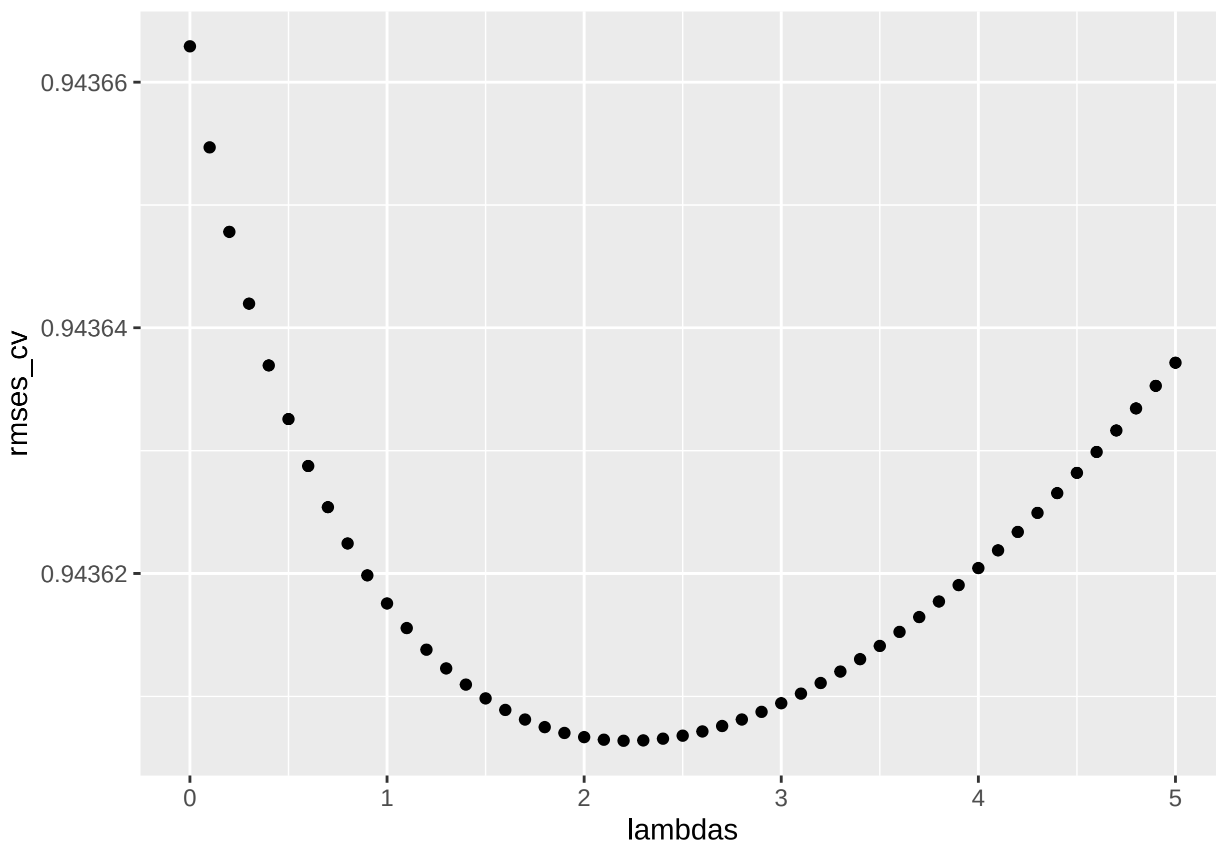
Perform 10-fold cross validation to determine the optimal lambda

```
for(k in 1:10) {
  train_set <- edx[cv_splits[[k]],]
  test_set <- edx[-cv_splits[[k]],]

# Make sure userId and movieId in test set are also in the train set

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

# Add rows removed from validation set back into edx set

removed <- anti_join(test_set, test_final)
  train_final <- rbind(train_set, removed)

 mu <- mean(train_final$rating)
  just_the_sum <- train_final %>%
    group_by(movieId) %>%
```

```r
    summarize(s = sum(rating - mu), n_i = n())

 rmses[k,] <- sapply(lambdas, function(l){
    predicted_ratings <- test_final %>%
      left_join(just_the_sum, by='movieId') %>%
      mutate(b_i = s/(n_i+l)) %>%
      mutate(pred = mu + b_i) %>%
      pull(pred)
    return(RMSE(predicted_ratings, test_final$rating))
  })
}
```

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

```r
rmses_cv <- colMeans(rmses)
qplot(lambdas,rmses_cv)
```




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

```r
mu <- mean(edx$rating)
movie_reg_avgs <- edx %>%
  group_by(movieId) %>%
  summarize(b_i = sum(rating - mu)/(n()+lambda), n_i = n())
predicted_ratings_5 <- validation %>%
  left_join(movie_reg_avgs, by = "movieId") %>%
  mutate(pred = mu + b_i) %>%
  pull(pred)
model_5_rmse <- RMSE(predicted_ratings_5, validation$rating)   # 0.943852 not too much improved
rmse_results <- bind_rows(rmse_results,
                          data_frame(Model="Regularized Movie Effect Model",
                                     RMSE = model_5_rmse))
rmse_results
```

```
## # A tibble: 5 x 2
##   Model                                  RMSE
##   <chr>                                 <dbl>
## 1 Basic Average                         1.06
## 2 Age Effect Model                      1.05
## 3 Movie Effect Model                    0.944
## 4 User Effects Model+Movie Effect Model 0.866
## 5 Regularized Movie Effect Model        0.944
```

This model did not improve the RMSE. This time, we will use the same lambdas for both movie and user effects.

Define a matrix to store the results of cross validation

```r
lambdas <- seq(0, 8, 0.1)
rmses_2 <- matrix(nrow=10,ncol=length(lambdas))
```

Perform 10-fold cross validation to determine the optimal lambda

```r
for(k in 1:10) {
  train_set <- edx[cv_splits[[k]],]
  test_set <- edx[-cv_splits[[k]],]


# Make sure userId and movieId in test set are also in the train set

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

# Add rows removed from validation set back into edx set

removed <- anti_join(test_set, test_final)
  train_final <- rbind(train_set, removed)

  mu <- mean(train_final$rating)

  rmses_2[k,] <- sapply(lambdas, function(l){
    b_i <- train_final %>%
      group_by(movieId) %>%
      summarize(b_i = sum(rating - mu)/(n()+l))
    b_u <- train_final %>%
```

```
      left_join(b_i, by="movieId") %>%
      group_by(userId) %>%
      summarize(b_u = sum(rating - b_i - mu)/(n()+l))
    predicted_ratings <-
      test_final %>%
      left_join(b_i, by = "movieId") %>%
      left_join(b_u, by = "userId") %>%
      mutate(pred = mu + b_i + b_u) %>%
      pull(pred)
    return(RMSE(predicted_ratings, test_final$rating))
  })
}
```

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

```
rmses_2
```

```
##             [,1]      [,2]      [,3]      [,4]      [,5]      [,6]      [,7]
##  [1,] 0.8647088 0.8646742 0.8646420 0.8646118 0.8645833 0.8645564 0.8645309
##  [2,] 0.8651128 0.8650784 0.8650467 0.8650171 0.8649894 0.8649633 0.8649386
##  [3,] 0.8669753 0.8669396 0.8669068 0.8668763 0.8668478 0.8668210 0.8667957
##  [4,] 0.8638918 0.8638631 0.8638360 0.8638104 0.8637861 0.8637629 0.8637409
##  [5,] 0.8656241 0.8655923 0.8655627 0.8655349 0.8655086 0.8654838 0.8654601
##  [6,] 0.8651945 0.8651618 0.8651313 0.8651026 0.8650755 0.8650497 0.8650252
##  [7,] 0.8658194 0.8657916 0.8657655 0.8657408 0.8657175 0.8656953 0.8656742
##  [8,] 0.8655540 0.8655181 0.8654846 0.8654532 0.8654234 0.8653952 0.8653683
##  [9,] 0.8662122 0.8661788 0.8661478 0.8661187 0.8660912 0.8660652 0.8660404
## [10,] 0.8665222 0.8664908 0.8664612 0.8664331 0.8664065 0.8663811 0.8663568
##             [,8]      [,9]     [,10]     [,11]     [,12]     [,13]     [,14]
##  [1,] 0.8645066 0.8644835 0.8644615 0.8644404 0.8644203 0.8644011 0.8643827
##  [2,] 0.8649151 0.8648928 0.8648715 0.8648512 0.8648319 0.8648134 0.8647957
##  [3,] 0.8667717 0.8667488 0.8667271 0.8667063 0.8666865 0.8666676 0.8666495
##  [4,] 0.8637198 0.8636997 0.8636805 0.8636621 0.8636446 0.8636278 0.8636118
##  [5,] 0.8654376 0.8654161 0.8653956 0.8653759 0.8653572 0.8653392 0.8653220
##  [6,] 0.8650018 0.8649794 0.8649580 0.8649376 0.8649180 0.8648992 0.8648812
##  [7,] 0.8656541 0.8656349 0.8656165 0.8655990 0.8655822 0.8655662 0.8655509
##  [8,] 0.8653427 0.8653183 0.8652949 0.8652726 0.8652512 0.8652307 0.8652111
##  [9,] 0.8660168 0.8659942 0.8659727 0.8659520 0.8659323 0.8659134 0.8658953
## [10,] 0.8663336 0.8663114 0.8662901 0.8662698 0.8662502 0.8662315 0.8662136
##            [,15]     [,16]     [,17]     [,18]     [,19]     [,20]     [,21]
##  [1,] 0.8643651 0.8643483 0.8643322 0.8643169 0.8643022 0.8642882 0.8642749
##  [2,] 0.8647788 0.8647627 0.8647473 0.8647326 0.8647186 0.8647052 0.8646925
##  [3,] 0.8666322 0.8666157 0.8665999 0.8665849 0.8665705 0.8665567 0.8665436
##  [4,] 0.8635965 0.8635819 0.8635680 0.8635548 0.8635421 0.8635301 0.8635187
##  [5,] 0.8653056 0.8652899 0.8652749 0.8652605 0.8652469 0.8652338 0.8652213
##  [6,] 0.8648640 0.8648475 0.8648318 0.8648167 0.8648022 0.8647884 0.8647753
```

```
##  [7,] 0.8655363 0.8655224 0.8655091 0.8654964 0.8654844 0.8654729 0.8654620
##  [8,] 0.8651924 0.8651744 0.8651571 0.8651406 0.8651249 0.8651098 0.8650953
##  [9,] 0.8658780 0.8658614 0.8658455 0.8658303 0.8658158 0.8658019 0.8657886
## [10,] 0.8661964 0.8661799 0.8661641 0.8661490 0.8661346 0.8661208 0.8661076
##           [,22]     [,23]     [,24]     [,25]     [,26]     [,27]     [,28]
##  [1,] 0.8642622 0.8642500 0.8642385 0.8642275 0.8642171 0.8642072 0.8641978
##  [2,] 0.8646804 0.8646688 0.8646578 0.8646474 0.8646376 0.8646282 0.8646194
##  [3,] 0.8665310 0.8665191 0.8665077 0.8664969 0.8664867 0.8664769 0.8664677
##  [4,] 0.8635079 0.8634976 0.8634879 0.8634787 0.8634700 0.8634618 0.8634542
##  [5,] 0.8652095 0.8651982 0.8651875 0.8651773 0.8651676 0.8651585 0.8651499
##  [6,] 0.8647627 0.8647507 0.8647392 0.8647284 0.8647180 0.8647082 0.8646989
##  [7,] 0.8654516 0.8654418 0.8654325 0.8654238 0.8654155 0.8654077 0.8654004
##  [8,] 0.8650815 0.8650683 0.8650558 0.8650438 0.8650324 0.8650215 0.8650112
##  [9,] 0.8657760 0.8657639 0.8657524 0.8657414 0.8657310 0.8657211 0.8657118
## [10,] 0.8660950 0.8660829 0.8660715 0.8660606 0.8660502 0.8660403 0.8660310
##           [,29]     [,30]     [,31]     [,32]     [,33]     [,34]     [,35]
##  [1,] 0.8641890 0.8641806 0.8641727 0.8641653 0.8641583 0.8641518 0.8641458
##  [2,] 0.8646110 0.8646032 0.8645958 0.8645889 0.8645825 0.8645765 0.8645709
##  [3,] 0.8664590 0.8664507 0.8664429 0.8664356 0.8664288 0.8664223 0.8664163
##  [4,] 0.8634470 0.8634403 0.8634340 0.8634282 0.8634228 0.8634179 0.8634134
##  [5,] 0.8651418 0.8651341 0.8651269 0.8651202 0.8651139 0.8651081 0.8651027
##  [6,] 0.8646900 0.8646817 0.8646738 0.8646664 0.8646594 0.8646529 0.8646468
##  [7,] 0.8653935 0.8653871 0.8653811 0.8653756 0.8653705 0.8653658 0.8653615
##  [8,] 0.8650014 0.8649921 0.8649833 0.8649750 0.8649671 0.8649598 0.8649528
##  [9,] 0.8657029 0.8656945 0.8656866 0.8656792 0.8656722 0.8656656 0.8656595
## [10,] 0.8660221 0.8660138 0.8660059 0.8659984 0.8659915 0.8659849 0.8659788
##           [,36]     [,37]     [,38]     [,39]     [,40]     [,41]     [,42]
##  [1,] 0.8641401 0.8641349 0.8641301 0.8641257 0.8641217 0.8641181 0.8641148
##  [2,] 0.8645657 0.8645610 0.8645566 0.8645527 0.8645491 0.8645460 0.8645432
##  [3,] 0.8664108 0.8664056 0.8664008 0.8663964 0.8663925 0.8663888 0.8663856
##  [4,] 0.8634093 0.8634055 0.8634022 0.8633993 0.8633967 0.8633945 0.8633927
##  [5,] 0.8650977 0.8650931 0.8650889 0.8650851 0.8650817 0.8650787 0.8650760
##  [6,] 0.8646411 0.8646359 0.8646310 0.8646266 0.8646225 0.8646188 0.8646154
##  [7,] 0.8653576 0.8653541 0.8653510 0.8653482 0.8653459 0.8653438 0.8653421
##  [8,] 0.8649464 0.8649403 0.8649347 0.8649295 0.8649247 0.8649203 0.8649163
##  [9,] 0.8656538 0.8656486 0.8656437 0.8656392 0.8656352 0.8656315 0.8656282
## [10,] 0.8659731 0.8659678 0.8659630 0.8659585 0.8659544 0.8659507 0.8659473
##           [,43]     [,44]     [,45]     [,46]     [,47]     [,48]     [,49]
##  [1,] 0.8641119 0.8641094 0.8641072 0.8641054 0.8641039 0.8641027 0.8641019
##  [2,] 0.8645407 0.8645386 0.8645369 0.8645355 0.8645344 0.8645337 0.8645332
##  [3,] 0.8663827 0.8663802 0.8663780 0.8663761 0.8663746 0.8663734 0.8663725
##  [4,] 0.8633912 0.8633901 0.8633893 0.8633889 0.8633887 0.8633889 0.8633894
##  [5,] 0.8650737 0.8650718 0.8650702 0.8650689 0.8650680 0.8650674 0.8650671
##  [6,] 0.8646124 0.8646098 0.8646075 0.8646056 0.8646040 0.8646027 0.8646018
##  [7,] 0.8653408 0.8653398 0.8653391 0.8653387 0.8653387 0.8653389 0.8653395
##  [8,] 0.8649126 0.8649094 0.8649065 0.8649039 0.8649017 0.8648998 0.8648983
##  [9,] 0.8656252 0.8656226 0.8656204 0.8656185 0.8656169 0.8656157 0.8656147
## [10,] 0.8659443 0.8659417 0.8659394 0.8659374 0.8659358 0.8659345 0.8659336
##           [,50]     [,51]     [,52]     [,53]     [,54]     [,55]     [,56]
##  [1,] 0.8641013 0.8641011 0.8641012 0.8641016 0.8641023 0.8641033 0.8641046
##  [2,] 0.8645331 0.8645333 0.8645338 0.8645346 0.8645357 0.8645371 0.8645388
##  [3,] 0.8663720 0.8663717 0.8663718 0.8663721 0.8663727 0.8663736 0.8663748
##  [4,] 0.8633902 0.8633913 0.8633927 0.8633944 0.8633964 0.8633986 0.8634011
##  [5,] 0.8650671 0.8650674 0.8650681 0.8650690 0.8650702 0.8650717 0.8650735
```

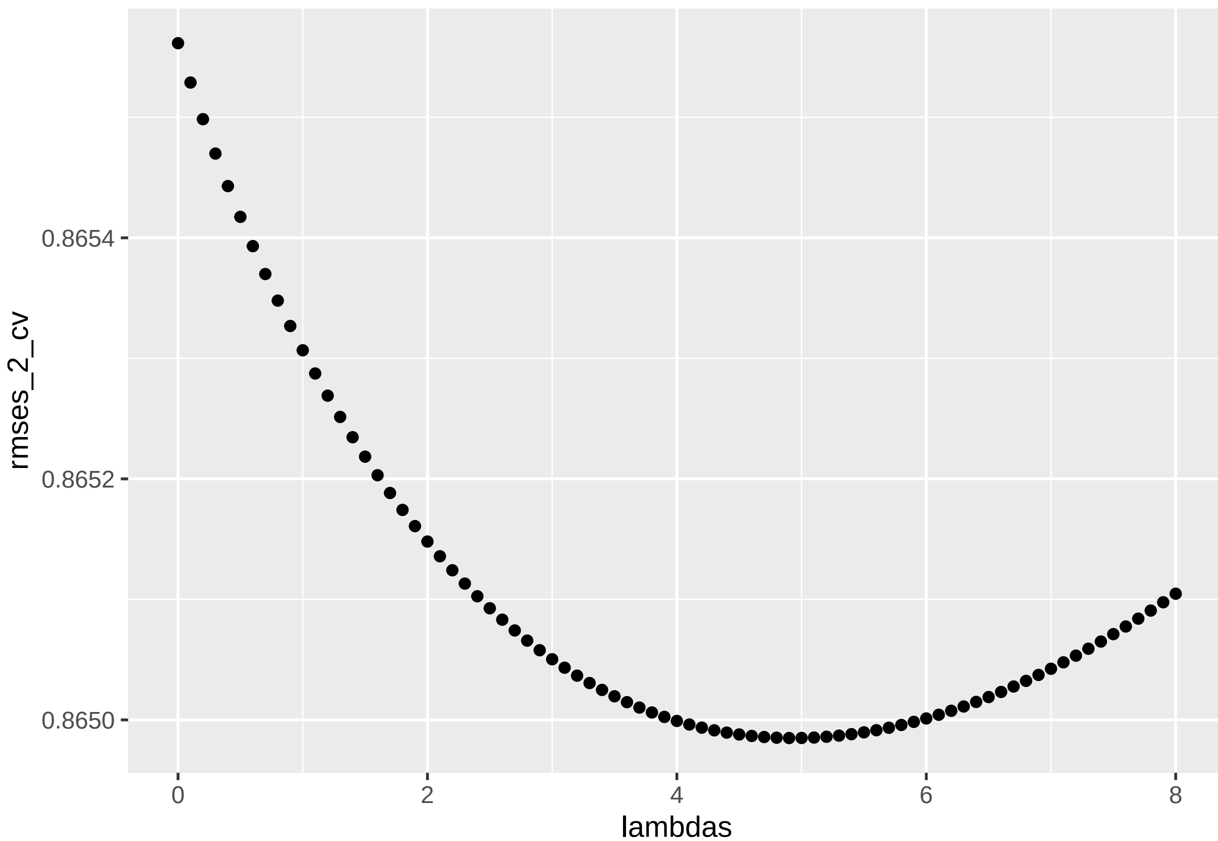
```
##  [6,] 0.8646011 0.8646008 0.8646008 0.8646010 0.8646016 0.8646024 0.8646035
##  [7,] 0.8653404 0.8653415 0.8653430 0.8653447 0.8653467 0.8653489 0.8653515
##  [8,] 0.8648971 0.8648962 0.8648957 0.8648954 0.8648955 0.8648958 0.8648965
##  [9,] 0.8656142 0.8656139 0.8656139 0.8656142 0.8656148 0.8656158 0.8656169
## [10,] 0.8659329 0.8659326 0.8659325 0.8659328 0.8659333 0.8659342 0.8659353
##           [,57]     [,58]     [,59]     [,60]     [,61]     [,62]     [,63]
##  [1,] 0.8641061 0.8641079 0.8641100 0.8641123 0.8641150 0.8641178 0.8641209
##  [2,] 0.8645407 0.8645429 0.8645453 0.8645481 0.8645511 0.8645543 0.8645578
##  [3,] 0.8663763 0.8663780 0.8663800 0.8663823 0.8663848 0.8663875 0.8663905
##  [4,] 0.8634039 0.8634070 0.8634103 0.8634138 0.8634176 0.8634217 0.8634260
##  [5,] 0.8650755 0.8650779 0.8650805 0.8650833 0.8650864 0.8650898 0.8650933
##  [6,] 0.8646049 0.8646066 0.8646085 0.8646107 0.8646131 0.8646158 0.8646187
##  [7,] 0.8653543 0.8653573 0.8653606 0.8653642 0.8653680 0.8653720 0.8653762
##  [8,] 0.8648974 0.8648986 0.8649001 0.8649018 0.8649039 0.8649061 0.8649087
##  [9,] 0.8656184 0.8656202 0.8656222 0.8656244 0.8656270 0.8656298 0.8656328
## [10,] 0.8659367 0.8659383 0.8659402 0.8659424 0.8659448 0.8659475 0.8659505
##           [,64]     [,65]     [,66]     [,67]     [,68]     [,69]     [,70]
##  [1,] 0.8641243 0.8641279 0.8641317 0.8641358 0.8641400 0.8641446 0.8641493
##  [2,] 0.8645615 0.8645654 0.8645696 0.8645740 0.8645787 0.8645835 0.8645886
##  [3,] 0.8663938 0.8663972 0.8664010 0.8664049 0.8664090 0.8664134 0.8664180
##  [4,] 0.8634305 0.8634353 0.8634403 0.8634455 0.8634509 0.8634566 0.8634624
##  [5,] 0.8650972 0.8651013 0.8651056 0.8651101 0.8651149 0.8651198 0.8651250
##  [6,] 0.8646219 0.8646253 0.8646290 0.8646328 0.8646369 0.8646413 0.8646458
##  [7,] 0.8653807 0.8653855 0.8653904 0.8653956 0.8654009 0.8654065 0.8654123
##  [8,] 0.8649115 0.8649145 0.8649178 0.8649213 0.8649250 0.8649290 0.8649332
##  [9,] 0.8656361 0.8656396 0.8656433 0.8656473 0.8656515 0.8656560 0.8656606
## [10,] 0.8659536 0.8659570 0.8659607 0.8659645 0.8659686 0.8659729 0.8659775
##           [,71]     [,72]     [,73]     [,74]     [,75]     [,76]     [,77]
##  [1,] 0.8641542 0.8641594 0.8641648 0.8641704 0.8641761 0.8641821 0.8641883
##  [2,] 0.8645939 0.8645994 0.8646051 0.8646110 0.8646171 0.8646235 0.8646300
##  [3,] 0.8664228 0.8664278 0.8664331 0.8664385 0.8664441 0.8664499 0.8664559
##  [4,] 0.8634685 0.8634748 0.8634812 0.8634879 0.8634948 0.8635018 0.8635091
##  [5,] 0.8651304 0.8651361 0.8651419 0.8651479 0.8651541 0.8651605 0.8651672
##  [6,] 0.8646505 0.8646555 0.8646607 0.8646660 0.8646716 0.8646774 0.8646833
##  [7,] 0.8654183 0.8654245 0.8654309 0.8654374 0.8654442 0.8654512 0.8654583
##  [8,] 0.8649376 0.8649423 0.8649471 0.8649522 0.8649575 0.8649630 0.8649686
##  [9,] 0.8656655 0.8656706 0.8656759 0.8656814 0.8656871 0.8656930 0.8656991
## [10,] 0.8659822 0.8659872 0.8659923 0.8659977 0.8660033 0.8660090 0.8660150
##           [,78]     [,79]     [,80]     [,81]
##  [1,] 0.8641946 0.8642012 0.8642079 0.8642149
##  [2,] 0.8646366 0.8646435 0.8646506 0.8646578
##  [3,] 0.8664621 0.8664685 0.8664751 0.8664818
##  [4,] 0.8635165 0.8635241 0.8635319 0.8635398
##  [5,] 0.8651740 0.8651809 0.8651881 0.8651955
##  [6,] 0.8646895 0.8646958 0.8647023 0.8647090
##  [7,] 0.8654656 0.8654731 0.8654808 0.8654886
##  [8,] 0.8649745 0.8649806 0.8649869 0.8649933
##  [9,] 0.8657054 0.8657119 0.8657185 0.8657254
## [10,] 0.8660211 0.8660274 0.8660339 0.8660406
```

```r
rmses_2_cv <- colMeans(rmses_2)
rmses_2_cv
```

```
##  [1] 0.8655615 0.8655289 0.8654984 0.8654699 0.8654429 0.8654174 0.8653931
##  [8] 0.8653700 0.8653479 0.8653268 0.8653067 0.8652874 0.8652690 0.8652514
```

```
## [15] 0.8652345 0.8652184 0.8652030 0.8651883 0.8651742 0.8651608 0.8651480
## [22] 0.8651358 0.8651241 0.8651131 0.8651026 0.8650926 0.8650832 0.8650742
## [29] 0.8650658 0.8650578 0.8650503 0.8650433 0.8650367 0.8650306 0.8650249
## [36] 0.8650196 0.8650147 0.8650102 0.8650061 0.8650024 0.8649991 0.8649962
## [43] 0.8649936 0.8649913 0.8649894 0.8649879 0.8649867 0.8649858 0.8649852
## [50] 0.8649849 0.8649850 0.8649853 0.8649860 0.8649869 0.8649881 0.8649896
## [57] 0.8649914 0.8649935 0.8649958 0.8649983 0.8650012 0.8650042 0.8650075
## [64] 0.8650111 0.8650149 0.8650189 0.8650232 0.8650277 0.8650324 0.8650373
## [71] 0.8650424 0.8650478 0.8650533 0.8650590 0.8650650 0.8650711 0.8650775
## [78] 0.8650840 0.8650907 0.8650976 0.8651047
```

```r
qplot(lambdas,rmses_2_cv)
```




```r
lambda <- lambdas[which.min(rmses_2_cv)]
```

From the 10-fold cross validation, we get an optimized value of lambda: 4.9. Regularized User Effects Model+Movie Effect Model. Now we use this parameter lambda to predict the validation dataset and evaluate the RMSE.

```r
mu <- mean(edx$rating)
b_i_reg <- edx %>%
  group_by(movieId) %>%
  summarize(b_i = sum(rating - mu)/(n()+lambda))
b_u_reg <- edx %>%
  left_join(b_i_reg, by="movieId") %>%
  group_by(userId) %>%
  summarize(b_u = sum(rating - b_i - mu)/(n()+lambda))
predicted_ratings_6 <-
  validation %>%
```

```r
  left_join(b_i_reg, by = "movieId") %>%
  left_join(b_u_reg, by = "userId") %>%
  mutate(pred = mu + b_i + b_u) %>%
  pull(pred)
model_6_rmse <- RMSE(predicted_ratings_6, validation$rating)
rmse_results <- bind_rows(rmse_results,
                          data_frame(Model="Regularized User Effects Model+Movie Effect Model",
                                     RMSE = model_6_rmse))
rmse_results
```

```
## # A tibble: 6 x 2
##   Model                                              RMSE
##   <chr>                                             <dbl>
## 1 Basic Average                                     1.06
## 2 Age Effect Model                                  1.05
## 3 Movie Effect Model                                0.944
## 4 User Effects Model+Movie Effect Model             0.866
## 5 Regularized Movie Effect Model                    0.944
## 6 Regularized User Effects Model+Movie Effect Model 0.865
```

There is a slight improvement here.

Let's see what matrix factorization does on the regularized Movie + User Effect Model because it gives the lowest RMSE. At this point, we need to calculate the residual. We need to still use the training set edx.

```r
lambda <- 4.9
mu <- mean(edx$rating)
b_i_reg <- edx %>%
  group_by(movieId) %>%
  summarize(b_i = sum(rating - mu)/(n()+lambda))
b_u_reg <- edx %>%
  left_join(b_i_reg, by="movieId") %>%
  group_by(userId) %>%
  summarize(b_u = sum(rating - b_i - mu)/(n()+lambda))
predicted_ratings_6_edx <-
  edx %>%
  left_join(b_i_reg, by = "movieId") %>%
  left_join(b_u_reg, by = "userId") %>%
  mutate(pred = mu + b_i + b_u) %>%
  pull(pred)
model_6_rmse_edx <- RMSE(predicted_ratings_6_edx, edx$rating)
model_6_rmse_edx
```

```
## [1] 0.8569964
```

```r
lambda <- 4.9
mu <- mean(edx$rating)
b_i_reg <- edx %>%
  group_by(movieId) %>%
  summarize(b_i = sum(rating - mu)/(n()+lambda))
b_u_reg <- edx %>%
  left_join(b_i_reg, by="movieId") %>%
  group_by(userId) %>%
  summarize(b_u = sum(rating - b_i - mu)/(n()+lambda))
predicted_ratings_6_edx <-
  edx %>%
```

```
  left_join(b_i_reg, by = "movieId") %>%
  left_join(b_u_reg, by = "userId") %>%
  mutate(pred = mu + b_i + b_u) %>%
  pull(pred)
model_6_rmse_edx <- RMSE(predicted_ratings_6_edx, edx$rating)
model_6_rmse_edx
```

```
## [1] 0.8569964
```

```
edx_residual <- edx %>%
  left_join(b_i_reg, by = "movieId") %>%
  left_join(b_u_reg, by = "userId") %>%
  mutate(residual = rating - mu - b_i - b_u) %>%
  select(userId, movieId, residual)
head(edx_residual)
```

```
##   userId movieId  residual
## 1      1     122 0.7485488
## 2      1     185 0.4831327
## 3      1     231 0.6762559
## 4      1     292 0.1966434
## 5      1     316 0.2627925
## 6      1     329 0.2795181
```

As Matrix

```
edx_for_mf <- as.matrix(edx_residual)
validation_for_mf <- validation %>%
  select(userId, movieId, rating)
validation_for_mf <- as.matrix(validation_for_mf)
```

Write edx_for_mf and validation_for_mf tables on disk

```
write.table(edx_for_mf , file = "trainset.txt" , sep = " " , row.names = FALSE, col.names = FALSE)
write.table(validation_for_mf, file = "validset.txt" , sep = " " , row.names = FALSE, col.names = FALSE
```

Use data_file() to specify a data set from a file in the hard disk.

```
set.seed(2019)
train_set <- data_file("trainset.txt")
valid_set <- data_file("validset.txt")
```

Build a recommender object

```
r <-Reco()
```

Tuning training set

```
opts <- r$tune(train_set, opts = list(dim = c(10, 20, 30), lrate = c(0.1, 0.2),
                                      costp_l1 = 0, costq_l1 = 0,
                                      nthread = 1, niter = 10))
opts
```

```
## $min
## $min$dim
## [1] 30
##
## $min$costp_l1
## [1] 0
```

```
##
## $min$costp_l2
## [1] 0.01
##
## $min$costq_l1
## [1] 0
##
## $min$costq_l2
## [1] 0.1
##
## $min$lrate
## [1] 0.1
##
## $min$loss_fun
## [1] 0.7937419
##
##
## $res
##    dim costp_l1 costp_l2 costq_l1 costq_l2 lrate  loss_fun
## 1   10        0     0.01        0     0.01   0.1 0.8069039
## 2   20        0     0.01        0     0.01   0.1 0.8112675
## 3   30        0     0.01        0     0.01   0.1 0.8208123
## 4   10        0     0.10        0     0.01   0.1 0.8046307
## 5   20        0     0.10        0     0.01   0.1 0.8020993
## 6   30        0     0.10        0     0.01   0.1 0.8029930
## 7   10        0     0.01        0     0.10   0.1 0.8038485
## 8   20        0     0.01        0     0.10   0.1 0.7956248
## 9   30        0     0.01        0     0.10   0.1 0.7937419
## 10  10        0     0.10        0     0.10   0.1 0.8245860
## 11  20        0     0.10        0     0.10   0.1 0.8236922
## 12  30        0     0.10        0     0.10   0.1 0.8231815
## 13  10        0     0.01        0     0.01   0.2 0.8096443
## 14  20        0     0.01        0     0.01   0.2 0.8224669
## 15  30        0     0.01        0     0.01   0.2 0.8371108
## 16  10        0     0.10        0     0.01   0.2 0.8053554
## 17  20        0     0.10        0     0.01   0.2 0.8050432
## 18  30        0     0.10        0     0.01   0.2 0.8070896
## 19  10        0     0.01        0     0.10   0.2 0.8024461
## 20  20        0     0.01        0     0.10   0.2 0.7997289
## 21  30        0     0.01        0     0.10   0.2 0.7992491
## 22  10        0     0.10        0     0.10   0.2 0.8232183
## 23  20        0     0.10        0     0.10   0.2 0.8221175
## 24  30        0     0.10        0     0.10   0.2 0.8205350
```

```r
r$train(train_set, opts = c(opts$min, nthread = 1, niter = 20))
```

```
## iter      tr_rmse          obj
##    0       0.8593   6.9602e+06
##    1       0.8352   6.4546e+06
##    2       0.8162   6.2645e+06
##    3       0.7984   6.0908e+06
##    4       0.7835   5.9519e+06
##    5       0.7713   5.8414e+06
##    6       0.7610   5.7506e+06
##    7       0.7526   5.6808e+06
```

```
##    8       0.7454   5.6216e+06
##    9       0.7391   5.5715e+06
##   10       0.7336   5.5283e+06
##   11       0.7287   5.4923e+06
##   12       0.7243   5.4583e+06
##   13       0.7204   5.4303e+06
##   14       0.7168   5.4039e+06
##   15       0.7136   5.3814e+06
##   16       0.7107   5.3606e+06
##   17       0.7081   5.3414e+06
##   18       0.7056   5.3245e+06
##   19       0.7033   5.3081e+06
```

Making prediction on validation set and calculating RMSE.

```
pred_file <- tempfile()
r$predict(valid_set, out_file(pred_file))
```

```
## prediction output generated at /var/folders/m7/mqtw78w13fzf7324by81zj7r0000gn/T//RtmpEa2sno/file6f34
```

```
predicted_residuals_mf <- scan(pred_file)
predicted_ratings_mf <- predicted_ratings_6 + predicted_residuals_mf
rmse_mf <- RMSE(predicted_ratings_mf,validation$rating)
rmse_results <- bind_rows(rmse_results,
                          data_frame(Model="Matrix Factorization",
                                     RMSE = rmse_mf))
rmse_results
```

```
## # A tibble: 7 x 2
##   Model                                             RMSE
##   <chr>                                            <dbl>
## 1 Basic Average                                    1.06
## 2 Age Effect Model                                 1.05
## 3 Movie Effect Model                               0.944
## 4 User Effects Model+Movie Effect Model            0.866
## 5 Regularized Movie Effect Model                   0.944
## 6 Regularized User Effects Model+Movie Effect Model 0.865
## 7 Matrix Factorization                             0.787
```

Results

For the average movie rating model that we generated first, the result was 1.0606506. After accounting for movie effects, we lowered the average to .944. In order to lower the RMSE even more, we added both the movie and user effects with the result of .866 We used regularization to penalize samples with few ratings and got a result of .94. We continued with adding user effects and reduced the RMSE to .865. Finally, we used matrix factorization to get the lowest RMSE of .787

Conclusion

In conclusion, we downloaded the data set and prepared it for analysis. We looked for various insights and created a simple model from the mean of the observed ratings. After that, we added the movie and user effects in an attempt to model user behavior. Finally, we conducted regularization that added a penalty for the movies and users with the least number of ratings. We achieved a model with an RMSE of 0.865. We decided to conduct matrix factorization on the lowest RMSE, which occured when we calculated the RMSE in model 6, which was the Regularized Movie + User Effect Model.