Movielens Report

**by Lam Vu**

5/16/2019

### SUMMARY

GroupLens Research has collected and made available rating data sets from the MovieLens web site (<http://movielens.org>). The data sets were collected over various periods of time, depending on the size of the set.

In this exercise, MovieLens 10M dataset was used to build movie rating preditions. Then the predictions will be compared to the true ratings in the validation set using RMSE.

The **objectives** of this exercise are to gain insights with **movielens** dataset through exploration, and visualization, and modeling approach to achieve the **RMSE** <= 0.87750.

### Dataset

The following code is used to generate datasets. Algorithm was develop using **edx** set. For a final test of the algorithm, predict movie ratings in the **validation** set as if they were unknown.

#### INTRODUCTION ####  
  
###################################  
# Create edx set and validation set  
###################################  
  
# 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")  
  
# 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 <- 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.numeric(levels(movieId))[movieId],  
 title = as.character(title),  
 genres = as.character(genres))  
  
movielens <- left\_join(ratings, movies, by = "movieId")  
  
# Validation set will be 10% of MovieLens data  
#used caret pkg  
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)

### 

### DATA ANALYSIS

Explorations that didn’t lead to any insights are not included in this report.

#exploring edx dataset  
class(edx)

## [1] "data.frame"

head(edx)

## userId movieId rating timestamp title  
## 1 1 122 5 838985046 Boomerang (1992)  
## 2 1 185 5 838983525 Net, The (1995)  
## 4 1 292 5 838983421 Outbreak (1995)  
## 5 1 316 5 838983392 Stargate (1994)  
## 6 1 329 5 838983392 Star Trek: Generations (1994)  
## 7 1 355 5 838984474 Flintstones, The (1994)  
## genres  
## 1 Comedy|Romance  
## 2 Action|Crime|Thriller  
## 4 Action|Drama|Sci-Fi|Thriller  
## 5 Action|Adventure|Sci-Fi  
## 6 Action|Adventure|Drama|Sci-Fi  
## 7 Children|Comedy|Fantasy

#number of rows and columns in edx dataset  
dim(edx)

## [1] 9000055 6

#number of ratings were given as zero(0) or three(3)  
edx %>% filter(rating == 0) %>% tally()

## n  
## 1 0

edx %>% filter(rating == 3) %>% tally()

## n  
## 1 2121240

### 

### Number of distinct userIds and movieIds

edx %>% summarize(n\_users = n\_distinct(userId),  
 n\_movies = n\_distinct(movieId))

## n\_users n\_movies  
## 1 69878 10677

### 

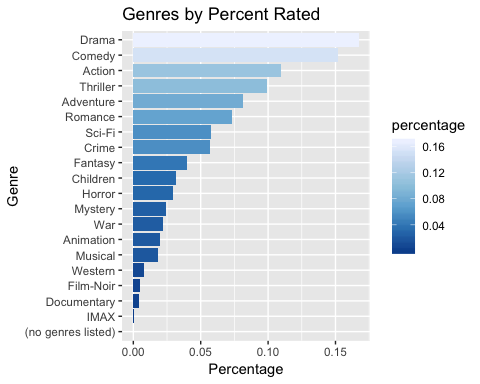
### Table of number of ratings by genres after separating combined genres.

#table of number of ratings by genres after separating combined genres  
library(tidyr)  
g\_ratings <- edx %>% separate\_rows(genres, sep = "\\|") %>%  
 group\_by(genres) %>%  
 summarize(count = n(), avg = mean(rating)) %>%  
 arrange(desc(count))

## # A tibble: 20 x 3  
## genres count avg  
## <chr> <int> <dbl>  
## 1 Drama 3910127 3.67  
## 2 Comedy 3540930 3.44  
## 3 Action 2560545 3.42  
## 4 Thriller 2325899 3.51  
## 5 Adventure 1908892 3.49  
## 6 Romance 1712100 3.55  
## 7 Sci-Fi 1341183 3.40  
## 8 Crime 1327715 3.67  
## 9 Fantasy 925637 3.50  
## 10 Children 737994 3.42  
## 11 Horror 691485 3.27  
## 12 Mystery 568332 3.68  
## 13 War 511147 3.78  
## 14 Animation 467168 3.60  
## 15 Musical 433080 3.56  
## 16 Western 189394 3.56  
## 17 Film-Noir 118541 4.01  
## 18 Documentary 93066 3.78  
## 19 IMAX 8181 3.77  
## 20 (no genres listed) 7 3.64

### 

### Distribution of Genres by Percent Rated



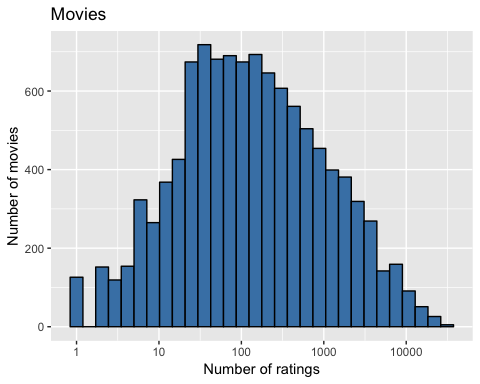
### Here are the top 10 movie titles with the greatest number of ratings including average rating.

#top 10 movie titles with the greatest number of ratings  
top\_titles <- edx %>%   
 group\_by(movieId) %>%  
 summarize(n = n(),  
 title = title[1],  
 avg = mean(rating)) %>%  
 top\_n(10, n) %>%  
 arrange(desc(n))   
top\_titles

## # A tibble: 10 x 4  
## movieId n title avg  
## <dbl> <int> <chr> <dbl>  
## 1 296 31362 Pulp Fiction (1994) 4.15  
## 2 356 31079 Forrest Gump (1994) 4.01  
## 3 593 30382 Silence of the Lambs, The (1991) 4.20  
## 4 480 29360 Jurassic Park (1993) 3.66  
## 5 318 28015 Shawshank Redemption, The (1994) 4.46  
## 6 110 26212 Braveheart (1995) 4.08  
## 7 457 25998 Fugitive, The (1993) 4.01  
## 8 589 25984 Terminator 2: Judgment Day (1991) 3.93  
## 9 260 25672 Star Wars: Episode IV - A New Hope (a.k.a. Star War… 4.22  
## 10 150 24284 Apollo 13 (1995) 3.89

### Distribution of Movies

#Distribution of Movies  
edx %>% group\_by(movieId) %>%   
 summarize(n = n())%>%  
 ggplot(aes(n)) +  
 geom\_histogram(fill = "steelblue", color = "black", bins = 30) +  
 scale\_x\_log10() +  
 xlab("Number of ratings") +  
 ylab("Number of movies") +  
 ggtitle("Movies")



### Distribution of Users

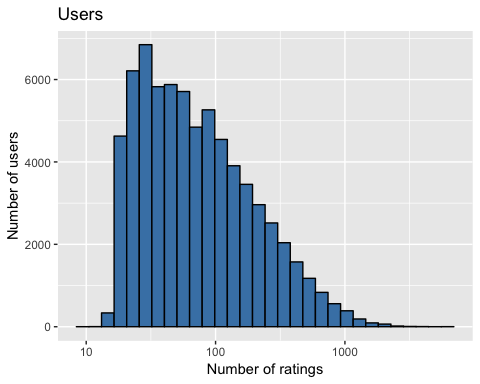
**Top ten (10) users with the most ratings**

#Top ten (10) users with the most ratings  
edx %>% group\_by(userId) %>% summarize(n = n())%>%top\_n(10, n) %>%  
 arrange(desc(n))

## # A tibble: 10 x 2  
## userId n  
## <int> <int>  
## 1 59269 6616  
## 2 67385 6360  
## 3 14463 4648  
## 4 68259 4036  
## 5 27468 4023  
## 6 19635 3771  
## 7 3817 3733  
## 8 63134 3371  
## 9 58357 3361  
## 10 27584 3142

**Distribution of users vs. ratings**

#Distribution of Users  
edx %>% group\_by(userId) %>% summarize(n = n()) %>%  
 ggplot(aes(n)) + geom\_histogram(fill = "steelblue", color = "black", bins = 30) +  
 scale\_x\_log10() +   
 xlab("Number of ratings") +  
 ylab("Number of users") +  
 ggtitle("Users")



### Top 10 movie titles with the greatest average rating

#top 10 movie titles with the greatest average rating  
top\_avg <- edx %>%   
 group\_by(movieId) %>%  
 summarize(n = n(),  
 title = title[1],  
 avg = mean(rating)) %>%  
 top\_n(10, n) %>%  
 arrange(desc(avg))   
top\_avg

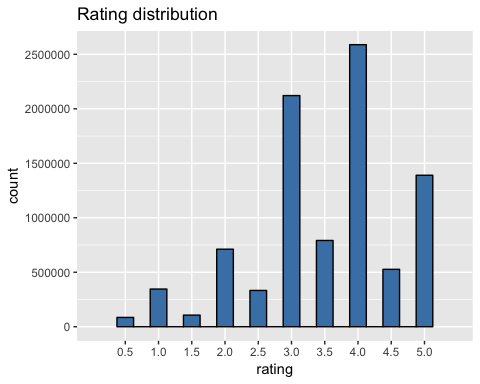
## # A tibble: 10 x 4  
## movieId n title avg  
## <dbl> <int> <chr> <dbl>  
## 1 318 28015 Shawshank Redemption, The (1994) 4.46  
## 2 260 25672 Star Wars: Episode IV - A New Hope (a.k.a. Star War… 4.22  
## 3 593 30382 Silence of the Lambs, The (1991) 4.20  
## 4 296 31362 Pulp Fiction (1994) 4.15  
## 5 110 26212 Braveheart (1995) 4.08  
## 6 356 31079 Forrest Gump (1994) 4.01  
## 7 457 25998 Fugitive, The (1993) 4.01  
## 8 589 25984 Terminator 2: Judgment Day (1991) 3.93  
## 9 150 24284 Apollo 13 (1995) 3.89  
## 10 480 29360 Jurassic Park (1993) 3.66

**In general, half star ratings are less common than whole star ratings. This can be observed at the table and graph below.**

#table of frequency of star ratings from most to least  
rating\_frequency <- edx %>% group\_by(rating) %>%   
 summarize(n = n()) %>%   
 arrange(desc(n))  
rating\_frequency

## # A tibble: 10 x 2  
## rating n  
## <dbl> <int>  
## 1 4 2588430  
## 2 3 2121240  
## 3 5 1390114  
## 4 3.5 791624  
## 5 2 711422  
## 6 4.5 526736  
## 7 1 345679  
## 8 2.5 333010  
## 9 1.5 106426  
## 10 0.5 85374

### Rating distribution



### MODELS

Root-mean-square error (RMSE) is used to measure the differences between values predicted by a model and the values observed.

RMSE <- function(true\_ratings, predicted\_ratings){  
 sqrt(mean((true\_ratings - predicted\_ratings)^2))  
}

### Simple Model

Predicting same rating for all movies across all users

### Simplest Model: predict same rating for all movies across all users  
#Yu,i = mu + Eu,i  
mu <- mean(edx$rating)  
mu

**## [1] 3.512465**

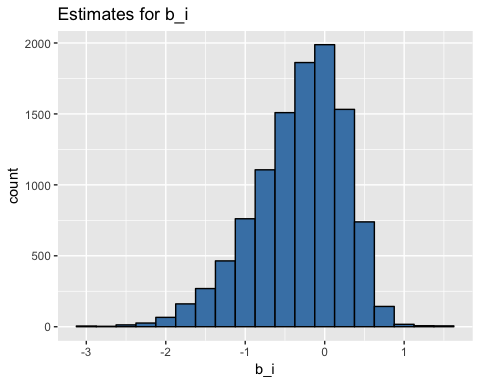
#test results based on simple prediction  
naive\_rmse <- RMSE(validation$rating, mu)  
  
#create a table that's going to store the results   
rmse\_results <- data\_frame(method = "Just the average", RMSE = naive\_rmse)  
rmse\_results %>% knitr::kable()

|  |  |
| --- | --- |
| method | **RMSE** |
| Just the average | 1.061202 |

### 

### Movie Effect Model

#Yu,i = mu + b\_i + Eu,i where b\_i = the average rating for movie i or as "bias  
movie\_avgs <- edx %>%  
 group\_by(movieId)%>%  
 summarize(b\_i = mean(rating - mu))  
  
#plot movie averages b\_i  
movie\_avgs %>%  
 ggplot(aes(b\_i )) +  
 geom\_histogram(fill = "steelblue",binwidth = .25, color = "black") +  
 ggtitle("Estimates for b\_i")



###Model  
predicted\_ratings <- mu + validation %>%  
 left\_join(movie\_avgs, by='movieId') %>%  
 .$b\_i  
model\_1\_rmse <- RMSE(predicted\_ratings, validation$rating)  
  
rmse\_results <- bind\_rows(rmse\_results,  
 data\_frame(method="Movie Effect Model",   
 RMSE = model\_1\_rmse ))  
rmse\_results %>% knitr::kable()

|  |  |
| --- | --- |
| method | **RMSE** |
| Just the average | 1.0612018 |
| Movie Effect Model | 0.9439087 |

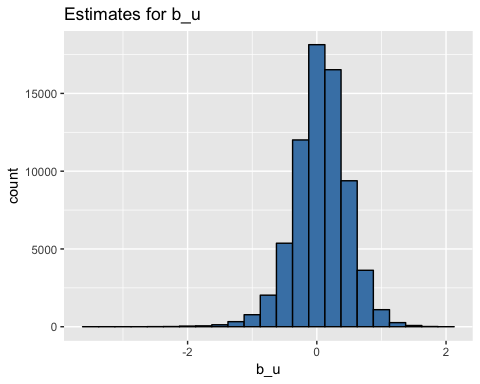
### 

### Movie and User Effect Model

###user averages, b\_u  
user\_avgs <- edx %>%  
 left\_join(movie\_avgs, by='movieId') %>%  
 group\_by(userId) %>%  
 summarize(b\_u = mean(rating - mu - b\_i))

Plot user averages b\_u

user\_avgs %>%  
 ggplot(aes(b\_u )) +  
 geom\_histogram(fill = "steelblue", binwidth = .25, color = "black") +  
 ggtitle("Estimates for b\_u ")



**Model**

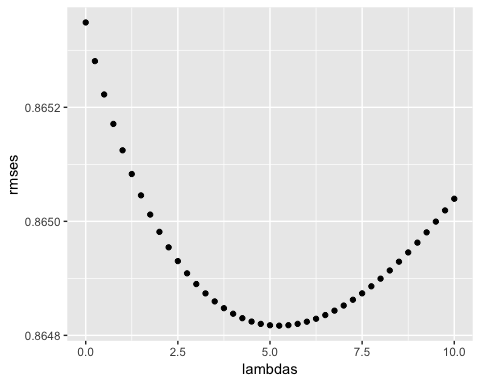
###Movie and User Effect Model  
predicted\_ratings <- validation %>%  
 left\_join(movie\_avgs, by='movieId') %>%  
 left\_join(user\_avgs, by='userId') %>%  
 mutate(pred = mu + b\_i + b\_u) %>%  
 .$pred  
  
model\_2\_rmse <- RMSE(predicted\_ratings, validation$rating)  
  
rmse\_results <- bind\_rows(rmse\_results,  
 data\_frame(method="Movie + User Effect Model",   
 RMSE = model\_2\_rmse ))  
rmse\_results %>% knitr::kable()

|  |  |
| --- | --- |
| method | **RMSE** |
| Just the average | 1.0612018 |
| Movie Effect Model | 0.9439087 |
| Movie + User Effect Model | 0.8653488 |

### Regularization Model

Regularization permits us to penalize large estimates that come from small sample sizes. This model includes the parameters for both **movie and user** effects. Cross-validation is also used to pick lambda.

# lambda is a tuning parameter  
# use cross-validation to find the lambda with lowest rmse   
lambdas <- seq(0, 10, 0.25)  
  
# For each lambda, find b\_i & b\_u, followed by rating prediction & testing  
rmses <- sapply(lambdas, function(lambda){  
   
 mu <- mean(edx$rating)  
   
 b\_i <- edx %>%   
 group\_by(movieId) %>%  
 summarize(b\_i = sum(rating - mu)/(n()+lambda))  
   
 b\_u <- edx %>%   
 left\_join(b\_i, by="movieId") %>%  
 group\_by(userId) %>%  
 summarize(b\_u = sum(rating - b\_i - mu)/(n()+lambda))  
   
 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(predicted\_ratings, validation$rating))  
})

**Plot rmses vs lambdas to select the optimal lambda** 

**Optimal lambda**

## [1] **5.25**

### RESULTS

Result of all models included in the table below.

|  |  |
| --- | --- |
| method | **RMSE** |
| Just the average | **1.0612018** |
| Movie Effect Model | **0.9439087** |
| Movie + User Effect Model | **0.8653488** |
| Regularized Movie and User Effect Model | **0.8648170** |

### CONCLUSION

Residual mean squared error is used to evaluate how close the predictions are to the true values in the validation set. RMSE of the **Regularized Models** has improved from **0.8653488** to **0.8648170** compared to the **Movie and User Effect Model** . Both models meet the objective of this exercise of achieving RMSE <= 0.87750.