Project: MovieLens Recommendation System

Abraham Verde

March/4/2022

INTRODUCTION

For this project, I will be creating a movie recommendation system using the MovieLens dataset. The version of movielens included in the dslabs package is a small subset of a major dataset. A movie recomendation system pretends to predict the preference of a user. I'm going to use the rating values and movie genres in the dataset to make models for this prediction.

The data set used in this project is loaded using the code provided in the course. This code split the data into a train set named edx and test set named validation. It's important to notice, the test set (validation)is roughly 10% the train set (edx).

```
##########################################################
# Create edx set, validation set (final hold-out test set)
##########################################################

# Note: this process could take a couple of minutes

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

```
## Loading required package: tidyverse

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

## v ggplot2 3.3.5     v purrr   0.3.4
## v tibble  3.1.1     v dplyr   1.0.6
## v tidyr   1.1.3     v stringr 1.4.0
## v readr   1.4.0     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)

# 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 3.6 or earlier:
movies <- as.data.frame(movies) %>% mutate(movieId = as.numeric(levels(movieId))[movieId],
                                           title = as.character(title),
                                           genres = as.character(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)
```

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

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

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

MY SOURCE CODE

PRIMARY DATA EXPLORATION

Once I loaded the data set, is necessary to perform some exploration to get familiar with the data structures of the involved objects. In this first sight we can easily watch the edx and validation objects, both of them has six columns with the main information, such as: userId, movieId, rating, timestamp, title and movie geners.

```
library(dplyr)
library(ggplot2)
library(tidyr)

#Exploration of main information
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
```

```
summary(edx)
```

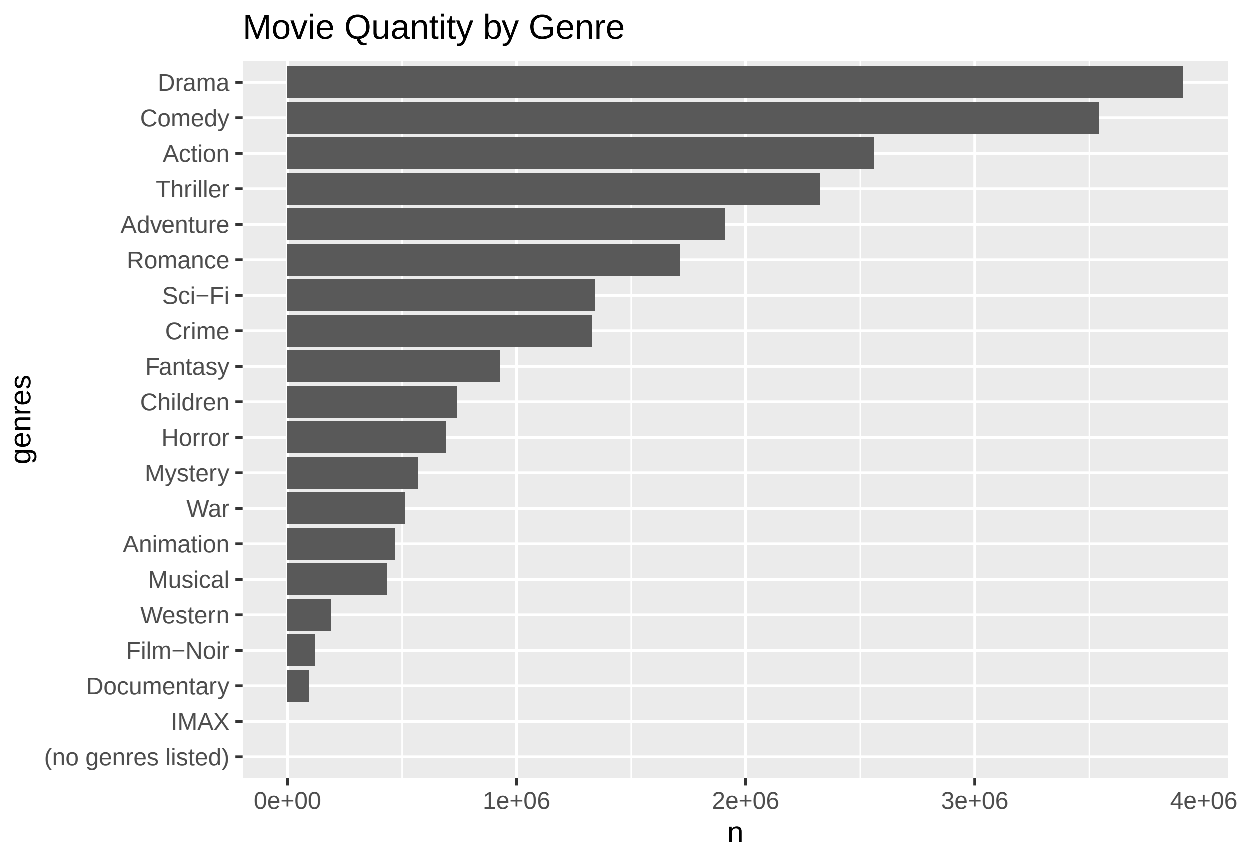
```
##      userId         movieId          rating        timestamp
##  Min.   :    1   Min.   :    1   Min.   :0.500   Min.   :7.897e+08
##  1st Qu.:18124   1st Qu.:  648   1st Qu.:3.000   1st Qu.:9.468e+08
##  Median :35738   Median : 1834   Median :4.000   Median :1.035e+09
##  Mean   :35870   Mean   : 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
##
##
##
```

Reviewing genre movie information It's quite useful to get the genres information per movies. In the dataset this information comes non splited in the field genres. To get genres information per movies is necessary split it and create a new data frame with this information. Once I have the data frame already populated, I pretend to show this information in a bar graphs to watch the information in a confortable way.

```r
dataGR <- edx %>% separate_rows(genres, sep = "\\|")
GR_Rating <- dataGR %>% group_by(genres) %>% summarize(n=n()) %>% arrange(n)
##Convert genres to factor, to get order in graph
GR_Rating$genres <- factor(GR_Rating$genres, levels = GR_Rating$genres[order(GR_Rating$n)])
```

```r


graph_1<-ggplot(data=GR_Rating, aes(x=n, y=genres)) +
  geom_bar(stat="identity") +
  ggtitle("Movie Quantity by Genre")
graph_1
```



Graph genre movies

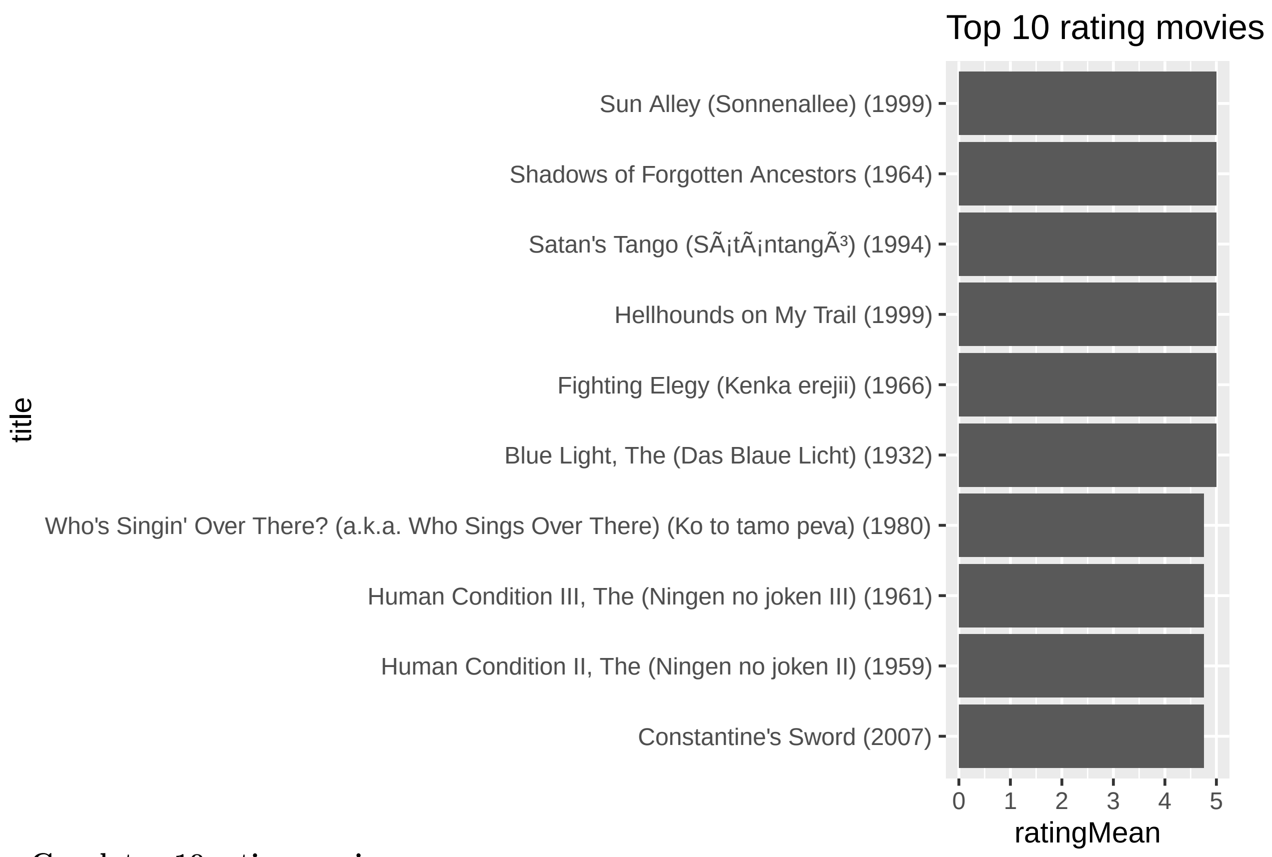
Detect top 10 rating movies Other information that is good to have is about top rated movies. In this case, I'm going to show only the first 10 highest ratings.

```
top10 <- edx %>% group_by(title) %>% summarize(ratingMean=mean(rating)) %>% arrange(desc(ratingMean))

##Convert title to factor, to get order in graph
top10$title <- factor(top10$title, levels = top10$title[order(top10$ratingMean)])
top10[1:10,]
```

```
## # A tibble: 10 x 2
##    title                                                           ratingMean
##    <fct>                                                                <dbl>
##  1 Blue Light, The (Das Blaue Licht) (1932)                             5
##  2 Fighting Elegy (Kenka erejii) (1966)                                 5
##  3 Hellhounds on My Trail (1999)                                        5
##  4 Satan's Tango (SÃ¡tÃ¡ntangÃ³) (1994)                                 5
##  5 Shadows of Forgotten Ancestors (1964)                                5
##  6 Sun Alley (Sonnenallee) (1999)                                       5
##  7 Constantine's Sword (2007)                                           4.75
##  8 Human Condition II, The (Ningen no joken II) (1959)                  4.75
##  9 Human Condition III, The (Ningen no joken III) (1961)                4.75
## 10 Who's Singin' Over There? (a.k.a. Who Sings Over There) (Ko to ta~       4.75
```

```r
graph_2<- top10[1:10, ] %>% ggplot(aes(x=ratingMean, y=title)) +
  geom_bar(stat="identity") +
  ggtitle("Top 10 rating movies")
graph_2
```



Graph top 10 rating movies

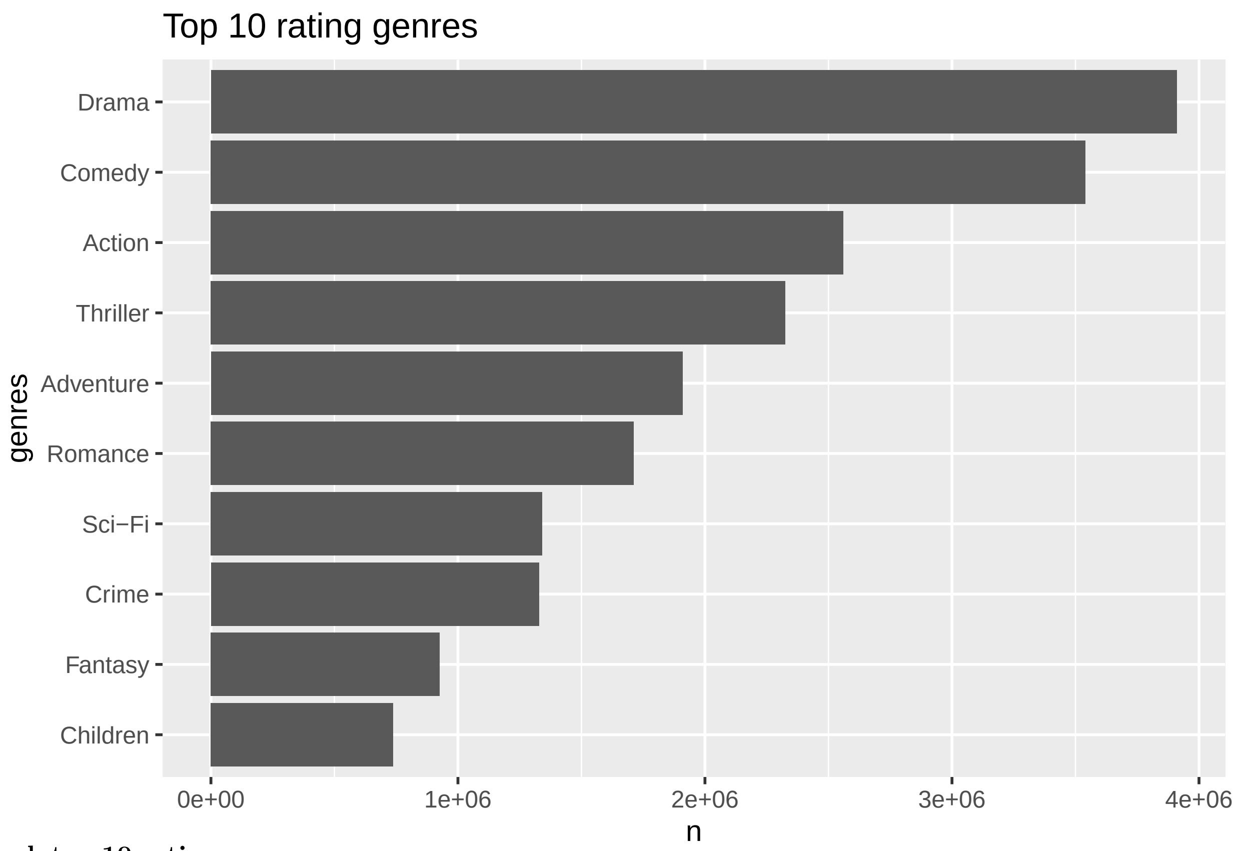
Detect top 10 rating genre As the previous graph, is good to know the top movie rating grouping by genres. In this case, I'm going to show only the first 10 highest rating per genres.

```r
top10_rating <- GR_Rating %>% group_by(genres) %>% arrange(desc(n))

##Convert genres to factor, to get order in graph
top10_rating$genres <- factor(top10_rating$genres, levels = top10_rating$genres[order(top10_rating$n)])
top10_rating[1:10, ]
```

```
## # A tibble: 10 x 2
## # Groups:   genres [10]
##    genres         n
##    <fct>      <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
```

```
graph_3<- top10_rating[1:10, ] %>% ggplot(aes(x=n, y=genres)) +
  geom_bar(stat="identity") +
  ggtitle("Top 10 rating genres")
graph_3
```



Graph top 10 rating genre

ANALYSIS

The next step after the exploration data analysis, is to build some models that help us to get the prediction.

Define RMSE function By definition The root mean square error (RMSE) allows us to measure how far predicted values are from observed values in a regression analysis. I build a function to get RMSE value.

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

SPLIT THE DATASET EDX. It's not allowed use validations set to train, so i'm going to split edx dataset into train and test set

```
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
```

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

First Approach - just rating average The simplest approach is just the rating average, I'm going to take the average of the rating values from the train set.

```
FirstApproach_Model <- mean(train_set$rating);
FirstApproach_Model
```

```
## [1] 3.512457
```

PREDICT - First Approach Once I have the "model", get the RMSE values.

```
FirstApproach_Predict<- RMSE(test_set$rating,FirstApproach_Model)
FirstApproach_Predict
```

```
## [1] 1.060056
```

```


rmse_results <- tibble(method = "First Approach - Average", RMSE = FirstApproach_Predict)
```

Save in table the first approach

```


rmse_results
```

Show results in a table for comparason

```
## # A tibble: 1 x 2
##   method                    RMSE
##   <chr>                    <dbl>
## 1 First Approach - Average  1.06
```

Second Approach - using movie effect As second approach I'm going to add some features to the model to try to increase the accuracy of the prediction. In this case following the approach using in the course: https://rafalab.github.io/dsbook/large-datasets.html#modeling-movie-effects, I'm going to add bs effect to the model.

```
movie_avgs <- train_set %>% group_by(movieId) %>% summarize(b_i = mean(rating - FirstApproach_Model))
```

```
SecondApproach_Model <- FirstApproach_Model + test_set %>% left_join(movie_avgs, by='movieId') %>% pull
```

Second Approach - using movie effect

PREDICT - Second Approach - using movie effect In the next, I'm going to have the RMSE values for the second approach model.

```
SecondApproach_Predict<- RMSE(test_set$rating,SecondApproach_Model)
SecondApproach_Predict
```

```
## [1] 0.9429615
```

```
rmse_results <- bind_rows(rmse_results, data_frame(method="Second Approach - Movie Effect",  RMSE = Sec
```

Save in table the second approach

```
rmse_results
```

Show results in a table for comparason

```
## # A tibble: 2 x 2
##   method                          RMSE
##   <chr>                          <dbl>
## 1 First Approach - Average       1.06
## 2 Second Approach - Movie Effect 0.943
```

CHECK WITH VALIDATION SET

```
validation_Mean <- mean(validation$rating)
validation_Movie_avg <- validation %>% group_by(movieId) %>% summarize(b_i = mean(rating - validation_M
validation_Model <- validation_Mean + validation %>% left_join(validation_Movie_avg, by='movieId') %>% p
validation_RMSE <- RMSE(validation$rating,validation_Model)
validation_RMSE
```

```
## [1] 0.9383091
```

```
rmse_results <- bind_rows(rmse_results, data_frame(method="Validation",  RMSE = validation_RMSE))
```

Show results in a table for comparasion

SHOW FINAL COMPARISON TABLE

We can easily detect that the second approach is way better than the first one, in this case, I suggest to use the second approach to predict the recomendation system.

```
rmse_results
```

```
## # A tibble: 3 x 2
##   method                         RMSE
##   <chr>                         <dbl>
## 1 First Approach - Average       1.06
## 2 Second Approach - Movie Effect 0.943
## 3 Validation                     0.938
```

CONCLUSION

I already have developed two diferents approaches. The first one was only de rating average, this model or approach, give me a RMSE value higher than 1, in this case, I considered using a second approach looking for increasing the accuracy of the prediction.

The second approach included the bs term that represent the average rating for each movie. With this approach, the RMSE value that I got was lower than 1.

In consequence, the second approach is quite better than the first one and it's recommended to use it.