Capstone I: MovieLens Recommender Project

HarvardX: PH125.9x Data Science

Mamadi Fofana

January 2020

I. Introduction

For this project, we will be creating a movie recommendation system using the MovieLens dataset. The version of MovieLens included in the dslabs package generated by the GroupLens research lab. We will be creating our own recommendation system using the 10M version of the MovieLens dataset to make the computation a little easier.

We will adopt the prediction version of recommender problem which aims is to predict the rating value for a user-item combination.

We are going to train our algorithms using the inputs in edx dataset and to predict movie ratings in the validation set.

The train data has 9,000,055 records in 6 variables and the validate data 999,999 records for the same variables as in the train

For this project we will be focusing on regression models using RMSE to evaluate our model

We will follow below steps:

- loading of the two datasets,
- pre-processing data if necessary,
- exploration and visualization of data
- modeling approach
- results obtained
- conclusion

II. Analysis

1. Loading data

```
#################################
# Create edx set, 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")
if(!require(data.table)) install.packages("data.table", 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 <- 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")
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

set.seed(1, sample.kind="Rounding")
# if using R 3.5 or earlier, use `set.seed(1)` instead
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)
```

Our loading code create two datasets:

- edx for training dataset
- validation for validation dataset

Below is the structure of the datasets

```
> glimpse(edx)
Observations: 9,000,055
Variables: 6
$ userId    <int> 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, ...
$ movieId   <dbl> 122, 185, 292, 316, 329, 355, 356, 362, 364, 370, 377, 42...
$ rating    <dbl> 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, ...
$ timestamp <int> 838985046, 838983525, 838983421, 838983392, 838983392, 83...
$ title     <chr> "Boomerang (1992)", "Net, The (1995)", "Outbreak (1995)",...
$ genres    <chr> "Comedy|Romance", "Action|Crime|Thriller", "Action|Drama|...
> glimpse(validation)
Observations: 999,999
Variables: 6
$ userId    <int> 1, 1, 1, 2, 2, 2, 3, 3, 4, 4, 4, 5, 5, 5, 5, 5, 5, 5, 5, ...
$ movieId   <dbl> 231, 480, 586, 151, 858, 1544, 590, 4995, 34, 432, 434, 8...
$ rating    <dbl> 5.0, 5.0, 5.0, 3.0, 2.0, 3.0, 3.5, 4.5, 5.0, 3.0, 3.0, 3....
$ timestamp <int> 838983392, 838983653, 838984068, 868246450, 868245645, 86...
$ title     <chr> "Dumb & Dumber (1994)", "Jurassic Park (1993)", "Home Alo...
$ genres    <chr> "Comedy", "Action|Adventure|Sci-Fi|Thriller", "Children|C...
> 
```

The column “rating” we want to predict represents a rating given by one user to one movie in each row.

2. Preprocessing

Missing data

```
> which(is.na(edx))
integer(0)
> which(is.na(validation))
integer(0)
> 
```

It looks like there is no missing data

Data conversion

Visualization of data suggest timestamp need to be converted

```
> edx <- edx %>% mutate(timestamp = as_datetime(timestamp))
> str(edx)
'data.frame':	9000055 obs. of  6 variables:
 $ userId   : int  1 1 1 1 1 1 1 1 1 1 ...
 $ movieId  : num  122 185 292 316 329 355 356 362 364 370 ...
 $ rating   : num  5 5 5 5 5 5 5 5 5 5 ...
 $ timestamp: POSIXct, format: "1996-08-02 11:24:06" "1996-08-02 10:58:45" "1996-08-02 10:57:01" ...
 $ title    : chr  "Boomerang (1992)" "Net, The (1995)" "Outbreak (1995)" "Stargate (1994)" ...
 $ genres   : chr  "Comedy|Romance" "Action|Crime|Thriller" "Action|Drama|Sci-Fi|Thriller" "Action|Adventure|Sci-Fi" ...

> validation <- validation %>% mutate(timestamp = as_datetime(timestamp))
> str(validation)
'data.frame':	999999 obs. of  6 variables:
 $ userId   : int  1 1 1 2 2 2 3 3 4 4 ...
 $ movieId  : num  231 480 586 151 858 ...
 $ rating   : num  5 5 5 3 2 3 3.5 4.5 5 3 ...
 $ timestamp: POSIXct, format: "1996-08-02 10:56:32" "1996-08-02 11:00:53" "1996-08-02 11:07:48" "1997-07-07 03:34:10" .
 $ title    : chr  "Dumb & Dumber (1994)" "Jurassic Park (1993)" "Home Alone (1990)" "Rob Roy (1995)" ...
 $ genres   : chr  "Comedy" "Action|Adventure|Sci-Fi|Thriller" "Children|Comedy" "Action|Drama|Romance|War" ...
> 
```

3. Initial Data Exploration

Individual Feature Statistics

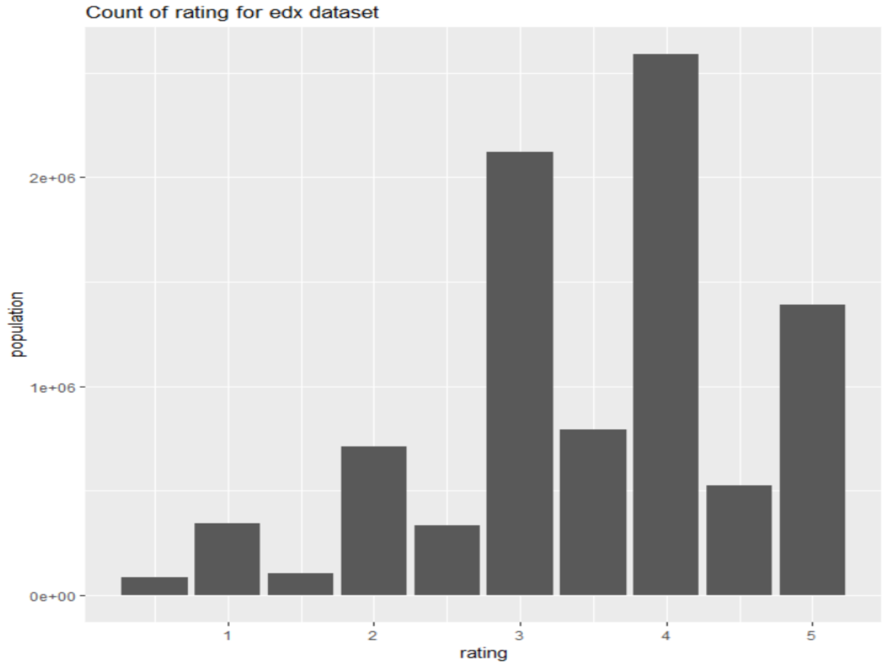
```
> summary(edx)
     userId         movieId          rating        timestamp                         title              genres         
 Min.   :    1   Min.   :    1   Min.   :0.500   Min.   :1995-01-09 11:46:49   Length:9000055     Length:9000055    
 1st Qu.:18124   1st Qu.:  648   1st Qu.:3.000   1st Qu.:2000-01-01 23:11:23   Class :character   Class :character  
 Median :35738   Median : 1834   Median :4.000   Median :2002-10-24 21:11:58   Mode  :character   Mode  :character  
 Mean   :35870   Mean   : 4122   Mean   :3.512   Mean   :2002-09-21 13:45:07                                        
 3rd Qu.:53607   3rd Qu.: 3626   3rd Qu.:4.000   3rd Qu.:2005-09-15 02:21:21                                        
 Max.   :71567   Max.   :65133   Max.   :5.000   Max.   :2009-01-05 05:02:16                                        
> summary(validation)
     userId         movieId          rating        timestamp                         title              genres         
 Min.   :    1   Min.   :    1   Min.   :0.500   Min.   :1995-01-09 11:46:49   Length:999999      Length:999999     
 1st Qu.:18096   1st Qu.:  648   1st Qu.:3.000   1st Qu.:1999-12-31 20:53:33   Class :character   Class :character  
 Median :35768   Median : 1827   Median :4.000   Median :2002-10-22 00:34:22   Mode  :character   Mode  :character  
 Mean   :35870   Mean   : 4108   Mean   :3.512   Mean   :2002-09-20 11:12:59                                        
 3rd Qu.:53621   3rd Qu.: 3624   3rd Qu.:4.000   3rd Qu.:2005-09-13 19:50:56                                        
 Max.   :71567   Max.   :65133   Max.   :5.000   Max.   :2009-01-05 04:50:28                                        
> |
```

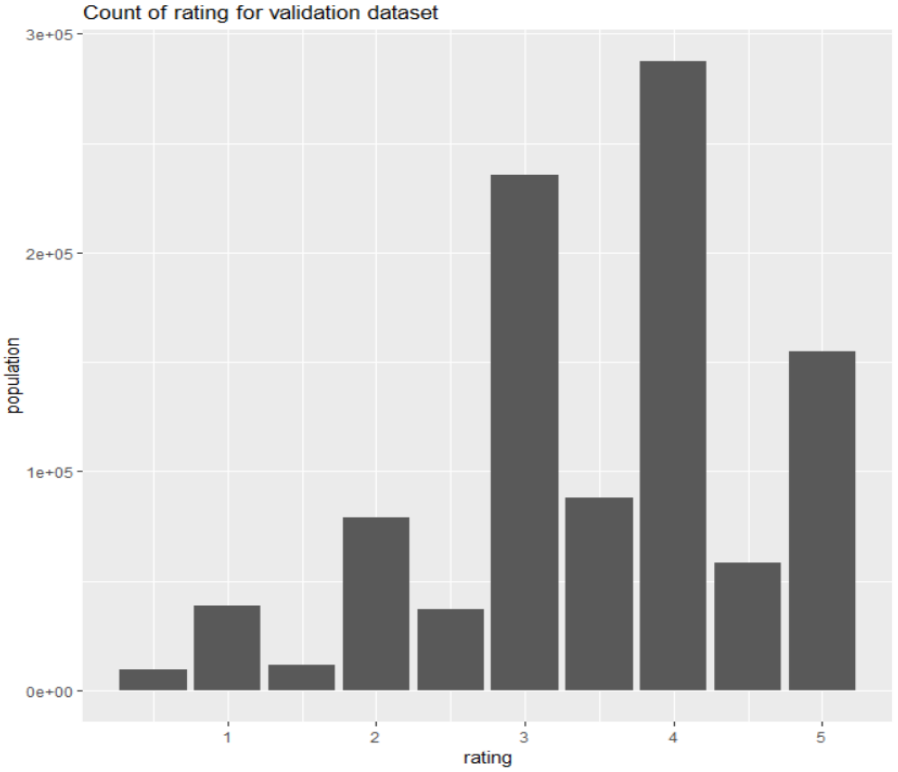
Statistic of movies and users

```
> n_distinct(edx$movieId)
[1] 10677
> n_distinct(edx$userId)
[1] 69878
```

We have a total of 10677 different movies for 69878 different users

Statistic of rating variable

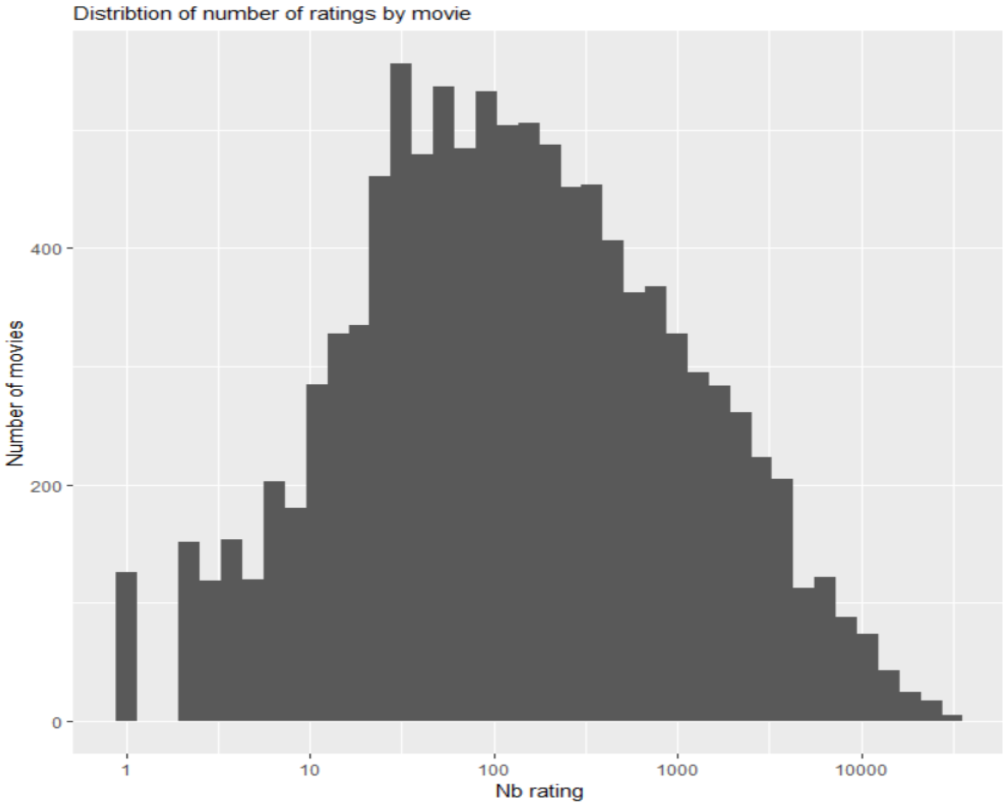




From this graph, we noticed That

- None zeros were given as ratings in the edx dataset
- In general, half star ratings are less common than whole star ratings (e.g., there are fewer ratings of 3.5 than there are ratings of 3 or 4, etc.).
- Both have similar distributions.

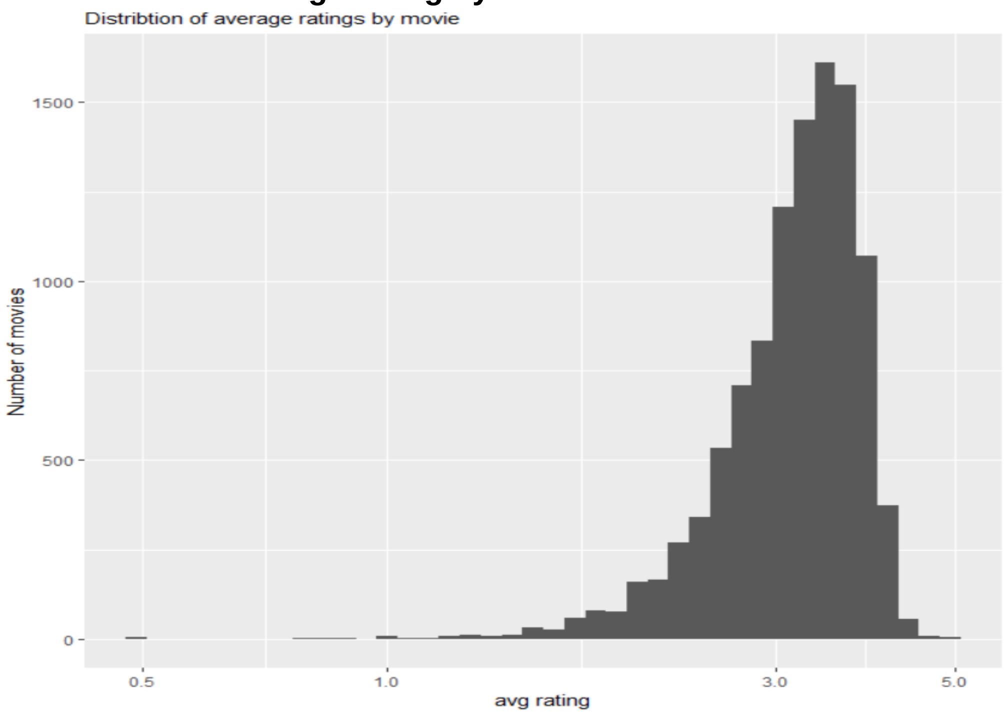
Distribution of number of rating by movie



We can see that some movies get more ratings than others

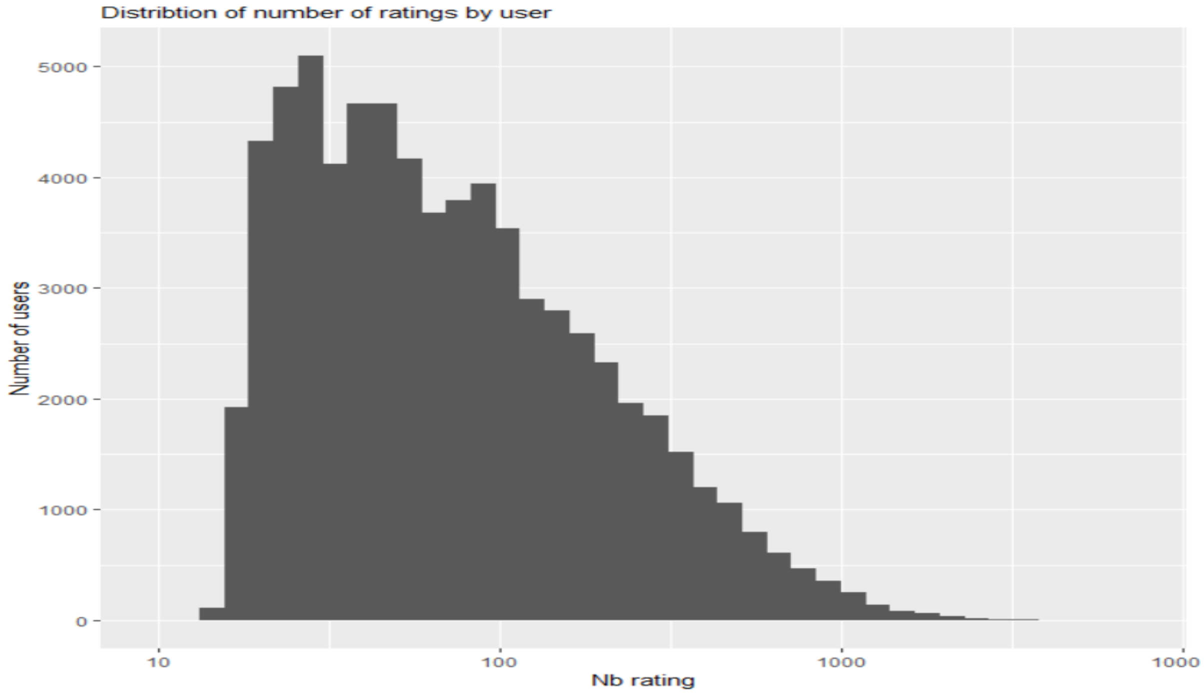
This is intuitive, since some movies are more popular than others.

Distribution of average rating by movie



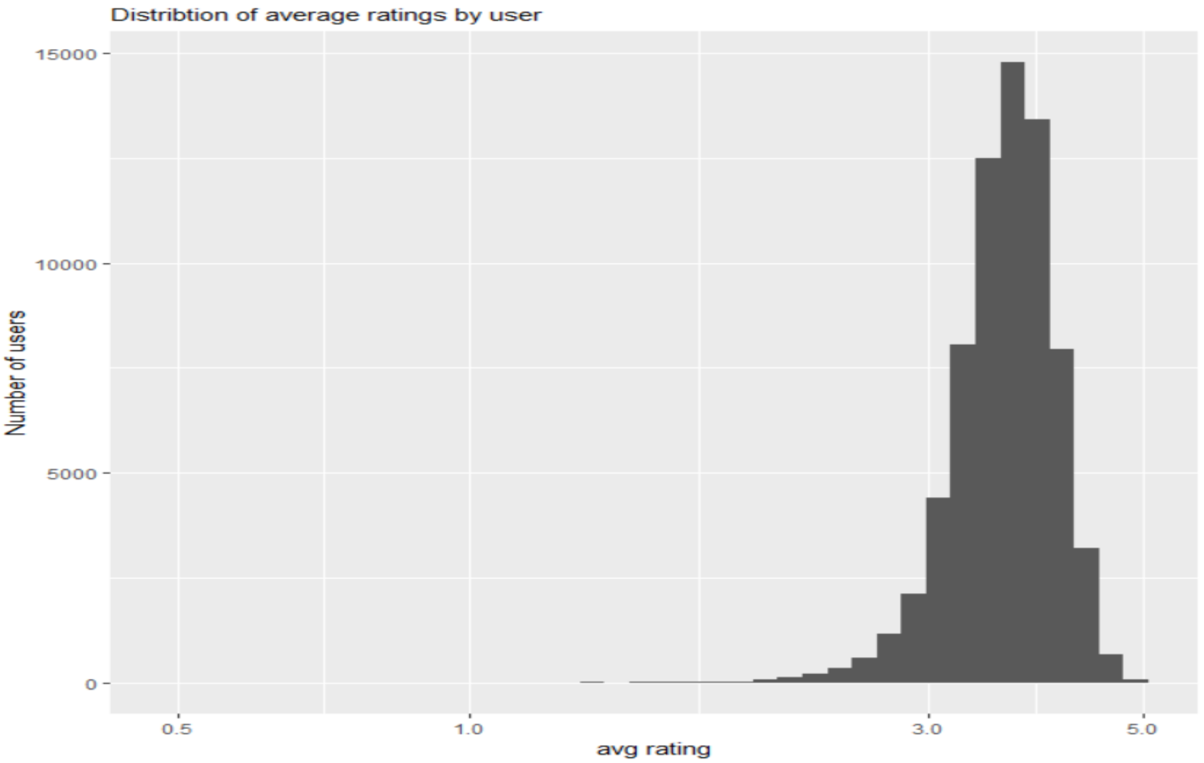
This chart shows that the average rating given to a movie is a distorted bell curve around the mean (3.5).

Distribution of rating by userId



We can see that some users are more active than others

Distribution of average rating by userId



This chart shows the number of users for whom a given rating is their average rating. This clearly shows that while the average ratings across users form a normal curve around the mean (3.5), there is a variation in average rating across users.

Distribution of movies by genres

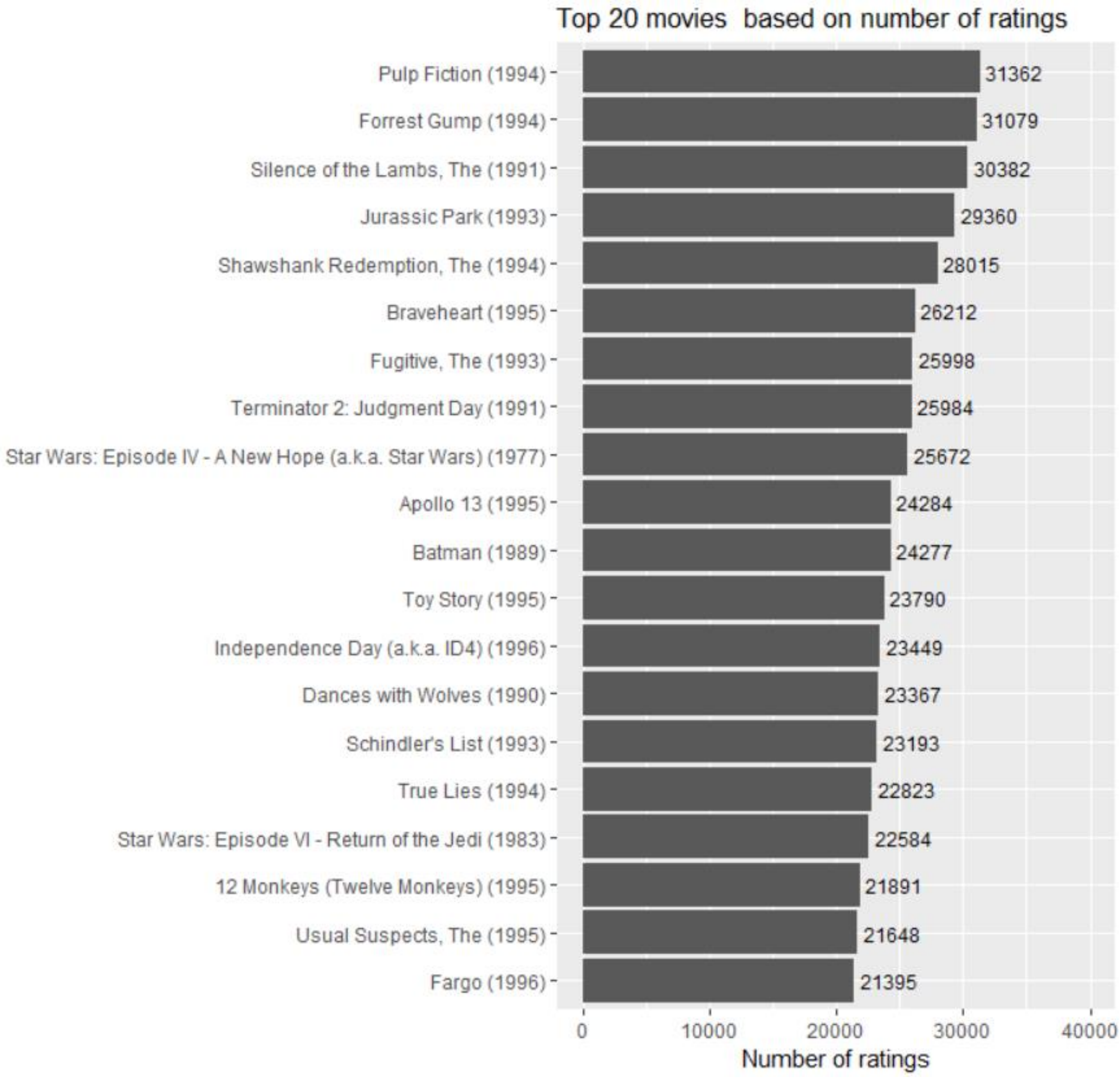
```
> top_genre <- edx %>% separate_rows(genres, sep = "\\|") %>%
+   group_by(genres) %>%
+   summarize(count = n()) %>%
+   arrange(desc(count))
> wordcloud(words=top_genre$genres,freq=top_genre$count,min.freq=50,
+           max.words = 30,random.order=FALSE,random.color=FALSE,
+           rot.per=0.35,colors = brewer.pal(8,"Dark2"),scale=c(5,.2),
+           family="plain",font=2,
+           main = "Top genres ")
```


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

we notice that the “Drama” genre has the top number of movies ratings, followed by the “Comedy” and the “Action” genres.

Movie with the greatest number of ratings

```
> edx %>%
+     group_by(title) %>%
+     summarize(count=n()) %>%
+     top_n(20,count) %>%
+     arrange(desc(count)) %>%
+     ggplot(aes(x=reorder(title, count), y=count)) +
+     geom_bar(stat='identity') + coord_flip(y=c(0, 40000)) +
+     labs(x="", y="Number of ratings") +
+     geom_text(aes(label= count), hjust=-0.1, size=3) +
+     labs(title="Top 20 movies  based on number of ratings")
```



Rating by time release

Let's try first to add a new field about year of release

```
edx <- mutate(edx, year = as.numeric(substr(edx$title,nchar(edx$title)-4, nchar(edx$title)-1)))
```

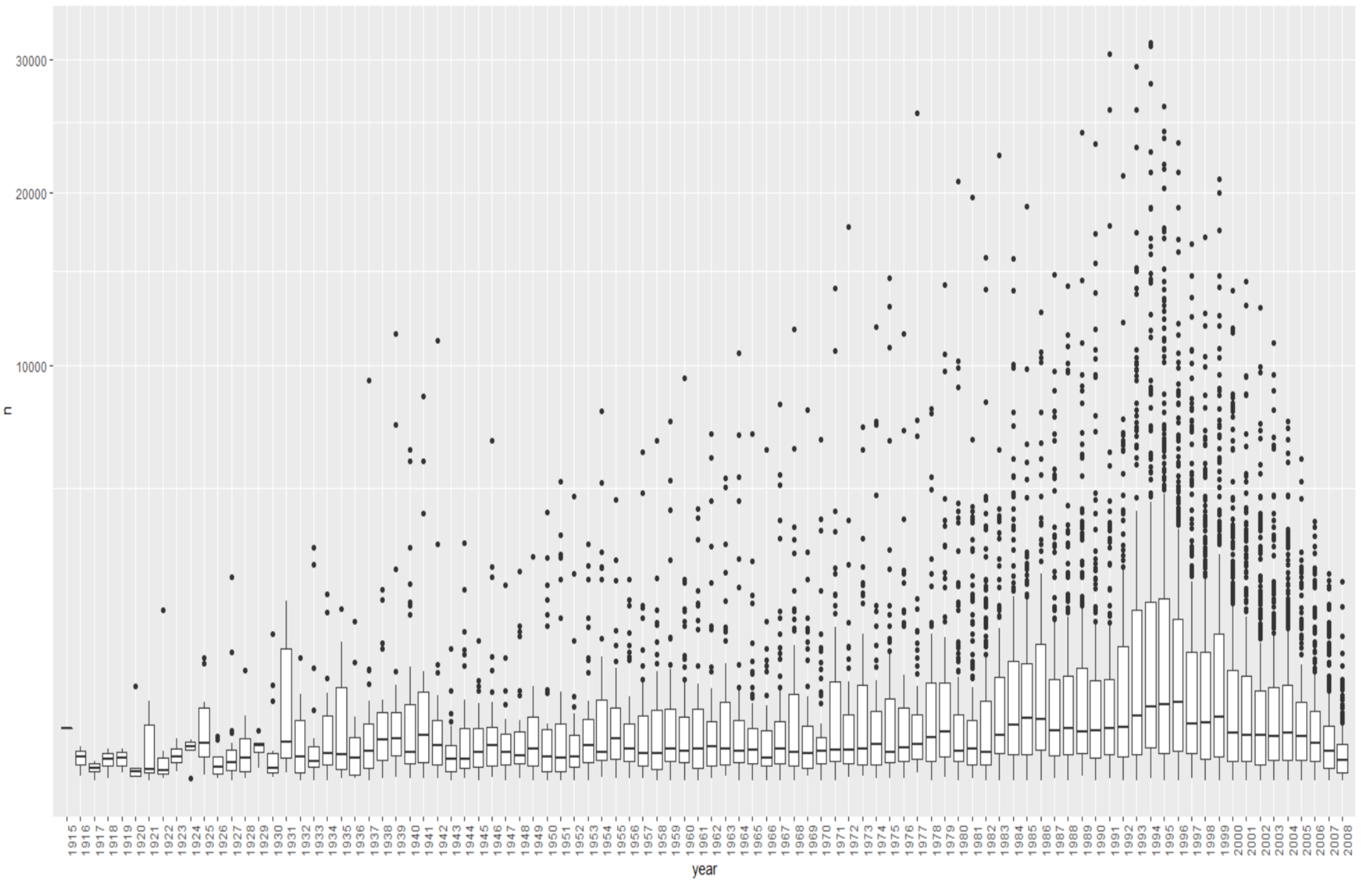
```
validation <- mutate(validation,year =as.numeric(substr(validation$title,nchar(validation$title) -4,nchar(validation$title)-1)))
```

Then, let's compute the number of ratings for each movie and then plot it against the year the movie came out. Using the square root transformation on the counts.

```
> edx %>% group_by(movieId) %>%
+ summarize(n = n(), year = as.character(first(year))) %>%
+ qplot(year, n, data = ., geom = "boxplot") +
+ coord_trans(y = "sqrt") +
+ theme(axis.text.x = element_text(angle = 90, hjust = 1))
```

From the plot, you can see that the year with the highest median number of ratings is 1995.

We see that, on average, movies that came out after 1993 get more ratings. We also see that with newer movies, starting in 1993, the number of ratings decreases with year: the more recent a movie is, the less time users have had to rate it.

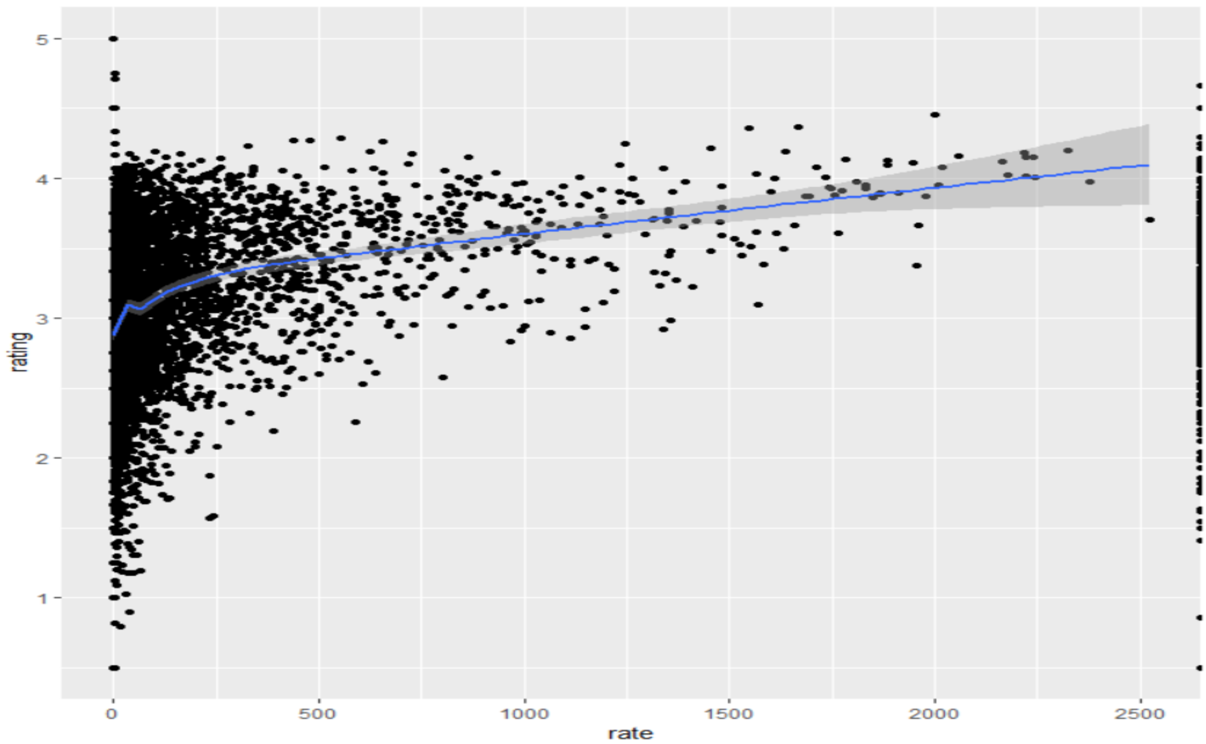


The top 15 movies with the most ratings per year, along with their average ratings, can be found using the following code:

```
> edx %>%
+   filter(year >= 1993) %>%
+   group_by(movieId) %>%
+   summarize(n = n(), years = 2018 - first(year),
+             title = title[1],
+             rating = mean(rating)) %>%
+   mutate(rate = n/years) %>%
+   top_n(25, rate) %>%
+   arrange(desc(rate))
# A tibble: 25 x 6
   movieId     n years title                                          rating  rate
     <dbl> <int> <dbl> <chr>                                           <dbl> <dbl>
 1     296 31362    24 Pulp Fiction (1994)                              4.15 1307.
 2     356 31079    24 Forrest Gump (1994)                              4.01 1295.
 3     480 29360    25 Jurassic Park (1993)                             3.66 1174.
 4     318 28015    24 Shawshank Redemption, The (1994)                 4.46 1167.
 5     110 26212    23 Braveheart (1995)                                4.08 1140.
 6    2571 20908    19 Matrix, The (1999)                               4.20 1100.
 7     780 23449    22 Independence Day (a.k.a. ID4) (1996)             3.38 1066.
 8     150 24284    23 Apollo 13 (1995)                                 3.89 1056.
 9    2858 19950    19 American Beauty (1999)                           4.19 1050
10     457 25998    25 Fugitive, The (1993)                             4.01 1040.
# ... with 15 more rows
> |
```

From the table constructed previously, we can see that the most frequently rated movies tend to have above average ratings. This is not surprising: more people watch popular movies. To confirm this, stratify the post-1993 movies by ratings per year and compute their average ratings. Make a plot of average rating versus ratings per year and show an estimate of the trend.

```
> edx %>%
+   filter(year >= 1993) %>%
+   group_by(movieId) %>%
+   summarize(n = n(), years = 2008 - first(year),
+             title = title[1],
+             rating = mean(rating)) %>%
+   mutate(rate = n/years) %>%
+   ggplot(aes(rate, rating)) +
+   geom_point() +
+   geom_smooth()
```

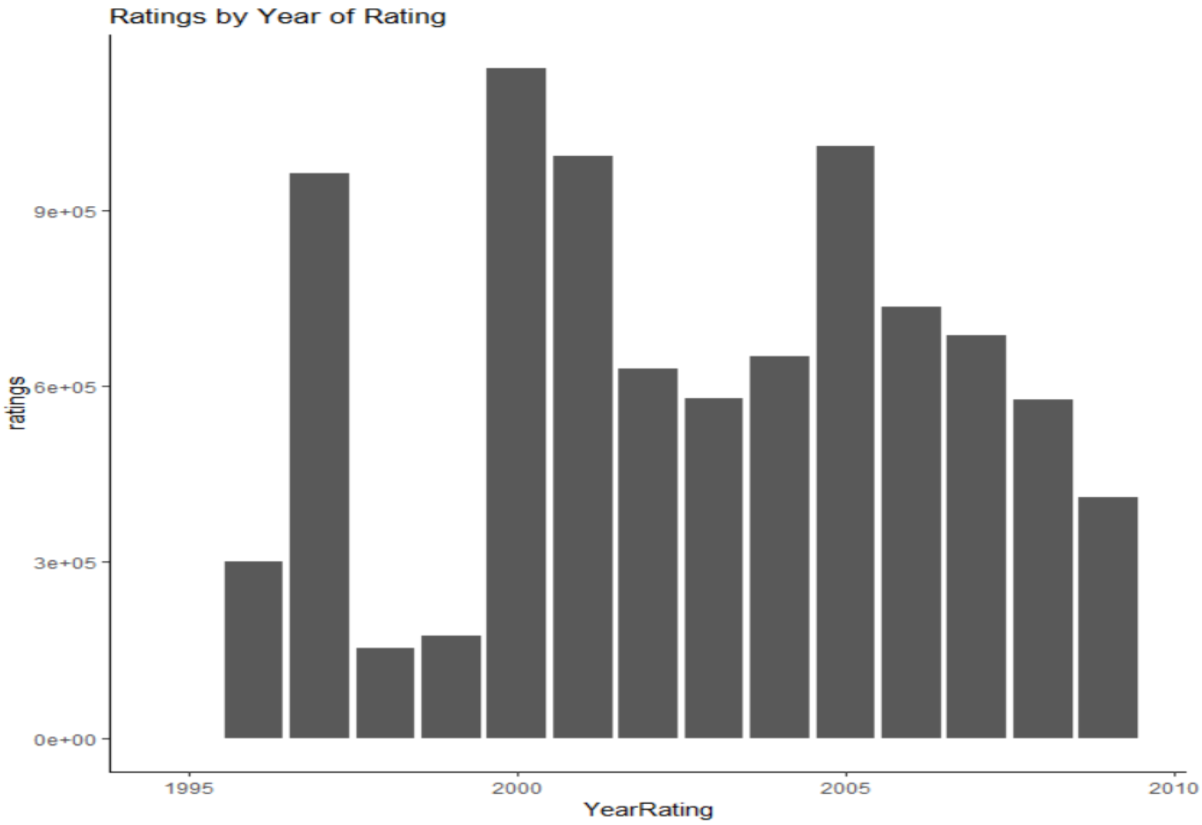


We see that the trend is that the more often a movie is rated, the higher its average rating.

We can notice that there is some evidence of a time effect in the plot, but there is not a strong effect of time.

Rating by year of rating

```
> edx <- mutate(edx, YearRating = round_date(timestamp, unit = "year"))
>  edx %>% group_by(YearRating) %>% summarize(ratings= n()) %>%
+   ggplot(aes(YearRating, ratings)) +
+   geom_bar(stat = "identity") +
+   theme_classic() +
+   ggtitle("Ratings by Year of Rating")
> 
```

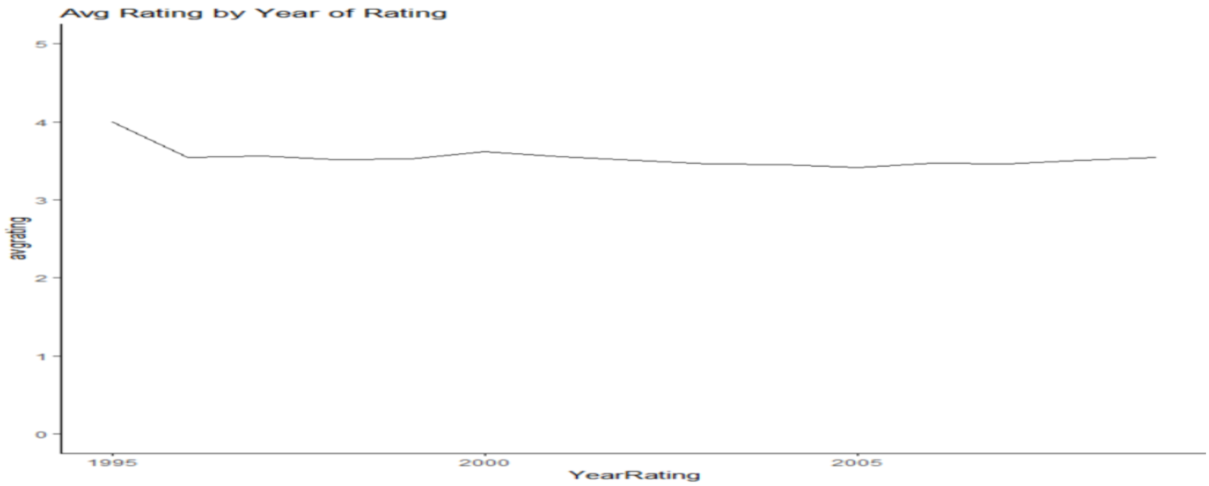


This shows that more ratings were given in the 2000s, though the ratings are distributed across years.

Let’s finally see if there is a difference in the average rating given by time

Average rating by Year of rating

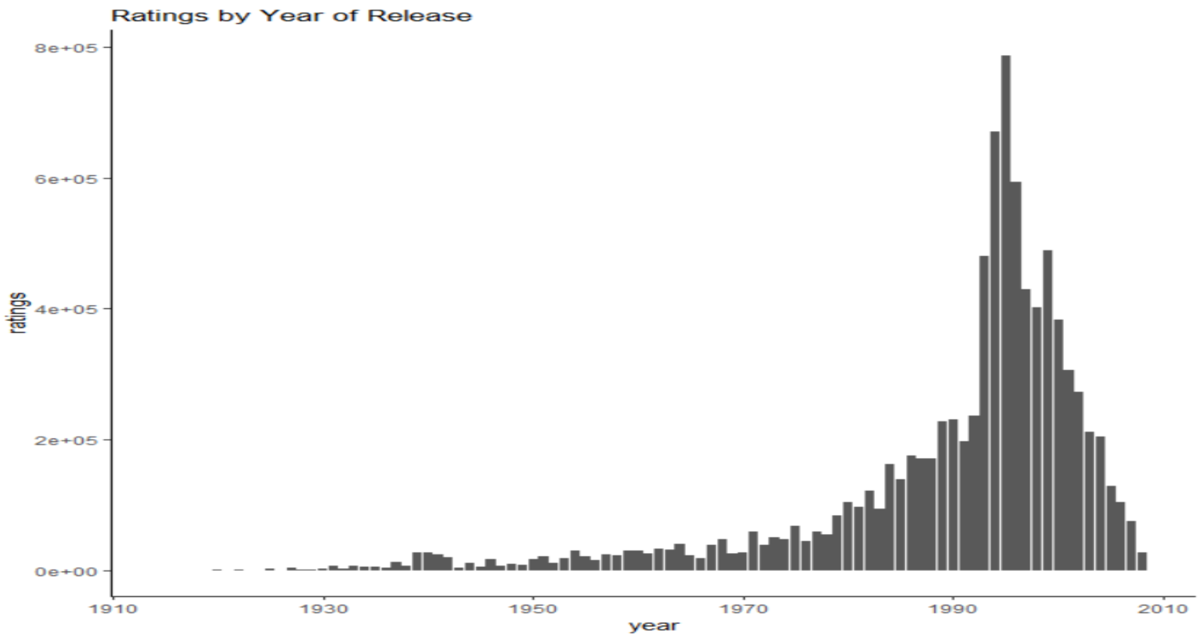
```
> edx %>% group_by(YearRating) %>% summarize(avgrating = mean(rating)) %>%
+
+   ggplot(aes(YearRating, avgrating)) +
+    geom_line() +
+    ylim(c(0,5)) +
+    theme_classic() +
+    ggtitle("Avg Rating by Year of Rating")
```



This chart shows that there is a variation in average rating with time, though this is a very small variation

Rating by year of release

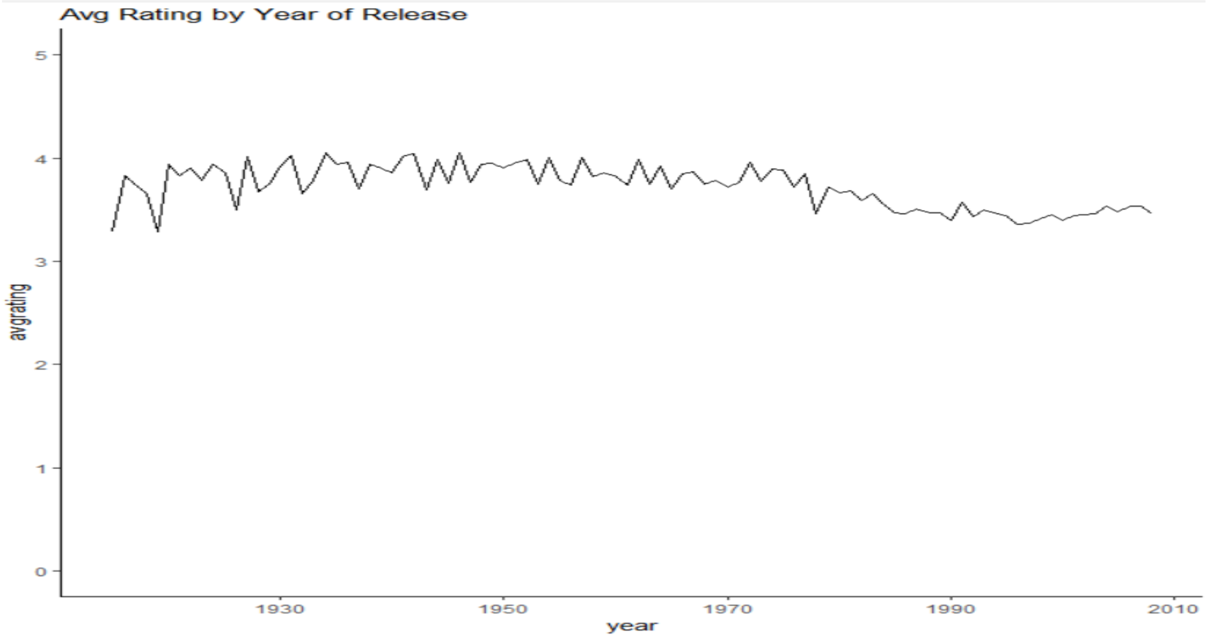
```
> edx %>% group_by(year) %>% summarize(avgrating = mean(rating))  %>%
+    ggplot(aes(year, avgrating)) +
+    geom_line() +
+    ylim(c(0,5)) +
+    theme_classic() +
+    ggtitle("Avg Rating by Year of Release")
```



Thus, we see that there are a lot more ratings given to movies released post mid-1990s. This makes sense since this dataset has ratings given from 1995 onwards, and there would be more ratings expected by users for current movies than for older movies.

Average rating by year of release

```
> edx %>% group_by(YearRating) %>% summarize(avgrating = mean(rating)) %>%
+
+   ggplot(aes(YearRating, avgrating)) +
+     geom_line() +
+     ylim(c(0,5)) +
+     theme_classic() +
+     ggtitle("Avg Rating by Year of Rating")
```



This chart shows that there is a variation in average rating by year of release of the movie, with a drop in average rating for movies released later. However, this variation is small.

4. Methods and analysis

In this section, we are going to explain the methodology over different Machine Learning algorithms we used and present the metric for the model performance evaluation.

According to the above analyzes, we will limit ourselves to three effects:

- User
- Movies
- Time

Regression Models

- **Modelling effects**

As in Irizarry,R 2018 *Recommender systems*,github page,accessed 5 January 2019, https://rafalab.github.io/dsbook/recommendation-systems.html, we followed the same approach to build our linear regression models as the simplest possible recommendation systems. We started from considering the same rating for all movies and users with all the differences explained by random variation Yu,i=μ+εu,iYu,i=μ+εu,i and thus, modelling successively the different effects.

movie effects: since we know that some movies are generally rated higher than others, we can augment our previous model by adding the term bi to represent average ranking for movie ii:

$$Y_{u,i} = \mu + b_i + \varepsilon_{u,i}(1)$$

where:

° μ the "true" rating for all movies
° bi effects or bias, movie-specific effect.
° εu,i independent errors sampled from the same distribution centered at 0

movie + user effects: We also know that some users are more active than others at rating movies. This implies that an additional improvement to our model may be:

$$Y_{u,i} = \mu + b_i + b_u + \varepsilon_{u,i}(2)$$

where :

° μ, bi , εu,i are defined as in (1)
° bu user-specific effect

movie + user + time effects. As in data exploration we showed some evidence of time effect, if we define with du,i as the day for user's u rating of movie i the new model is the following :

$$Y_{u,i} = \mu + b_i + b_u + f(d_{u,i}) + \varepsilon_{u,i} (3)$$

We will use a simple regression model using the two most correlates variables to rating: user and movie. Those two variables will be sufficient to get a good RMSE.

For user *u* and movie *i*, our regression function will be:

Y(u,i) = avgRating + avgRatingI + avgRatingU

With :

avgRating : all average rating

avgRatingI : bias rating for movie i

avgRatingU : bias rating for user u

III. Results

We used below code to train our model on training set and predict records in validation set which lead to a RMSE of: 0.8653488

```
library(Metrics)
library(tidyverse)
library(caret)


edx <- readRDS("C:/Users/mamadi.fofana/Desktop/FOAD/Harvard Data Science/HarvardX_Capstone_MovieLens/edx.rds")
validation <- readRDS("C:/Users/mamadi.fofana/Desktop/FOAD/Harvard Data Science/HarvardX_Capstone_MovieLens/validation.rds")


# avgRating get the average of all ratngs of the trainng set
avgRating <- mean(edx$rating)

# movieAVG get bias rating for each movie on the training set
movieAVG <- edx %>%
  group_by(movieId) %>%
  summarize(avgRatingI = mean(rating - avgRating))

#userAVG get bias rating for each user on the training set
userAVG <- edx %>%
  left_join(movieAVG, by='movieId') %>%
  group_by(userId) %>%
  summarize(avgRatingU = mean(rating - avgRating - avgRatingI))

#Predicted ratings on validation set
predictedRatings <- validation %>%
  left_join(movieAVG, by='movieId') %>%
  left_join(userAVG, by='userId') %>%
  mutate(pred = avgRating + avgRatingI + avgRatingU) %>%
  .$pred

#rmse get root mean squere errors on validation test
rmse <- rmse(validation$rating,predictedRatings)
rmse
```

IV. Conclusion

Analysing the MovieLens dataset gave many interesting insights into the movie business during data exploration.

Our baseline regression model got a RMSE 0.8653488 on validation data which is already a good score.

However, there are possibilities to improve that model adding regularization on our model or adding another interesting variable (Timestamp etc).

We have also possibility to use recommender engine or ensemble methods to go further in our analysis.