HarvardX-Capstone Project: MovieLens

Jocelyn Carmen Dumlao
16/04/2021

Introduction

This report is part of the Data Science capstone project of the edX course 'HarvardX. The goal is to demonstrate that the student acquired skills with the R programming language in the field of data science to actually solve real-world problems. The task is to analyse a dataset called 'MovieLens' which contains of millions of movie ratings by users. The insights from this analysis are used to generate predictions of movies. By developing algorithm using the 'edx set', the generated predicted results are compared with the actual ratings to check the quality of the algorithm. For a final test of my final algorithm, the expected movie ratings from the validation set (the final hold-out test set) as if they were unknown will be assessed using Residual Mean Square Error (RMSE) to determine the closeness of true values from the train set.

The full MovieLens dataset is relatively large, which can be found here: "http://files.grouplens.org/datasets/movielens/ml-10m.zip". I explored the "10Million version" of MovieLens dataset to make the computation easier for this project.

After investigating the movies through graphical representations and calculating RMSE, I found the best ratings predictor was 'movieId',' userId'. The results were a decrease in the Residual Mean Square Error (RMSE) which yield a more accurate prediction of future film ratings.

The final hold-out test set- RMSE is : **0.86490**

Executive Summary

The report is composed of three (3) general sections and a conclusion. First, the dataset load and preformat for further analysis. Second, the exploratory data analysis which helps to understand the dataset structure. Third, develop and employ machine learning algorithm to generate predictions which are then exported for a final test (final hold-out test set). An initial Exploratory Data Analysis of the 10Million version dataset was done using our set of quiz questions provided by edx. The plot indicates strong correlation of user, movie and genre as calculated by Residual Mean Square Error (RMSE). The conclusion is also presented.

Methodology

At first, I downloaded the 10 Million version of "MovieLens" dataset and organized into format that is ready to explore. There are some steps to perform for the data analysing:

(1) set the MovieLens data validation at 10%, (2)make sure 'userId' and 'movieId' were included in edx set, (3) Add rows removed from validation set back into edx set, (4) perform a Data Cleaning, Data Exploration and Data Visualization, (5) calculate and evaluate movieId, userId, using age and year rating average, (6) -Age of movies, genres, userId effects using the validation set, and we could saw some graphics, tibbles,Discription and their results, (7) evaluate algorithm performance by using the Residual Mean Square Error (RMSE) and compare forecasting error of different models for a particular dataset (the final hold-out test set) and (8) calculate movie rating predictions and compare to the true ratings in the train set and validation set (the final hold-out test set) by using Residual Mean Square Error (RMSE) where a penalty lambda is taken into account.

Once a prediction is made, transform the prediction into a continuous number from 0.5 to 5.0 scales. The accuracy is determined by comparing the derived prediction and actual values.

A. Dataset

A.1. Download the MovieLens dataset.

The dataset 'movielens' gets split into two parts, namely: (1) training- test set called 'edx' and (2) validation set (final hold-out test set).

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")
if(!require(caret)) install.packages("caret", repos = "http://cran.us.r-proje
ct.org")
if(!require(data.table)) install.packages("data.table", repos = "http://cran.
us.r-project")
require(corrplot)
library(finalfit)
library(tidyverse)
library(caret)
library(data.table)
library(lubridate)
library(stringr)
library(kableExtra)
library(RColorBrewer)
library(purrr)
library(Matrix)
library(caret)
library(ggthemes)
library(finalfit)

dl <- tempfile()
download.file("http://files.grouplens.org/datasets/movielens/ml-10m.zip", dl)
```

```
ratings <- read.table(text = gsub("::", "\t", readLines(unzip( dl,   "ml-10M1
00K/ratings.dat"))),col.names = c("userId", "movieId", "rating", "timestamp")
)
```

I use the following code to download the dataset movielens

A.2 Build the Dataset

```
movies <- str_split_fixed(readLines(unzip(dl, "ml-10M100K/movies.dat")), "\\:
:", 3) colnames(movies) <- c("movieId", "title", "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))
```

A.2.a. Explore the size of Dataset

```
movielens <- left_join(ratings, movies, by = "movieId")
nrow(movielens)# 10000054

n_distinct(movielens$movieId)# 10677

n_distinct(movielens$genres)# 797

n_distinct(movielens$userId) # 69878
```

A.2.b. Set the MovieLens data Validation @ 10%

```
#Validation set will be 10% of MovieLens data
set.seed (1, sample.kind = "Rounding")
test_index <- createDataPartition(y = movielens$rating, times = 1, p = 0.1, l
ist = FALSE)
edx <- movielens[-test_index,]
temp <- movielens[test_index,]
```

A.2.c. Make sure' UserID' and 'MovieID' are included in Validation Set

```
#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")
```

A.2.d. 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)

A.2.e. Data Cleaning, Data Exploration and Data Visualization

In order to determine if age of the movie is a factor for predicting rating, the age of the movie is calculated against the genres effect and user ratings effect.

B. Exploratory Data Analysis

To get familiar with the dataset, the first rows of edx set as below. Which contain the six (6) variables “userId”, “movieId”, “rating”, “timestamp”, “title”, and “genres”. Each row represent a single rating of a user for a single movie.

B.1 -“UserId”,“MovieId”, “Ratings”

```
head(edx) %>%
  kbl() %>%
  kable_styling()
```

	userId	movieId	rating	timestamp	title	genres	year_rated
1	1	122	5	838985046	Boomerang (1992)	Comedy|Romance	1996
2	1	185	5	838983525	Net, The (1995)	Action|Crime|Thriller	1996
4	1	292	5	838983421	Outbreak (1995)	Action|Drama|Sci-Fi|Thriller	1996
5	1	316	5	838983392	Stargate (1994)	Action|Adventure|Sci-Fi	1996
6	1	329	5	838983392	Star Trek: Generations (1994)	Action|Adventure|Drama|Sci-Fi	1996
7	1	355	5	838984474	Flintstones, The (1994)	Children|Comedy|Fantasy	1996

Description df of edx Summary

A summary of the subset to confirm that there are no missing values.

```
summary(edx)%>%
  kbl() %>%
  kable_styling()
```

userId	movieId	rating	timestamp	title	genres	year_rated
Min. : 1	Min. : 1	Min. :0.500	Min. :7.897e+08	Length:9000055	Length:9000055	Min. :1995
1st Qu.:18124	1st Qu.: 648	1st Qu.:3.000	1st Qu.:9.468e+08	Class :character	Class :character	1st Qu.:2000
Median :35738	Median : 1834	Median :4.000	Median :1.035e+09	Mode :character	Mode :character	Median :2002
Mean :35870	Mean : 4122	Mean :3.512	Mean :1.033e+09	NA	NA	Mean :2002
3rd Qu.:53607	3rd Qu.: 3626	3rd Qu.:4.000	3rd Qu.:1.127e+09	NA	NA	3rd Qu.:2005
Max. :71567	Max. :65133	Max. :5.000	Max. :1.231e+09	NA	NA	Max. :2009

Number of Unique Movies and User in the edx dataset

```
edx %>%
  summarise(n_users = n_distinct(userId),
            n_movies = n_distinct(movieId))%>%
  kbl()%>%
  kable_styling()
```

n_users	n_movies
69878	10677

User have a preference to rate movies rather higher than lower as shown by the distribution of rating below.

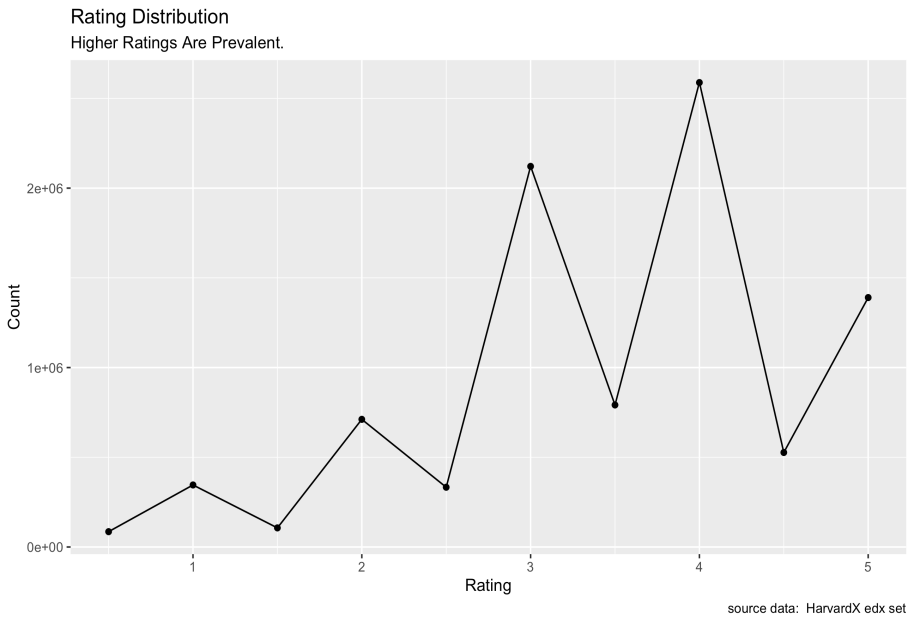
Tibble of Count the Number of Ratings

```
edx %>%
  group_by(rating) %>%
  summarize (n =n ())%>%
  kbl()%>%
  kable_styling()
```

rating	n
0.5	85374
1.0	345679
1.5	106426
2.0	711422
2.5	333010
3.0	2121240
3.5	791624
4.0	2588430
4.5	526736
5.0	1390114

Line Graph of Rating Distribution

```
edx %>%
  group_by(rating) %>%
  summarize(count =n ()) %>%
  ggplot(aes (x= rating, y=count)) +
  geom_line() +
  geom_point() +
  ggtitle ("Rating Distribution", subtitle = "Higher Ratings Are Prevalent.")
+
  labs( x="Rating", y = "Count", caption = "source data:  HarvardX edx set")
```



B.2 Data Structure of the Validation Set

```
glimpse(edx)
```

Rows: 9,000,055
Columns: 7
$ userId <int> 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 2, 2, 2, 2, 2, 2,...
$ movieId <dbl> 122, 185, 292, 316, 329, 355, 356, 362, 364, 370, 377, 420, 466, 520, 539,...
$ rating <dbl> 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 3, 3, 5, 2,...
$ timestamp <int> 838985046, 838983525, 838983421, 838983392, 838983392, 838984474, 83898365...
$ title <chr> "Boomerang (1992)", "Net, The (1995)", "Outbreak (1995)", "Stargate (1994)...
$ genres <chr> "Comedy|Romance", "Action|Crime|Thriller", "Action|Drama|Sci-Fi|Thriller",...
$ year_rated <dbl> 1996, 1996, 1996, 1996, 1996, 1996, 1996, 1996, 1996, 1996, 1996, 1996, 19...

Its class data.frame: with 999,999 obs. of 6 variable: its exactly the 10% of our validation set, and has the same 6 features.

A tibble of Number of Rating/top 10

```
numRating <- edx %>%
  group_by(movieId) %>%
  summarise(numRatings = n(), movieTitle =first(title))%>%
  arrange(desc(numRatings)) %>%
  top_n(10, numRatings)

edx %>%
  group_by(title) %>%
  summarise(number = n()) %>%
  arrange(desc(number))%>%
```

```
  kbl()%>%
kable_styling()
```

title	number
Pulp Fiction (1994)	31362
Forrest Gump (1994)	31079
Silence of the Lambs, The (1991)	30382
Jurassic Park (1993)	29360
Shawshank Redemption, The (1994)	28015
Braveheart (1995)	26212
Fugitive, The (1993)	25998
Terminator 2: Judgment Day (1991)	25984
Star Wars: Episode IV - A New Hope (a.k.a. Star Wars) (1977)	25672
Apollo 13 (1995)	24284
Batman (1989)	24277
Toy Story (1995)	23790

The Pulp Fiction (1994) movie has the greatest number of ratings.

```
head(sort(-table(edx$rating)),5)%>%
  kbl() %>%
  kable_styling()
```

Var1	Freq
4	-2588430
3	-2121240
5	-1390114
3.5	-791624
2	-711422

4,3,5,3.5,2 are the five most given ratings in order from most to least.

```
table(edx$rating)%>%
  kbl() %>%
  kable_styling()
```

Var1	Freq
0.5	85374
1	345679
1.5	106426
2	711422
2.5	333010
3	2121240
3.5	791624
4	2588430
4.5	526736
5	1390114

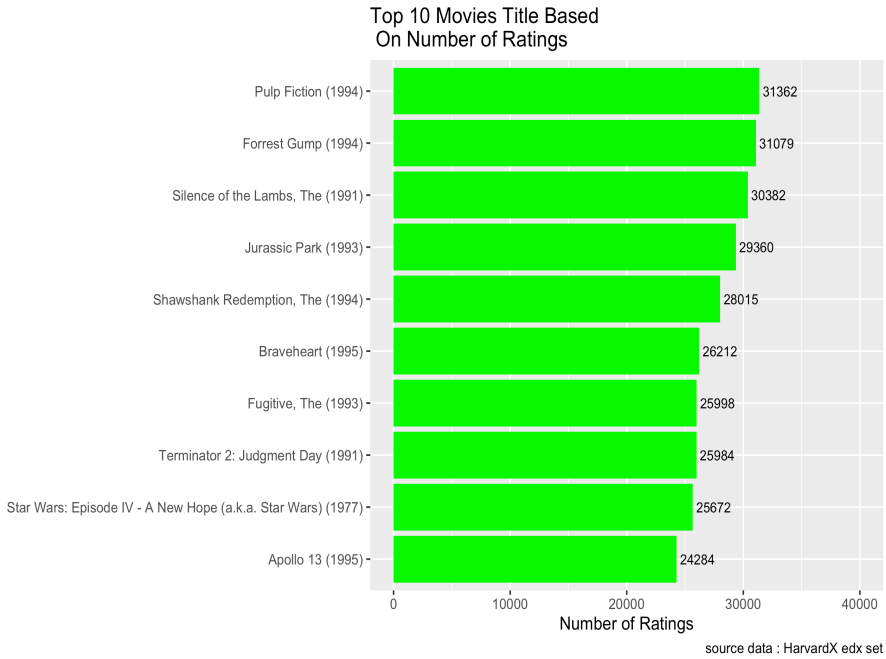
A Tibble of Top 10 Movies- Number of Ratings The data frame top_title contains the top 10 movies which count the major number of ratings

```
top_title <- edx %>%
  group_by(title) %>%
  summarize (count = n ()) %>%
  top_n (10, count) %>%
  arrange(desc (count))

# with the head function i output the top 5
kable(head(edx %>%
             group_by(title, genres) %>%
             summarize(count = n ( )) %>%
             top_n(10,count) %>%
             arrange(desc (count)), 10)) %>%
  kable_styling(bootstrap_options = "bordered", full_width = F, position = "c
enter") %>%
  column_spec(1, bold = T)%>%
  column_spec(2, bold = T) %>%
  column_spec(3, bold = T)
```

title	genres	count
Pulp Fiction (1994)	Comedy|Crime|Drama	31362
Forrest Gump (1994)	Comedy|Drama|Romance|War	31079
Silence of the Lambs, The (1991)	Crime|Horror|Thriller	30382
Jurassic Park (1993)	Action|Adventure|Sci-Fi|Thriller	29360
Shawshank Redemption, The (1994)	Drama	28015
Braveheart (1995)	Action|Drama|War	26212
Fugitive, The (1993)	Thriller	25998
Terminator 2: Judgment Day (1991)	Action|Sci-Fi	25984
Star Wars: Episode IV - A New Hope (a.k.a. Star Wars) (1977)	Action|Adventure|Sci-Fi	25672
Apollo 13 (1995)	Adventure|Drama	24284

Bar Chart of Top 10 Movies Number of Ratings/ top_title

```
top_title %>%
  ggplot(aes(x=reorder(title,count), y=count)) +
  geom_bar (stat='identity', fill="green") +
  coord_flip( y=c(0, 40000)) +
  labs( x = "", y = "Number of Ratings") +
  geom_text (aes(label = count), hjust = -0.1, size =3) +
  labs(titles = "Top 10 Movies Title Based \n On Number of Ratings", caption
= "source data : HarvardX edx set")
```



The movies which have the highest number of ratings are in the top genres categories: movies like 1. Pulp fiction (1994), 2.Forrest Gump(1994), 3.Jurrasic Park (1993), 4.Shawshank Redemption, The (1994), 5. Braveheart (1995), 6. Fugitive, The (1993), 7, Terminator 2: Judgment Day (1991), 8. Star Wars: 9. Episode IV - A New Hope (a.k.a. Star Wars) (1977), 10. Apollo 13 (1995) which are in the top 10 of movies ratings number, are part of the Drama, Comedy or Action genres.

B.3 Distinct Movies, Users and Genres

How many distinct movie, users and generes

```
dim(edx) #9000055           6
n_distinct(edx$movieId) # 10677
n_distinct(edx$title)  # 10676
n_distinct(edx$userId)   #69878
n_distinct(edx$movieId) * n_distinct(edx$userId)     # 746087406
n_distinct(edx$movieId) * n_distinct(edx$userId)/ dim(edx) [ 1 ]  # 83
```

As shown above, the edx dataset had **10677** distinct movies, **69878** distinct users, **10676** distinct title : there might be movies of different IDs with the same title,we have 10677 x 69878 = **746087406** this would be the ratings.

User There are **69878** different users are in the edx set. The majority of users rate few movies, while a few users rate more than a thousand movies 5% users rated less than 20 movies.

A Tibble of UserId

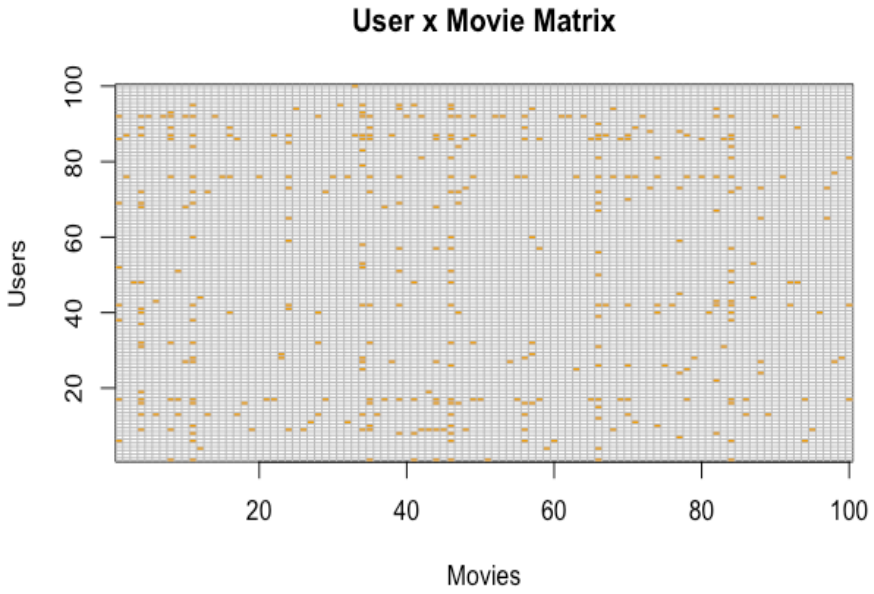
```
edx %>%
  group_by(userId) %>%
  summarize(n = n()) %>%
  arrange (n) %>%
  head()%>%
  kable()%>%
  kable_styling()
```

userId	n
62516	10
22170	12
15719	13
50608	13
901	14
1833	14

B.4. Scatter Plot of User x Movies Matrix

Matrix for a random sample of 100 movies and 100 users with yellow Indicating a user/movie combination for which we have a rating.

```
users <- sample(unique(edx$userId), 100)
edx %>%
  filter(userId %in% users) %>%
  select(userId, movieId,rating) %>%
  mutate(rating = 1) %>%
  spread(movieId, rating) %>%
  select(sample(ncol(.), 100)) %>%
  as.matrix() %>%
  t(.) %>%
  image(1:100, 1:100, . , xlab = "Movies", ylab = "Users")
abline(h = 0:100 + 0.5, v=0:100 + 0.5, col= "grey")
title("User x Movie Matrix")
```



Description df of Extracting the Year Released

```
#Extract the year_released

year_released <- stringi::stri_extract(edx$title,regex = "(\\d{4})", comments
=TRUE) %>%
  as.numeric()

#add the year_released
edx_with_title_dates <- edx %>%
  mutate(year_released = year_released)
head(edx_with_title_dates)%>%
  kbl() %>%
  kable_styling()
```

	userId	movieId	rating	timestamp	title	genres	year_rated	year_released
1	1	122	5	838985046	Boomerang (1992)	Comedy|Romance	1996	1992
2	1	185	5	838983525	Net, The (1995)	Action|Crime|Thriller	1996	1995
4	1	292	5	838983421	Outbreak (1995)	Action|Drama|Sci-Fi|Thriller	1996	1995
5	1	316	5	838983392	Stargate (1994)	Action|Adventure|Sci-Fi	1996	1994
6	1	329	5	838983392	Star Trek: Generations (1994)	Action|Adventure|Drama|Sci-Fi	1996	1994
7	1	355	5	838984474	Flintstones, The (1994)	Children|Comedy|Fantasy	1996	1994

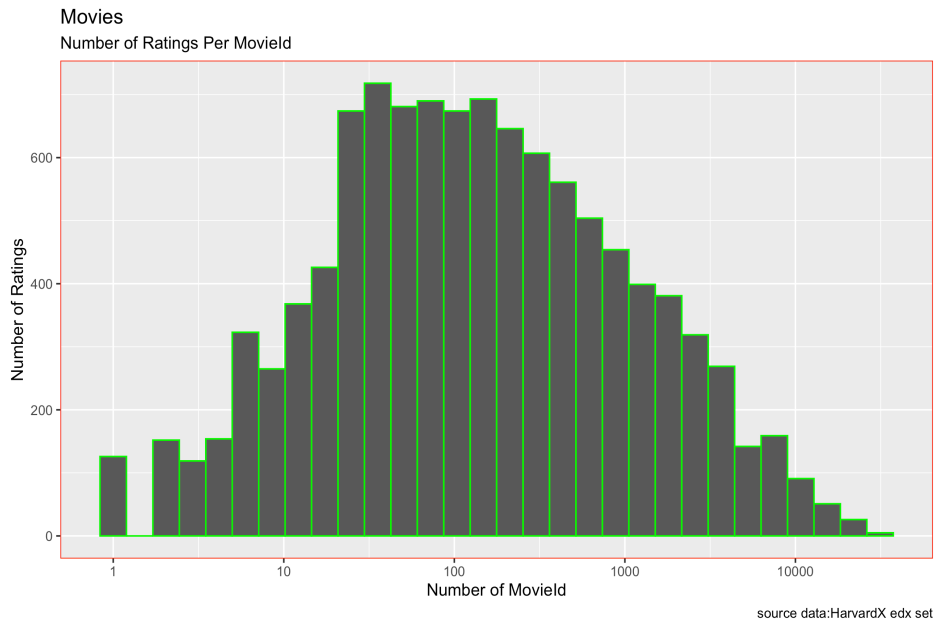
Description df of Extracting the Age at Rating

```
#Extract the age_at_rating

age_at_rating <- stringi::stri_extract(edx$title,regex = "(\\d{4})", comments
=TRUE) %>%
  as.numeric()

#add the age_at_rating
edx_with_title_dates <- edx %>%
  mutate(age_at_rating= rating)
head(edx_with_title_dates)%>%
  kbl()%>%
  kable_styling()
```

	userId	movieId	rating	timestamp	title	genres	year_rated	age_at_rating
1	1	122	5	838985046	Boomerang (1992)	Comedy|Romance	1996	5
2	1	185	5	838983525	Net, The (1995)	Action|Crime|Thriller	1996	5
4	1	292	5	838983421	Outbreak (1995)	Action|Drama|Sci-Fi|Thriller	1996	5
5	1	316	5	838983392	Stargate (1994)	Action|Adventure|Sci-Fi	1996	5
6	1	329	5	838983392	Star Trek: Generations (1994)	Action|Adventure|Drama|Sci-Fi	1996	5
7	1	355	5	838984474	Flintstones, The (1994)	Children|Comedy|Fantasy	1996	5

A Histogram of Number of Ratings Per MovieId

```
edx %>%
  count(movieId)%>%
  ggplot(aes(n)) +
  geom_histogram(bins = 30, color = "Green")+
  scale_x_log10() +
  ggtitle ("Movies") +
  labs(subtitle = "Number of Ratings Per MovieId",
       x ="Number of MovieId",
       y ="Number of Ratings",caption ="source data:HarvardX edx set") +
  theme(panel.border = element_rect(colour =" red", fill = NA))
```

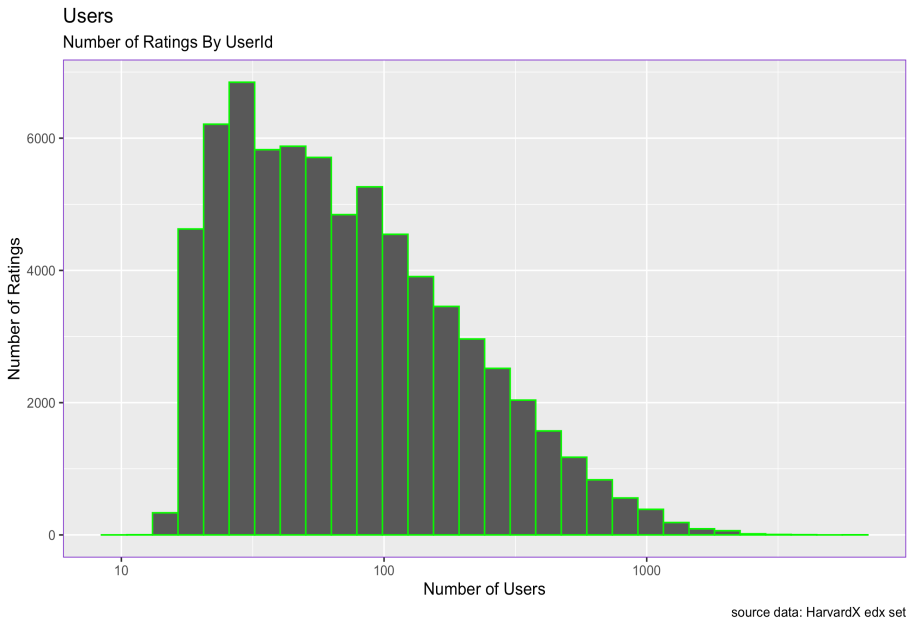


A Tibble of 20 Movies rated only once

```
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()
```

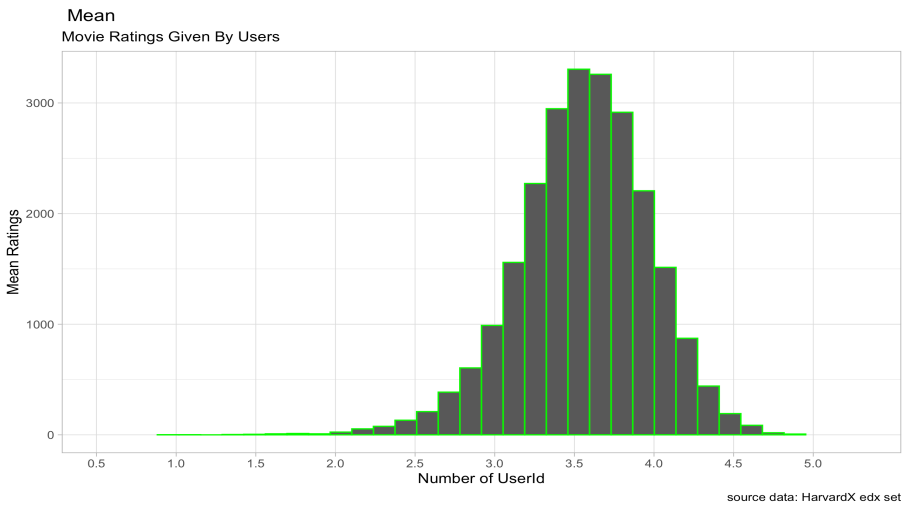
A Histogram of Number of Ratings by UserID

```
edx %>%
  count(userId)%>%
  ggplot(aes(n)) +
  geom_histogram(bins = 30, color = "Green")+
  scale_x_log10() +
  ggtitle("Users") +
  labs(subtitle="Number of Ratings By UserId",
       x = "Number of Users",
       y = "Number of Ratings",
       caption = "source data: HarvardX edx set") +
  theme (panel.border = element_rect(colour = "purple3", fill =NA))
```



A histogram of Mean Movie Rating Given By Users

```
edx %>%
  group_by (userId) %>%
  filter(n () >= 100) %>%
  summarize(b_u = mean(rating)) %>%
  ggplot(aes(b_u)) +
  geom_histogram(bins = 30, color = "Green") +
  labs(subtitle= "Movie Ratings Given By Users",x="Number of UserId", y= "Mea
n Ratings", caption = "source data: HarvardX edx set") +
  ggtitle(" Mean") +
  scale_x_discrete(limits = c (seq(0.5,5,0.5))) +
  theme_light( )
```



A Description df of Convert Timestamp to Year

```
# convert timestamp to year
edx  <- mutate(edx, year_rated =year(as_datetime(timestamp)))
head(edx)%>%
  kbl() %>%
  kable_styling()
```

	userId	movieId	rating	timestamp	title	genres	year_rated
1	1	122	5	838985046	Boomerang (1992)	Comedy|Romance	1996
2	1	185	5	838983525	Net, The (1995)	Action|Crime|Thriller	1996
4	1	292	5	838983421	Outbreak (1995)	Action|Drama|Sci-Fi|Thriller	1996
5	1	316	5	838983392	Stargate (1994)	Action|Adventure|Sci-Fi	1996
6	1	329	5	838983392	Star Trek: Generations (1994)	Action|Adventure|Drama|Sci-Fi	1996
7	1	355	5	838984474	Flintstones, The (1994)	Children|Comedy|Fantasy	1996

A Description df of Premier Date

```
#Extract the premier date and calculate the age of the movie.

premier <- stringi::stri_extract(edx$title,regex = "(\\d{4})", comments =TRUE
) %>%
  as.numeric()

#add the premier date
edx_with_title_dates <- edx %>%
  mutate(premier_date =premier)
```

```
head(edx_with_title_dates)%>%
  kbl() %>%
  kable_styling()
```

	userId	movieId	rating	timestamp	title	genres	year_rated	premier_date
1	1	122	5	838985046	Boomerang (1992)	Comedy|Romance	1996	1992
2	1	185	5	838983525	Net, The (1995)	Action|Crime|Thriller	1996	1995
4	1	292	5	838983421	Outbreak (1995)	Action|Drama|Sci-Fi|Thriller	1996	1995
5	1	316	5	838983392	Stargate (1994)	Action|Adventure|Sci-Fi	1996	1994
6	1	329	5	838983392	Star Trek: Generations (1994)	Action|Adventure|Drama|Sci-Fi	1996	1994
7	1	355	5	838984474	Flintstones, The (1994)	Children|Comedy|Fantasy	1996	1994

A Description df of drop the timestamp

```
#drop the timestamp
edx_with_title_dates <-edx_with_title_dates %>%
  select(-timestamp)
head(edx_with_title_dates)%>%
  kbl() %>%
  kable_styling()
```

	userId	movieId	rating	title	genres	year_rated	premier_date
1	1	122	5	Boomerang (1992)	Comedy|Romance	1996	1992
2	1	185	5	Net, The (1995)	Action|Crime|Thriller	1996	1995
4	1	292	5	Outbreak (1995)	Action|Drama|Sci-Fi|Thriller	1996	1995
5	1	316	5	Stargate (1994)	Action|Adventure|Sci-Fi	1996	1994
6	1	329	5	Star Trek: Generations (1994)	Action|Adventure|Drama|Sci-Fi	1996	1994
7	1	355	5	Flintstones, The (1994)	Children|Comedy|Fantasy	1996	1994

A Tibble of Premier Date > 2018

```
edx_with_title_dates %>%
  filter(premier_date > 2018) %>%
  group_by(movieId, title, premier_date) %>%
  summarize(n= n())%>%
  kbl() %>%
  kable_styling()
```

movieId	title	premier_date	n
671	Mystery Science Theater 3000: The Movie (1996)	3000	3280
2308	Detroit 9000 (1973)	9000	22
4159	3000 Miles to Graceland (2001)	3000	714
5310	Transylvania 6-5000 (1985)	5000	195
8864	Mr. 3000 (2004)	3000	146
27266	2046 (2004)	2046	426

A Tibble of Premier Date < 1900

```
edx_with_title_dates %>%
  filter(premier_date < 1900) %>%
  group_by(movieId, title, premier_date) %>%
  summarize(n= n())%>%
  kbl() %>%
  kable_styling()
```

movieId	title	premier_date	n
1422	Murder at 1600 (1997)	1600	1566
4311	Bloody Angels (1732 Høtten: Marerittet Har et Postnummer) (1998)	1732	9
5472	1776 (1972)	1776	185
6290	House of 1000 Corpses (2003)	1000	367
6645	THX 1138 (1971)	1138	464
8198	1000 Eyes of Dr. Mabuse, The (Tausend Augen des Dr. Mabuse, Die) (1960)	1000	24
8905	1492: Conquest of Paradise (1992)	1492	134
53953	1408 (2007)	1408	466

Fix the incorrect dates

```
edx_with_title_dates[edx_with_title_dates$movieId == "27266","premier_date"]
<- 2004
edx_with_title_dates[edx_with_title_dates$movieId == "671","premier_date"] <-
1996
edx_with_title_dates[edx_with_title_dates$movieId == "2308","premier_date"] <
- 1973
edx_with_title_dates[edx_with_title_dates$movieId == "4159","premier_date"] <
- 2001
```

```
edx_with_title_dates[edx_with_title_dates$movieId == "5310","premier_date"] <
- 1985
edx_with_title_dates[edx_with_title_dates$movieId == "8864","premier_date"] <
- 2004
edx_with_title_dates[edx_with_title_dates$movieId == "1422","premier_date"] <
- 1997
edx_with_title_dates[edx_with_title_dates$movieId == "4311","premier_date"] <
- 1998
edx_with_title_dates[edx_with_title_dates$movieId == "5472","premier_date"] <
- 1972
edx_with_title_dates[edx_with_title_dates$movieId == "6290","premier_date"] <
- 1998
edx_with_title_dates[edx_with_title_dates$movieId == "6645","premier_date"] <
- 1971
edx_with_title_dates[edx_with_title_dates$movieId == "8198","premier_date"] <
- 1960
edx_with_title_dates[edx_with_title_dates$movieId == "8905","premier_date"] <
- 1992
edx_with_title_dates[edx_with_title_dates$movieId == "53953","premier_date"]
<- 2007
```

Description df of 2018 Premier Date, Rating Date Range, Year Rated

Calculate the age of movie

```
edx_with_title_dates <- edx_with_title_dates %>%
  mutate(age_of_movie = 2018 -premier_date, rating_date_range = year_rated -
premier_date)
head(edx_with_title_dates)%>%
  kbl()%>%
  kable_styling()
```

	userId	movieId	rating	title	genres	year_rated	premier_date	age_of_movie	rating_date_range
1	1	122	5	Boomerang (1992)	Comedy|Romance	1996	1992	26	4
2	1	185	5	Net, The (1995)	Action|Crime|Thriller	1996	1995	23	1
4	1	292	5	Outbreak (1995)	Action|Drama|Sci-Fi|Thriller	1996	1995	23	1
5	1	316	5	Stargate (1994)	Action|Adventure|Sci-Fi	1996	1994	24	2
6	1	329	5	Star Trek: Generations (1994)	Action|Adventure|Drama|Sci-Fi	1996	1994	24	2
7	1	355	5	Flintstones, The (1994)	Children|Comedy|Fantasy	1996	1994	24	2

A Tibble of Movie Rating Average

```
movie_avgs <- edx_with_title_dates %>%
  group_by(movieId) %>%
  summarize(avg_movie_rating = mean(rating))
head(movie_avgs)%>%
  kbl() %>%
  kable_styling()
```

userId	avg_user_rating
1	5.000000
2	3.294118
3	3.935484
4	4.057143
5	3.918919
6	3.948718

A Tibble of UserId Average

```
user_avgs <- edx_with_title_dates %>%
  group_by(userId) %>%
  summarize(avg_user_rating = mean(rating))
head(user_avgs)%>%
  kbl() %>%
  kable_styling()
```

userId	movieId	rating	title	genres	year_rated	premier_date	age_of_movie	rating_date_range
1	122	5	Boomerang (1992)	Comedy	1996	1992	26	4
1	122	5	Boomerang (1992)	Romance	1996	1992	26	4
1	185	5	Net, The (1995)	Action	1996	1995	23	1
1	185	5	Net, The (1995)	Crime	1996	1995	23	1
1	185	5	Net, The (1995)	Thriller	1996	1995	23	1
1	292	5	Outbreak (1995)	Action	1996	1995	23	1

A Tibble of Year Average

```
year_avgs <- edx_with_title_dates %>%
  group_by(year_rated) %>%
```

```
  summarize(avg_rating_by_year = mean (rating))
head(year_avgs)%>%
  kbl() %>%
  kable_styling()
```

year_rated	avg_rating_by_year
1995	4.000000
1996	3.545286
1997	3.585703
1998	3.506144
1999	3.617354
2000	3.578044

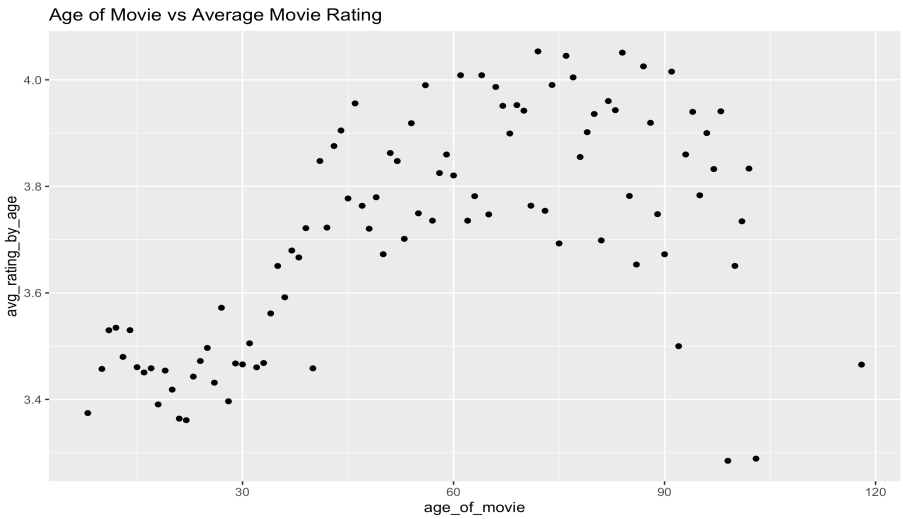
A Tibble of Age Average

```
age_avgs <- edx_with_title_dates %>%
  group_by(age_of_movie) %>%
  summarize(avg_rating_by_age = mean(rating)) # age of movie
head(age_avgs)%>%
  kbl() %>%
  kable_styling()
```

age_of_movie	avg_rating_by_age
8	3.374355
10	3.457238
11	3.529879
12	3.534620
13	3.479861
14	3.530074

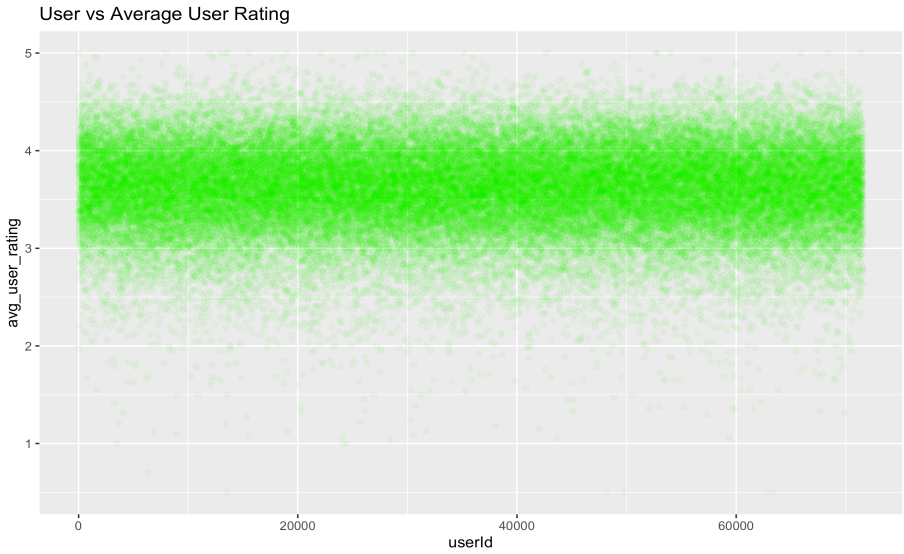
A Scatter Plot of Age of Movie Vs Average Movie Rating

```
age_avgs %>%
  ggplot(aes(age_of_movie,avg_rating_by_age)) +
  geom_point() +
  ggtitle("Age of Movie vs Average Movie Rating")
```



A Graph of User Vs Average User Rating

```
user_avgs %>%
  ggplot(aes(userId, avg_user_rating)) +
  geom_point(alpha = 1/20, colour = "green") +
  ggtitle("User vs Average User Rating")
```



As we can see the graph above the average ratings by user are pretty consistent between 2.5 and 4.5

B.4. Calculate the Linear Model Age of Movie

Calculate the lm of the age of a movie vs average rating

```
summary(lm(avg_rating_by_age ~ age_of_movie, data = age_avgs))
```

Call:

lm(formula = avg_rating_by_age ~ age_of_movie, data = age_avgs)

Residuals:
```
    Min      1Q  Median      3Q     Max
-0.61683 -0.10391  0.00274  0.12759  0.28507
```

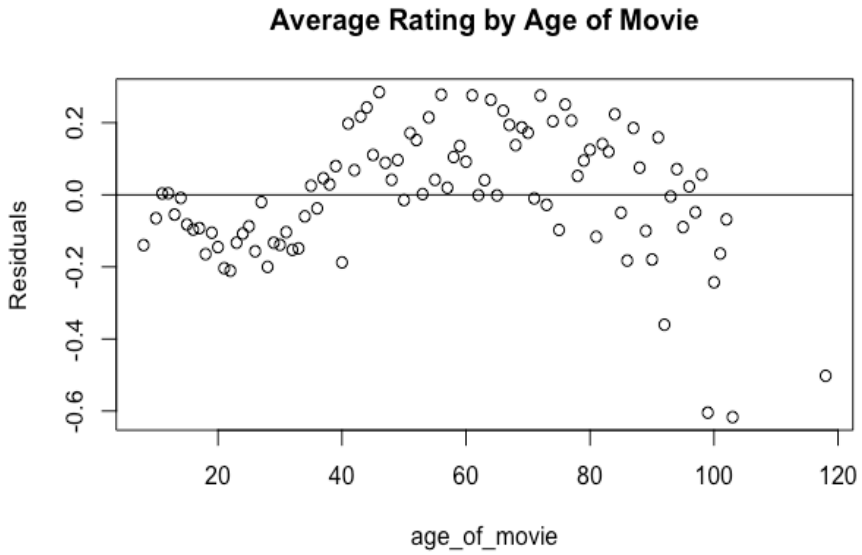
Coefficients:
```
             Estimate Std. Error t value Pr(>|t|)
(Intercept)  3.4809725  0.0409981  84.906  < 2e-16 ***
age_of_movie 0.0041237  0.0006489   6.355  7.4e-09 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

Residual standard error: 0.1781 on 94 degrees of freedom
Multiple R-squared: 0.3005,	Adjusted R-squared: 0.2931
F-statistic: 40.39 on 1 and 94 DF, p-value: 7.396e-09

we can saw that R-square is small at **0.30**

A Bubble Chart of Residual : Average Rating by Age of Movie

```
avg_rating.lm <- lm(avg_rating_by_age ~ age_of_movie, data = age_avgs)
avg_rating.res <-resid(avg_rating.lm)

plot(age_avgs$age_of_movie, avg_rating.res,
     ylab = 'Residuals', xlab = 'age_of_movie', main = 'Average Rating by Age
of Movie') +
  abline(0,0)
```



Average Rating Residual integer (0)

B.5. A Tibble of Genres Split The Data Into Single Genres

```
dat <- edx_with_title_dates %>%
       separate_rows(genres, sep ="\\|")
head (dat)%>%
  kbl()%>%
  kable_styling()
```

userId	movieId	rating	title	genres	year_rated	premier_date	age_of_movie	rating_date_range
1	122	5	Boomerang (1992)	Comedy	1996	1992	26	4
1	122	5	Boomerang (1992)	Romance	1996	1992	26	4
1	185	5	Net, The (1995)	Action	1996	1995	23	1
1	185	5	Net, The (1995)	Crime	1996	1995	23	1
1	185	5	Net, The (1995)	Thriller	1996	1995	23	1
1	292	5	Outbreak (1995)	Action	1996	1995	23	1

A Tibble of Count Number of Movies Using MovieId in Each Genres

```
genre_count_by_movieId <- dat %>%
    group_by(movieId, genres) %>%
    summarize ( n = n( ))
 head (genre_count_by_movieId)%>%
   kbl()%>%
   kable_styling()
```

movieId	genres	n
1	Adventure	23790
1	Animation	23790
1	Children	23790
1	Comedy	23790
1	Fantasy	23790
2	Adventure	10779

A Tibble of Genres List and Movie ratings are in each of the following genres in the edx dataset.

```
genres_list <- c( 'Drama', 'Comedy', 'Thriller', 'Romance')
genre_counts <- sapply (genres_list, function(g){
  edx %>%
    filter(str_detect(genres,g)) %>%
    tally()
})

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

```
  group_by(genres) %>%
  summarise(count = n())%>%
  kbl()%>%
kable_styling()
```

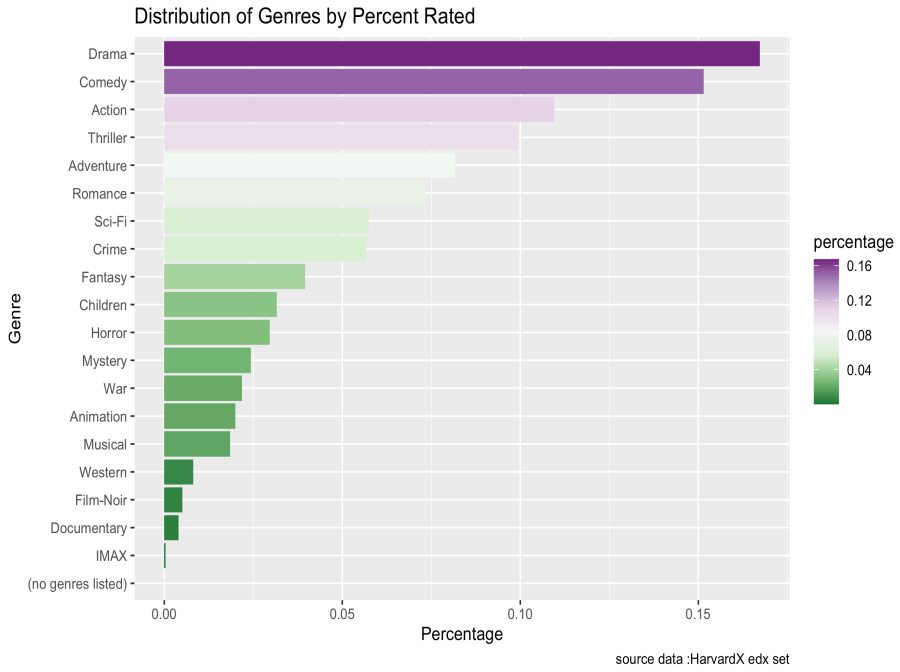
genres	count
(no genres listed)	7
Action	2560545
Adventure	1908892
Animation	467168
Children	737994
Comedy	3540930
Crime	1327715
Documentary	93066
Drama	3910127
Fantasy	925637
Film-Noir	118541
Horror	691485
IMAX	8181
Musical	433080

Bar Graph of Distribution of Genre by Percent Rated

Explore the distribution of ratings by genre

```
temp <- dat %>%
    group_by (genres) %>%
    summarize (n=n ( ) ) %>%
    ungroup( ) %>%
    mutate(sumN = sum(n), percentage = n/sumN) %>%
    arrange (-percentage)

temp %>%
  ggplot(aes(reorder(genres, percentage), percentage, fill= percentage)) +
  geom_bar(stat = "identity") +
  coord_flip() +
  scale_fill_distiller(palette = "PRGn") +
  labs( y = "Percentage", x = "Genre",caption = "source data :HarvardX edx se
```

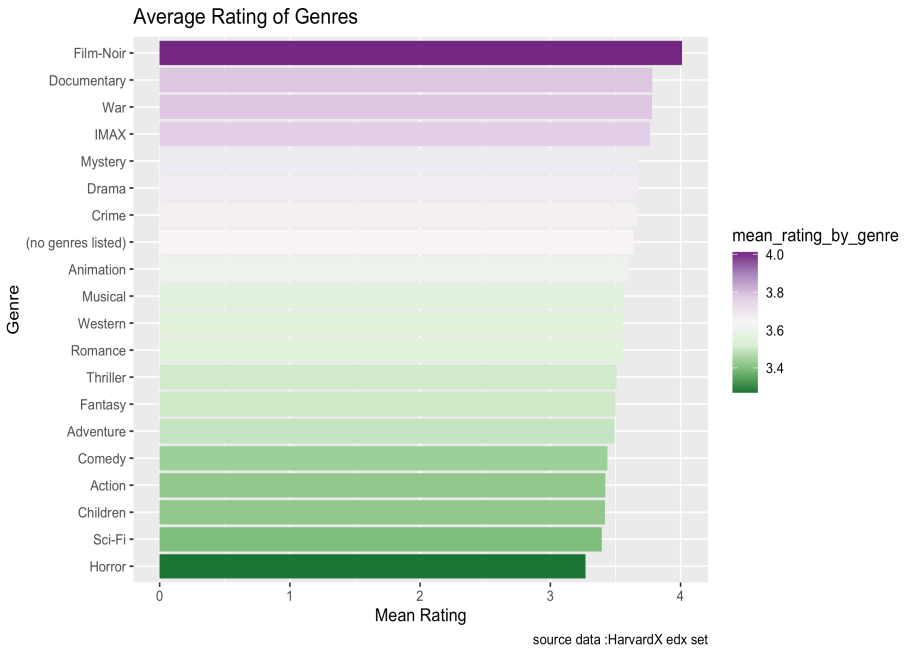
```
t") +
  ggtitle ("Distribution of Genres by Percent Rated")
```



As the graph shows that Drama has the highest percentage of ratings.

Bar Graph of Genre's Mean Rating

```
temp <- dat %>%
  group_by(genres) %>%
  summarize(mean_rating_by_genre = mean(rating)) %>%
  arrange(-mean_rating_by_genre)

temp %>%
  ggplot(aes(reorder (genres,mean_rating_by_genre), mean_rating_by_genre, fil
l= mean_rating_by_genre)) +
geom_bar(stat = "identity") +
  coord_flip() +
  scale_fill_distiller(palette = "PRGn") +
  labs( y = "Mean Rating", x = "Genre", caption = "source data :HarvardX edx
set") +
  ggtitle("Average Rating of Genres")
```



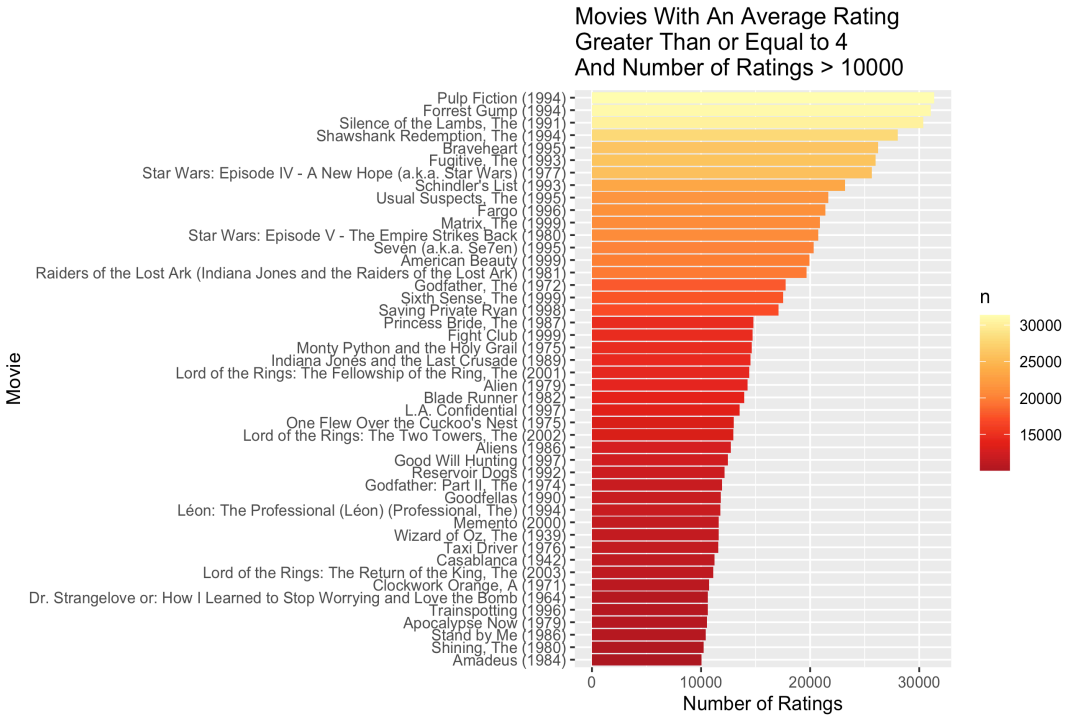
The film Horror had the lowest average rating, while film Noir had the highest average rating.

B.6. A Bar Graph on Movies With An Average Rating Greater Than or Equal to 4 and Number of Ratings > 10000

Explore movie ratings based on number of ratings and value of the rating

Graph on movies with more than 10000 ratings and mean rating greater than 4.

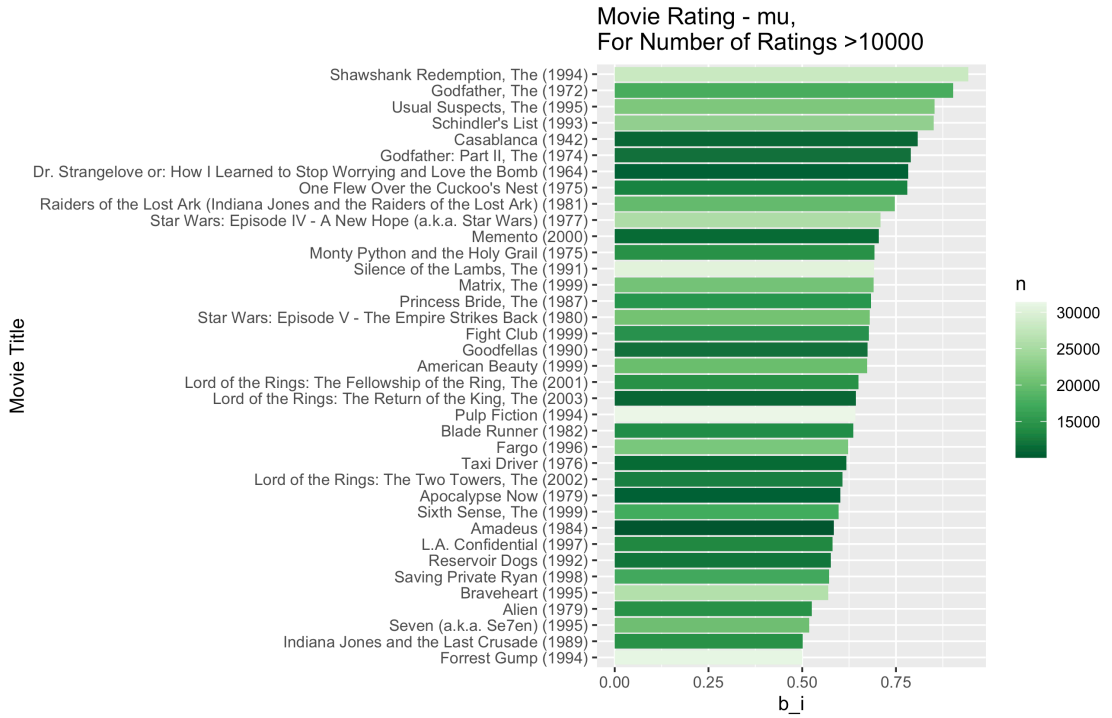
```
avg_rating_greater_than_4 <- edx %>%
  group_by (title) %>%
  summarize(mean_rating = mean(rating), n = n()) %>%
  filter (mean_rating >= 4) %>%
  arrange(desc (n, mean_rating))

avg_rating_greater_than_4 %>%
  filter(n >=10000) %>%
  ggplot(aes(reorder(title,n), n, fill = n)) +
  geom_bar(stat = "identity") +
  coord_flip () +
  scale_fill_distiller (palette= "YlOrRd") +
  xlab("Movie") +
  ylab("Number of Ratings") +
  ggtitle("Movies With An Average Rating\nGreater Than or Equal to 4\nAnd Num
ber of Ratings > 10000")
```



B.7. A Bar Graph of Movie Rating - mu and for Number of Ratings >10000

compute the least squares for movieId Which movies have a large number of ratings and a rating larger than the average mu

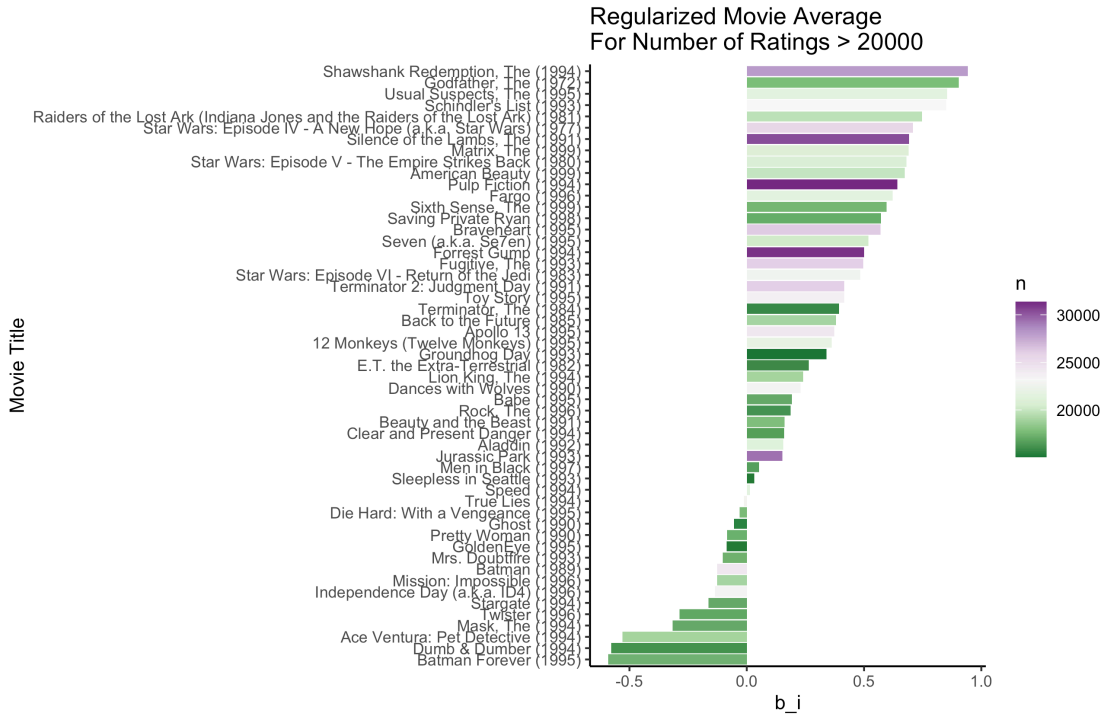
```
mu <- mean(edx$rating)
mu
[1] 3.512465

#A Bar Graph of Movie Rating - mu and for Number of Ratings >10000
edx %>%
  group_by(title) %>%
  summarize(b_i = mean(rating -mu), n = n ()) %>%
  filter(b_i >0.5, n >10000) %>%
  ggplot(aes(reorder(title, b_i), b_i, fill= n))+
  geom_bar(stat = "identity") +
  coord_flip() +
  scale_fill_distiller(palette ="Y1OrRd") +
  ggtitle("") +
  xlab("Movie Title") +
  ggtitle("Movie Rating - mu, \nFor Number of Ratings >10000") +
  theme_grey()
```



Bar Graph of Regularized Movie Average and For Number of Ratings >20000

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

movie_title <- edx %>%
  select(movieId, title) %>%
  distinct ()

movie_reg_avgs <- edx %>%
  group_by(movieId) %>%
  summarize(b_i = sum (rating - mu)/ (n () + 1), n_i = n())

edx_with_avgs <- edx %>%
  group_by(title, movieId) %>%
  summarize( n= n()) %>%
  left_join(movie_reg_avgs, by = "movieId") %>%
  arrange(desc(b_i, n))

edx_with_avgs %>%
  filter(n > 15000) %>%
  ggplot(aes(reorder(title, b_i), b_i, fill= n)) +
  geom_bar(stat = "identity") +
  coord_flip() +
  scale_fill_distiller(palette = "PRGn") +
```

```
  ggtitle("") +
  xlab("Movie Title")+
  ggtitle('Regularized Movie Average\nFor Number of Ratings > 20000') +
  theme_classic()
```



A Tibble of edx with avgs

```
head(edx_with_avgs)%>%
  kbl() %>%
  kable_styling()
```

title	movieId	n	b_i	n_i
More (1998)	4454	7	1.0515929	7
Satan's Tango (Sátántangó) (1994)	33264	2	0.9916899	2
Human Condition II, The (Ningen no joken II) (1959)	26048	4	0.9900278	4
Human Condition III, The (Ningen no joken III) (1961)	26073	4	0.9900278	4
Who's Singin' Over There? (a.k.a. Who Sings Over There) (Ko to tamo peva) (1980)	5194	4	0.9900278	4
Shawshank Redemption, The (1994)	318	28015	0.9426323	28015

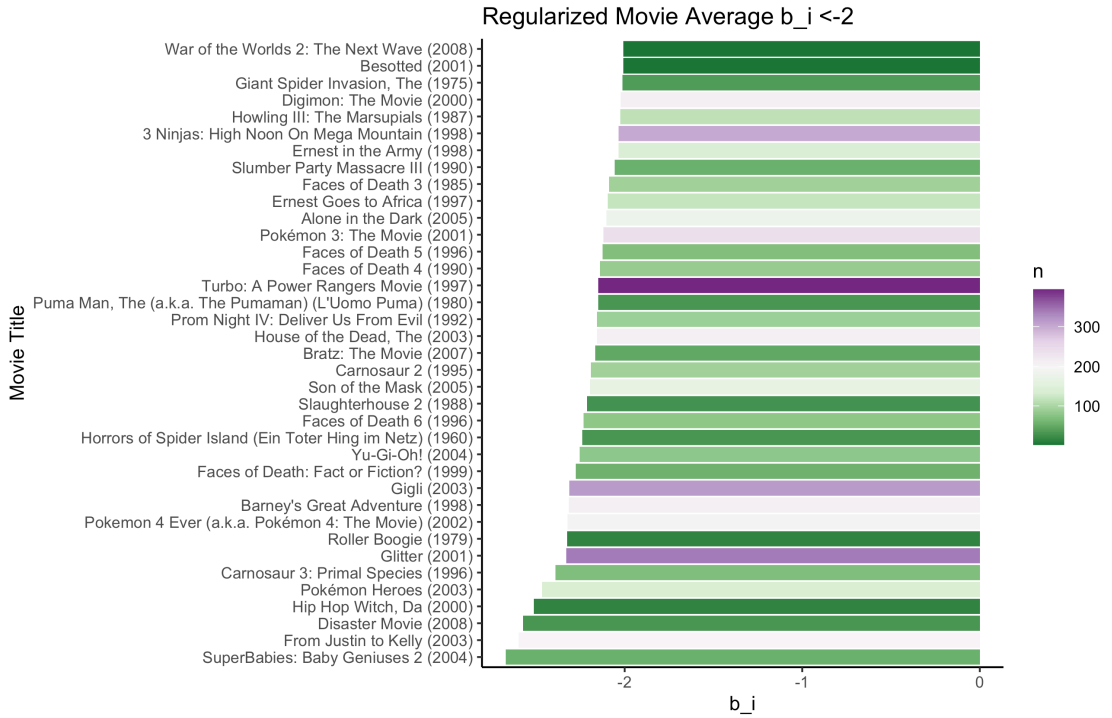
A Tibble of percentage

```
p <- edx_with_avgs %>%
  arrange(b_i) %>%
  filter(b_i <  -2) %>%
  arrange((b_i))
p %>%
  kbl()%>%
  kable_styling()
```

title	movieId	n	b_i	n_i
SuperBabies: Baby Geniuses 2 (2004)	8859	56	-2.670141	56
From Justin to Kelly (2003)	6483	199	-2.597403	199
Disaster Movie (2008)	61348	32	-2.572694	32
Hip Hop Witch, Da (2000)	7282	14	-2.511634	14
Pokémon Heroes (2003)	6371	137	-2.465273	137
Carnosaur 3: Primal Species (1996)	3574	68	-2.389096	68
Glitter (2001)	4775	339	-2.330076	339
Roller Boogie (1979)	8856	15	-2.324186	15
Pokemon 4 Ever (a.k.a. Pokémon 4: The Movie) (2002)	5672	202	-2.322749	202
Barney's Great Adventure (1998)	1826	208	-2.313841	208
Gigli (2003)	6587	313	-2.311789	313
Faces of Death: Fact or Fiction? (1999)	5740	58	-2.274966	58
Yu-Gi-Oh! (2004)	8811	80	-2.253052	80
Horrors of Spider Island (Ein Toter Hing im Netz) (1960)	4051	30	-2.237869	30

A Bar Graph Regularized Movie Average b_1 < -2

```
p %>%
  ggplot(aes(reorder(title, b_i), b_i, fill= n )) +
  geom_bar(stat = "identity") +
  coord_flip() +
  scale_fill_distiller(palette = "PRGn") +
  ggtitle("") +
  xlab("Movie Title") +
  ggtitle('Regularized Movie Average b_i <-2') +
  theme_classic()
```



B.8. Explore Correlation Between Rating, UserId,MovieId , Age of Movie And Number of Ratings

Explore correlation between rating, usersId, movieId, age of movie and number of ratings

```
#Number of movie ratings per movie

n_movies_ratings <- edx_with_title_dates %>%
  group_by(movieId) %>%
  summarize( n = n())

#Average Movie Rating for each movie
avg_movie_rat <- edx_with_title_dates %>%
  group_by(movieId) %>%
  summarize(avg_m_r = mean(rating))
```

A Description df of Correlation

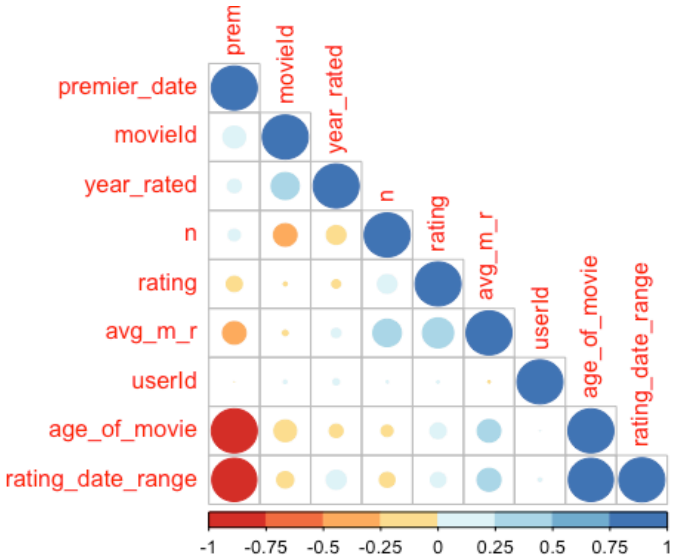
```
cor_dat <- edx_with_title_dates %>%
  select(rating,movieId, userId, year_rated, age_of_movie, rating_date_range,
premier_date) %>%
  left_join(n_movies_ratings, by = "movieId") %>%
  left_join(avg_movie_rat, by = "movieId")
head(cor_dat)%>%
```

```
  kbl() %>%
  kable_styling()
```

rating	movieId	userId	year_rated	age_of_movie	rating_date_range	premier_date	n	avg_m_r
5	122	1	1996	26	4	1992	2178	2.858586
5	185	1	1996	23	1	1995	13469	3.129334
5	292	1	1996	23	1	1995	14447	3.418011
5	316	1	1996	24	2	1994	17030	3.349677
5	329	1	1996	24	2	1994	14550	3.337457
5	355	1	1996	24	2	1994	4831	2.487787

corplot of Correlation

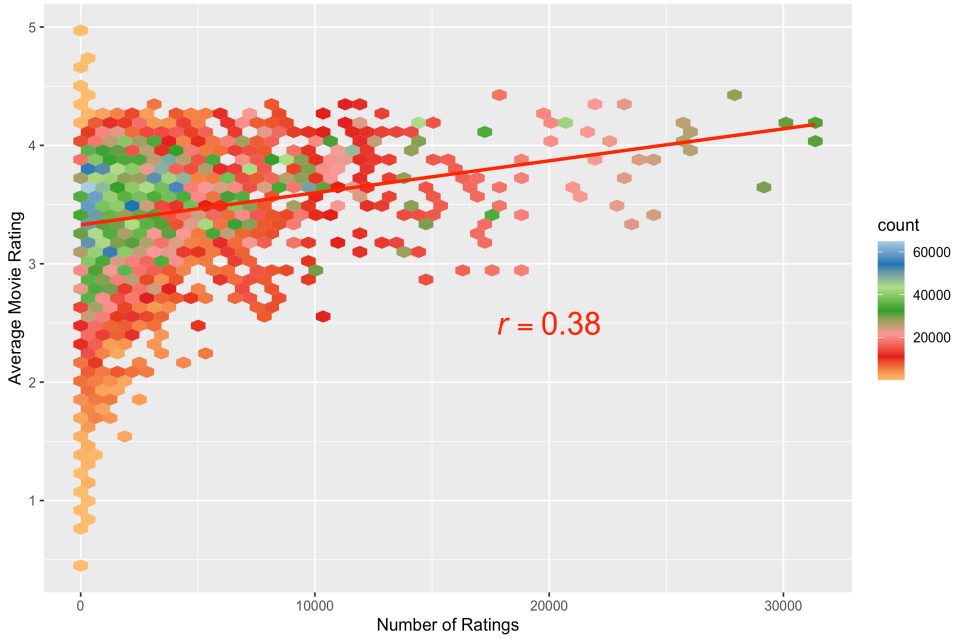
```
temp <- cor_dat %>%
  select( one_of("rating", "movieId", "userId", "year_rated", "age_of_movie",
"rating_date_range", "premier_date", "n", "avg_m_r")) %>%
  as.matrix()
M <- cor(temp,use =  "pairwise.complete.obs")

corrplot (M, order = "hclust", addrect = 2, type ="lower", col = brewer.pal(
n = 8, name= "RdYlBu"))
```



A Coorplot of Number of Rating Vs Avg Movie Ratings

Number of rating vs avg movie ratings

```
get_cor <- function(df) {
  m <- cor(df$x, df$y, use="pairwise.complete.obs");
  eq <- substitute(italic(r) == cor,list(cor=format(m, digits =2)))
  as.character(as.expression(eq));
}

cor_dat %>%
  ggplot( aes(n, avg_m_r)) +
  stat_bin_hex(bins = 50) +
  scale_fill_distiller (palette ="Paired") +
  stat_smooth(method = "lm", color= "Red", size = 1) +
  annotate("text", x = 20000, y = 2.5, label = get_cor (data.frame(x = cor_da
t$n, y = cor_dat$avg_m_r)), parse = TRUE, color ="red", size= 7) +
  ylab("Average Movie Rating") + xlab ("Number of Ratings")
```



B.9. Penalized Least Square With Regularization

Create train and test set with caret package to assess the accuracy of the models. The Netflix challenge used the Residual Mean Squared Error(RMSE) metrics to fin the winner based on a test set. The RMSE is then defined as a statistical method.

$$RMSE = \sqrt{\frac{1}{N}\sum_{u,i}(\widehat{y_{u,i}} - y_{u,i})2}$$

Where $y_{u,i}$ as the rating for movie I by user u and denote our prediction with $\widehat{y_{u,i}}$ and N is being the number of user / movie combinations and. The sum occurring over all these combination

In generally movielens dataset was investigate with method including but not limited to data cleaning by eliminating unwanted column and data visualization by creating tables and graphs of various formats

We can use this equation:

$$\frac{1}{N}\sum_{u,i}(y_{u,i-\mu-b_i})2 + \lambda \sum_{i} b \frac{3}{i}$$

The first term is the least square and the second term is the penalty that gets larger when many b_i are large. Using Calculus we can actually show that the values of the b_i that minimize this equation are:

$$\hat{b}\ (\lambda) = \frac{1}{\lambda + n_i}\sum_{y=1}^{n_i} (Y_{u,i} - \hat{\mu})$$

Where n_i is the number of ratings made for movie i.

A linear model with average rating and different BIASES are used as predictors:

$$\hat{Y} = \mu + b_i + b_u + b_a$$

μ = Average rating of all movies

b_i= Bias based on Movies

b_u= Bias based on Users

b_a= Bias based on Age

```
# Define Residual Mean Squared Error (RMSE)

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

B.9.a. Model 1

First model : use average rating for all movies regardless of user

In the first model, just based on the ratings itself, to minimize the RMSE, the best prediction of ratings for each movie will be the overall average of all ratings.

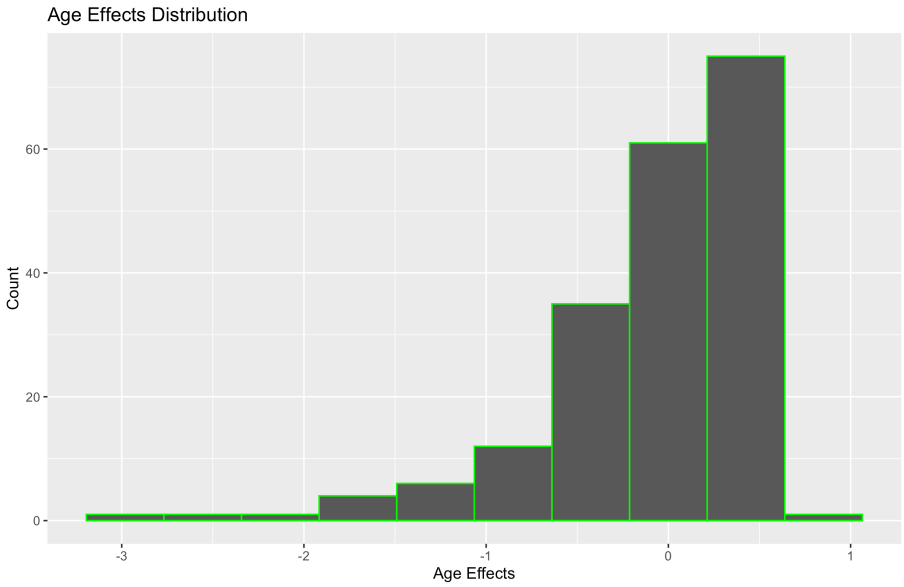
```
 mu <- mean(edx$rating)
mu
[1] 3.512465

naive_rmse <- RMSE(edx$rating, mu) #compute Residual Mean Standard Error(RMSE
)

naive_rmse
[1] 1.060331

#Just the Average
rmse_model1 <- RMSE(edx$rating,mu)

rmse_model1
[1] 1.060331
```

B.9.b. Mode1 2

Age Effects : Adding b_a to represent ratings on movies. At first lets calculate the age bias and take a look at its distribution. Then we will make predictions and evaluate the RMSE using the validation set.

```
edx_1 <- edx %>%
  mutate(year_rated = year(as_datetime(timestamp)))
premier <- stringi::stri_extract(edx$title,regex = "(\\d{4})", comments =TRUE
) %>%
  as.numeric()
```

```
#add the premier date
edx_with_title_dates <- edx %>%
  mutate(premier_date =premier)

age_at_rating <- stringi::stri_extract(edx$title,regex = "(\\d{4})", comments
=TRUE) %>%
  as.numeric()

# convert timestamp to year
edx  <- mutate(edx, year_rated =year(as_datetime(timestamp)))
head(edx)

# extract the release year of the movie
#edx_1 has year_rated, year_released, age_at_rating, and titles without year
information
edx_1 <- edx_with_title_dates %>%
  mutate(title = str_replace(title, "^(.+)\\s\\((\\d{4})\\)$","\\1__\\2")) %>
%
  separate(title, c("title", "year_released"), "__") %>%
  select(-timestamp)
edx_1 <- edx_with_title_dates %>%
  mutate(age_at_rating = as.numeric(year_rated) -as.numeric(year_released))

age_effect <- edx_1%>%
  group_by(age_at_rating) %>%
  summarize(b_a = mean (rating)- mu)

age_effect %>%
  ggplot(aes(x = b_a)) +
  geom_histogram (bins =10, color = I ("Green")) +
  ggtitle("Age Effects Distribution") +
  xlab( "Age Effects") +
  ylab("Count")
```




```
validation_1 <- validation %>%
  mutate(year_rated = year(as_datetime(timestamp)))
age_at_rating <- stringi::stri_extract(edx$title,regex = "(\\d{4})", comments
=TRUE) %>%
  as.numeric()

validation_1 <- edx_with_title_dates%>%
  mutate(title = str_replace(title, "^(.+)\\s\\((\\d{4})\\)$","\\1__\\2")) %>
%
  separate(title, c("title", "year_released"), "__") %>%
  select(-timestamp)
validation_1 <- validation_1 %>%
  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)

rmse_model2 <- RMSE(validation$rating, predicted_ratings_2)

rmse_model2
[1] 1.069718
```

B.9.c. Model 3

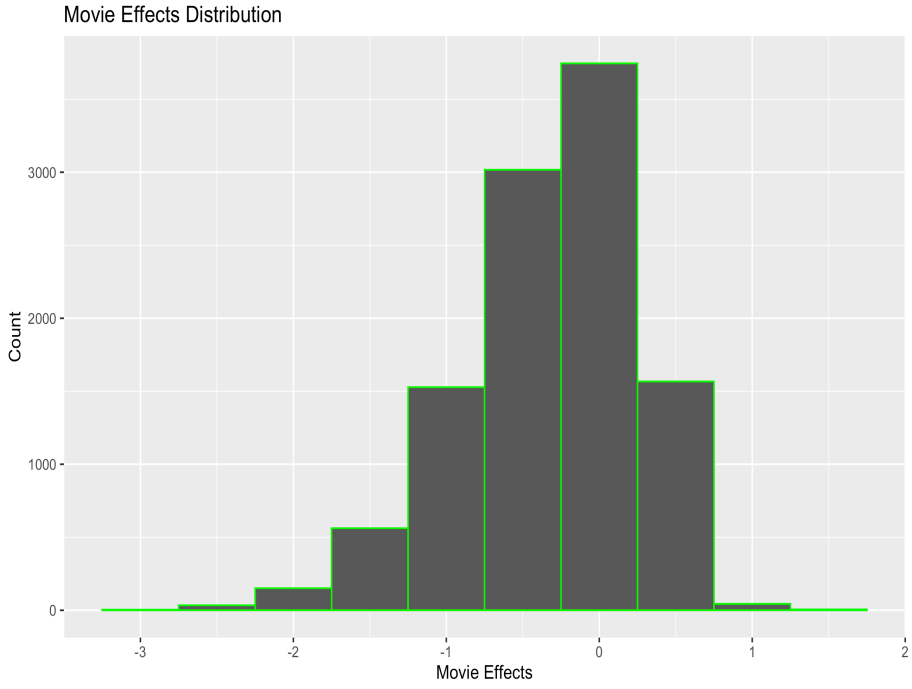
Movie effects: adding b_i to represent average ranking for movie i

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

```r
#calculate b_i for validation set
movie_avgs <- edx %>%
  group_by(movieId) %>%
  summarize(b_i = mean(rating-mu))

#predicted ratings
predicted_ratings_bi <- mu + validation %>%
  left_join(movie_avgs, by = 'movieId') %>%
  .$b_i

rmse_model3 <- RMSE(validation$rating, predicted_ratings_bi)

rmse_model3
```

```
[1] 0.9439087
```

```r
movie_avgs <- edx %>%
  group_by(movieId) %>%
  summarize(b_i = mean(rating - mu))

bi %>%
  ggplot(aes(x = b_i)) +
  geom_histogram (bins =10, color = I ("Green")) +
  ggtitle("Movie Effects Distribution") +
  xlab( "Movie Effects") +
  ylab("Count")
```

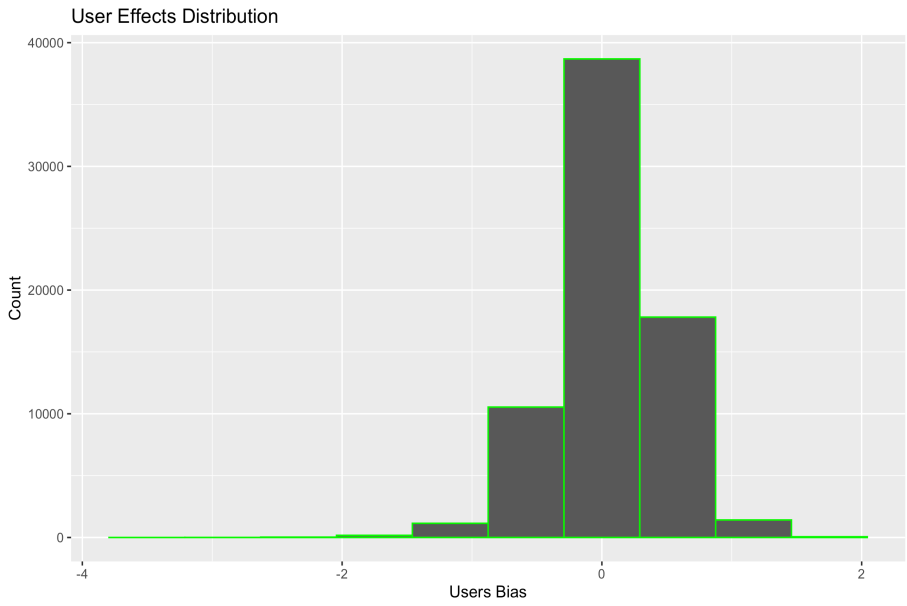


We have predicted movie rating based on the fact that movie are rated differently by adding the bi to mu. If an individual movie is on average rated worse that the average rating of all movies mu, we predict that if will rated lower that mu by bi, the difference of the individual movie average from the total average.

B.9.d. Model 4

Movie + User Effects : Adding b_u to represent average ranking for user u. We now further add the bias of user (b_u) to movie effects model.

```
#calculate b_u for validation set

user_avgs <- edx %>%
  left_join(movie_avgs, by= 'movieId') %>%
  group_by(userId) %>%
  summarize(b_u = mean(rating - mu- b_i))

#predicted ratings
predicted_ratings_bu <- validation %>%
  left_join(movie_avgs, by = 'movieId') %>%
  left_join(user_avgs, by = 'userId') %>%
  mutate(pred = mu + b_i + b_u) %>%
  .$pred


rmse_model4 <- RMSE(validation$rating, predicted_ratings_bu)

rmse_model4

[1] 0.8653488

movie_avgs <- edx %>%
  group_by(movieId) %>%
  summarize(b_u = mean(rating - mu))

bu %>%
  ggplot(aes(x = b_u)) +
  geom_histogram (bins =10, color = I ("Green")) +
  ggtitle("User Effects Distribution") +
  xlab( "Users Bias") +
  ylab("Count")
```



B.9.e. Model 5

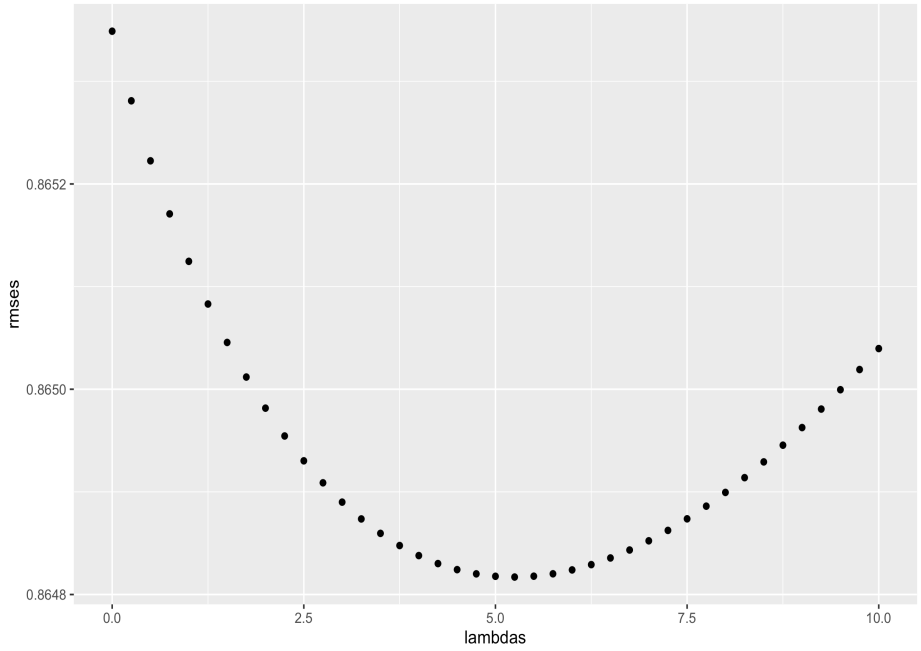
Regularization Movie + User Effects Model on final hold-out test set.

To estimates of bi and bu are caused by movies with very few ratings and in some users that only rated a very small number of movies. The use of the regularization permits to penalize these aspects. We use the value of lambda that is a tuning parameter that will minimize the RMSE. This shrinks the bi and bu in case of small number of rating.

remembering (5), λ is a tuning parameter. We use cross- validation to choose it.

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

lambda <- seq(0, 10, 0.25)

rmses <- sapply(lambda,function(l){
  mu <- mean(edx$rating)

  b_i <- edx %>%
    group_by (movieId) %>%
    summarize(b_i = sum(rating - mu) / (n() + l))

  b_u <- edx %>%
    left_join (b_i, by =  "movieId") %>%
    group_by(userId)  %>%
    summarize(b_u = sum(rating - b_i - mu) / (n () + l))

  predicted_ratings_b_i_u <- validation %>%
    left_join(b_i, by = "movieId")%>%
```

```
    left_join(b_u, by = "userId") %>%
    mutate(pred = mu + b_i +  b_u) %>%
    .$pred

  return(RMSE( validation$rating, predicted_ratings_b_i_u))
})


#we plot RMSE with lambdas to select the optimal lambda.

qplot(lambdas, rmses)
```




```
#For the full model,  the optimal lambda is: 5.25

lambda <- lambdas[which.min(rmses)]

lambda
[1] 5.25
```

Summarize of RMSE Results

```
rmse_results <- data.frame(Methods = c("Just The Average Model ","Age Effects
Model","Movie Effects Model", "Movie + User Effects Model on final  hold-out
test set", "Regularized Movie + User Effects Model on final hoid-out test set
"), RMSE = c(rmse_model1, rmse_model2,rmse_model3, rmse_model4, rmse_model5 )
)
```

```
kable(rmse_results) %>%
  kable_styling(bootstrap_options = "striped", full_width = F, position = "ce
nter") %>%
  kable_styling(bootstrap_options = "bordered", full_width = F, position = "c
enter") %>%
  column_spec(1,bold = T) %>%
  column_spec(2,bold = T, color = "white", background = "#D7261E")
```

Methods	RMSE
Just The Average Model	1.0603313
Age Effects Model	1.0697185
Movie Effects Model	0.9439087
Movie + User Effects Model on final hold-out test set	0.8653488
Regularized Movie + User Effects Model on final hoid-out test set	0.8648170

We find the lowest value of RMSE to be at **0.86490**

C. Conclusion

MovieLens is a classical dataset for recommendation system and represent a challenge for development of better machine learning algorithm. In this project: Model 1. Just the Average Model had RMSE of **1.060331**, Model 2. Age Effects Model had RMSE of **1.069718**, Model 3. Movie Effects Model had RMSE of **0.9439087**, Model 4. Movie + User Effects Model on final hold-out test set had RMSE of **0.8653488**, Model 5. Regularized Movie + User Effects Model on final hold-out test set had RMSE of **0.8648170**.

The linear Model achieved the RMSE of **0.8648170**, successfully passing the target of **0.86490**.

Therefore, the data model continued to improved as variety of factors were being accounted for and included. Future analysis should be conducted with a recommendation to parse data by film year and genre type to best determine how genre or time frame influences film ratings.