MovieLens Project: Rating Scores Prediction

March 11, 2019

A. INTRODUCTION

The goal of this project is to predict the rating given a user and a movie, using linear regression that included user and movie features on the MovieLens 10M data set. The data set was provided by MovieLens. A brief data analysis was conducted to see the data distribution and factors that might influnced how the users rate movie. Then, we compared the RMSEs (Root Mean Square Error) of linear regression model that only included movie effects and linear regression model that only included both user and movie effects. After going through this case study, you'll be able to:

- Analyze data to develop your own version of a recommendation engine, which forms the basis of content systems used at companies like Netflix, Pandora, Spotify, etcetera.
- Experience a hands-on approach to advance your data science skills.
- Access to a series of resources and tools, including sample data basis, that will enable you to build your recommendation system.

A.1. About the Data

We will use the 10M dataset provided by MovieLens. This dataset set consists of:

- 10000054 ratings and 95580 tags applied to 10681 movies by 71567 users. We are not using the tag file for the simplicity of this project.
- Users were selected at random and had rated at least 20 movies.

A.2. Getting Data

Using the code below, which is provided in the Data Science: Capstone course by Professor Rafael Irizarry, we can download and join the movies and rating files from the 10M dataset. Then, the data will be slpitted into two datasets: edx (contains 90% of data) and validation (contains 10% of data). We will use edx dataset as our known dataset and ignore validation dataset as if it is unknown for our future rating score prediction.

```
#############################################################
# Create edx set, validation set, and submission file
#############################################################
# Note: this process could take a couple of minutes

#Install packages if not installed, but I will just load the packages since I already installed them
#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")
library(caret)
library(tidyverse)

# 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)
```

```r
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_10M <- left_join(ratings, movies, by = "movieId")

# Validation set will be 10% of MovieLens data
set.seed(1)
test_index <- createDataPartition(y = movielens_10M$rating, times = 1, p = 0.1, list = FALSE)
edx <- movielens_10M[-test_index,]
temp <- movielens_10M[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)

# Learners will develop their algorithms on the edx set
# For grading, learners will run algorithm on validation set to generate ratings
validation <- validation %>% select(-rating)

# Ratings will go into the CSV submission file below:
#write.csv(validation %>% select(userId, movieId) %>% mutate(rating = NA),"submission.csv", na = "", ro

#Remove data
rm(dl, ratings, movies, test_index, temp, removed, movielens_10M)
```

B. DATA ANALYSIS

In this section, I will analyze edx dataset to understand the dataset better and to figure the effects that influenced the rating scores for the movie.We need ignore the validation dataset completely as if it is unknown. Hence, we will only explore the edx dataset to get a better understand of our known data.

B.1. Load packages

Import packages that we will need later

```r
#training classification and regression models
library(caret)
#toolkits for iteration
library(purrr)
#data manipulation
library(tidyverse)
```

```r
#data visualization
library(ggplot2)
#date-times
library(lubridate)
#data wrangling
library(dplyr)
#convert rmarkdown to other formats
library(rmarkdown)
#dynamic report generation
library(knitr)
```

B.2. Data Summary

As shown below, the edx dataset has 9000055 observations and 6 variables (userId, movieId, rating, timestamp, title and genres). The averge rating score and median score are 3.512 and 4.0 respectively.

```r
#Get data info for edx set
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: int  838985046 838983525 838983421 838983392 838983392 838984474 ...
##  $ title    : chr  "Boomerang (1992)" "Net, The (1995)" "Outbreak (1995)" ...
##  $ genres   : chr  "Comedy|Romance" "Action|Crime|Thriller" ...
```

```r
#Summary of edx set
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
```

B.3. Data Exploratory

B.3.a. Number of Unique Movies and Users

There are 10677 unique movies and 69878 different users in this dataset.

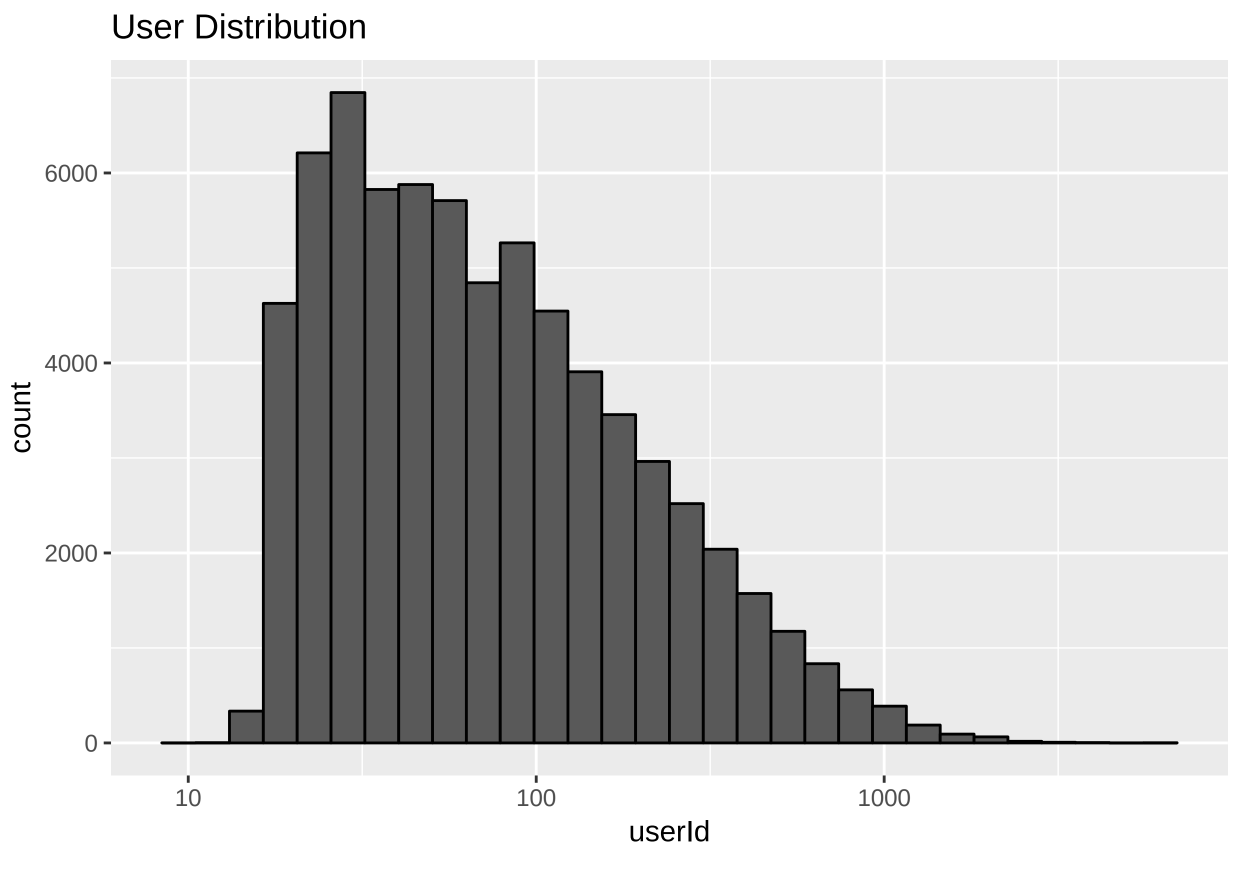
```r
#Number of different users and movies
edx %>% summarize(n_movies = n_distinct(movieId),n_users = n_distinct(userId))
```

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

B.3.b. User Distribution

Some users are more active than others at rating movies. The users with most number of ratings and their average scores and median scores are shown below.

```
# User distribution
edx%>%
  count(userId) %>%
  ggplot(aes(n)) +
  geom_histogram(bins = 30, color = "black") +
  scale_x_log10() +
  ggtitle("User Distribution") +
  xlab("userId")
```



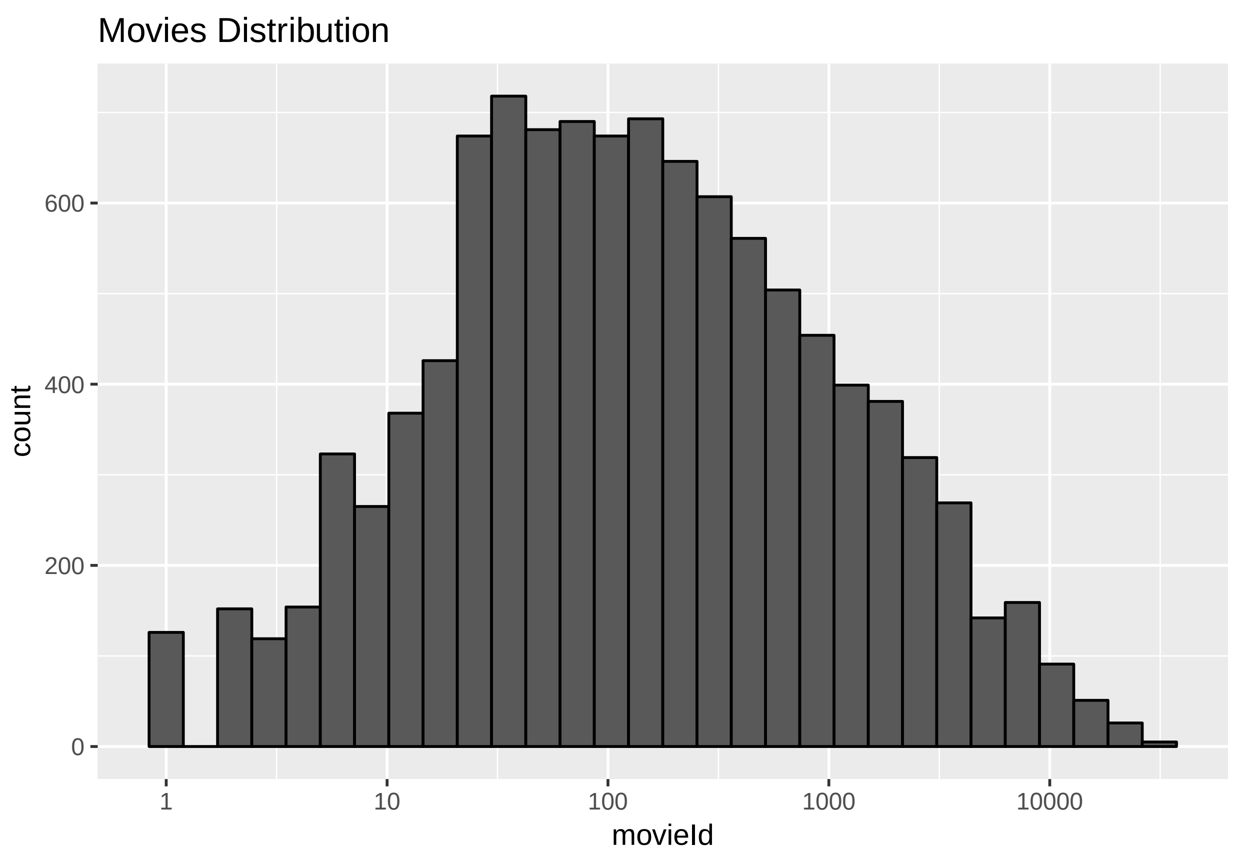
```
#Popular users with highest number of rating with their average and median scores
edx %>%
  select(userId,rating) %>%
  group_by(userId) %>%
  summarise(count = n(),
            avg_score = mean(rating,na.rm = TRUE),
            median_score = median(rating,na.rm = TRUE)
            ) %>%
  ungroup() %>%
  arrange(desc(count))
```

```
## # A tibble: 69,878 x 4
##    userId count avg_score median_score
##     <int> <int>     <dbl>        <dbl>
##  1  59269  6616      3.26          3
##  2  67385  6360      3.20          3
##  3  14463  4648      2.40          2
##  4  68259  4036      3.58          4
##  5  27468  4023      3.83          4
##  6  19635  3771      3.50          3.5
##  7   3817  3733      3.11          3
##  8  63134  3371      3.27          3.5
##  9  58357  3361      3.00          3
## 10  27584  3142      3.00          3
## # ... with 69,868 more rows
```

B.3.c. Movies Distribution

We can see that some movies get rated more than others. The most popular movie is Pulp Fiction (1994) with 31,362 ratings and an average rating score of 4.154789.

```
# Movies distribution
edx%>%
  count(movieId) %>%
  ggplot(aes(n)) +
  geom_histogram(bins = 30, color = "black") +
  scale_x_log10() +
  labs(title="Movies Distribution",x ="movieId")
```



Most Popular Movies

```
# Most popular movies
edx %>%
  mutate(year=year(as.Date(as_datetime(timestamp), format = "%m/%d/%Y")),month=month(as.Date(as_datetim
  select(title,rating,year,month) %>%
  group_by(title) %>%
  summarise(count = n(), #total number of ratings
            avg_score = mean(rating,na.rm = TRUE) #average rating score
            ) %>%
  ungroup() %>%
  arrange(desc(count))
```

```
## # A tibble: 10,676 x 3
##    title                                                    count avg_score
##    <chr>                                                    <int>     <dbl>
##  1 Pulp Fiction (1994)                                      31362      4.15
##  2 Forrest Gump (1994)                                      31079      4.01
##  3 Silence of the Lambs, The (1991)                         30382      4.20
##  4 Jurassic Park (1993)                                     29360      3.66
##  5 Shawshank Redemption, The (1994)                         28015      4.46
##  6 Braveheart (1995)                                        26212      4.08
##  7 Fugitive, The (1993)                                     25998      4.01
##  8 Terminator 2: Judgment Day (1991)                        25984      3.93
##  9 Star Wars: Episode IV - A New Hope (a.k.a. Star Wars) (~ 25672      4.22
## 10 Apollo 13 (1995)                                         24284      3.89
## # ... with 10,666 more rows
```

Most Popular Genres Drama is the most popular genre with 3,910,127 ratings and an average rating score of 3.673131. Followed by Comedy, Action and Adventure.

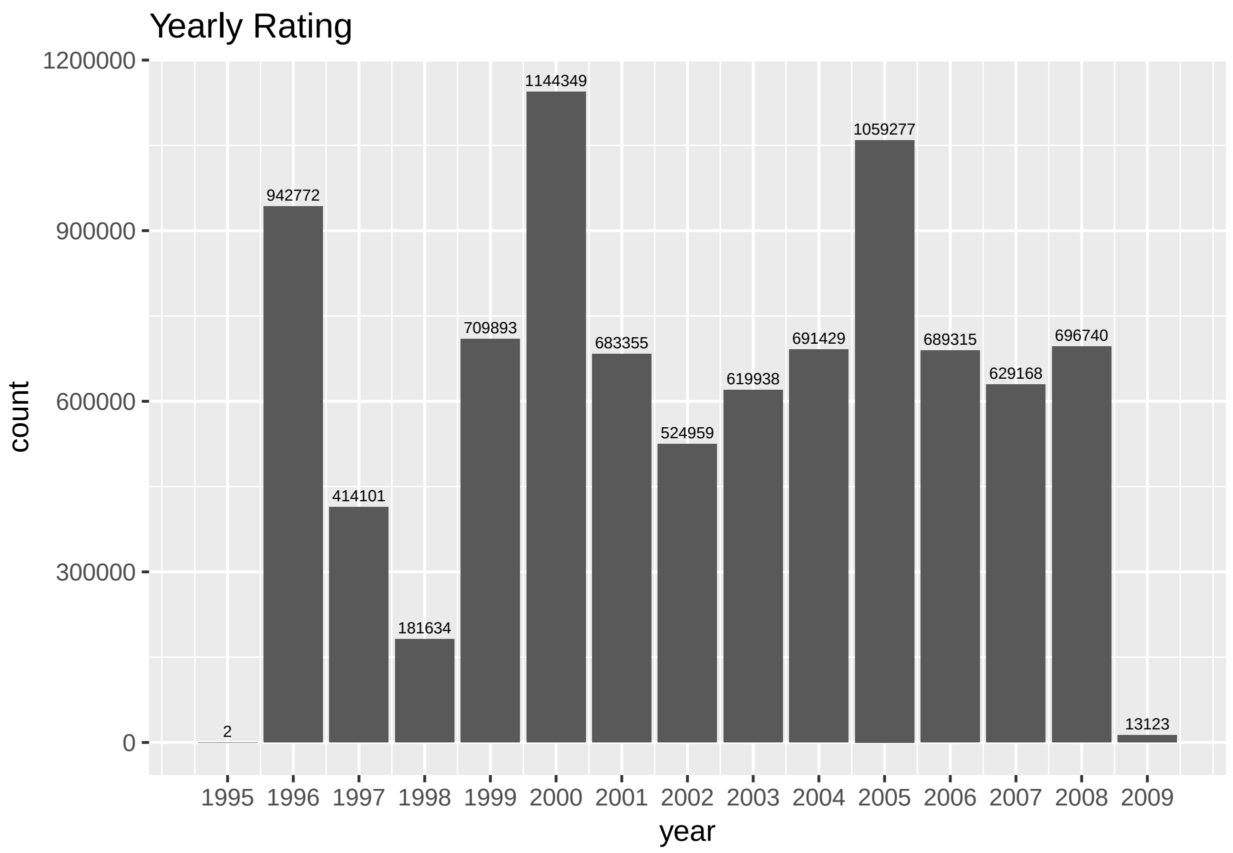
```
#Number of ratings for each genres
edx %>%
  separate_rows(genres, sep = "\\|") %>%
  select(genres, rating) %>%
  group_by(genres) %>%
  summarise(count = n(),
            avg_score = mean(rating,na.rm = TRUE),
            median_score = median(rating,na.rm = TRUE)
            ) %>%
  ungroup() %>%
  arrange(desc(count))
```

```
## # A tibble: 20 x 4
##    genres               count avg_score median_score
##    <chr>                <int>     <dbl>        <dbl>
##  1 Drama              3910127      3.67          4
##  2 Comedy             3540930      3.44          3.5
##  3 Action             2560545      3.42          3.5
##  4 Thriller           2325899      3.51          3.5
##  5 Adventure          1908892      3.49          3.5
##  6 Romance            1712100      3.55          4
##  7 Sci-Fi             1341183      3.40          3.5
##  8 Crime              1327715      3.67          4
##  9 Fantasy             925637      3.50          3.5
## 10 Children            737994      3.42          3.5
## 11 Horror              691485      3.27          3.5
## 12 Mystery             568332      3.68          4
## 13 War                 511147      3.78          4
## 14 Animation           467168      3.60          4
## 15 Musical             433080      3.56          4
## 16 Western             189394      3.56          4
## 17 Film-Noir           118541      4.01          4
## 18 Documentary          93066      3.78          4
## 19 IMAX                  8181      3.77          4
## 20 (no genres listed)       7      3.64          3.5
```

B.3.d. Time Distribution

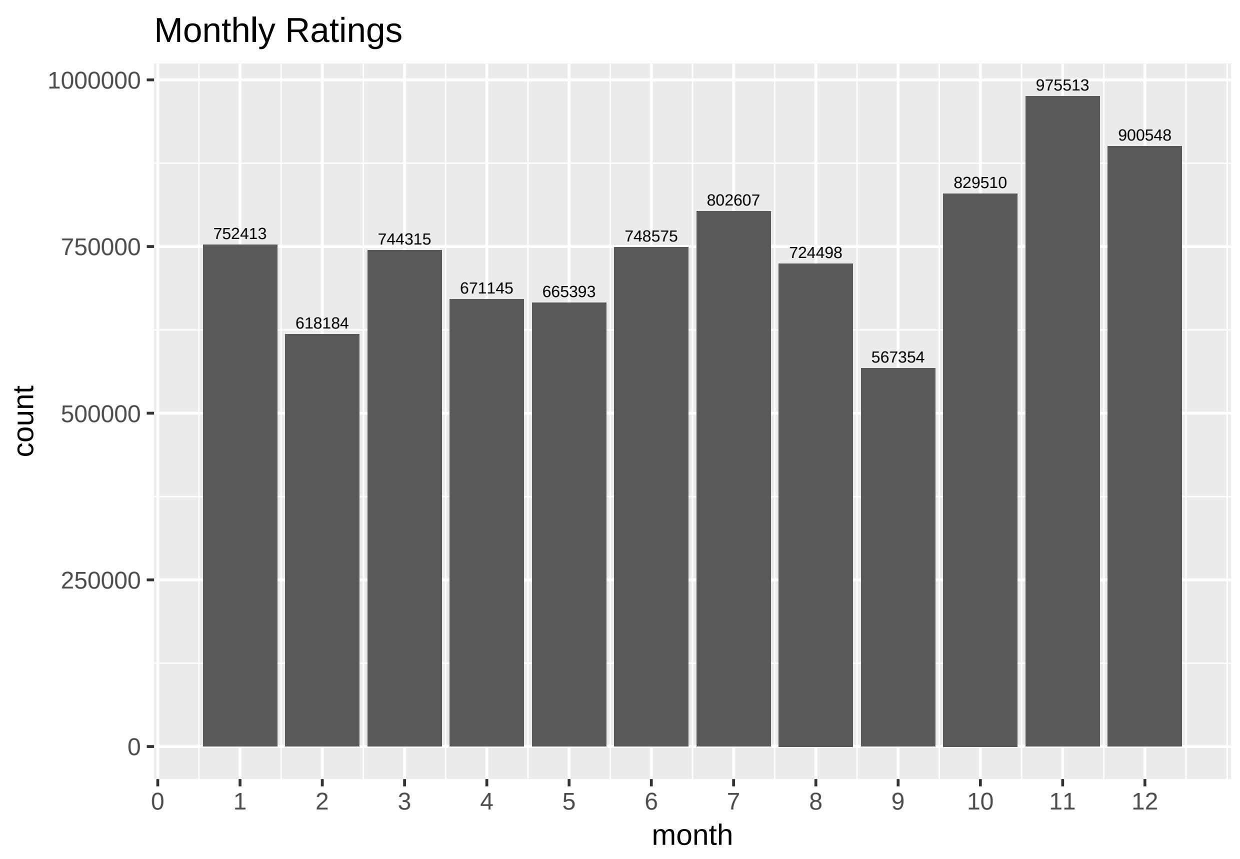
Year with Largest Number of Ratings The year of 2000 has the highest number of ratings of 1,144,349. We can see that some years have larger number of reviews than others.

```
#Number of ratings in each year
edx %>% mutate(year=year(as.Date(as_datetime(timestamp), format = "%m/%d/%Y"))) %>%
  ggplot(aes(x=year)) +
  geom_bar() +
  geom_text(stat='count', aes(label=..count..), vjust= -0.5,size=2) +
  scale_x_continuous(breaks=seq(1995, 2009, 1)) +
  labs(title="Yearly Rating")
```



Month with Largest Number of Ratings There is also an uneven distibution of reviews over the months. November has the largest number of reviews at 975,513.

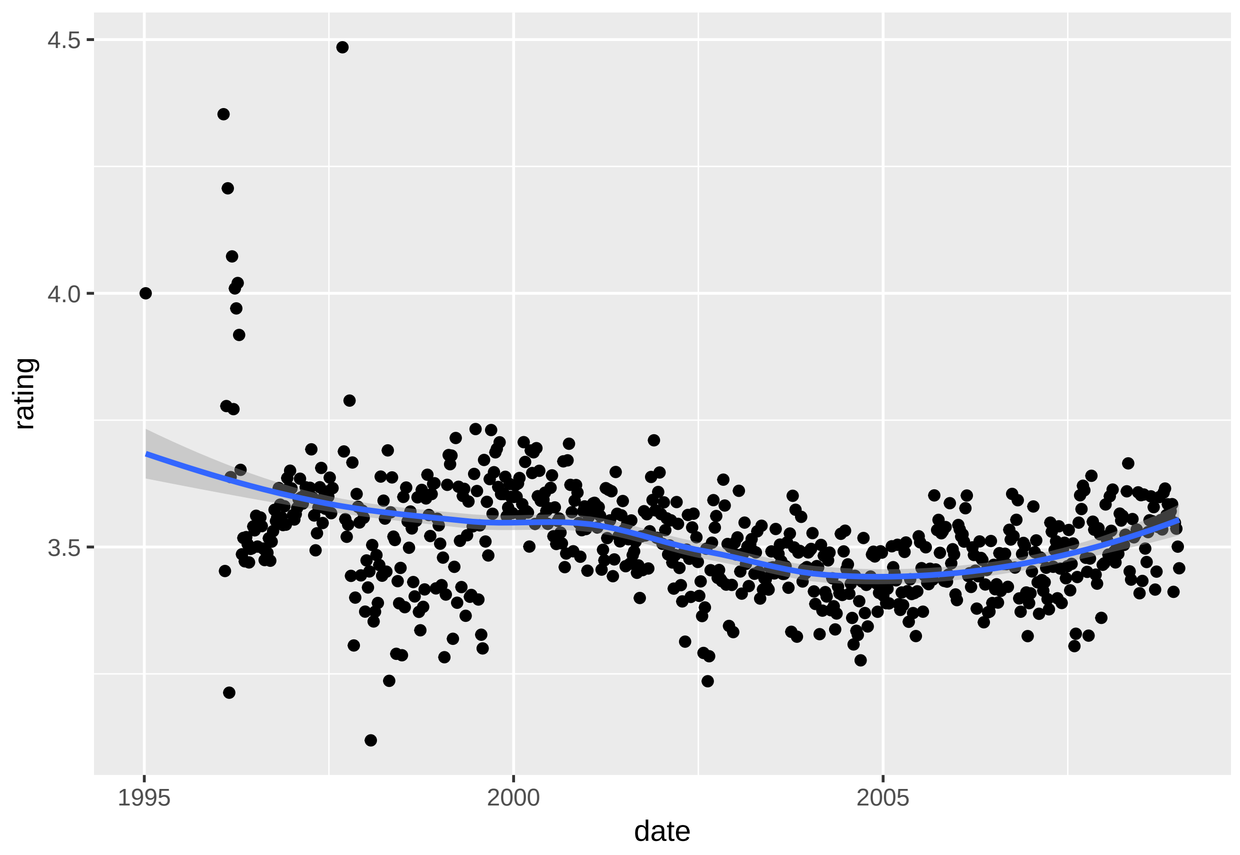
```
edx %>% mutate(month=month(as.Date(as_datetime(timestamp), format = "%m/%d/%Y"))) %>%
  ggplot(aes(x=month)) +
  geom_bar() +
  geom_text(stat='count', aes(label=..count..), vjust= -0.5,size=2) +
  scale_x_continuous(breaks=seq(0, 12, 1)) +
  labs(title="Monthly Ratings")
```



Week with Largest Number of Ratings The average rating for each week is shown below:

```
edx %>% mutate(date = round_date(as.Date(as_datetime(timestamp), format = "%m/%d/%Y"), unit = "week"))
    group_by(date) %>%
    summarize(rating = mean(rating)) %>%
    ggplot(aes(date, rating)) +
    geom_point() +
    geom_smooth()
```

```
## `geom_smooth()` using method = 'loess' and formula 'y ~ x'
```



C. PREDICTION MODELS

C.1. Train and Test Sets for EDX Dataset

We will split the edx dataset into train and test sets. Test set will have 10% of data from edx dataset.

```r
# Split edx data into train and test sets
set.seed(1)
test_index_edx <- createDataPartition(y = edx$rating, times = 1, p = 0.1, list = FALSE)
train_set <- edx[-test_index_edx,] #train set will be 90% of edx dataset
test_set <- edx[test_index_edx,] #test set will be 10% of edx dataset
#Remove unused data
rm(test_index_edx)
```

To make sure we do not include users and movies in the test set that do not appear in the training set, we remove these entries using the semi_join function.

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

C.2. The Residual Mean Squared Error (RMSE)

The RMSE is then defined as below, with N being the number of user/movie combinations and the sum occurring over all these combinations. This number in our case should be less than 1 (star).

$$RMSE = \sqrt{\frac{1}{N}\sum_{u,i}(\hat{y}_{u,i} - y_{u,i})^2}$$

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

C.3. Modeling Movie Effects

In Professor Rafalab's lectures and textbook, multiple effects was tested. Below, I would like to test the movie effects and user effects on predicting the rating scores. The equation below accounted for the movie effects or bias (b_i represents average ranking for movie i), where μ is the "true" rating for all movies, $\epsilon_{u,i}$ is independent errors sampled from the same distribution centered at 0.

$$Y_{i,j} = \mu + b_i + \epsilon_{u,i}$$

We can use least squared to estimate the b_i in the following way:

```
movieMod <- lm(rating ~  as.factor(movieId), data=train_set)
```

However, ecause there are thousands of b_i, each movie gets one, the `lm()` function will be very large and slow. Therefore, we are not using this function. However, the least square estimate $\hat{b}_i$ is just the average of $Y_{u,i} - \hat{\mu}$ for each movie i. So we can compute them this way (we will drop the hat notation in the code to represent estimates going forward):

```r
mu <- mean(train_set$rating) #Average rating

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

predicted_ratings <- mu + test_set %>%
  left_join(movie_avgs, by='movieId') %>%
  pull(b_i)
```

```r
# Calculate RMSE
model_1_rmse <- RMSE(test_set$rating, predicted_ratings)
rmse_results <- tibble(method = "Movie Effect Model", RMSE = model_1_rmse)
rmse_results %>% knitr::kable()
```

method	RMSE
Movie Effect Model	0.9429615

C.4. Modeling User effects

As we saw from the data analysis, some users rate most movie highly while others rate most movies very low. Hence, we can add another effect or bias b_u (user-specific effect) to out model:

$$Y_{i,j} = \mu + b_i + b_u + \epsilon_{u,i}$$

To fit this model, we could again use `lm` like below:

```
movie_userMod <- lm(rating ~  as.factor(movieId) +  as.factor(userId), data=train_set)
```

However, for the reasons described earlier, we won't. Instead, we will compute an approximation but computing $\hat{\mu}$ and $\hat{b}_i$ and estimating $\hat{b}_u$ as the avergage of $y_{u,i} - \hat{\mu} - \hat{b}_i$ as below:

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

We can construct predictors and see how the RMSE workout:

```
predicted_ratings <- test_set %>%
  left_join(movie_avgs, by='movieId') %>%
  left_join(user_avgs, by='userId') %>%
  mutate(pred = mu + b_i + b_u) %>%
  pull(pred)
```

```
# Calculate RMSE
model_2_rmse <- RMSE(predicted_ratings,test_set$rating)
rmse_results <- bind_rows(rmse_results,
                          tibble(method="Movie + User Effects Model",
                                       RMSE = model_2_rmse ))
```

D. RESULTS: Movie User Effects Model for Prediction

The model that includes both movie and user effects has been improved with a lower RMSE (of 0.8646844) than the model that only included movie effect (with RMSE of 0.9429615).

```
rmse_results %>% knitr::kable()
```

method	RMSE
Movie Effect Model	0.9429615
Movie + User Effects Model	0.8646844

E. CONCLUSION and RATING SCORES PREDICTION

Through the data analysis, the following points can be made for the 10M MovieLens dataset: + Top five most popular movie titles are: Pulp Fiction (1994), Forrest Gump (1994), Silence of the Lambs, The (1991), Jurassic Park (1993) and Shawshank Redemption, The (1994). + Top five most popular genres are: Drama, Comedy, Action, Thriller and Adventure.

According to the RMSEs resulted from previous tests, we should account both movie and user effect (with RMSE of 0.8646844) in our prediction. Therefore, I am going to apply use the User and Movie Effects Model for predicting the rating score for our testing set from validation dataset as below. Then, we save the file as submission.csv file.

```
#Predict the rating scores
predicted_rating <- validation %>%
  left_join(movie_avgs, by='movieId') %>%
  left_join(user_avgs, by='userId') %>%
  mutate(pred = mu + b_i + b_u) %>%
  pull(pred)

output <- cbind(validation, predicted_rating) #combine validation and predicted_rating
#Save the output as submission.csv without timestamp. title and genres columns
write.csv(output %>% select(-c(timestamp,title,genres)),"submission.csv", na = "", row.names=FALSE)

#Show top 15 preditec scores
output %>% select(userId,movieId,predicted_rating)%>%head(15)%>% knitr::kable()
```

userId	movieId	predicted_rating
1	231	4.606139
1	480	5.334134
1	586	4.720980
2	151	3.095836
2	858	3.982417
2	1544	2.518681
3	590	4.010929
3	4995	4.168978
4	34	4.400189
4	432	3.429102
4	434	3.770718
5	85	3.589380
5	171	3.699314
5	232	4.174758
5	242	3.512619