Movielens

Rahul Saran

16/06/2019

Introduction

About the dataset

The dataset used for this project is titled 'movielens'. The '10M version' of this dataset is used, which has 10 million observations. This is a dataset of movie ratings. Specifically, this dataset contains 10 million observations, each of which represents a rating given by one user for one movie. The dataset is already in tidy format with no missing data. Specifically, we use 2 datasets - a training set called 'edx' and a test set called 'validation' which have already been created from the movielens dataset.

Goal of the project

The goal is to build a model that predicts the rating that will be given by a particular user to a particular movie. This can be used to develop a recommendation system. A recommendation system suggests movies to users which it predicts users will rate highly.

Key steps that were performed

1) Data exploration and visualization to understand the variables and data provided.
2) Exploratory analysis to understand the impact of various factors on rating.
3) Modelling - the prediction model is built
4) Results from both exploratory analysis & modelling are noted & conclusions identified.

Analysis

This section has 3 major parts as follows

1) Setting up - loading the required packages & data
2) Exploratory Data Analysis & Visualization
3) Modelling

Important Note

This code was written using R 3.6.0.

Setting up

Let's load the required packages.

```
#Install the required packages
if(!require(tidyverse))
  install.packages("tidyverse", repos = "http://cran.us.r-project.org")
if(!require(ggthemes))
  install.packages("ggthemes", repos = "http://cran.us.r-project.org")
if(!require(stringr))
  install.packages("stringr", repos = "http://cran.us.r-project.org")
if(!require(lubridate))
  install.packages("lubridate", repos = "http://cran.us.r-project.org")
```

```
#Load the required packages
library(tidyverse)
library(ggthemes)
library(stringr)
library(lubridate)
```

Let's load the dataset. Ways to load the dataset: 1) I have included the code I have used to download the dataset on my system. 2) Since you may already have the edx & validation datasets on your system, the best way is to load the edx dataset into a variable called 'edx' and the validation dataset into a variable called 'validation' by adding the relevant code here. That will minimize the scope of errors. You may simply replace the filepath for these files on my system with the filepath on your system. 3) I have also included the code to download the dataset and load it as provided. However, this does not work if using R 3.6.0.

```
#The code below loads the data on my system.
#Replace the below filepath with the filepath for these files on your system to load the dataset

edx <- readRDS("C:\\Users\\Rahul Saran\\Desktop\\Data Science\\Capstone\\edx.rds")
validation <- readRDS("C:\\Users\\Rahul Saran\\Desktop\\Data Science\\Capstone\\validation.rds")

#I have also provided below the link to the github repository with the files
# https://github.com/rahsaran/Movielens.git.

#The code below is an alternative to download and loads the data. Do not use this if using R 3.6.0.
#If using this code, please uncomment it (remove one hashtag from each line of the code).

####################################
## Create edx set and validation set
####################################

## Note: this process could take a couple of minutes

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

## MovieLens 10M dataset:
## https://grouplens.org/datasets/movielens/10m/
## http://files.grouplens.org/datasets/movielens/ml-10m.zip

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

#ratings <- read.table(text = gsub("::", "\t", readLines(unzip(dl, "ml-10M100K/ratings.dat"))),
#                      col.names = c("userId", "movieId", "rating", "timestamp"))

#movies <- str_split_fixed(readLines(unzip(dl, "ml-10M100K/movies.dat")), "\\::", 3)
#colnames(movies) <- c("movieId", "title", "genres")
#movies <- as.data.frame(movies) %>% mutate(movieId = as.numeric(levels(movieId))[movieId],
#                                           title = as.character(title),
#                                           genres = as.character(genres))

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

## Validation set will be 10% of MovieLens data
```

```
#set.seed(1) # if using R 3.6.0: set.seed(1, sample.kind = "Rounding")
#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)
```

Exploratory data analysis & visualization

The aims of this major part are - 1) Understanding the structure of the dataset 2) Understanding each of the variables provided 3) Understanding the relationship of the different variables with rating

Overall Dataset

Let's understand the datasets.

```
#Top-level look at data
head(edx)
```

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

```
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 838983653 838984885 83
##  $ title    : chr  "Boomerang (1992)" "Net, The (1995)" "Outbreak (1995)" "Stargate (1994)" ...
##  $ genres   : chr  "Comedy|Romance" "Action|Crime|Thriller" "Action|Drama|Sci-Fi|Thriller" "Action|Ac
```

```
dim(edx)
```

```
## [1] 9000055       6
```

```
dim(validation)
```

```
## [1] 999999      6
```

The edx dataset has 9 million observations. Each observation corresponds to the rating given by one user for one movie. There are 5 feature variables and a prediction/label variable 'rating'.

The validation dataset has 1 million observations.

Rating

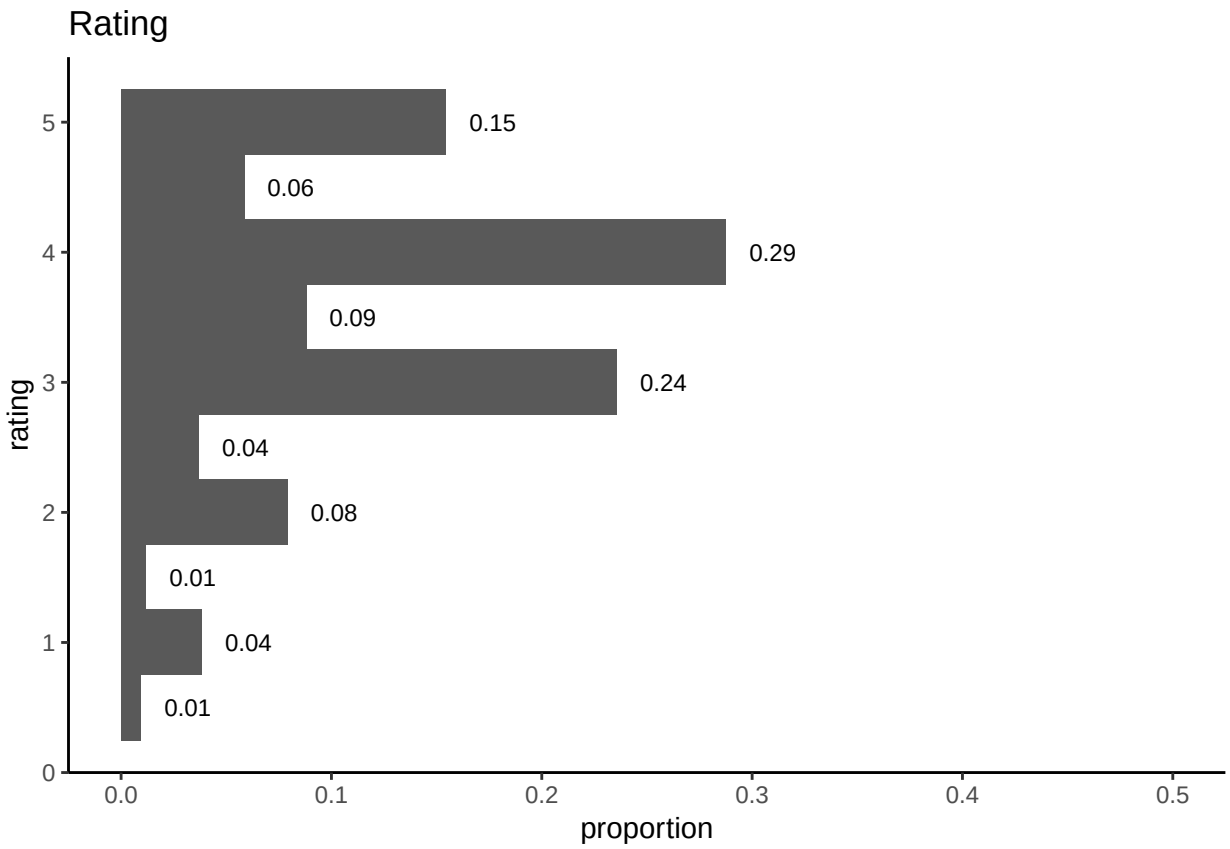
Let's now take a closer look at the 'rating' variable.

```
summary(edx$rating) #top-level look
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   0.500   3.000   4.000   3.512   4.000   5.000
```

```
ratingc <- edx %>% group_by(rating) %>% summarize(n = n(), proportion = n/nrow(edx))
#this creates, for each value of rating, the proportion of movies rated at the value.

#the below is a graphical representation of the above
ratingc %>% ggplot(aes(rating, proportion)) +
  geom_bar(stat = "identity", width = 0.5) +
  ylim(c(0,0.5)) +
  coord_flip() +
  geom_text(label = round(ratingc$proportion, 2), hjust = -0.5, color = "black", size = 3) +
  theme_classic() +
  ggtitle("Rating")
```



rating is a numerical variable, with 9 possible values - 0.5 to 5, at each half-point interval. This corresponds to the number of 'stars' given by users to movies, and higher number of stars are taken as better ratings. The chart shows that ratings of 3 and 4 are the most common, accounting for 53% of observations. The mean rating is 3.51. Also, full-number ratings are more common than half-number ratings.

userId

```
class(edx$userId)
```

```
## [1] "integer"
```

```
length(unique(edx$userId)) #see number of unique users
```

```
## [1] 69878
```

User Id is an integer, denoting a unique ID number for each user. There are 69878 unique users in the edx dataset.

Given that there are 9 million ratings in the edx dataset, this implies that multiple movies may be rated by each user. Let's see the variation in number of movies rated by each user.

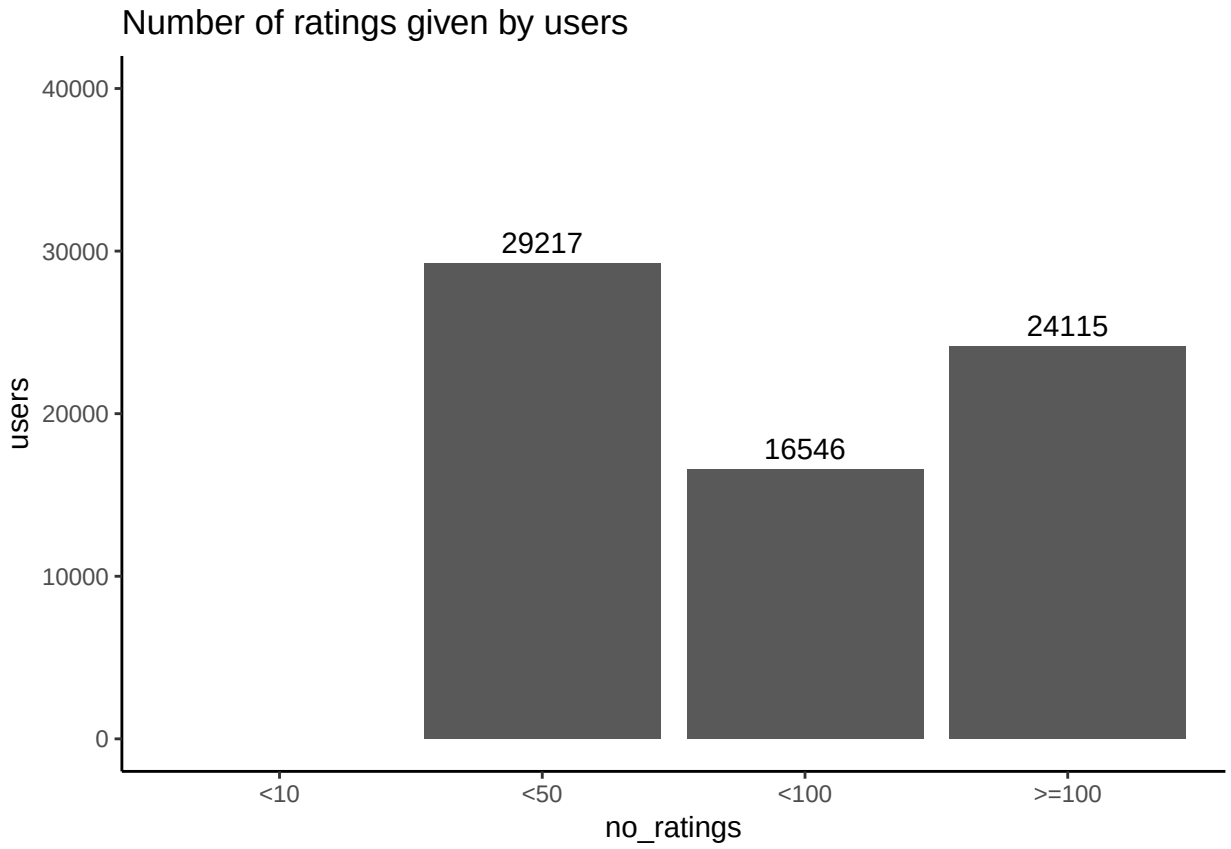
```
userc <- edx %>% group_by(userId) %>% summarize(ratings = n())
#this gives the number of ratings given by each user.
#since there are ~70000 users, it is not useful to see a graphical representation
#of 70000 different values. Instead, it is useful to group users into buckets
#wherein each bucket has users who have given a similar number of ratings.

#the below code creates 4 user buckets - those who have given <10 ratings,
```

```
#those who have given 10-50, those who have given 50-100, and those
#who have given more than 100.
userc <- mutate(userc, no_ratings =
                  ifelse(ratings < 10, "<10",
                         ifelse(ratings < 50, "<50",
                                ifelse(ratings < 100, "<100", ">=100"))))

#the below code gives the number of users in each bucket.
userc2 <-userc %>% group_by(no_ratings) %>% summarize (users = n())

#this code makes a graphical representation of this information.
userc2 %>%
  ggplot(aes(no_ratings, users)) +
  geom_bar(stat = "identity") +
  ylim(c(0,40000)) +
  geom_text(label = userc2$users, vjust = -0.5, color = "black", size = 4) +
  scale_x_discrete(limits = c("<10","<50","<100", ">=100")) +
  theme_classic() +
  ggtitle("Number of ratings given by users")
```



The above exercise shows that each user gave at least 10 movie ratings. About 40% users rated between 10 and 50 movies, 20% rated between 50 and 100, and another 40% rated over 100 movies.

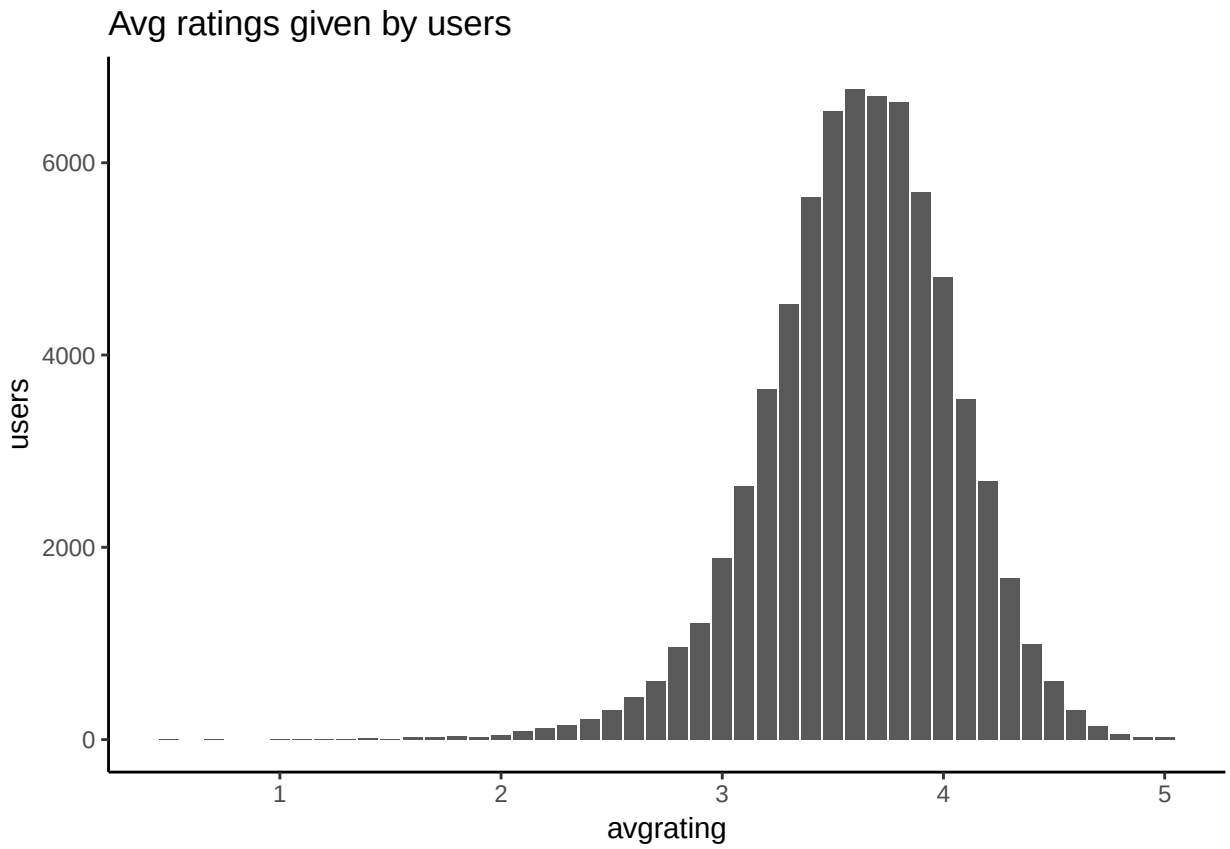
Let's finally see if there is a difference in average rating given by users across the ratings given by them.

```
userc3 <- edx %>% group_by(userId) %>% summarize(avgrating = round(mean(rating),1))
#this code gives the average rating by each user Id.
```

```
#Now since there are ~70000 users, it is not informative to see a graph of average rating
#by each user. Instead, it is useful to see number of users who have given a particular
#average rating.

#The below code calculates number of users who have a particular average rating.
userc4 <- userc3 %>% group_by(avgrating) %>% summarize(users = n())

#The below code makes a graph of this information
userc4 %>%
  ggplot(aes(avgrating, users)) +
  geom_bar(stat = "identity") +
  theme_classic() +
  ggtitle("Avg ratings given by users")
```



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. Several users have average rating of 3 or lower - these can be interpreted as 'critical' or 'selective' users who on average tend to rate movies low. On the other hand, several users have average rating of over 4 - these can be interpreted as users who have a positive feeling towards most movies.

movieId

```
class(edx$movieId)
```

```
## [1] "numeric"
```

```r
length(unique(edx$movieId))
```

```
## [1] 10677
```

movieId is numeric, denoting a unique ID number for each movie. There are 10677 unique movies in the edx dataset.

With ~70000 users and ~10000 movies, if every user were to rate every movie, there would be over 700 million observations in the data. However, since we have about 9 million, this implies that not every user rated every movie; in fact, only about 1-2% of all such possible user-movie combinations are available in the data.

Let's see the variation in number of ratings received by each movie.

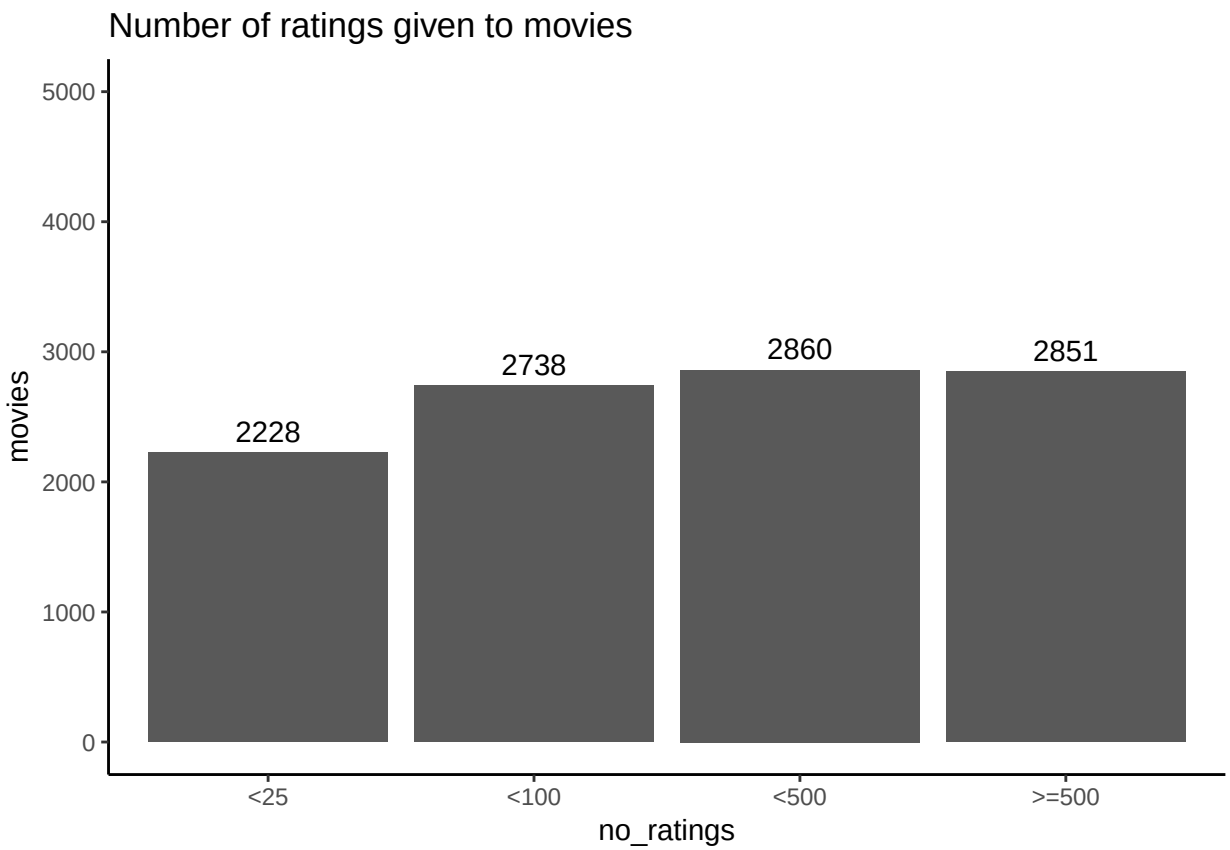
```r
moviec <- edx %>% group_by(movieId) %>% summarize(ratings = n())
#this gives the number of ratings given by each movie.
#since there are ~10000 users, it is not useful to see a graphical representation
#of 10000 different values. Instead, it is useful to group movies into buckets
#wherein each bucket has movies who have received a similar number of ratings.

#the below code creates 4 movie buckets - those who have got <25 ratings,
#those who have got 25-100, those who have got 100-500, and those
#who have got more than 500.

moviec <- mutate(moviec, no_ratings =
                   ifelse(ratings < 25, "<25",
                          ifelse(ratings < 100, "<100",
                                 ifelse(ratings < 500, "<500", ">=500"))))

#the below code gives the number of movies in each bucket.
moviec2 <-moviec %>% group_by(no_ratings) %>% summarize (movies = n())

#The below code makes a graph of this information
moviec2 %>%
  ggplot(aes(no_ratings, movies)) +
  geom_bar(stat = "identity") +
  ylim(c(0,5000)) +
  geom_text(label = moviec2$movies, vjust = -0.5, color = "black", size = 4) +
  scale_x_discrete(limits = c("<25","<100","<500", ">=500")) +
  theme_classic() +
  ggtitle("Number of ratings given to movies")
```



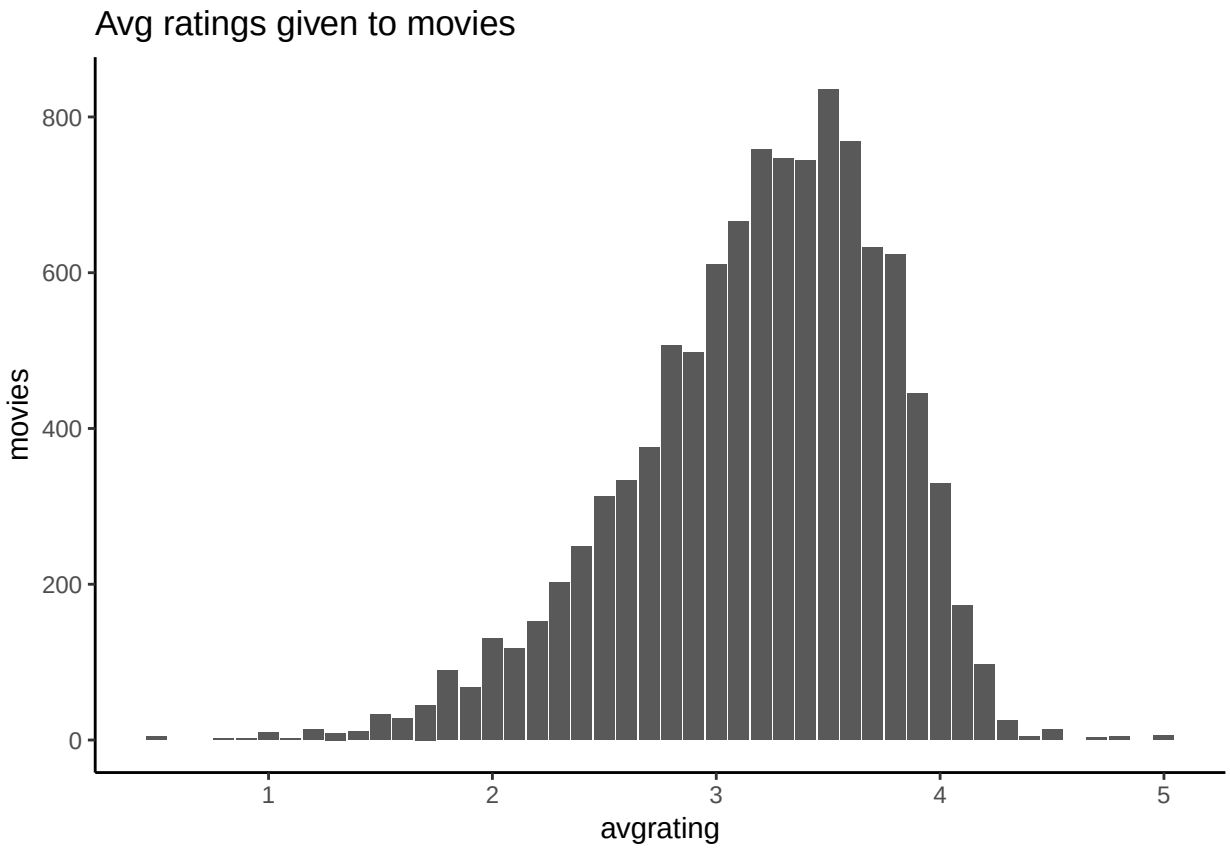
Thus, there is significant variation in number of ratings given to each movie. This is intuitive, since some movies are more popular than others. The data is roughly even divided between movies with <25, 25-100, 100-500, and over 500 ratings.

Let's finally see the variation in average rating given to each movie.

```
moviec3 <- edx %>% group_by(movieId) %>% summarize(avgrating = round(mean(rating),1))
#this code gives the average rating by each movie Id.
#Now since there are ~10000 movies, it is not informative to see a graph of average rating
#by each movie. Instead, it is useful to see number of movies who have got a particular
#average rating.

#The below code calculates number of movies who have got a particular average rating.
moviec4 <- moviec3 %>% group_by(avgrating) %>% summarize(movies = n())

#The below code makes a graph of this information
moviec4 %>%
  ggplot(aes(avgrating, movies)) +
  geom_bar(stat = "identity") +
  theme_classic() +
  ggtitle("Avg ratings given to movies")
```



This chart shows that the average rating given to a movie is a distorted bell curve around the mean (3.5). However, there is significant variation in average rating given to a movie, from 1 to 5. This is intuitive since some movies are perceived in general to be 'good' while others are perceived to be 'bad'.

timestamp

```
class(edx$timestamp)
```

```
## [1] "integer"
```

```
length(unique(edx$timestamp))
```

```
## [1] 6519590
```

Timestamp is an integer, denoting the specific time at which the given user rated the given movie in that observation. There are over 6.5 million unique values. However, this time is not easy to understand in its current format. Let's convert it to another variable in a better format.

```
#this code converts timestamp to a date-time format
edx <- mutate(edx, date = as_datetime(timestamp))
validation <- mutate(validation, date = as_datetime(timestamp))
head(edx$date)
```

```
## [1] "1996-08-02 11:24:06 UTC" "1996-08-02 10:58:45 UTC"
## [3] "1996-08-02 10:57:01 UTC" "1996-08-02 10:56:32 UTC"
## [5] "1996-08-02 10:56:32 UTC" "1996-08-02 11:14:34 UTC"
```

We can clearly see now that the date as well as exact time for when each rating was given is available to us, which is stored in the variable 'date'.

```r
min(edx$date)
```

```
## [1] "1995-01-09 11:46:49 UTC"
```

```r
max(edx$date)
```

```
## [1] "2009-01-05 05:02:16 UTC"
```

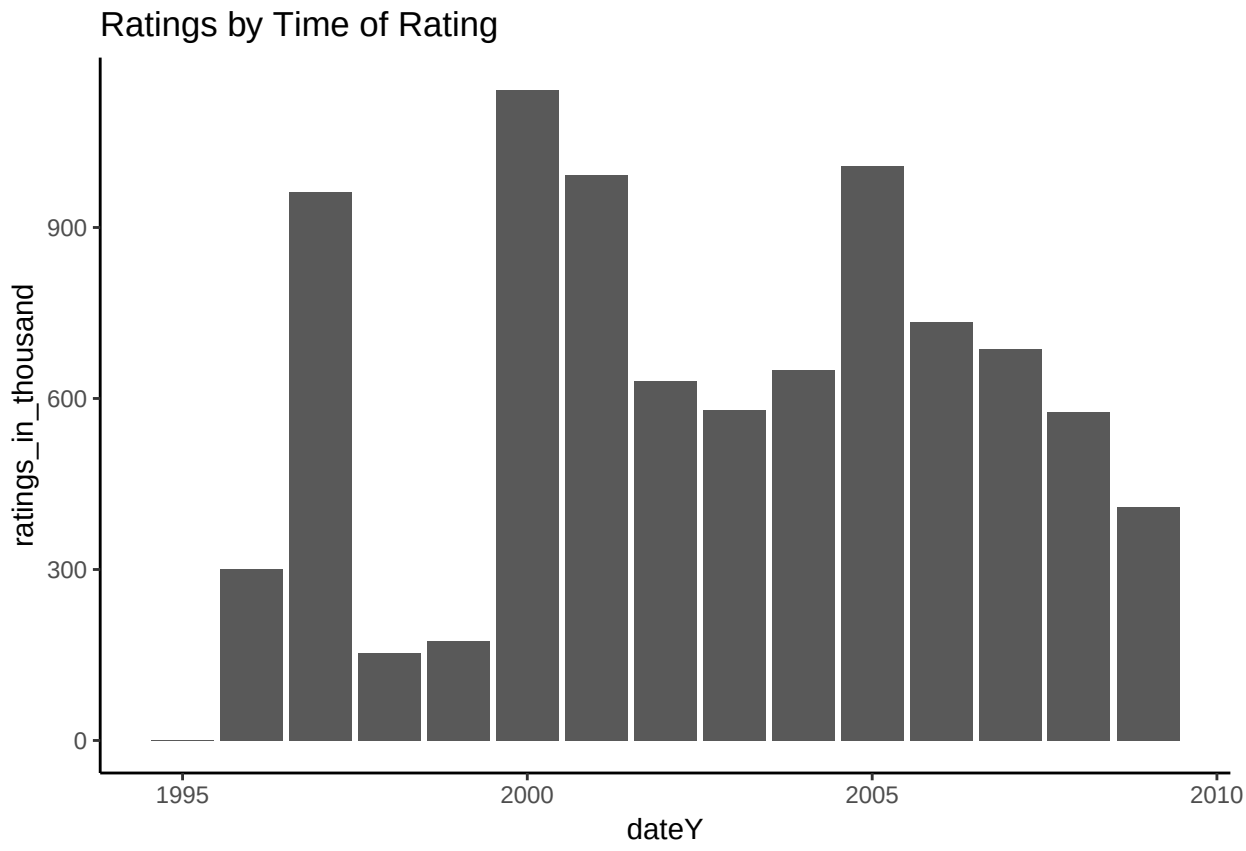
Thus, the dataset has ratings given over a time period from September 1995 to May 2009 - a 14-year period!

Let's now see if the number of movies rated varies by time in our dataset. Since there are millions of unique observations, we group time by year.

```r
edx <- mutate(edx, dateY = round_date(date, unit = "year"))
#this converts each date to the nearest year

dateYc <- edx %>% group_by(dateY) %>% summarize(ratings_in_thousand = n()/(10^3))
#this gives the number of ratings given in each year.

#this makes a chart of this information
dateYc %>%
  ggplot(aes(dateY, ratings_in_thousand)) +
  geom_bar(stat = "identity") +
  theme_classic() +
  ggtitle("Ratings by Time of Rating")
```

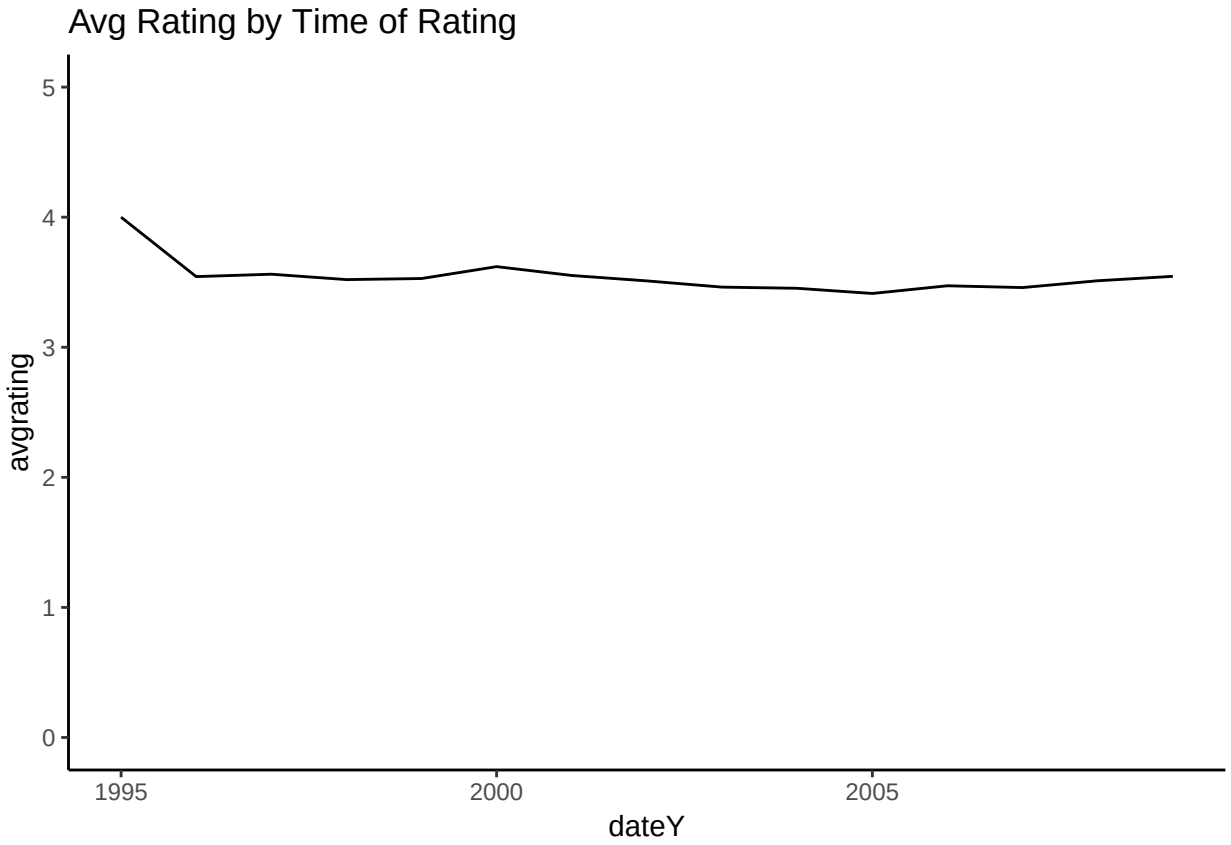


This shows that more ratings were given in the 2000s, though the ratings are distributed across years.

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

```r
dateYc2 <- edx %>% group_by(dateY) %>% summarize(avgrating = mean(rating))
#this gives the average rating given in each year.

#this makes a chart of this information.
dateYc2 %>%
  ggplot(aes(dateY, avgrating)) +
  geom_line() +
  ylim(c(0,5)) +
  theme_classic() +
  ggtitle("Avg Rating by Time of Rating")
```



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

title

```r
class(edx$title)
```

```
## [1] "character"
```

```r
length(unique(edx$title))
```

```
## [1] 10676
```

Title is a character variable. There are 10676 unique movie titles (for some reason, this is one less than the number of unique movie IDs. This implies that 2 movie IDs have been given to the same movie in 1 case. However, this should not affect our anlaysis much). Title clearly contains the name of each movie, along with its year of release. It is actually thus containing 2 different parameters in one column. It may be useful to

separate the year of release of the movie into a separate variable.

```
#this code separates out the year, which is always the 5th-last to 2nd-last characters in title.
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)))
head(edx$year)
```

```
## [1] 1992 1995 1995 1994 1994 1994
```

Now we have the year of release as a separate variable 'year'. Let's examine this variable.

```
min(edx$year)
```

```
## [1] 1915
```

```
max(edx$year)
```

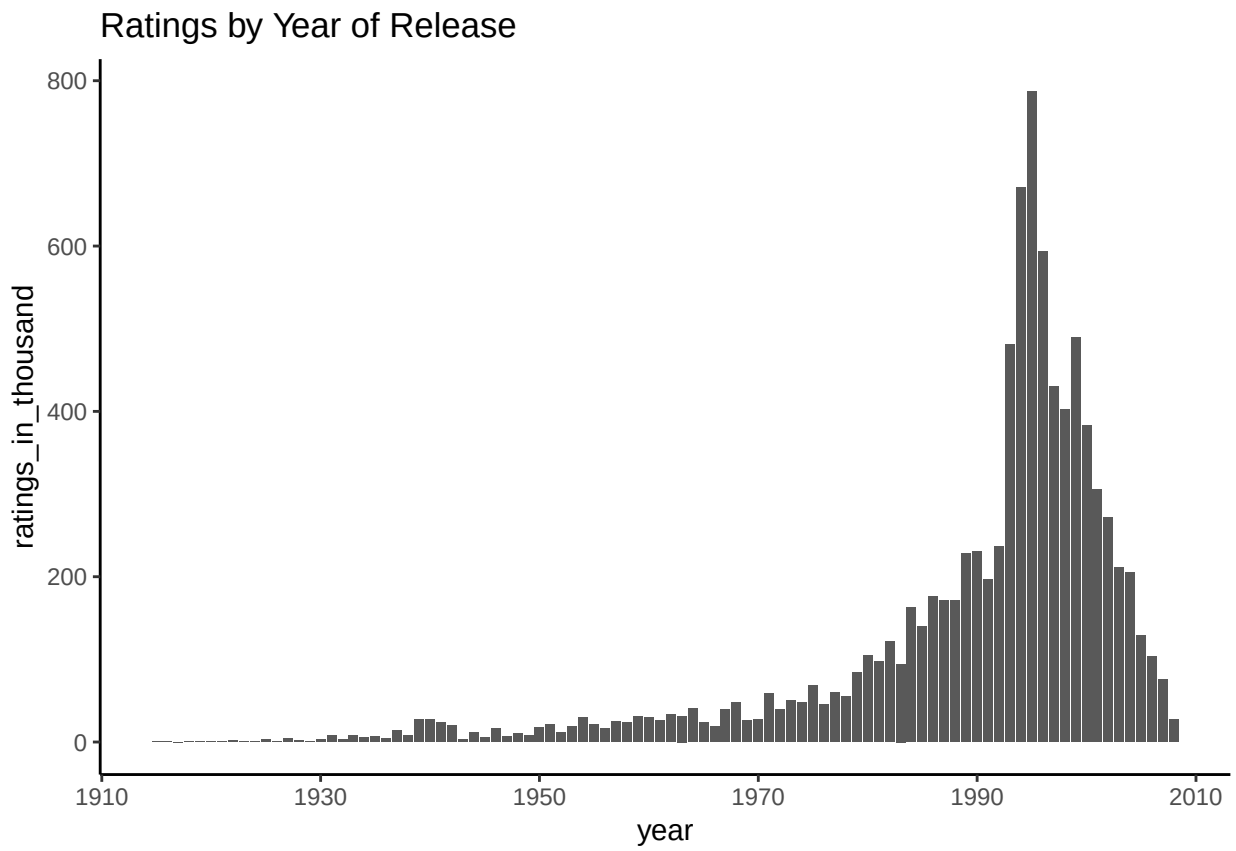
```
## [1] 2008
```

This shows that the dataset has movies released from as early as 1915 to as recent as 2008.

Let's see if the number of ratings given to movies varies by their year of release.

```
yearc <- edx %>% group_by(year) %>% summarize(ratings_in_thousand = n()/(10^3))
#this code gives the number of ratings by each year.

#this code makes a graph of this information
yearc %>%
  ggplot(aes(year, ratings_in_thousand)) +
  geom_bar(stat = "identity") +
  theme_classic() +
  ggtitle("Ratings by Year of Release")
```

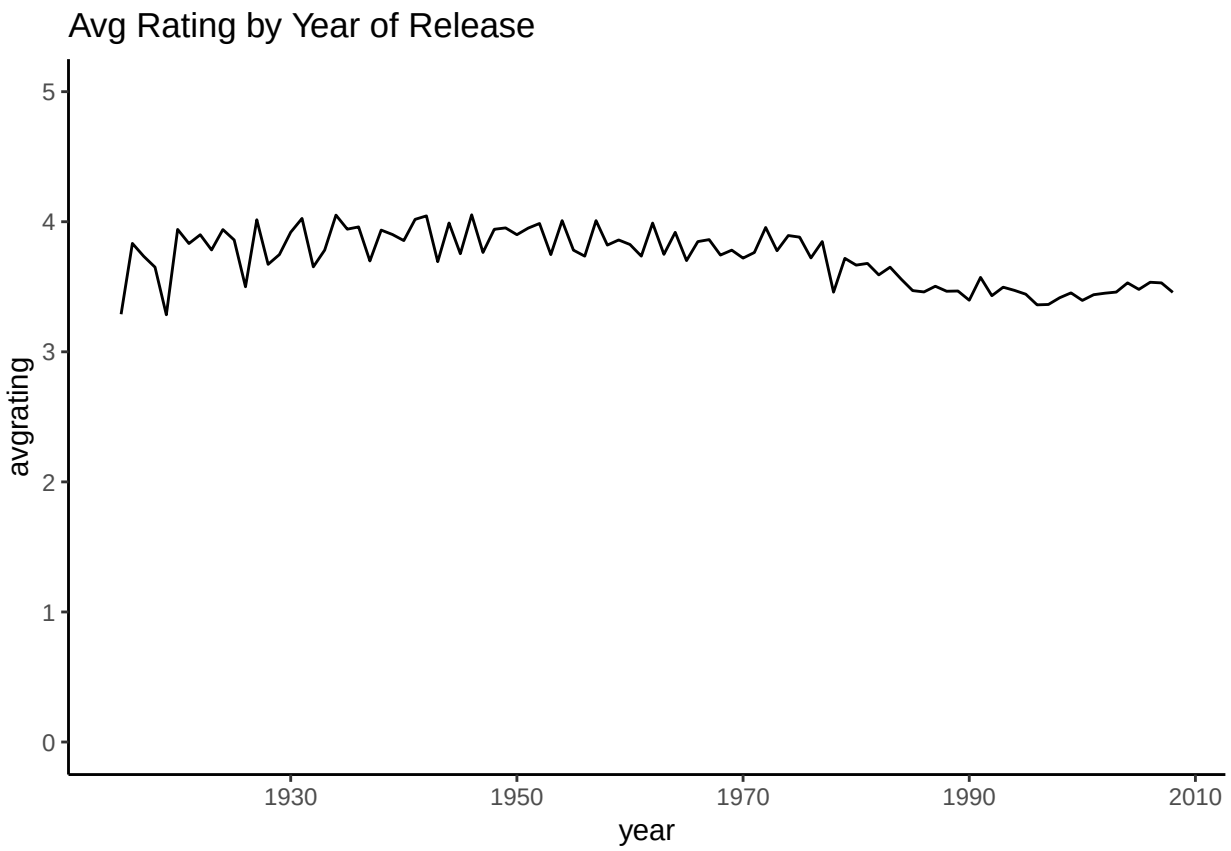


Thus, we see that there are a lot more ratings given to movies released post mid-1990s. This makes sense since this dataset has ratings given from 1995 onwards, and there would be more ratings expected by users for current movies than for older movies.

Let's finally see if the average rating given varies by year of release of the movie.

```
yearc2 <- edx %>% group_by(year) %>% summarize(avgrating = mean(rating))
#this code gives the average rating by each year

#this code makes a graph of this information
yearc2 %>%
  ggplot(aes(year, avgrating)) +
  geom_line() +
  ylim(c(0,5)) +
  theme_classic() +
  ggtitle("Avg Rating by Year of Release")
```



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.

genres

```
class(edx$genres)
```

```
## [1] "character"
```

```
head(edx$genres)
```

```
## [1] "Comedy|Romance"                "Action|Crime|Thriller"
## [3] "Action|Drama|Sci-Fi|Thriller"  "Action|Adventure|Sci-Fi"
## [5] "Action|Adventure|Drama|Sci-Fi" "Children|Comedy|Fantasy"
```

```
length(unique(edx$genres))
```

```
## [1] 797
```

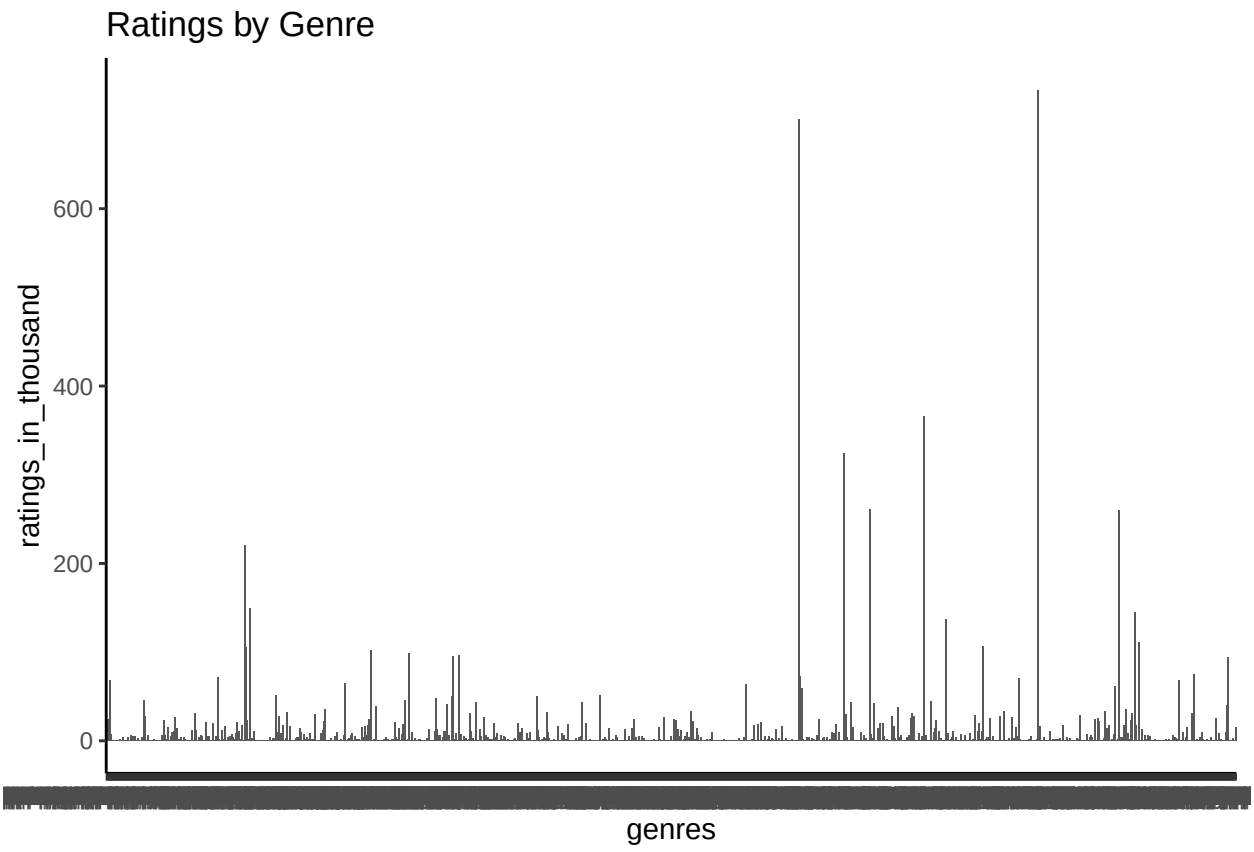
genres is a character variable. The 'genres' variable contains all the genres that that particular movie can be said to belong to. This number of genres for each movie varies anywhere from 1 to 4. There are thus 797 unique 'combinations' of genres across the movies.

Let's see the average number of ratings for each combination of genres.

```
genrec <- edx %>% group_by(genres) %>% summarize(ratings_in_thousand = n()/(10^3))
#this gives the number of ratings for each value of 'genres'

#this code gives a graph of this information
```

```
genrec %>%
  ggplot(aes(genres, ratings_in_thousand)) +
  geom_bar(stat = "identity") +
  theme_classic() +
  ggtitle("Ratings by Genre")
```



Thus, the average number of ratings for each value of 'genres' is low due to the high number of combinations (800). Thus, many of these may be affected by small bases and be difficult to use in our analysis.

On further study of the genres variable, all the combinations are made of 17 'base' genres. I attempted to break down the data into the constitutent genres using the following code, but it led to my computer crashing.

```
# the following code breaks down the data for a movie into a separate row for each 'genre'
# in the 'genres' variable. However, it may lead to crashing of the computer.
# edx %>% separate_rows(genres, sep = "\\|")
```

What is possible is to identify for each movie whether one of these 17 constituent genres is one of the movie's genres.

```
#the below code identifies whether the given text, example "Action", is part of the character
#list in the 'genres' column.

#edx <- mutate(edx,
#                   Action = grepl("Action", genres, fixed = TRUE),
#                   Adventure = grepl("Adventure", genres, fixed = TRUE),
#                   Animation = grepl("Animation", genres, fixed = TRUE),
#                   Children = grepl("Children", genres, fixed = TRUE),
#                   Comedy = grepl("Comedy", genres, fixed = TRUE),
```

```
#                   Crime = grepl("Crime", genres, fixed = TRUE),
#                   Drama = grepl("Drama", genres, fixed = TRUE),
#                   Fantasy = grepl("Fantasy", genres, fixed = TRUE),
#                   FilmNoir = grepl("Film-Noir", genres, fixed = TRUE),
#                   Horror = grepl("Horror", genres, fixed = TRUE),
#                   Musical = grepl("Musical", genres, fixed = TRUE),
#                   Mystery = grepl("Mystery", genres, fixed = TRUE),
#                   Romance = grepl("Romance", genres, fixed = TRUE),
#                   SciFi = grepl("Sci-Fi", genres, fixed = TRUE),
#                   Thriller = grepl("Thriller", genres, fixed = TRUE),
#                   War = grepl("War", genres, fixed = TRUE),
#                   Western = grepl("Western", genres, fixed = TRUE)
# )
```

With this, it is possible to identify the number of ratings or average rating for a particular genre. However, since these are 17 separate variables, if these are all used in modelling, it again crashes the system given the size of the dataset.

Modelling

Let's now build our prediction model.

First, let's create a loss function RMSE, which will be our primary way of determining the effectiveness of our model.

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

This calculates the square root of the mean squared difference of the true rating & our predicted rating. This difference is essentially how many 'stars' our predicted rating is off from the true rating. The aim will be to make this difference as low as possible.

Now let's create a table to hold our results. We'll first test our model on the edx training set itself, and once our model is complete we will obtain results on the validation test set.

```
#create a results table
rmse_results <- data.frame(method = character(), RMSE = numeric())
```

Base

To start, let's create a base rating for all movies across all users. The best rating for this purpose is the mean rating.

```
#predict all as average rating
mu <- mean(edx$rating)
mu
```

```
## [1] 3.512465
```

Let's note the result.

```
rmse_1 <- RMSE(edx$rating, mu)
rmse_1
```

```
## [1] 1.060331
```

```
rmse_results <- data.frame(method = "Mean Rating", RMSE = rmse_1)
rmse_results %>% knitr::kable()
```

method	RMSE
Mean Rating	1.060331

Thus, the predicted rating is off by more than 1 star.

Add Movie Variation

Since the task is to predict the rating for a particular movie, it makes sense to take that movie's average rating into account. As noted in the exploratory analysis, the average rating for a movie varies significantly across movies - some are good, while others are bad. Let's add that variation to the model.

```
#add movie variation
movie_var <- edx %>%
  group_by(movieId) %>%
  summarize(b_i = mean(rating - mu))
predicted_ratings <- mu + edx %>%
  left_join(movie_var, by = "movieId") %>%
  .$b_i
```

The model now essentially predicts, for every movie, the average rating of that movie.

```
rmse_2 <- RMSE(predicted_ratings, edx$rating)
rmse_results <- bind_rows(rmse_results,
                          data.frame(method = "Movie Variation", RMSE = rmse_2))
rmse_results %>% knitr::kable()
```

method	RMSE
Mean Rating	1.0603313
Movie Variation	0.9423475

Thus, the RMSE improves to 0.942.

Add User Variation

Currently the model predicts, for every movie, the average rating of that movie. However, as noted in the exploratory analysis, the average rating of a user varies across users. Thus, the same movie may be rated differently by users based on their own tendencies (critical or favourable). Let's add the user variation to the model. It is important to identify the additional variation due to users over and above the variation already accounted for by movie.

```
#add user variation
user_var <- edx %>%
  left_join(movie_var, by = "movieId") %>%
  group_by(userId) %>%
  summarize(b_u = mean(rating - mu - b_i))
predicted_ratings <- edx %>%
  left_join(movie_var, by = "movieId") %>%
  left_join(user_var, by = "userId") %>%
  mutate(pred = mu + b_i + b_u) %>% .$pred
```

The model now accounts for the impact of whether a movie is good or bad, and whether a user is critical or favourable. Let's now note the results.

```
rmse_3 <- RMSE(predicted_ratings, edx$rating)
rmse_results <- bind_rows(rmse_results,
                          data.frame(method = "Movie + User Variation", RMSE = rmse_3))
rmse_results %>% knitr::kable()
```

method	RMSE
Mean Rating	1.0603313
Movie Variation	0.9423475
Movie + User Variation	0.8567039

The RMSE is now 0.856.

Add Time variation

As noted in the exploratory analysis, there is a small difference in the average rating depending on the time at which the rating was made. It may be useful to account for the additional impact of that variation in our model as well. However, there were 6.5 million unique values of the time of rating. Utilizing all those is neither feasible nor advisable. As seen in the exploratory chart, the variation was slight and over a long time period. Thus, we may not lose much info by averaging over a longer time period. A week is advisable, since that will be precise enough and yet feasible (over a 14-year period from 1995 to 2009, there will be approximately 700 weeks). Let's convert our date into weeks. We also do this for validation since it will be used later.

```
edx <- mutate(edx, dateW = round_date(date, unit = "week"))
validation <- mutate(validation, dateW = round_date(date, unit = "week"))
```

Let's now add the variation due to time of rating.

```
#add time variation
week_var <- edx %>%
  left_join(movie_var, by = "movieId") %>%
  left_join(user_var, by = "userId") %>%
  group_by(dateW) %>%
  summarize(b_t = mean(rating - mu - b_i - b_u))

predicted_ratings <- edx %>%
  left_join(movie_var, by = "movieId") %>%
  left_join(user_var, by = "userId") %>%
  left_join(week_var, by = "dateW") %>%
  mutate(pred = mu + b_i + b_u + b_t) %>% .$pred
```

Let's note the results.

```
rmse_4 <- RMSE(predicted_ratings, edx$rating)
rmse_results <- bind_rows(rmse_results,
                          data.frame(method = "Movie + User + Time Variation", RMSE = rmse_4))
rmse_results %>% knitr::kable()
```

method	RMSE
Mean Rating	1.0603313
Movie Variation	0.9423475
Movie + User Variation	0.8567039

method	RMSE
Movie + User + Time Variation	0.8566027

The RMSE improves only marginally, from 0.8567 to 0.8566.

Add Year variation

As noted in the exploratory analysis, there is a small difference in the average rating depending on the year the movie was released. It may be useful to account for the additional impact of that variation in our model as well.

```
#add year variation
year_var <- edx %>%
  left_join(movie_var, by = "movieId") %>%
  left_join(user_var, by = "userId") %>%
  left_join(week_var, by = "dateW") %>%
  group_by(year) %>%
  summarize(b_y = mean(rating - mu - b_i - b_u - b_t))

predicted_ratings <- edx %>%
  left_join(movie_var, by = "movieId") %>%
  left_join(user_var, by = "userId") %>%
  left_join(week_var, by = "dateW") %>%
  left_join(year_var, by = "year") %>%
  mutate(pred = mu + b_i + b_u + b_t + b_y) %>% .$pred

rmse_5 <- RMSE(predicted_ratings, edx$rating)
rmse_results <- bind_rows(rmse_results,
                          data.frame(method = "Movie + User + Time + Year Variation",
                                     RMSE = rmse_5))
rmse_results %>% knitr::kable()
```

method	RMSE
Mean Rating	1.0603313
Movie Variation	0.9423475
Movie + User Variation	0.8567039
Movie + User + Time Variation	0.8566027
Movie + User + Time + Year Variation	0.8562735

This improves our RMSE to 0.8562.

This completes our model.

Results

Let's now apply our model on the validation set and see the results.

```
#this code applies the entire set of processes carried out in the model together.

mu_val <- mean(validation$rating)

movie_var_val <- validation %>%
```

```
  group_by(movieId) %>%
  summarize(b_i_val = mean(rating - mu_val))

user_var_val <- validation %>%
  left_join(movie_var_val, by = "movieId") %>%
  group_by(userId) %>%
  summarize(b_u_val = mean(rating - mu_val - b_i_val))

week_var_val <- validation %>%
  left_join(movie_var_val, by = "movieId") %>%
  left_join(user_var_val, by = "userId") %>%
  group_by(dateW) %>%
  summarize(b_t_val = mean(rating - mu_val - b_i_val - b_u_val))

year_var_val <- validation %>%
  left_join(movie_var_val, by = "movieId") %>%
  left_join(user_var_val, by = "userId") %>%
  left_join(week_var_val, by = "dateW") %>%
  group_by(year) %>%
  summarize(b_y_val = mean(rating - mu_val - b_i_val - b_u_val - b_t_val))

predicted_ratings <- validation %>%
  left_join(movie_var_val, by = "movieId") %>%
  left_join(user_var_val, by = "userId") %>%
  left_join(week_var_val, by = "dateW") %>%
  left_join(year_var_val, by = "year") %>%
  mutate(pred = mu_val + b_i_val + b_u_val + b_t_val + b_y_val) %>% .$pred
```

Let's note the result of this model on the validation set.

```
rmse_final <- RMSE(predicted_ratings, validation$rating)
rmse_results <- bind_rows(rmse_results,
                          data.frame(method = "Final - Test", RMSE = rmse_final))
rmse_results %>% knitr::kable()
```

method	RMSE
Mean Rating	1.0603313
Movie Variation	0.9423475
Movie + User Variation	0.8567039
Movie + User + Time Variation	0.8566027
Movie + User + Time + Year Variation	0.8562735
Final - Test	0.8246169

Thus, this model gives us an RMSE of 0.824. The RMSE is even better than that obtained on the training edx set, which indicates no overfitting.

Conclusions

The movielens dataset gives the opportunity to create a movie recommendation system from past user rating data.

1) There is a definite variation in the average rating given among users. It appears that some users are on average critical about movies while some are pleased with most movies. This variation could also be

due to a difference in interpreting the movie rating scales - one user might regard 3 on 5 as a 'good' rating, while another user might regard it is an 'average' rating.

2) There is a definite variation in the average rating given to movies. This is intuitive, since some movies are generally agreed to be 'good' while other movies are generally agreed on being 'bad'.

3) There is a slight variation in rating given by the time of rating, though this variation does not appear to be significant. This is also shown in our modelling exercise when including the time of rating does not influence the model to a great extent.

4) There is a variation in rating given depending on year of release of the movie. In particular, movies released in the early 1900s and mid-1990s onwards are rated slightly lower, while movies rated in the time period 1950-1990 are rated slightly higher. The dataset has data of ratings made from 1996 onwards. While users may view current movies more, there is a tendency to take movies which have released some time in the past as 'classics' due to which the rating of earlier movies may be higher.

5) Accounting for these variations allows us to build a model with an error of about 0.825. This is a strong model given the few predictor variables used - it shows that just given the movie & user past history, as well as some time information, it is possible to develop a model that predicts user ratings within less than one star.

6) While we have a model which estimates a user's rating for a particular movie, there is scope to further analyze whether this is sufficient to build a movie recommendation system. For example, even if a model predicts that a particular user will highly rate 5 movies from the 1950s, it may not make sense to recommend all of these since a user may not have heard of any of them and actually feel that the movie recommendations are poor. Utilizing these ratings along with an understanding of what makes users likely to agree to watch a particular movie will lead to a strong recommendation system.