Movielens Recommendation System

Patricia Londono

January, 2019

```
knitr::opts_chunk$set(echo = TRUE)
```

Introduction

As per Wikipedia, "a recommender system or a recommendation system is a subclass of information filtering system that seeks to predict the"rating" or "preference" a user would give to an item". These systems are largely used in several industries to ensure customers satisfaction and increase sales. As an example of this, in 2006 Netflix offered a million-dollar prize to anyone capable of improving their recommender system by 10%.

This project is an attempt to build a recommendation system for the 10M Movielens Dataset. The data used for this project was downloaded from the Grouplens website:

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

Goal

Train a machine learning algorithm capable of predicting movie ratings for the MovieLens dataset.

Methodology

Five main steps will be followed:

1. Data preparation
2. Exploratory data analysis
3. Model Training
4. Analysis of Results
5. Conclusions

Data Split

Data will be split into training and validation sets (10%)

Metric

The RMSE (Root Mean Squared Deviation) will be the metric used to evaluate the quality of the model.

1. Data preparation

```
# Loading required libraries
library(tidyverse)
```

```
## -- Attaching packages ---------------------------------------------------------------------------
## v ggplot2 3.1.0     v purrr   0.2.5
## v tibble  1.4.2     v dplyr   0.7.7
## v tidyr   0.8.1     v stringr 1.3.1
## v readr   1.1.1     v forcats 0.3.0
```

```
## -- Conflicts -------------------------------------------------------------------------------------
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```
library(caret)
```

```
## Loading required package: lattice
```

```
##
## Attaching package: 'caret'
```

```
## The following object is masked from 'package:purrr':
##
##     lift
```

```
library(tidyr)
library(ggplot2)
library(lubridate)
```

```
##
## Attaching package: 'lubridate'
```

```
## The following object is masked from 'package:base':
##
##     date
```

```
# Downloading the dataset

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)
test_index <- createDataPartition(y = movielens$rating, times = 1, p = 0.1, list = FALSE)
edx <- movielens[-test_index,]
temp <- movielens[test_index,]

# Make sure userId and movieId in validation set are also in edx set

validation <- temp %>%
     semi_join(edx, by = "movieId") %>%
     semi_join(edx, by = "userId")

# Add rows removed from validation set back into edx set
```

```
removed <- anti_join(temp, validation)
```

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

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

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

#  Defining year column

edx <- edx %>%
  mutate(year = as.numeric(str_sub(title,-5,-2)))
validation <- validation %>%
  mutate(year = as.numeric(str_sub(title,-5,-2)))

# Defining test and validation sets

validation_t <- validation
validation <- validation %>% select(-rating)


# Defining RMSE Metric

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

2. Exploratory data analysis

The following table shows the structure of the dataset. Each row represents a rating given by one user to one movie.

```
head(edx)
```

```
##   userId movieId rating timestamp                          title
## 1      1     122      5 838985046              Boomerang (1992)
## 2      1     185      5 838983525               Net, The (1995)
## 3      1     292      5 838983421               Outbreak (1995)
## 4      1     316      5 838983392               Stargate (1994)
## 5      1     329      5 838983392 Star Trek: Generations (1994)
## 6      1     355      5 838984474       Flintstones, The (1994)
##                         genres year
## 1                Comedy|Romance 1992
## 2         Action|Crime|Thriller 1995
## 3  Action|Drama|Sci-Fi|Thriller 1995
## 4       Action|Adventure|Sci-Fi 1994
## 5 Action|Adventure|Drama|Sci-Fi 1994
## 6       Children|Comedy|Fantasy 1994
```

Dataset dimensions

```
dim(edx)
```

```
## [1] 9000055       7
```

```
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              year
##  Length:9000055     Length:9000055     Min.   :1915
##  Class :character   Class :character   1st Qu.:1987
##  Mode  :character   Mode  :character   Median :1994
##                                        Mean   :1990
##                                        3rd Qu.:1998
##                                        Max.   :2008
```

Number of users and movies

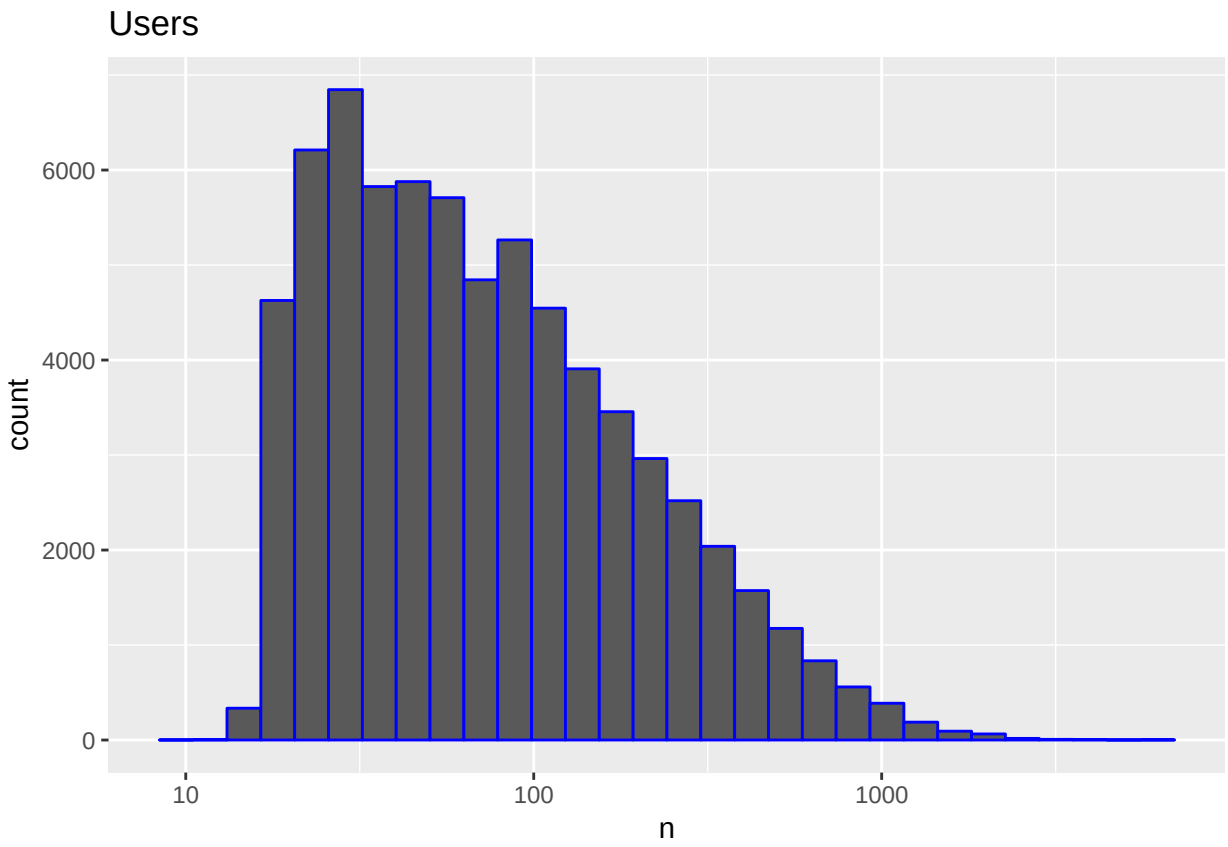
```
edx %>%
  summarize(n_users = n_distinct(userId),
            n_movies = n_distinct(movieId))
```

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

Distribituion of Users

This plot below shows that most users have rated less than 100 movies, with the number of movies rated per user peaking at around 40

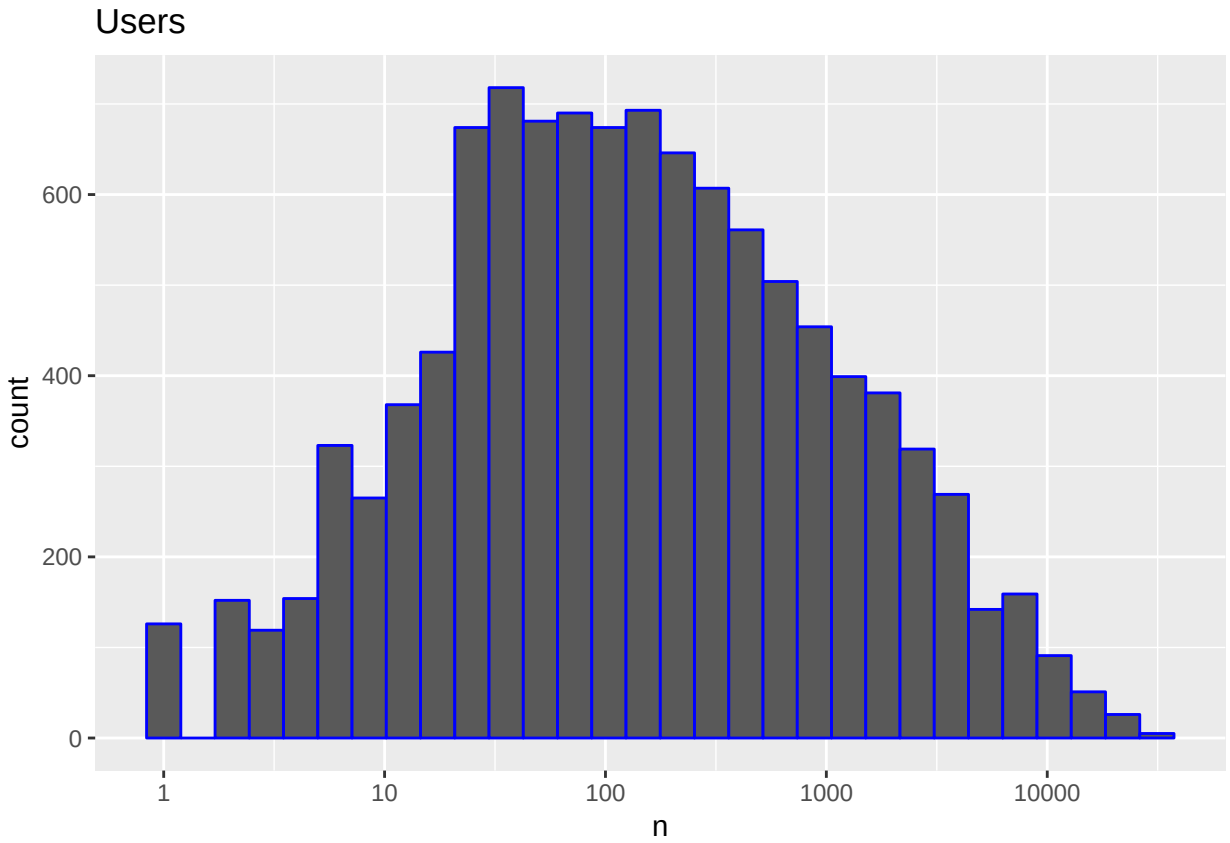
```
edx %>%
  count(userId) %>%
  ggplot(aes(n)) +
  geom_histogram(bins = 30, color = "blue") +
  scale_x_log10() +
  ggtitle("Users")
```



Distribution of Movies

This plot shows that most movies have been rated between 50 and 1000. This could be explained due to the popularity of certain blockbuster movies.

```
edx %>%
  count(movieId) %>%
  ggplot(aes(n)) +
  geom_histogram(bins = 30, color = "blue") +
  scale_x_log10() +
  ggtitle("Users")
```



Number of Genres

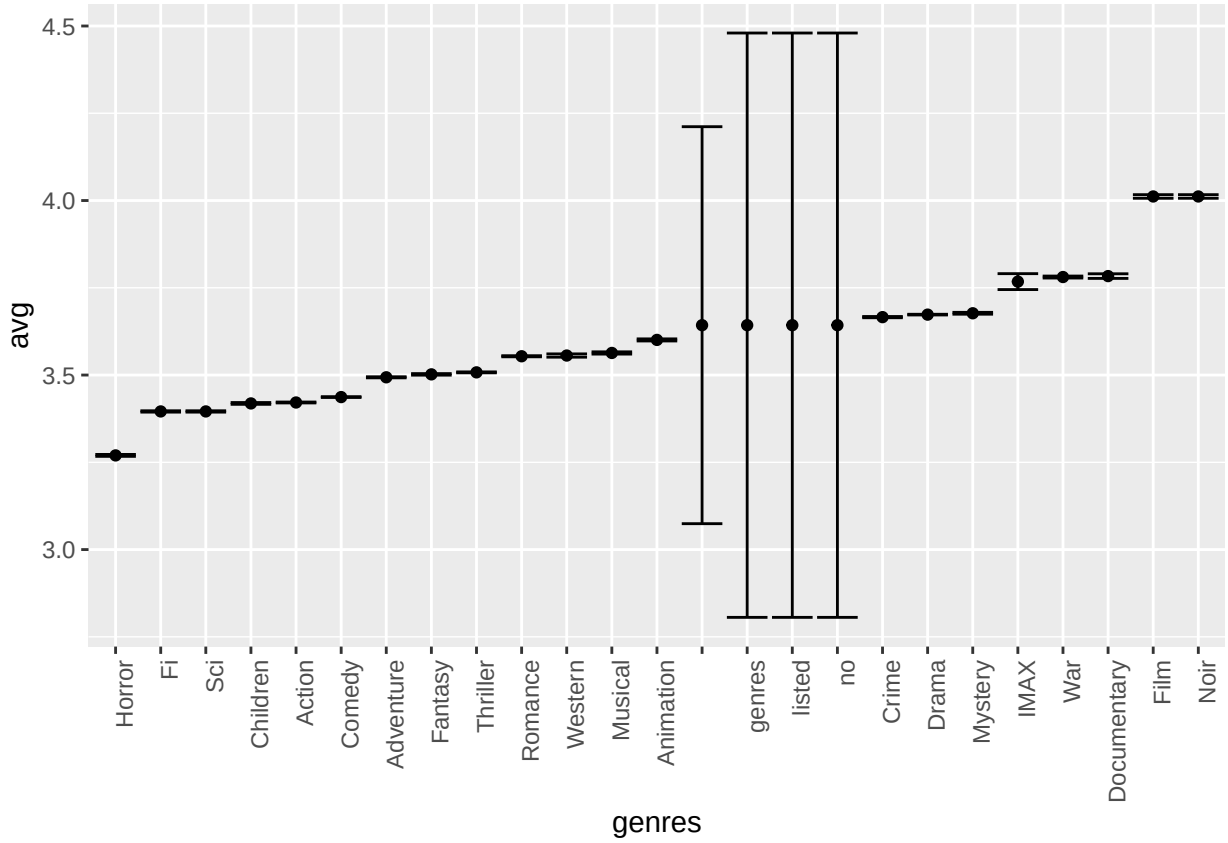
This table shows the number of movie ratings per genre

```
edx %>%
  separate_rows(genres) %>%
  group_by(genres) %>%
  summarize(n = n()) %>%
  arrange(desc(n))
```

```
## # A tibble: 25 x 2
##    genres          n
##    <chr>       <int>
##  1 Drama     3910127
##  2 Comedy    3540930
##  3 Action    2560545
##  4 Thriller  2325899
##  5 Adventure 1908892
##  6 Romance   1712100
##  7 Fi        1341183
##  8 Sci       1341183
##  9 Crime     1327715
## 10 Fantasy    925637
## # ... with 15 more rows
```

This plot shows how different genres are rated in average, some being more popular than others.

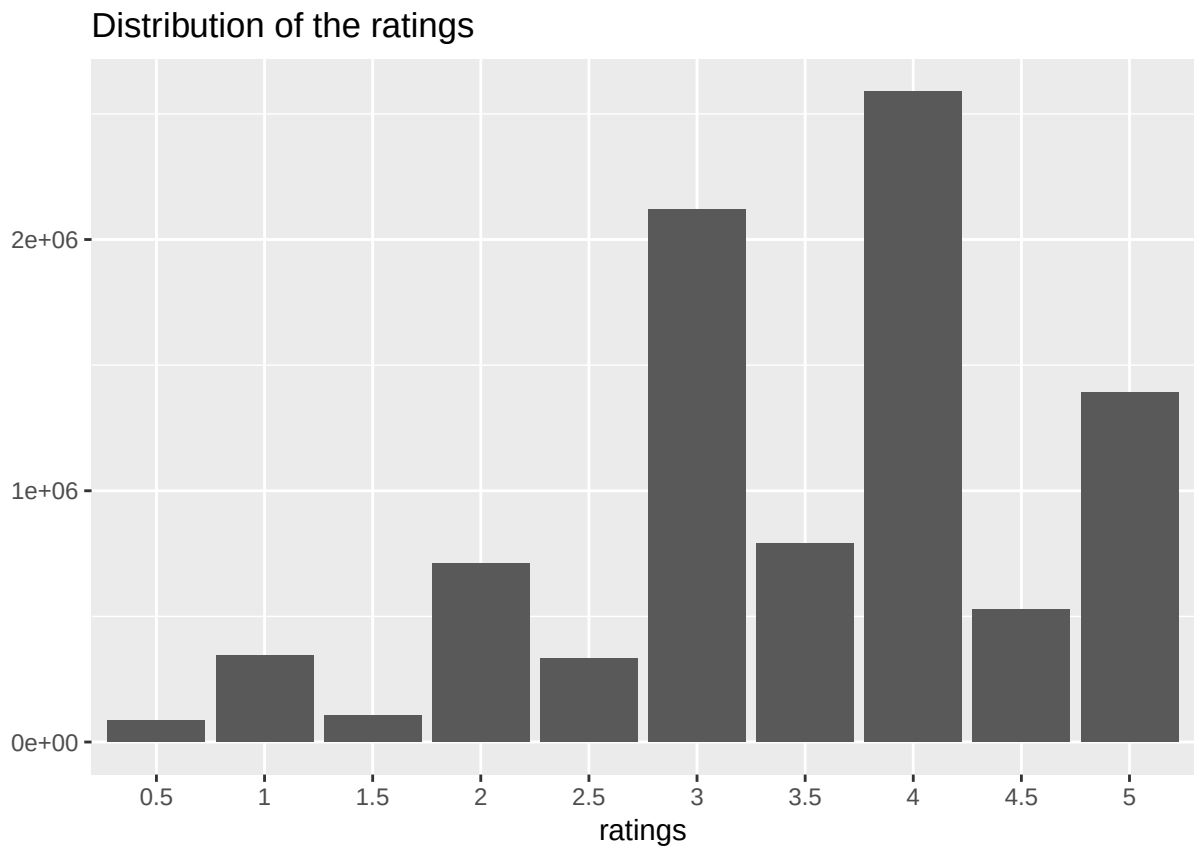
```
edx %>%
  select(movieId, genres, rating) %>%
  separate_rows(genres) %>%
  group_by(genres) %>%
  summarize(n = n(), avg = mean(rating), se = sd(rating)/sqrt(n())) %>%
  mutate(genres = reorder(genres, avg)) %>%
  ggplot(aes(x = genres, y = avg, ymin = avg - 2*se, ymax = avg + 2*se)) +
  geom_point() +
  geom_errorbar() +
  theme(axis.text.x = element_text(angle = 90, hjust = 1))
```



Distribution of Ratings

This plot shows that most movies are rated using whole numbers, with 3 and 4 being the most popular ratings given to movies across the dataset

```
ratings <- as.vector(edx$rating)
ratings <- ratings[ratings != 0]
ratings <- factor(ratings)
qplot(ratings) +
  ggtitle("Distribution of the ratings")
```

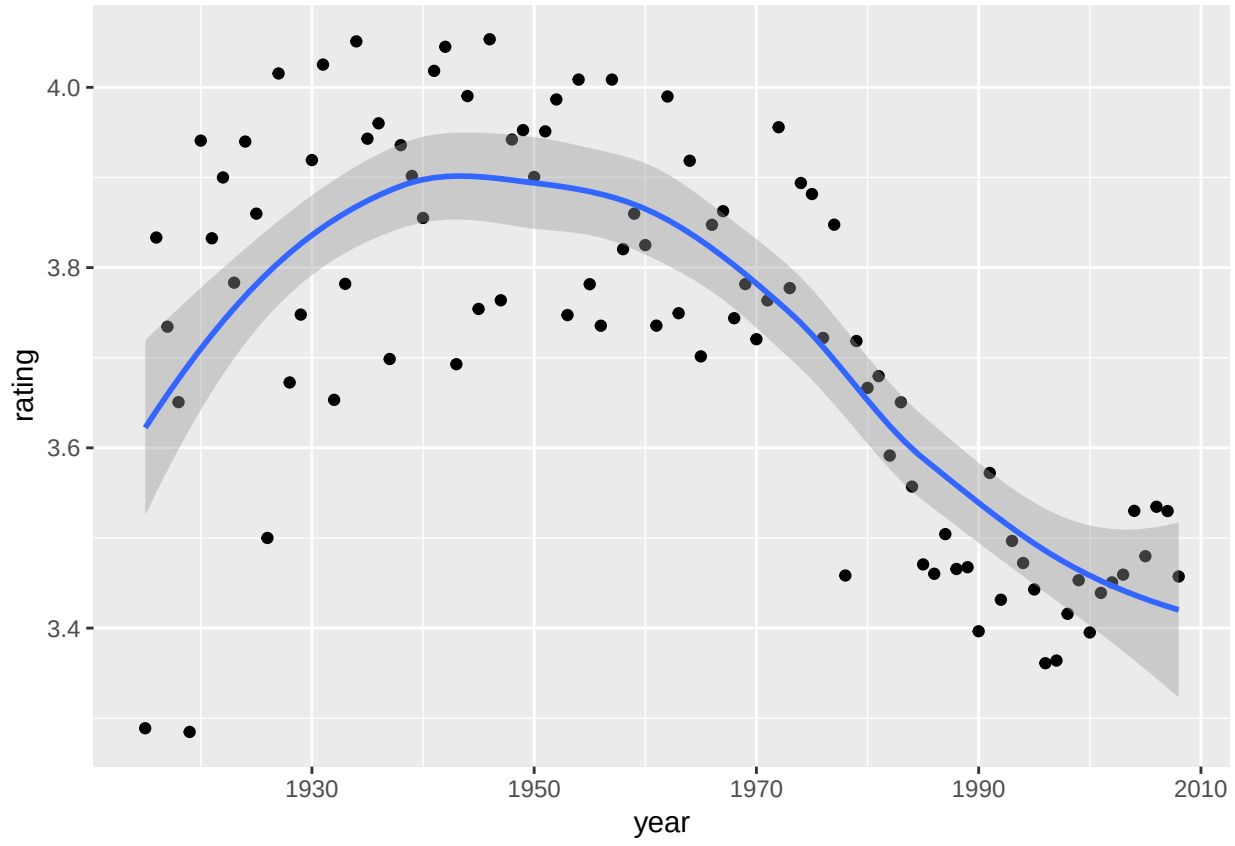


Ratings per Year

This plot shows that in general users seem to rate higher older movies over most recent ones.

```
edx %>% group_by(year) %>%
  summarize(rating = mean(rating)) %>%
  ggplot(aes(year, rating)) +
  geom_point() +
  geom_smooth()
```

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



Top 10 Most Rated Movies

```
edx %>% group_by(movieId, title) %>%
  summarize(n = n()) %>%
  arrange(desc(n))
```

```
## # A tibble: 10,677 x 3
## # Groups:   movieId [10,677]
##    movieId title                                                     n
##      <dbl> <chr>                                                 <int>
##  1     296 Pulp Fiction (1994)                                   31362
##  2     356 Forrest Gump (1994)                                   31079
##  3     593 Silence of the Lambs, The (1991)                      30382
##  4     480 Jurassic Park (1993)                                  29360
##  5     318 Shawshank Redemption, The (1994)                      28015
##  6     110 Braveheart (1995)                                     26212
##  7     457 Fugitive, The (1993)                                  25998
##  8     589 Terminator 2: Judgment Day (1991)                     25984
##  9     260 Star Wars: Episode IV - A New Hope (a.k.a. Star Wars) (1~ 25672
## 10     150 Apollo 13 (1995)                                      24284
## # ... with 10,667 more rows
```

3. Model Training

Results Table

```
# Creating results table
results <- data_frame()
```

Predicting results using the average rating

```
mu <- mean(edx$rating)
model_1_rmse <- RMSE(validation_t$rating, mu)
results <- data_frame(Model = 'Model 1', Method = "Using Average Ratings", RMSE = model_1_rmse)
results %>% knitr::kable()
```

Model	Method	RMSE
Model 1	Using Average Ratings	1.061202

Movie Bias

As shown above, predicting movie ratings based on previous overall average ratings doesn't produce good results. As noted in a previuos plot, some movies are rated more than other depending on their popularity, thus in order to achieve more accurate results it is important to take into account this efferct and remove movie bias.

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

predicted_ratings2 <- mu + validation_t %>%
  left_join(movie_avgs, by='movieId') %>%
  .$b_i

model_2_rmse <- RMSE(predicted_ratings2, validation_t$rating)

results <- bind_rows(results, data_frame(Model = 'Model 2', Method="Movie Bias", RMSE = model_2_rmse))
results %>% knitr::kable()
```

Model	Method	RMSE
Model 1	Using Average Ratings	1.0612018
Model 2	Movie Bias	0.9439087

User and Movie Bias

As we can see adding the movie effect to the model help achieve better results, let's see if we can do better by also accounting for user bias.

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

predicted_ratings3 <- validation %>%
  left_join(movie_avgs, by='movieId') %>%
  left_join(user_avgs, by='userId') %>%
  mutate(pred = mu + b_i + b_u) %>%
  .$pred
```

```
model_3_rmse <- RMSE(predicted_ratings3, validation_t$rating)

results <- bind_rows(results, data_frame(Model = 'Model 3', Method="User and Movie Bias", RMSE = model_3

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

Model	Method	RMSE
Model 1	Using Average Ratings	1.0612018
Model 2	Movie Bias	0.9439087
Model 3	User and Movie Bias	0.8292477

All Bias

As seen above, results are improving, but so far only user and movie bias have been considered. This time the training will be done by accounting for all bias (movie, user, year and genre).

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

genre_avgs <- validation_t %>%
  left_join(movie_avgs, by='movieId') %>%
  left_join(user_avgs, by='userId') %>%
  left_join(year_avgs, by='year') %>%
  group_by(genres) %>%
  summarize(b_g = mean(rating - mu - b_i - b_u - b_y))

predicted_ratings4 <- validation %>%
  left_join(movie_avgs, by='movieId') %>%
  left_join(user_avgs, by='userId') %>%
  left_join(year_avgs, by = 'year') %>%
  left_join(genre_avgs, by = 'genres') %>%
  mutate(pred = mu + b_i + b_u + b_y + b_g) %>%
  .$pred

model_4_rmse <- RMSE(predicted_ratings4, validation_t$rating)

results <- bind_rows(results, data_frame(Model = 'Model 4', Method="All Bias", RMSE = model_4_rmse))
results %>% knitr::kable()
```

Model	Method	RMSE
Model 1	Using Average Ratings	1.0612018
Model 2	Movie Bias	0.9439087
Model 3	User and Movie Bias	0.8292477
Model 4	All Bias	0.8282446

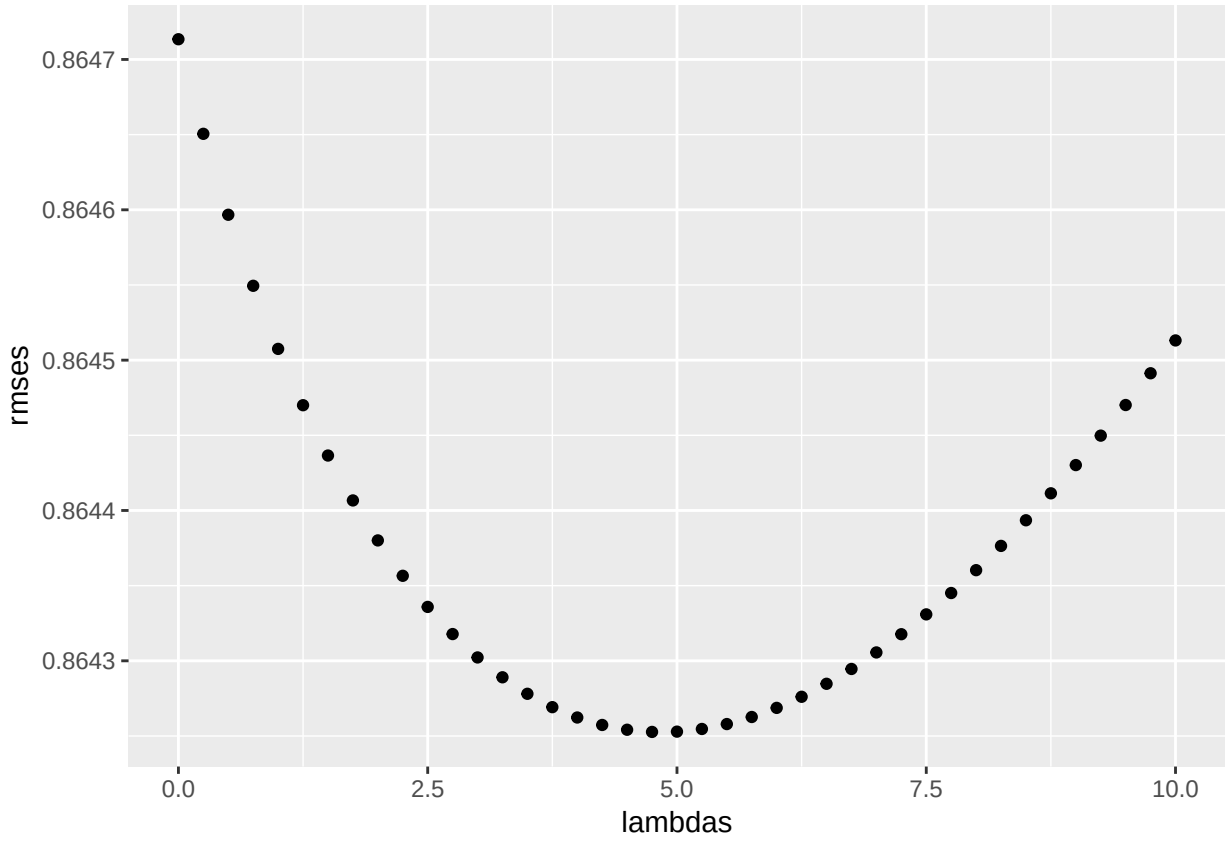
Regularizing model

Regularization is a technique that will help balance not only the bias in each of the variables but also its variance. First the optimal lambda will be found and the predictions will be made using the optimal lambda

```
lambdas <- seq(0, 10, 0.25)
rmses <- sapply(lambdas, 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))

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

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

  predicted_ratings5 <- validation %>%
    left_join(b_i, by='movieId') %>%
    left_join(b_u, by='userId') %>%
    left_join(b_y, by = 'year') %>%
    left_join(b_g, by = 'genres') %>%
    mutate(pred = mu + b_i + b_u + b_y + b_g) %>%
    .$pred

  return(RMSE(predicted_ratings5, validation_t$rating))
})
```

Finding Optimal Lambda

```
qplot(lambdas, rmses)
```



Plotting Lambdas vs RMSEs

```
lambda_opt <- lambdas[which.min(rmses)]
lambda_opt
```

```
## [1] 4.75
```

Predictions using optimal lambda

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

user_avgs_r <- edx %>%
  left_join(movie_avgs_r, by='movieId') %>%
  group_by(userId) %>%
  summarize(b_u = sum(rating - mu - b_i)/(n()+lambda_opt), n_u = n())

year_avgs_r <- edx %>%
  left_join(movie_avgs_r, by='movieId') %>%
  left_join(user_avgs_r, by='userId') %>%
  group_by(year) %>%
  summarize(b_y = sum(rating - mu - b_i - b_u)/(n()+lambda_opt), n_y = n())

genre_avgs_r <- edx %>%
  left_join(movie_avgs_r, by='movieId') %>%
  left_join(user_avgs_r, by='userId') %>%
```

```
  left_join(year_avgs_r, by = 'year') %>%
  group_by(genres) %>%
  summarize(b_g = sum(rating - mu - b_i - b_u - b_y)/(n()+lambda_opt), n_g = n())

predicted_ratings5 <- validation %>%
  left_join(movie_avgs_r, by='movieId') %>%
  left_join(user_avgs_r, by='userId') %>%
  left_join(year_avgs_r, by = 'year') %>%
  left_join(genre_avgs_r, by = 'genres') %>%
  mutate(pred = mu + b_i + b_u + b_y + b_g) %>%
  .$pred


model_5_rmse <- RMSE(predicted_ratings5, validation_t$rating)

results <- bind_rows(results, data_frame(Model = 'Model 5', Method="Regularized Effects", RMSE = model_5
results %>% knitr::kable()
```

Model	Method	RMSE
Model 1	Using Average Ratings	1.0612018
Model 2	Movie Bias	0.9439087
Model 3	User and Movie Bias	0.8292477
Model 4	All Bias	0.8282446
Model 5	Regularized Effects	0.8642527

4. Analysis of Results

Five different models were explored in order to predict movie ratings in the MovieLens dataset: predicting all ratings based on the observed average, penalizing the movie bias, penalizing user and movie bias, penalizing all bias and finally regularizing all movie bias using the most suitable lambda value. As per table above, the lowest RMSE is found with Model 4 - Penalizing All Bias (RMSE = 0.8282446)

5. Conclusions

Throughout the Data Science series several data cleaning, data visualization, data mining and machine learning techniques were explored, most of them were implemented in the development of this capstone project and they served to achieve a lower RMSE than te one reached by the winners of the 2006 Netflix challenge. During the execution of the project I learnt a lot about the Movielens dataset and recommendation systems, this knowledge will be valuable in the development of future projects.