Havardx Data Science - Movielens Capstone Project

Daniel Handojo

3/25/2021

1. Introduction

This data science project is about the 10M version of the movielens dataset that was collected by the GroupLens research lab. In the movielens dataset, GroupLens accumulated data about ratings of users for movies. This project applies data science and Machine Learning methods to clean, explore, visualize and finally make predictions about the data, that is, predicting the ratings of users for movies. The goodness of the models will be evaluated by comparing the RMSE of a holdout set. In order to make these predictions, we focus particularly on recommendation systems (matrix factorization) with regularization in order to decrease the variance and increase the predicting power.

1.1 Let's load our libraries and some data.

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

```
## Loading required package: tidyverse
```

```
## -- Attaching packages --------------------------------------- tidyverse 1.3.0 --
```

```
## v ggplot2 3.3.3     v purrr   0.3.4
## v tibble  3.0.6     v dplyr   1.0.4
## v tidyr   1.1.2     v stringr 1.4.0
## v readr   1.4.0     v forcats 0.5.1
```

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

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

```
## Loading required package: caret
```

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

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

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

```
if(!require(data.table)) install.packages("data.table", repos = "http://cran.us.r-project.org")
```

```
## Loading required package: data.table
```

```
##
## Attaching package: 'data.table'
```

```
## The following objects are masked from 'package:dplyr':
##
##     between, first, last
```

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

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

```
## Loading required package: lubridate
```

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

```
## The following objects are masked from 'package:data.table':
##
##     hour, isoweek, mday, minute, month, quarter, second, wday, week,
##     yday, year
```

```
## The following objects are masked from 'package:base':
##
##     date, intersect, setdiff, union
```

```
if(!require(ggplot2)) install.packages("ggplot2", repos = "http://cran.us.r-project.org")
if(!require(RColorBrewer)) install.packages("RColorBrewer", repos ="http://cran.us.r-project.org")
```

```
## Loading required package: RColorBrewer
```

```
library(tidyverse)
library(caret)
library(data.table)
library(lubridate)
library(ggplot2)
library(RColorBrewer)
options(digits=5)


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

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

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

```
# Join the movie ratings with the movie ID.
movies <- as.data.frame(movies) %>% mutate(movieId = as.numeric(movieId),
                                           title = as.character(title),
                                           genres = as.character(genres))


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

```
##    userId movieId rating timestamp                         title
## 1:      1     122      5 838985046              Boomerang (1992)
## 2:      1     185      5 838983525               Net, The (1995)
```

```
## 3:      1     231      5 838983392          Dumb & Dumber (1994)
## 4:      1     292      5 838983421               Outbreak (1995)
## 5:      1     316      5 838983392               Stargate (1994)
## 6:      1     329      5 838983392 Star Trek: Generations (1994)
##                         genres
## 1:                Comedy|Romance
## 2:         Action|Crime|Thriller
## 3:                        Comedy
## 4:  Action|Drama|Sci-Fi|Thriller
## 5:       Action|Adventure|Sci-Fi
## 6: Action|Adventure|Drama|Sci-Fi
```

```
# Create our validation set to evaluate our RMSE for our final model.
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)
```

1.2 Data description

```
class(edx)
```

```
## [1] "data.table" "data.frame"
```

We will work with 9M rows and 6 columns in order to build our model.

```
glimpse(edx)
```

```
## Rows: 9,000,061
## Columns: 6
## $ userId    <int> 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, ...
## $ movieId   <dbl> 122, 185, 231, 292, 316, 329, 355, 356, 362, 364, 370, 37...
## $ rating    <dbl> 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, ...
## $ timestamp <int> 838985046, 838983525, 838983392, 838983421, 838983392, 83...
## $ title     <chr> "Boomerang (1992)", "Net, The (1995)", "Dumb & Dumber (19...
## $ genres    <chr> "Comedy|Romance", "Action|Crime|Thriller", "Comedy", "Act...
```

```
names(edx)
```

```
## [1] "userId"    "movieId"   "rating"    "timestamp" "title"     "genres"
```

We have 10677 movies.

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

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

Which are rated by approximately 70000 users.

```
length(unique(edx$userId))
```

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

A first glimpse shows that there might be some interesting distribution for the movies in our genres.

```
#sapply with str_detect to check genres
genres <- c("Drama", "Comedy", "Thriller", "Romance")
sapply(genres, function(g){
  sum(str_detect(edx$genres, g))
})
```

```
##    Drama   Comedy Thriller  Romance
##  3909401  3541284  2325349  1712232
```

2. Analysis

After we defined our objective and load the data, we now will continue to go one step further and try to find some patterns in the data. This will be done by summarizing, aggregating, creating ratios and new time variables in order to find meaningful insights.

For that, it is necessary to sometimes clean and transform the data. While we can't drive useful information out of every plot, there will be plots that will define our model later.

First, we check out which movies were rated the most often. We see that after the title comes a year, which is most probably the release year. We will make use of that later. For now, we see popular movies within the first rows. This suggests our data seems reasonable.

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

```
## `summarise()` has grouped output by 'movieId'. You can override using the `.groups` argument.
```

```
## # A tibble: 10,677 x 3
## # Groups:   movieId [10,677]
##    movieId title                                                     count
##      <dbl> <chr>                                                     <int>
##  1     296 Pulp Fiction (1994)                                       31336
##  2     356 Forrest Gump (1994)                                       31076
##  3     593 Silence of the Lambs, The (1991)                          30280
##  4     480 Jurassic Park (1993)                                      29291
##  5     318 Shawshank Redemption, The (1994)                          27988
##  6     110 Braveheart (1995)                                         26258
##  7     589 Terminator 2: Judgment Day (1991)                         26115
##  8     457 Fugitive, The (1993)                                      26050
##  9     260 Star Wars: Episode IV - A New Hope (a.k.a. Star Wars) (1977) 25809
## 10     592 Batman (1989)                                             24343
## # ... with 10,667 more rows
```

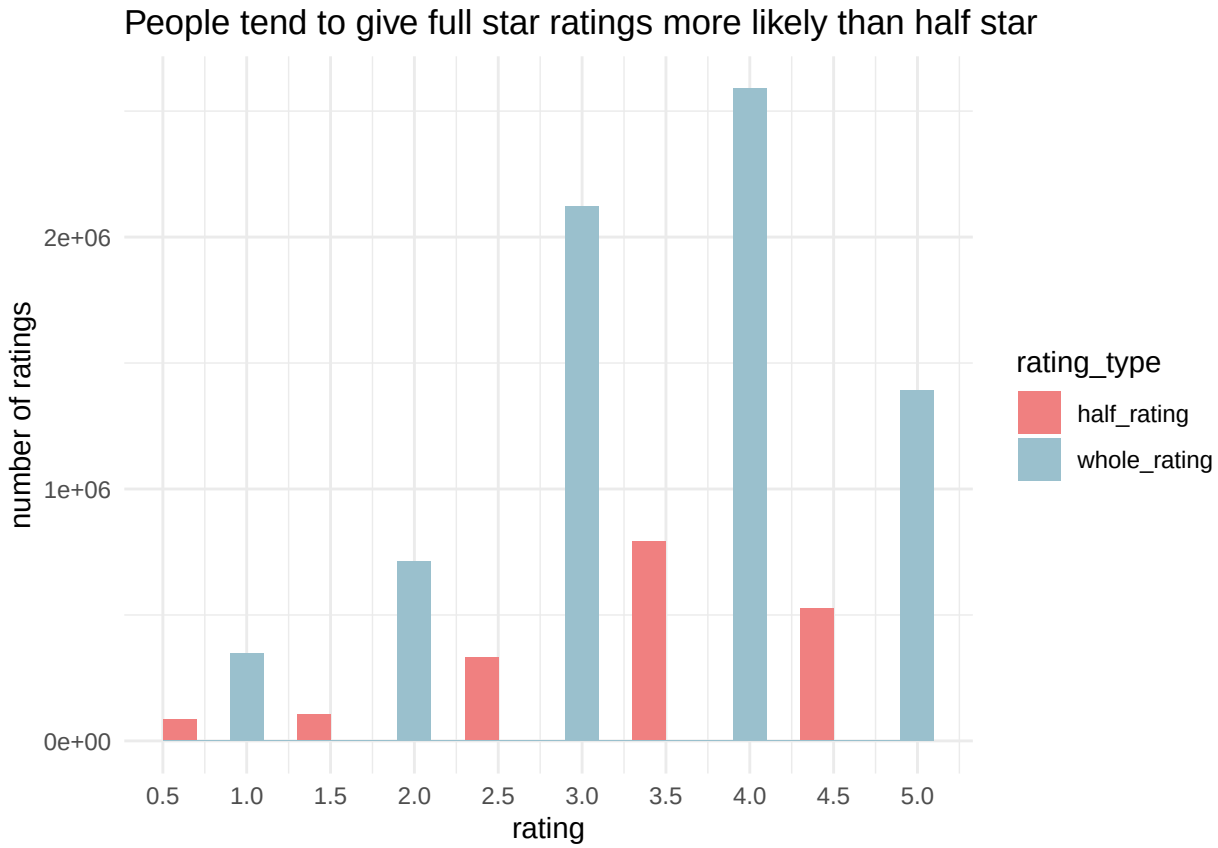
Sorting the ratings by their occurrence, we see that the higher ratings tend to be given more frequently than lower ratings.

```
edx %>% group_by(rating) %>%
  summarize(count = n()) %>%
  arrange(desc(count))
```

```
## # A tibble: 10 x 2
##    rating   count
##     <dbl>   <int>
##  1    4   2588021
##  2    3   2121638
##  3    5   1390541
##  4    3.5  792037
##  5    2    710998
##  6    4.5  526309
##  7    1    345935
##  8    2.5  332783
##  9    1.5  106379
## 10    0.5   85420
```

Visualizing this insight in a plot, makes it evident, that full ratings are more likely than half ratings with 4.0 the most popular rating.

```
rating_type <- ifelse((edx$rating == 1 |edx$rating == 2 | edx$rating == 3 |
                         edx$rating == 4 | edx$rating == 5) ,
                       "whole_rating",
                       "half_rating")

edx$rating_type <- rating_type


edx %>% ggplot(aes(x= rating, fill = rating_type)) +
  geom_histogram( binwidth = 0.2) +
  scale_x_continuous(breaks = seq(0.5,5, by = 0.5)) +
  scale_fill_manual(values =c("half_rating"="lightcoral", "whole_rating" ="lightblue3")) +
  labs(x="rating", y="number of ratings") +
  ggtitle("People tend to give full star ratings more likely than half star") +
  theme_minimal()
```



Let's add some time columns in order to check out some time trends regarding when the movie's were launched and at which time the most ratings were given.

```
#adding the dates for the rating and the release years of the movies
edx$date_rating <- as_datetime(edx$timestamp)
edx$year_rating <- as.integer(year(edx$date))
edx$year_release <- as.numeric(str_sub(edx$title, -5, -2))
```

In this dataset, the first movie rated was in 1995 while the first movie released in 1915.

```
data.frame(Name =c("First movie released", "First movie rating"),
           Year = c(min(edx$year_release),min(edx$year_rating)))
```

```
##                   Name Year
## 1 First movie released 1915
## 2   First movie rating 1995
```

Let's check out which was the first movie that was released. It was "The Birth of a Nation" and rated with full stars!

```
edx %>% filter(year_release == 1915) %>%
  group_by(movieId) %>% slice(1) %>%
  select(movieId, title, genres, rating)
```

```
## # A tibble: 1 x 4
## # Groups:   movieId [1]
##   movieId title                          genres    rating
##     <dbl> <chr>                          <chr>      <dbl>
## 1    7065 Birth of a Nation, The (1915) Drama|War      5
```

I was curious which was the release year that got the most ratings. Apparently in the year 2000 there were the most ratings. Could it be because of the internet?

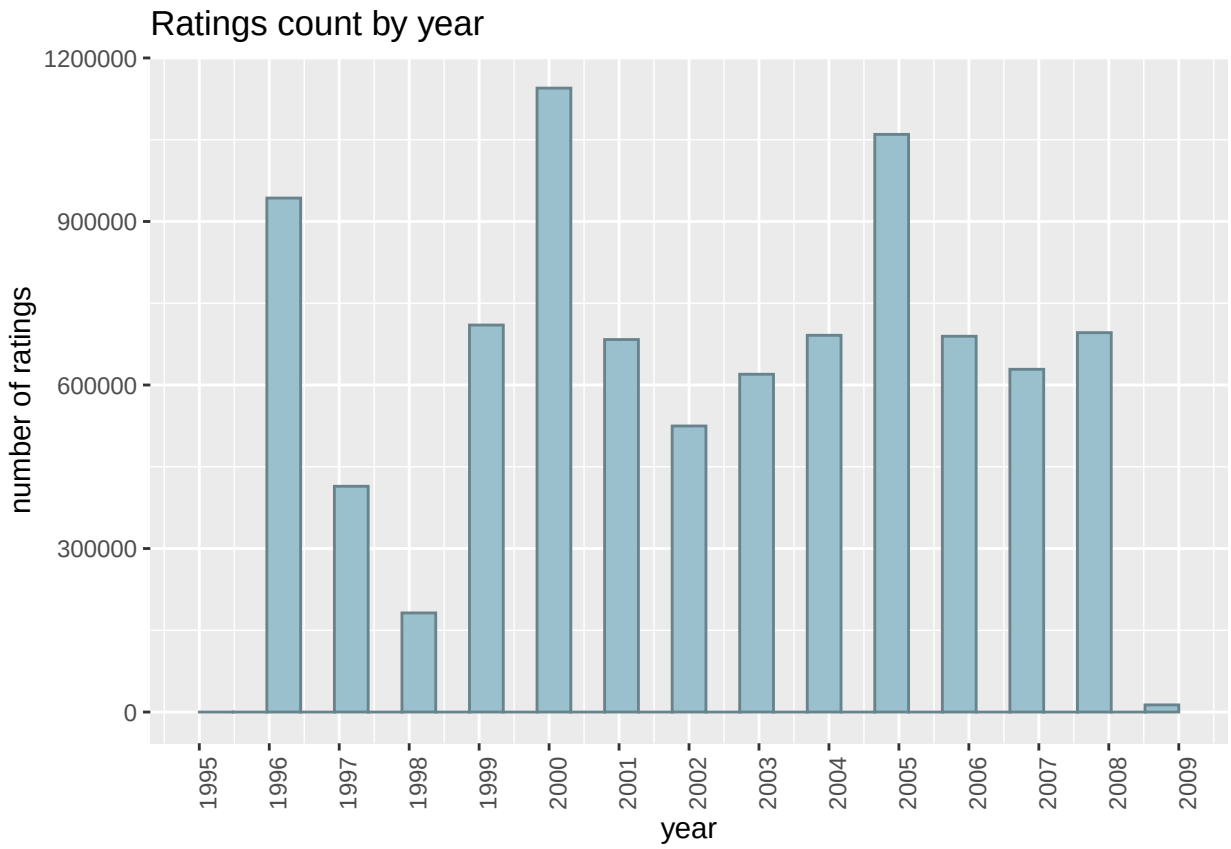
```
edx %>% group_by(year_rating) %>%
  mutate(n = n()) %>%
  select(year_rating, n) %>%
  slice(1) %>%
  arrange(desc(n))
```

```
## # A tibble: 15 x 2
## # Groups:   year_rating [15]
##    year_rating       n
##          <int>   <int>
##  1        2000 1144666
##  2        2005 1059807
##  3        1996  942976
##  4        1999  709978
##  5        2008  696027
##  6        2004  691191
##  7        2006  689447
##  8        2001  683412
##  9        2007  628845
## 10        2003  619707
## 11        2002  524826
## 12        1997  414218
## 13        1998  181845
## 14        2009   13114
## 15        1995       2
```

Plotting our ratings count by year, there is no significant trend recognizable. However, people liked to rate in the years 2000, 2005 and 1996.

```
edx %>% filter(year_rating>1970) %>%
  ggplot(aes(year_rating)) +
  geom_histogram(fill ="lightblue3", color="lightblue4") +
  scale_x_continuous(breaks = seq(1970,2010, by = 1)) +
  labs(x ="year", y="number of ratings") +
  ggtitle("Ratings count by year") +
  theme(axis.text.x = element_text(angle = 90, hjust = 1))
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```

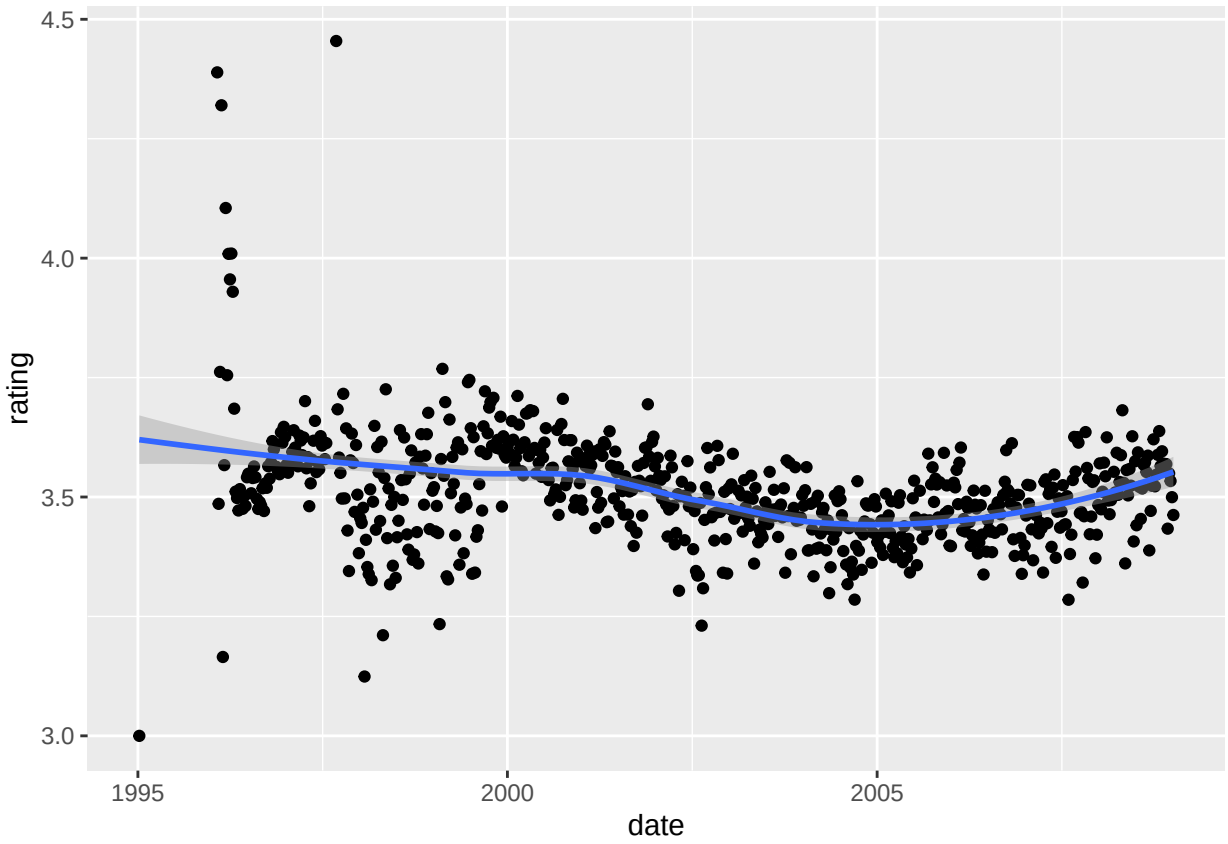


Utilizinig the timestamp a bit more, we check whether there is a correlation between the week of the rating and the rating itself.

And indeed, we clearly see that there is some kind of pattern between the date and the rating. We should also consider this later in our model.

```
edx %>% mutate(date = round_date(date_rating, 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'
```



As we have another year variable, that is the release year, we will see if there is a trend between release year vs. number of ratings.

Movies that were released in 1995 received the most ratings. We see, that 90s movies received a lot of ratings. This could be due to the fact, that the longer a movie is on the market, the more ratings it can gain.

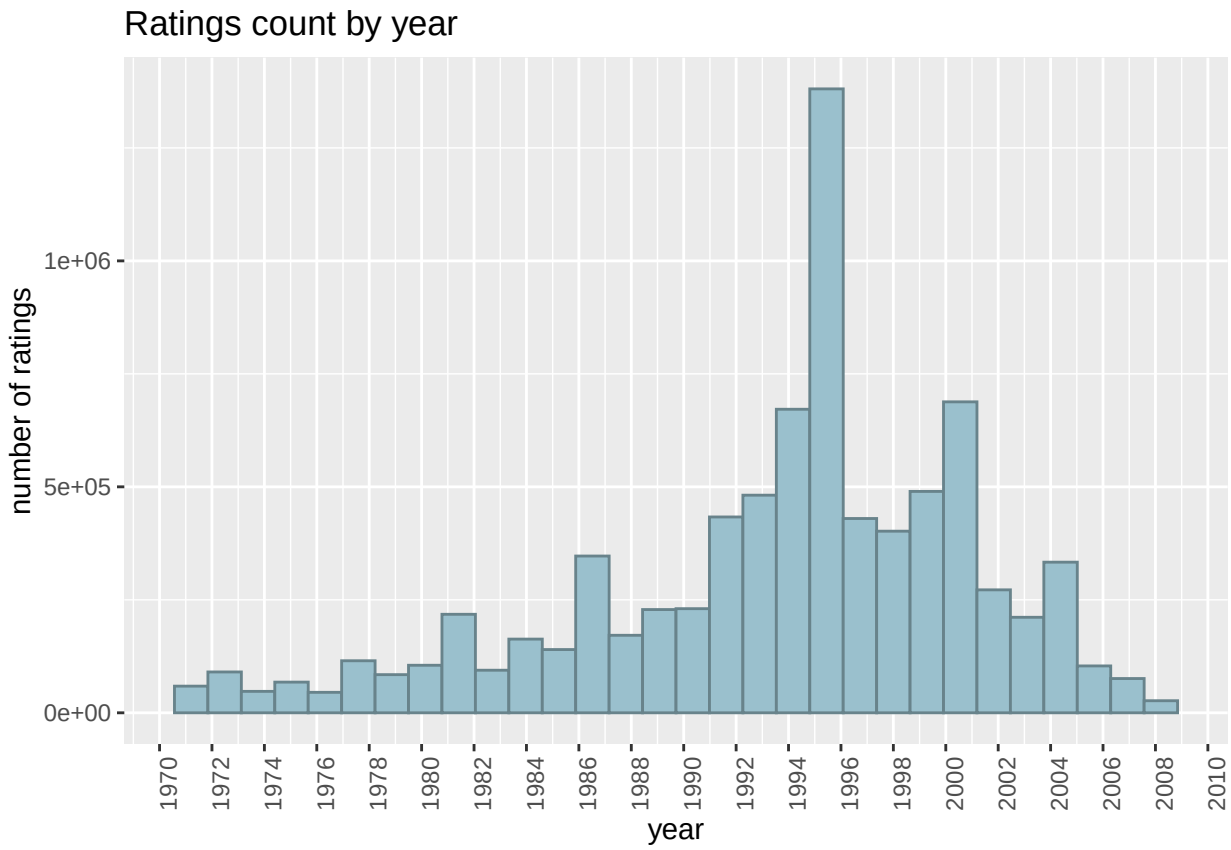
```
edx %>% group_by(year_release) %>%
  mutate(n = n()) %>%
  select(year_release, n) %>%
  slice(1) %>%
  arrange(desc(n))
```

```
## # A tibble: 94 x 2
## # Groups:   year_release [94]
##    year_release      n
##           <dbl>  <int>
##  1         1995 787116
##  2         1994 671676
##  3         1996 593442
##  4         1999 489899
##  5         1993 481557
##  6         1997 429928
##  7         1998 401905
##  8         2000 382510
##  9         2001 305561
## 10         2002 272061
## # ... with 84 more rows
```

The plot shows a left skewed histogram with a big spike in 1995.

```
edx %>% filter(year_release>1970) %>%
  ggplot(aes(year_release)) +
  geom_histogram(fill ="lightblue3", color="lightblue4") +
  scale_x_continuous(breaks = seq(1970,2010, by = 2)) +
  labs(x ="year", y="number of ratings") +
  ggtitle("Ratings count by year") +
  theme(axis.text.x = element_text(angle = 90, hjust = 1))
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



Checking out some movies released in 1995 that very mostly rated.

```
edx %>% filter(year_release==1995) %>%
  group_by(movieId) %>%
  mutate(count=n()) %>%
  select(title, count) %>%
  slice(1) %>%
  arrange(desc(count)) %>%
  top_n(10)
```

```
## Adding missing grouping variables: `movieId`
```

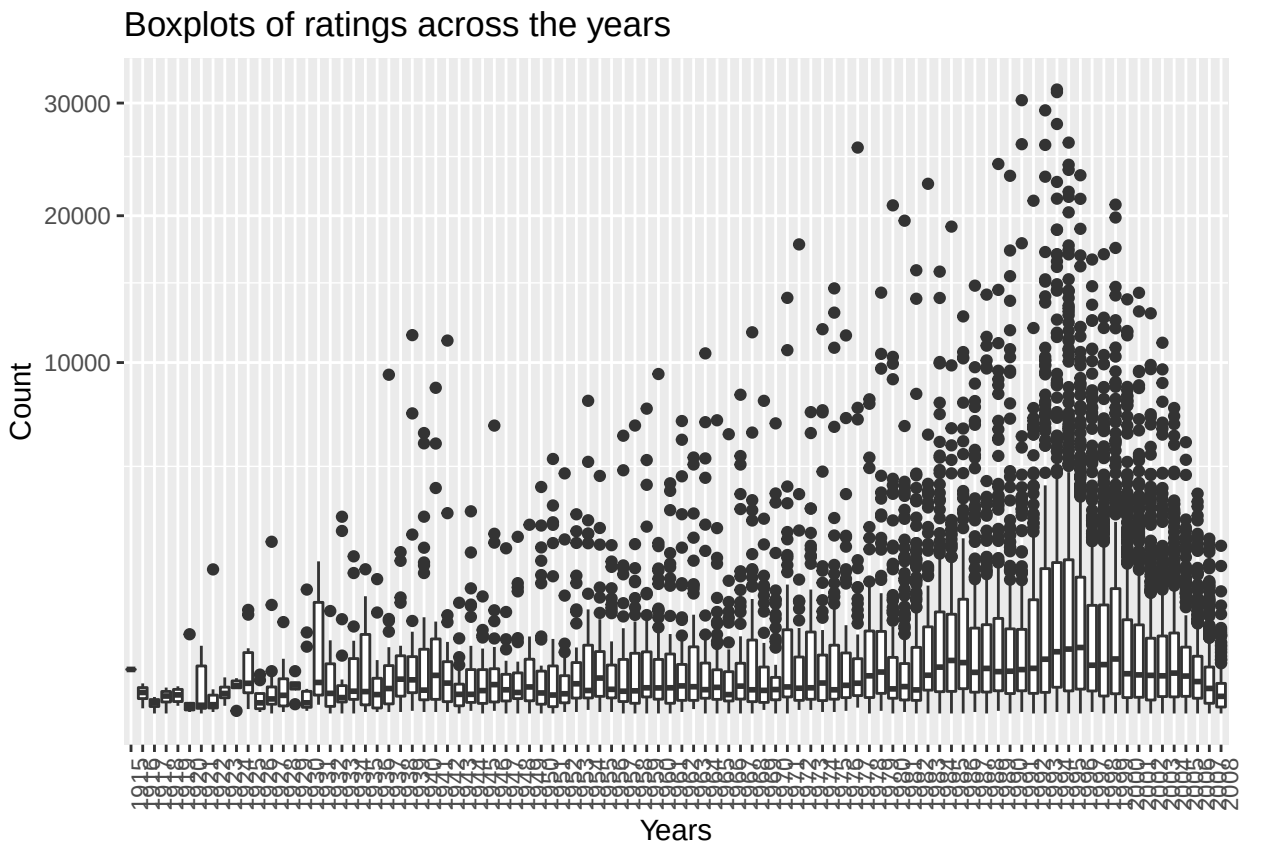
```
## Selecting by count
```

```
## # A tibble: 362 x 3
## # Groups:   movieId [362]
```

```
##    movieId title                                    count
##      <dbl> <chr>                                    <int>
##  1     110 Braveheart (1995)                        26258
##  2     150 Apollo 13 (1995)                         24277
##  3       1 Toy Story (1995)                         23826
##  4      32 12 Monkeys (Twelve Monkeys) (1995)       21959
##  5      50 Usual Suspects, The (1995)               21533
##  6      47 Seven (a.k.a. Se7en) (1995)              20271
##  7     165 Die Hard: With a Vengeance (1995)        17700
##  8     153 Batman Forever (1995)                    17358
##  9      34 Babe (1995)                              17035
## 10      10 GoldenEye (1995)                         15250
## # ... with 352 more rows
```

We can have a deeper understanding of the ratings across the release dates when plotting the boxplots.

```
edx %>% group_by(movieId) %>%
  summarize(n = n(), year = as.character(first(year_release))) %>%
  qplot(year, n, data = . , geom = "boxplot") +
  coord_trans(y="sqrt") +
  ggtitle("Boxplots of ratings across the years") +
  labs(x="Years", y ="Count") +
  theme(axis.text.x = element_text(angle = 90, hjust = 1))
```



Now, we create a ratio that checks how many ratings a movie got in a year and sort that rate in decreasing order.

Pulp Fiction, Forrest Gump and Jurassic Park received the highest amount of ratings per year.

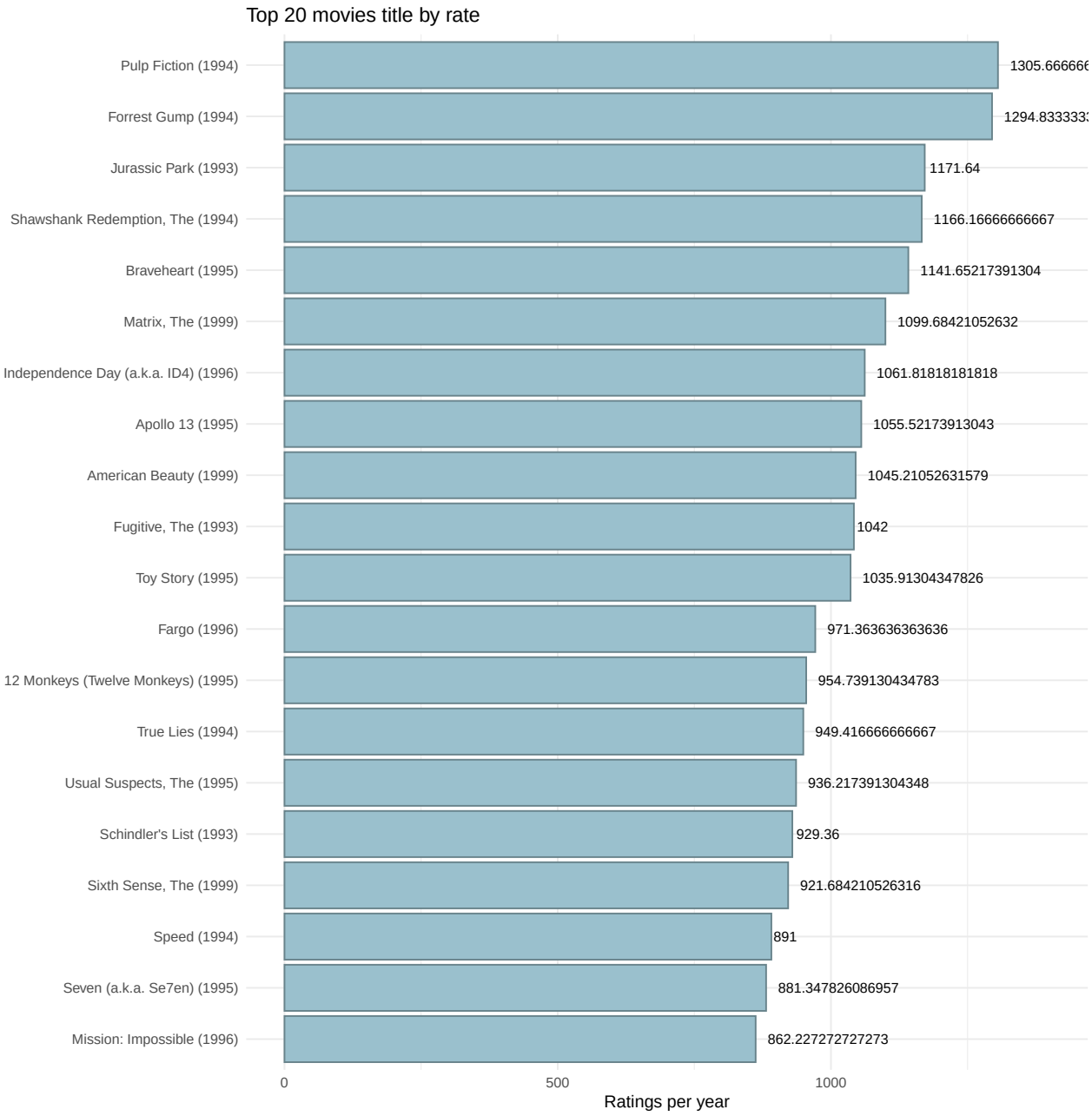
```
highest_rate_titles <- edx %>%  filter(year_release >= 1993) %>%
  group_by(movieId) %>%
  summarize(n = n(), years = 2018 - first(year_release),
            title = title[1],
            rating = mean(rating)) %>%
  mutate(rate = n/years) %>%
  top_n(25, rate) %>%
  arrange(desc(rate))


highest_rate_titles %>% top_n(10)
```

```
## Selecting by rate
```

```
## # A tibble: 10 x 6
##    movieId     n years title                                rating  rate
##      <dbl> <int> <dbl> <chr>                                 <dbl> <dbl>
##  1     296 31336    24 Pulp Fiction (1994)                    4.16 1306.
##  2     356 31076    24 Forrest Gump (1994)                    4.01 1295.
##  3     480 29291    25 Jurassic Park (1993)                   3.66 1172.
##  4     318 27988    24 Shawshank Redemption, The (1994)       4.46 1166.
##  5     110 26258    23 Braveheart (1995)                      4.08 1142.
##  6    2571 20894    19 Matrix, The (1999)                     4.21 1100.
##  7     780 23360    22 Independence Day (a.k.a. ID4) (1996)   3.38 1062.
##  8     150 24277    23 Apollo 13 (1995)                       3.89 1056.
##  9    2858 19859    19 American Beauty (1999)                 4.19 1045.
## 10     457 26050    25 Fugitive, The (1993)                   4.01 1042
```

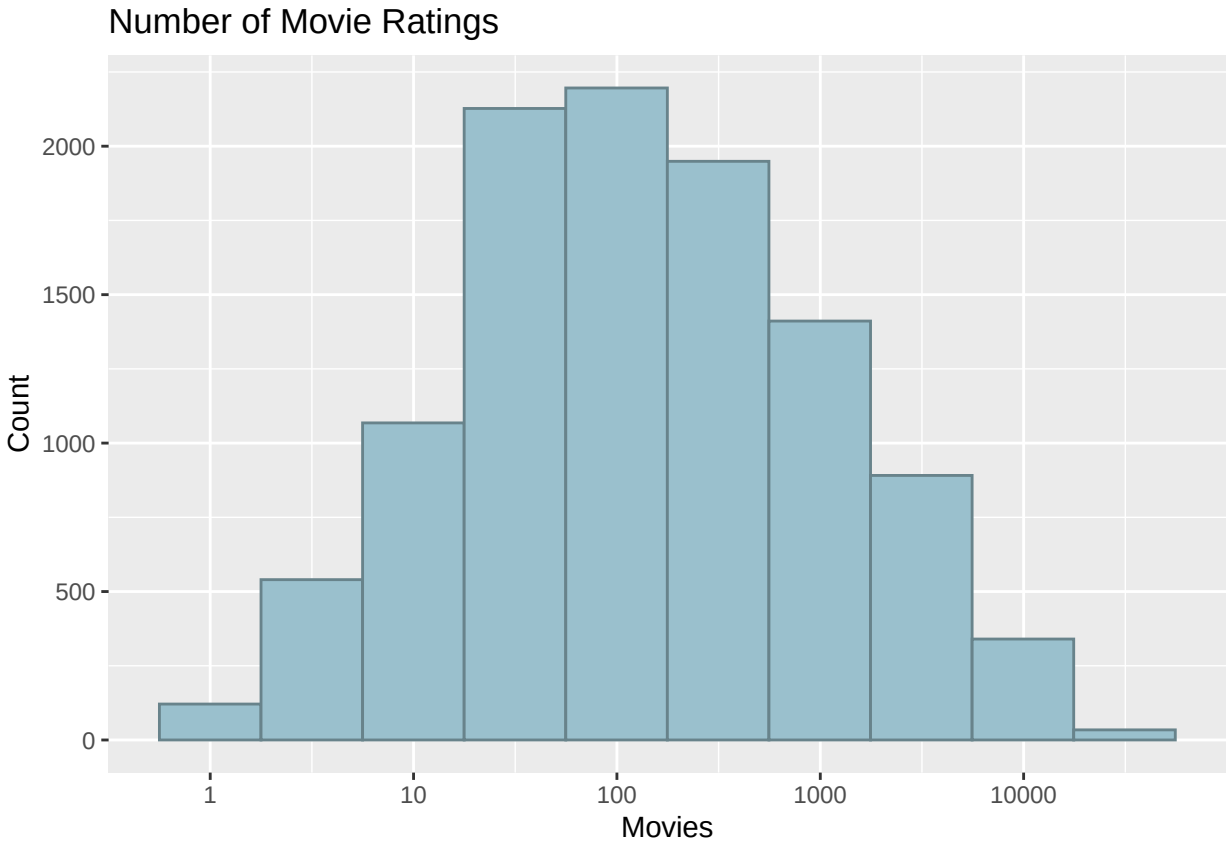
```
highest_rate_titles %>%
  top_n(20) %>%
  ggplot(aes(x=reorder(title, rate), y = rate)) +
  geom_bar(stat="identity", fill="lightblue3", color="lightblue4") +
  coord_flip(y=c(0,1400)) +
  labs(x="", y = "Ratings per year") +
  geom_text(aes(label = rate), hjust = -0.1, size = 3) +
  ggtitle("Top 20 movies title by rate") +
  theme_minimal()
```

```
## Selecting by rate
```



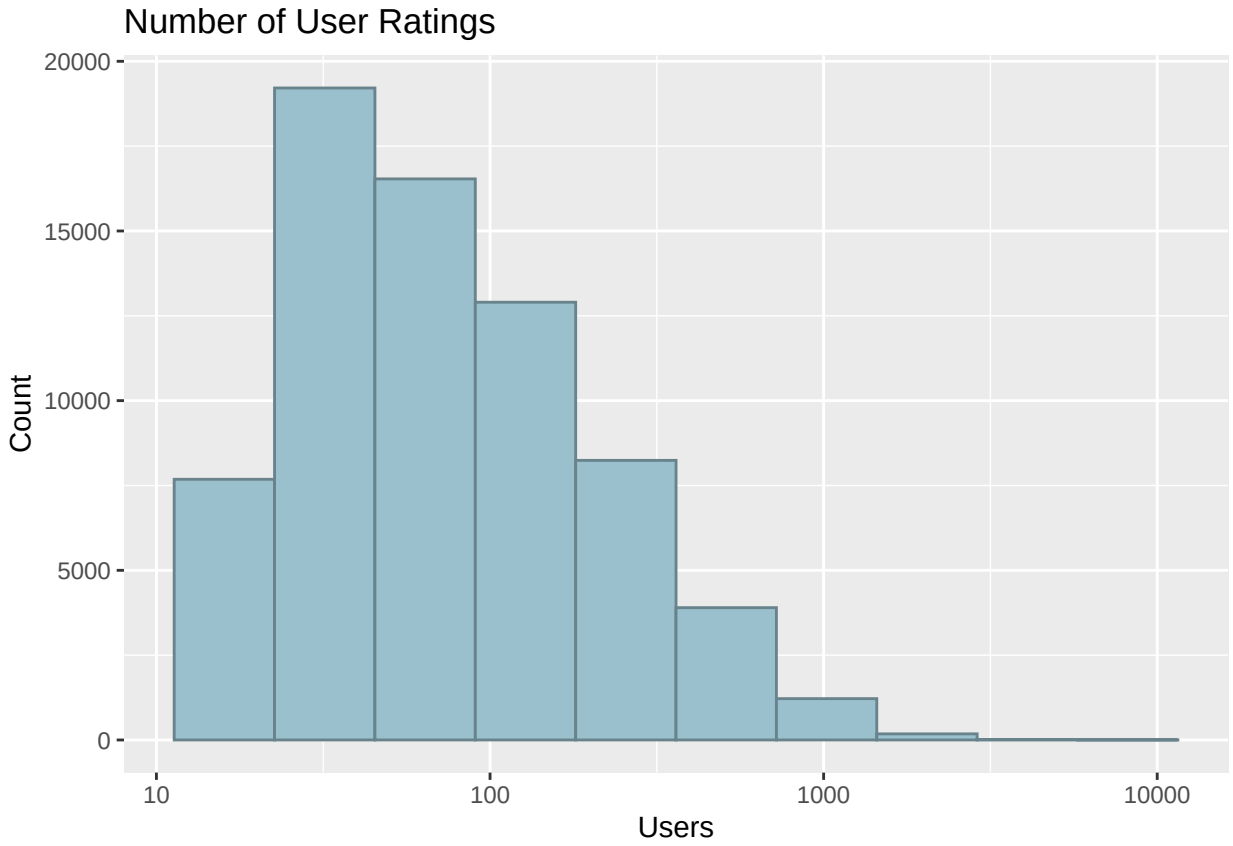
Plotting the number of ratings against movies while having a log-scale for the x-axis, we see that about 100 movies received the most ratings.

```
edx %>% group_by(movieId) %>%
  summarize(n = n()) %>%
  ggplot(aes(n)) +
  geom_histogram(fill="lightblue3", color = "lightblue4", bins = 10) +
  scale_x_log10() +
  labs(x = "Movies", y ="Count") +
  ggtitle("Number of Movie Ratings")
```



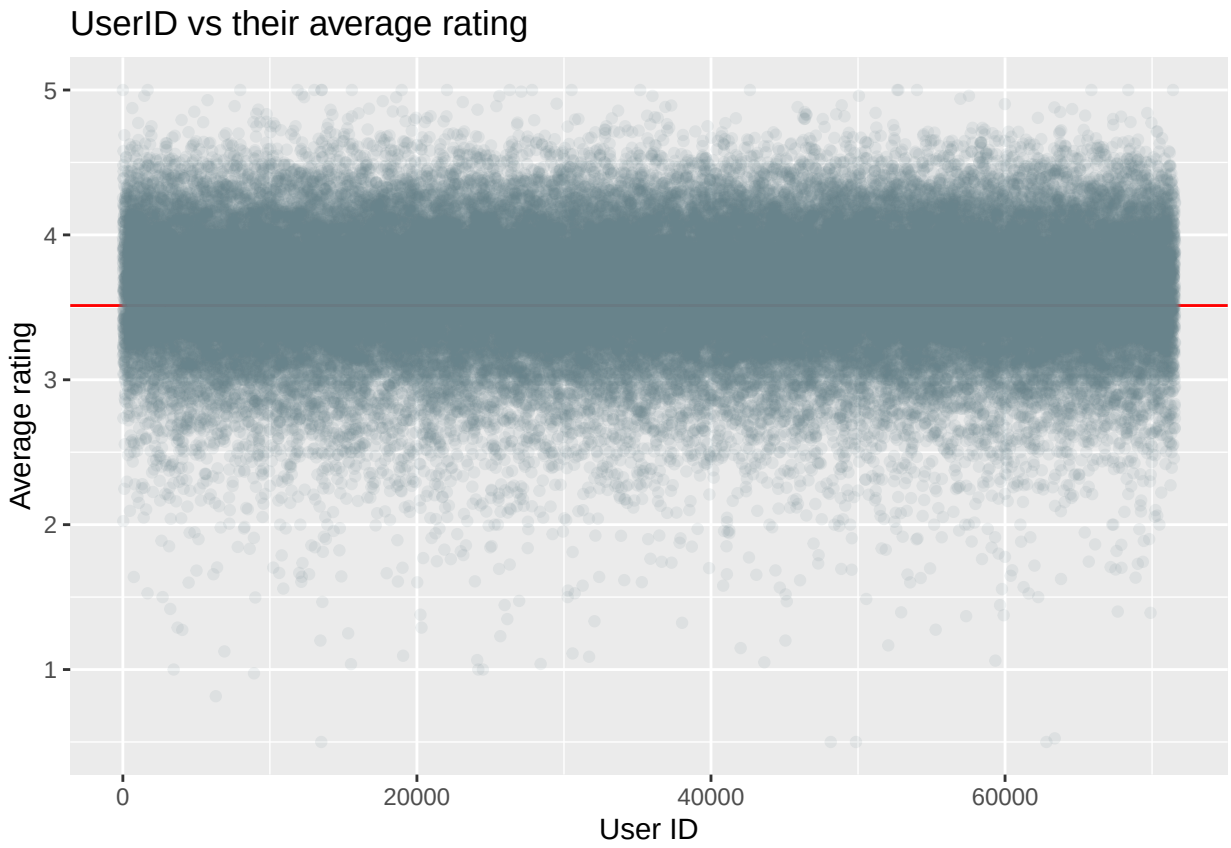
Doing the same for the user ID we see the same pattern.

```
edx %>% group_by(userId) %>%
  summarize(n = n()) %>%
  ggplot(aes(n)) +
  geom_histogram(fill="lightblue3", color = "lightblue4", bins = 10) +
  scale_x_log10() +
  labs(x = "Users", y ="Count") +
  ggtitle("Number of User Ratings")
```



While the average rating for all movies seems to be at 3.5, this rating doesn't seem to be the middle of that ratings cloud, created by the users. This is due to the fact, that users are more likely to give better ratings as seen before, while the bad ratings are less frequent. This will be relevant later when we want to derive the user effect for the model and do our regularization.

```
edx %>% group_by(userId) %>%
  summarize(avg_rating = mean(rating)) %>%
  ggplot(aes(x=userId, y=avg_rating)) +
  geom_hline(yintercept=mean(edx$rating), color="red") +
  geom_point(alpha= 0.1, color="lightblue4") +
  labs(x="User ID", y="Average rating") +
  ggtitle("UserID vs their average rating")
```



Finally, we still haven't touched our genre variable. Let's see which insights we can generate out of it.

First, we need to do some data cleaning.

```
#This will take a while
genres <- edx %>% separate_rows(genres, sep="\\|") %>%
  group_by(genres) %>%
  summarize(count = n()) %>%
  arrange(desc(count))
```

The majority are in the genre Drama, Comedy and Action.

```
genres %>% top_n(10)
```

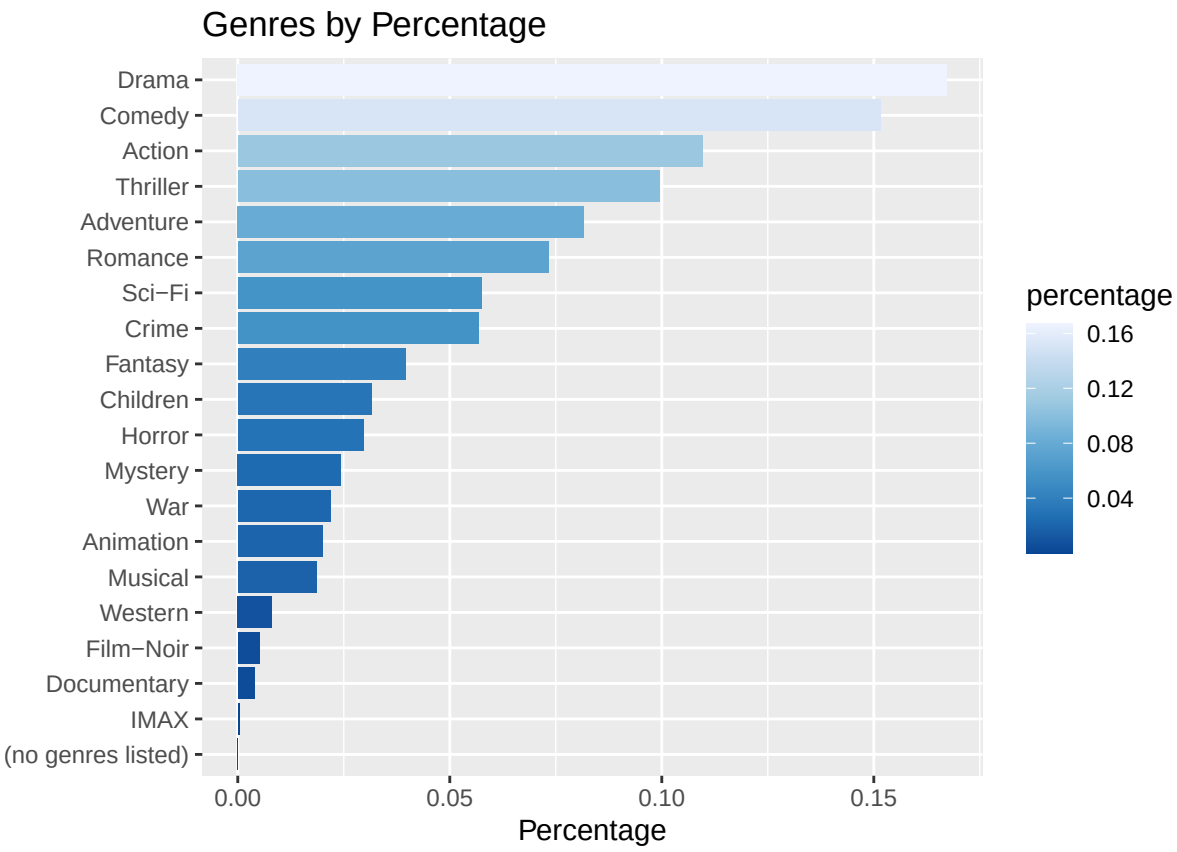
```
## Selecting by count
```

```
## # A tibble: 10 x 2
##    genres      count
##    <chr>       <int>
##  1 Drama     3909401
##  2 Comedy    3541284
##  3 Action    2560649
##  4 Thriller  2325349
##  5 Adventure 1908692
##  6 Romance   1712232
##  7 Sci-Fi    1341750
##  8 Crime     1326917
```

```
##  9 Fantasy    925624
## 10 Children   737851
```

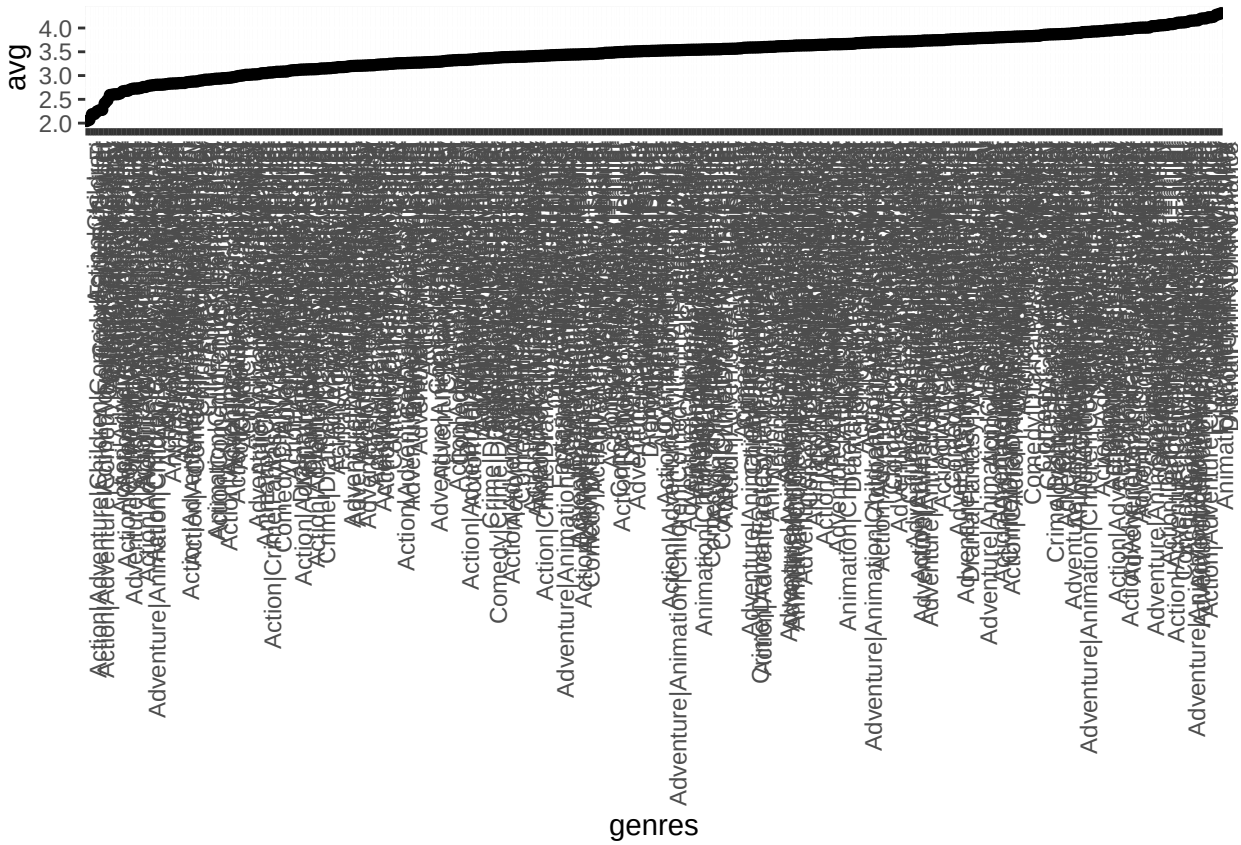
Plotting a visual allows us to see, that Drama and Comedy are the most popular by a margin.

```
genres <- genres %>% mutate(total = sum(count), percentage = count/total)

genres %>% ggplot(aes(reorder(genres, percentage), percentage, fill =percentage)) +
  geom_bar(stat="identity") +
  coord_flip() +
  scale_fill_distiller(palette ="Blues") +
  labs(x="", y="Percentage") +
  ggtitle("Genres by Percentage")
```



Even though the plot is unreadable as the genre combination are just too many, one can clearly see that there is a genre effect on rating. Here we plot the movies with the exact same genres and their average rating. We will take this effect into consideration for our later prediction.

```
edx %>% group_by(genres) %>%
  summarize(n = n(), avg = mean(rating), se = sd(rating)/sqrt(n())) %>%
  filter(n >= 1000) %>%
  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))
```


avg
4.0
3.5
3.0
2.5
2.0
genres

3. Methods

As we explored our data, we now try to consider all information and build a model out of it.

For this, we first split our data in train and test set. We do this, to check whether our model is under- or overfitting.

```
set.seed(1)
test_index <- createDataPartition(y = edx$rating, times = 1,
                                  p = 0.2, list = FALSE)
train_set <- edx[-test_index,]
test_set <- edx[test_index,]

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

We also build a function for the root mean squared error in order to compare our models.

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

In order to create predictions for ratings for all users and for all movies, we use a particular class of collaborative filtering called matrix factorization. Thus we will create different user-movie interaction matrices based on different effects and combine them afterwards. This method was also used for the Netflix prize.

3.1 Average Model

First, we just use the average rating of all movies in order to predict all ratings. We also save all our results in a table in order to have an easy comparison afterwards. As this is the first model, we can't say a lot yet.

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

naive_rmse <- RMSE(mu, test_set$rating)

rmse_results <- tibble(method = "Average", RMSE=naive_rmse)
rmse_results
```

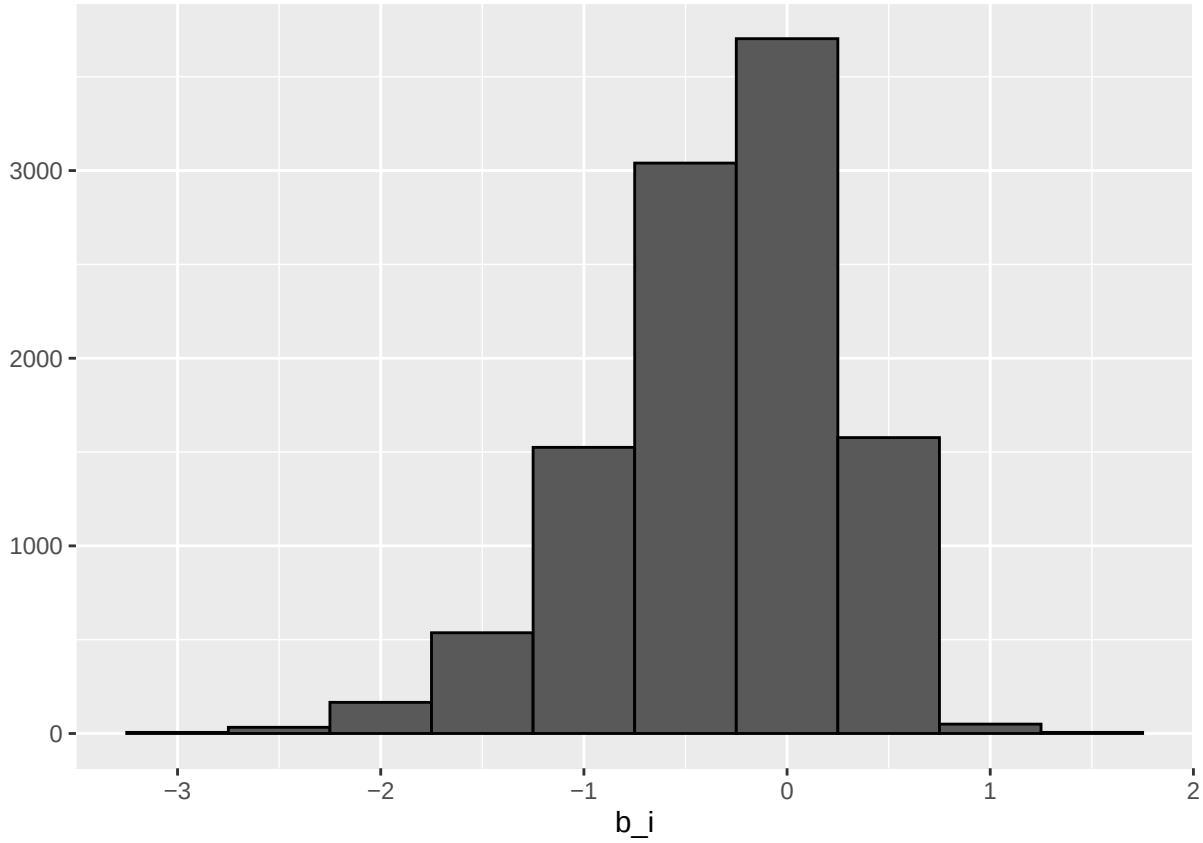
```
## # A tibble: 1 x 2
##   method   RMSE
##   <chr>   <dbl>
## 1 Average  1.06
```

3.2 Movie Effects Model

We saw earlier, that there are movies which are rated very good such as "GoodFellas" or "Shawshank Redemption". Therefore, we will consider a movie effect (b_i) in our model.

Indeed, we see a skewed qplot. This indicates that movies are generally rated worse than the average, based on the movie.

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


qplot(b_i, data = movie_avgs, bins = 10, color = I("black"))
```



We see, that we already could decrease our RMSE under one. Let's keep going.

```
predicted_ratings <- mu + test_set %>%
  left_join(movie_avgs, by = "movieId") %>%
  .$b_i

mu_bi_rmse <- RMSE(predicted_ratings, test_set$rating)

rmse_results <- rmse_results %>%
  add_row(method = "Average + Movie Effect", RMSE = mu_bi_rmse)
rmse_results
```

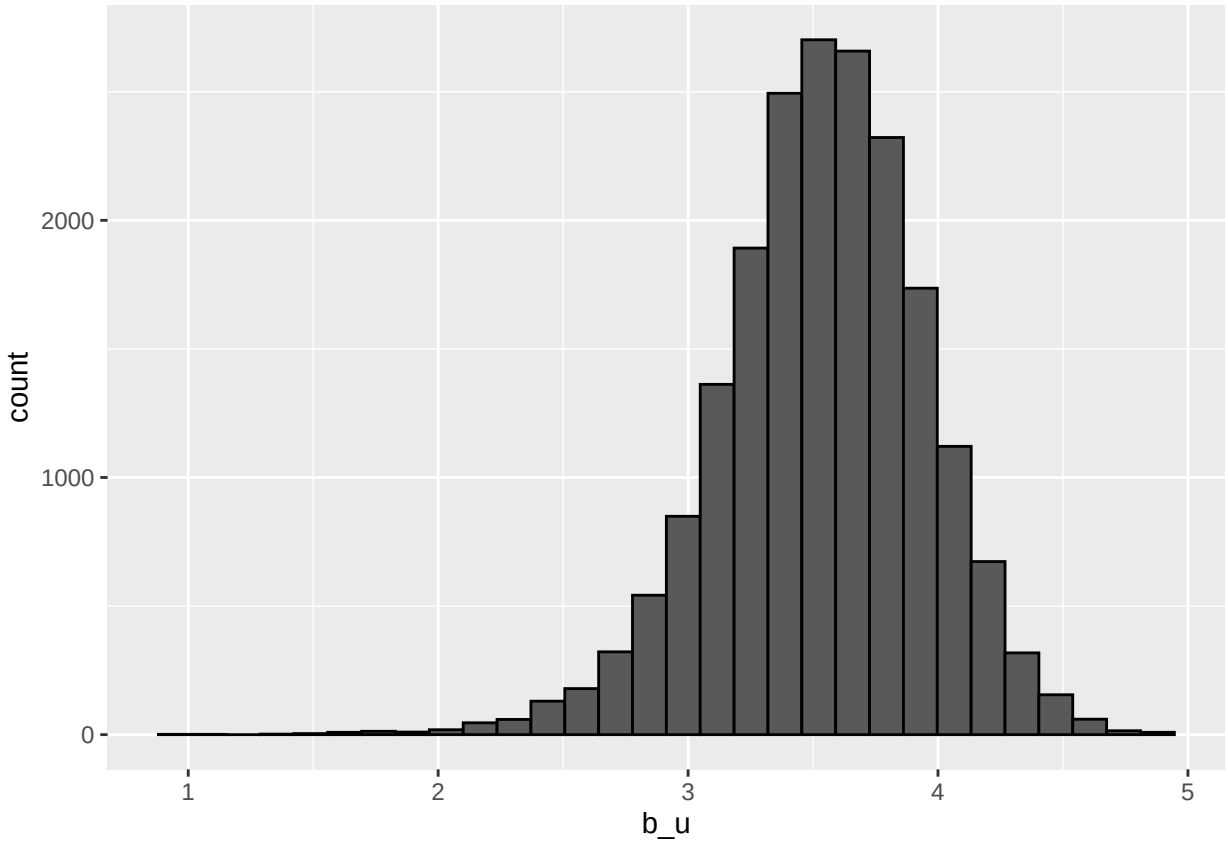
```
## # A tibble: 2 x 2
##   method                  RMSE
##   <chr>                  <dbl>
## 1 Average                1.06
## 2 Average + Movie Effect 0.943
```

3.3 User Effecs Model

Users generally give a subjective rating. Thus there are users who easily give high ratings and users who are more critical considering the rating. In order to take this into accocunt, we add a user effect (b_u) to our model.

In the plot we can see the distribution of the average rating of the users. As the overall average of all movies is about 3.5, it is not surprising, that the mode is also at about 3.5.

```
train_set %>%
  group_by(userId) %>%
  filter(n()>= 100) %>%
  summarize(b_u = mean(rating)) %>%
  ggplot(aes(b_u)) +
  geom_histogram(bins = 30, color = "black")
```



Adding the user effect to our model, we can decrase our RMSE to 0.866, which is already pretty good.

```
# create user effect variable
user_avgs <- train_set %>%
  left_join(movie_avgs, by = "movieId") %>%
  group_by(userId) %>%
  summarize(b_u =mean(rating - mu - b_i))


# predict movie effect + user effect
predicted_ratings <- test_set %>%
```

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


mu_bi_bu_rmse <- RMSE(test_set$rating, predicted_ratings)

rmse_results <- rmse_results %>%
  add_row(method ="+ User Effect", RMSE = mu_bi_bu_rmse)
rmse_results
```

```
## # A tibble: 3 x 2
##   method                  RMSE
##   <chr>                  <dbl>
## 1 Average                1.06
## 2 Average + Movie Effect 0.943
## 3 + User Effect          0.866
```

3.4 Week Effects Model

We saw earlier that time also played a role when users rated movies. Therefore, we will take into account the week in which users rated the movies.

```
time_avgs <- train_set %>%
  left_join(movie_avgs, by ="movieId") %>%
  left_join(user_avgs, by ="userId") %>%
  mutate(date = round_date(as_datetime(timestamp), unit ="week")) %>%
  group_by(date) %>%
  summarize(b_t = mean(rating - mu - b_i - b_u))


# adding week to train and test set
train_set_n <- train_set %>%
  mutate(date = round_date(as_datetime(timestamp), unit="week"))
test_set_n <- test_set %>%
  mutate(date = round_date(as_datetime(timestamp), unit ="week"))

predicted_ratings_bt <- test_set_n %>%
  left_join(movie_avgs, by ="movieId") %>%
  left_join(user_avgs, by ="userId") %>%
  left_join(time_avgs, by ="date") %>%
  mutate(pred = mu + b_i + b_u + b_t) %>%
  .$pred


mu_bi_bu_bt_rmse <- RMSE(test_set_n$rating, predicted_ratings_bt)

rmse_results <- rmse_results %>%
  add_row(method ="+ Week Effect", RMSE = mu_bi_bu_bt_rmse)
rmse_results
```

```
## # A tibble: 4 x 2
##   method                  RMSE
```

```
##   <chr>                  <dbl>
## 1 Average                1.06
## 2 Average + Movie Effect 0.943
## 3 + User Effect          0.866
## 4 + Week Effect          0.865
```

It seems that our RMSE didn't decrease in the table. However, calculating the min, we see that adding the week slightly decreased our RMSE.

```
rmse_results$method[which.min(rmse_results$RMSE)]
```

```
## [1] "+ Week Effect"
```

```
data.table(rmse_results)
```

```
##                     method    RMSE
## 1:                 Average 1.05964
## 2: Average + Movie Effect 0.94317
## 3:           + User Effect 0.86552
## 4:           + Week Effect 0.86543
```

3.5 Genres Effect Model

As the genres are combined in one cell and format the data frame in a tidy format would use too much computing power, we use the genres feature as given to create our genres effect (even not optimal).

```
genre_avgs <- train_set_n %>%
  left_join(movie_avgs, by ="movieId") %>%
  left_join(user_avgs, by ="userId") %>%
  left_join(time_avgs, by ="date") %>%
  group_by(genres) %>%
  summarize(b_g = mean(rating - mu - b_i - b_u - b_t))

predicted_ratings <- test_set_n %>%
  left_join(movie_avgs, by ="movieId") %>%
  left_join(user_avgs, by ="userId") %>%
  left_join(time_avgs, by ="date") %>%
  left_join(genre_avgs, by ="genres") %>%
  mutate(pred = mu + b_i + b_u + b_t + b_g) %>%
  .$pred

mu_bi_bu_bt_bg_rmse <- RMSE(test_set_n$rating, predicted_ratings)

rmse_results <- rmse_results %>%
  add_row(method ="+ Genres", RMSE = mu_bi_bu_bt_bg_rmse)
rmse_results
```

```
## # A tibble: 5 x 2
##   method                  RMSE
##   <chr>                  <dbl>
## 1 Average                1.06
## 2 Average + Movie Effect 0.943
## 3 + User Effect          0.866
## 4 + Week Effect          0.865
## 5 + Genres               0.865
```

We were again able to decrease our RMSE.

3.6 Regularization

To further improve our RMSE we use regularization. We constraint the total variability of the coefficients by penalizing large estimates from small sample sizes.

```
test_set %>%
  left_join(movie_avgs, by ="movieId") %>%
  mutate(residual = rating - (mu + b_i)) %>%
  arrange(desc(abs(residual))) %>%
  select(title, residual) %>% slice(1:40) %>% knitr::kable()
```

title	residual
From Justin to Kelly (2003)	4.0994
Pokémon Heroes (2003)	4.0762
Holy Mountain, The (Montaña sagrada, La) (1973)	-4.0000
Shawshank Redemption, The (1994)	-3.9596
Shawshank Redemption, The (1994)	-3.9596
Shawshank Redemption, The (1994)	-3.9596
Shawshank Redemption, The (1994)	-3.9596
Shawshank Redemption, The (1994)	-3.9596
Shawshank Redemption, The (1994)	-3.9596
Carnosaur 3: Primal Species (1996)	3.9554
Godfather, The (1972)	-3.9208
Godfather, The (1972)	-3.9208
Godfather, The (1972)	-3.9208
Godfather, The (1972)	-3.9208
Godfather, The (1972)	-3.9208
Godfather, The (1972)	-3.9208
Godfather, The (1972)	-3.9208
Godfather, The (1972)	-3.9208
Godfather, The (1972)	-3.9208
Godfather, The (1972)	-3.9208
Pokemon 4 Ever (a.k.a. Pokémon 4: The Movie) (2002)	3.9019
Pokemon 4 Ever (a.k.a. Pokémon 4: The Movie) (2002)	3.9019
Usual Suspects, The (1995)	-3.8715
Usual Suspects, The (1995)	-3.8715
Usual Suspects, The (1995)	-3.8715
Usual Suspects, The (1995)	-3.8715
Schindler's List (1993)	-3.8631
Schindler's List (1993)	-3.8631
Schindler's List (1993)	-3.8631
Schindler's List (1993)	-3.8631
Schindler's List (1993)	-3.8631
Schindler's List (1993)	-3.8631
Schindler's List (1993)	-3.8631
Schindler's List (1993)	-3.8631
Schindler's List (1993)	-3.8631
Schindler's List (1993)	-3.8631
Sunset Blvd. (a.k.a. Sunset Boulevard) (1950)	-3.8264

title	residual
Sunset Blvd. (a.k.a. Sunset Boulevard) (1950)	-3.8264
Sunset Blvd. (a.k.a. Sunset Boulevard) (1950)	-3.8264

Movies with movie effect received a high rating, even though they are not well known.

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

# movie that received high b_i effects are not well known
movie_avgs %>% left_join(movie_titles, by ="movieId") %>%
  arrange(desc(b_i)) %>%
  select(title, b_i) %>%
  slice(1:10) %>%
  knitr::kable()
```

title	b_i
Shanghai Express (1932)	1.4876
Satan's Tango (Sátántangó) (1994)	1.4876
Fighting Elegy (Kenka erejii) (1966)	1.4876
Sun Alley (Sonnenallee) (1999)	1.4876
Constantine's Sword (2007)	1.4876
Human Condition II, The (Ningen no joken II) (1959)	1.3210
Who's Singin' Over There? (a.k.a. Who Sings Over There) (Ko to tamo peva) (1980)	1.2376
Life of Oharu, The (Saikaku ichidai onna) (1952)	1.2376
I'm Starting From Three (Ricomincio da Tre) (1981)	1.1543
Human Condition III, The (Ningen no joken III) (1961)	1.1543

The same accounts for movies at the low end. These movies are also not well known.

```
movie_avgs %>% left_join(movie_titles, by ="movieId") %>%
  arrange(b_i) %>%
  select(title, b_i) %>%
  slice(1:10) %>%
  knitr::kable()
```

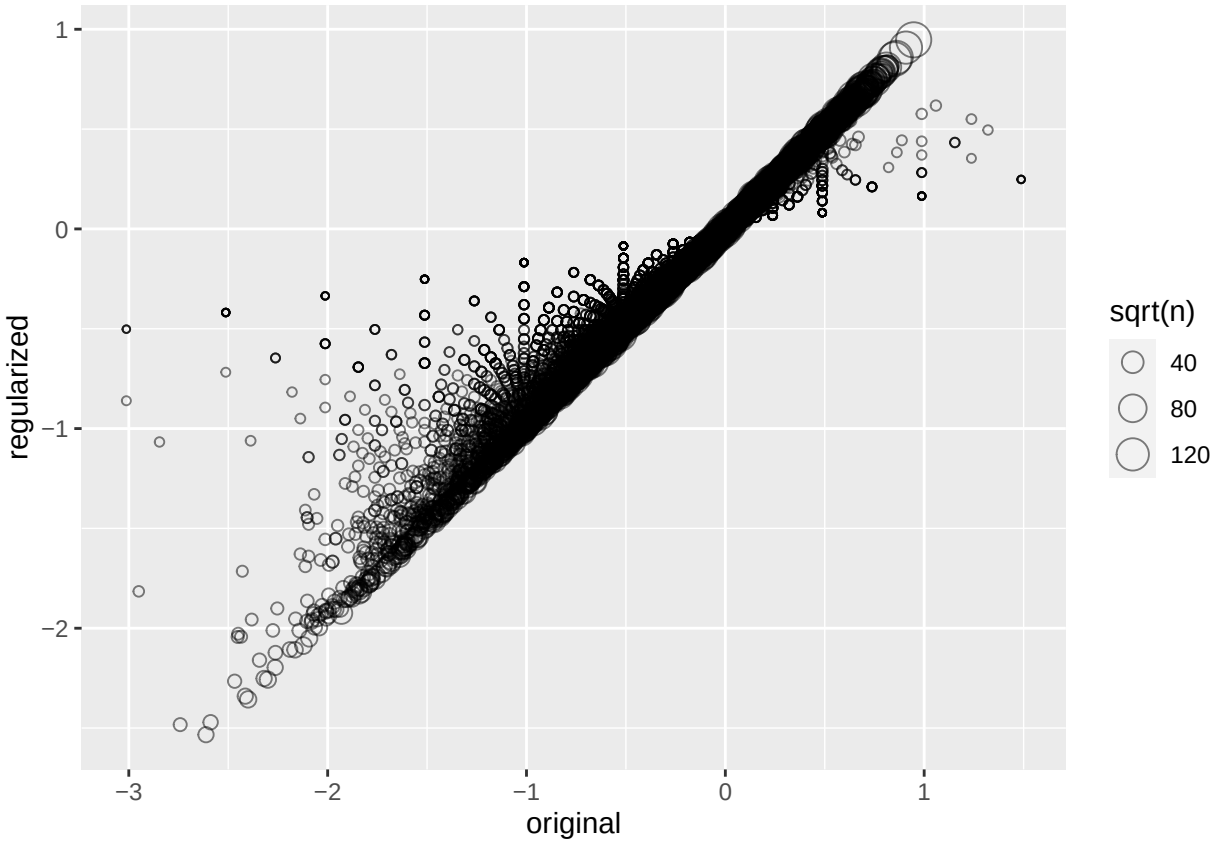
title	b_i
Besotted (2001)	-3.0124
Hi-Line, The (1999)	-3.0124
Uncle Nino (2003)	-3.0124
Accused (Anklaget) (2005)	-3.0124
Hip Hop Witch, Da (2000)	-2.9499
Karla (2006)	-2.8457
SuperBabies: Baby Geniuses 2 (2004)	-2.7416
From Justin to Kelly (2003)	-2.6118
Pokémon Heroes (2003)	-2.5886
Criminals (1996)	-2.5124

Movie Effect Regularization

To account for this phenomenon, we regularize our movie effect by penalizing movies that are not rated by many users.

```
lambda <- 5
mu <- mean(train_set$rating)
movie_reg_avgs <- train_set %>%
  group_by(movieId) %>%
  summarize(b_i = sum(rating- mu)/(n() + lambda), n_i = n())

data_frame(original = movie_avgs$b_i,
           regularized = movie_reg_avgs$b_i,
           n = movie_reg_avgs$n_i) %>%
  ggplot(aes(original, regularized, size = sqrt(n))) +
  geom_point(shape = 1, alpha = 0.5)
```

```
## Warning: `data_frame()` is deprecated as of tibble 1.1.0.
## Please use `tibble()` instead.
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_warnings()` to see where this warning was generated.
```



Regularizing for movie effect, we now see movies that are famous and are highly rated.

```
train_set %>%
  dplyr::count(movieId) %>%
  left_join(movie_reg_avgs) %>%
```

```
  left_join(movie_titles, by = "movieId") %>%
  arrange(desc(b_i)) %>%
  select(title, b_i, n) %>%
  slice(1:10) %>%
  knitr::kable()
```

```
## Joining, by = "movieId"
```

title	b_i	n
Shawshank Redemption, The (1994)	0.94703	22340
Godfather, The (1972)	0.90809	14322
Usual Suspects, The (1995)	0.85887	17163
Schindler's List (1993)	0.85046	18531
Double Indemnity (1944)	0.81396	1702
Sunset Blvd. (a.k.a. Sunset Boulevard) (1950)	0.81222	2315
Casablanca (1942)	0.80973	9000
Seven Samurai (Shichinin no samurai) (1954)	0.80308	4159
Rear Window (1954)	0.79647	6343
Godfather: Part II, The (1974)	0.79175	9616

The same applies to movies with low rating.

```
train_set %>%
  dplyr::count(movieId) %>%
  left_join(movie_reg_avgs) %>%
  left_join(movie_titles, by = "movieId") %>%
  arrange(b_i) %>%
  select(title, b_i, n) %>%
  slice(1:10) %>%
  knitr::kable()
```

```
## Joining, by = "movieId"
```

title	b_i	n
From Justin to Kelly (2003)	-2.5331	161
SuperBabies: Baby Geniuses 2 (2004)	-2.4829	48
Pokémon Heroes (2003)	-2.4709	105
Glitter (2001)	-2.3564	271
Pokemon 4 Ever (a.k.a. Pokémon 4: The Movie) (2002)	-2.3402	158
Carnosaur 3: Primal Species (1996)	-2.2655	56
Gigli (2003)	-2.2574	260
Barney's Great Adventure (1998)	-2.2518	166
House of the Dead, The (2003)	-2.1965	163
Yu-Gi-Oh! (2004)	-2.1599	59

Adding the regulation of the movie effects is better than adding the unregulated model.

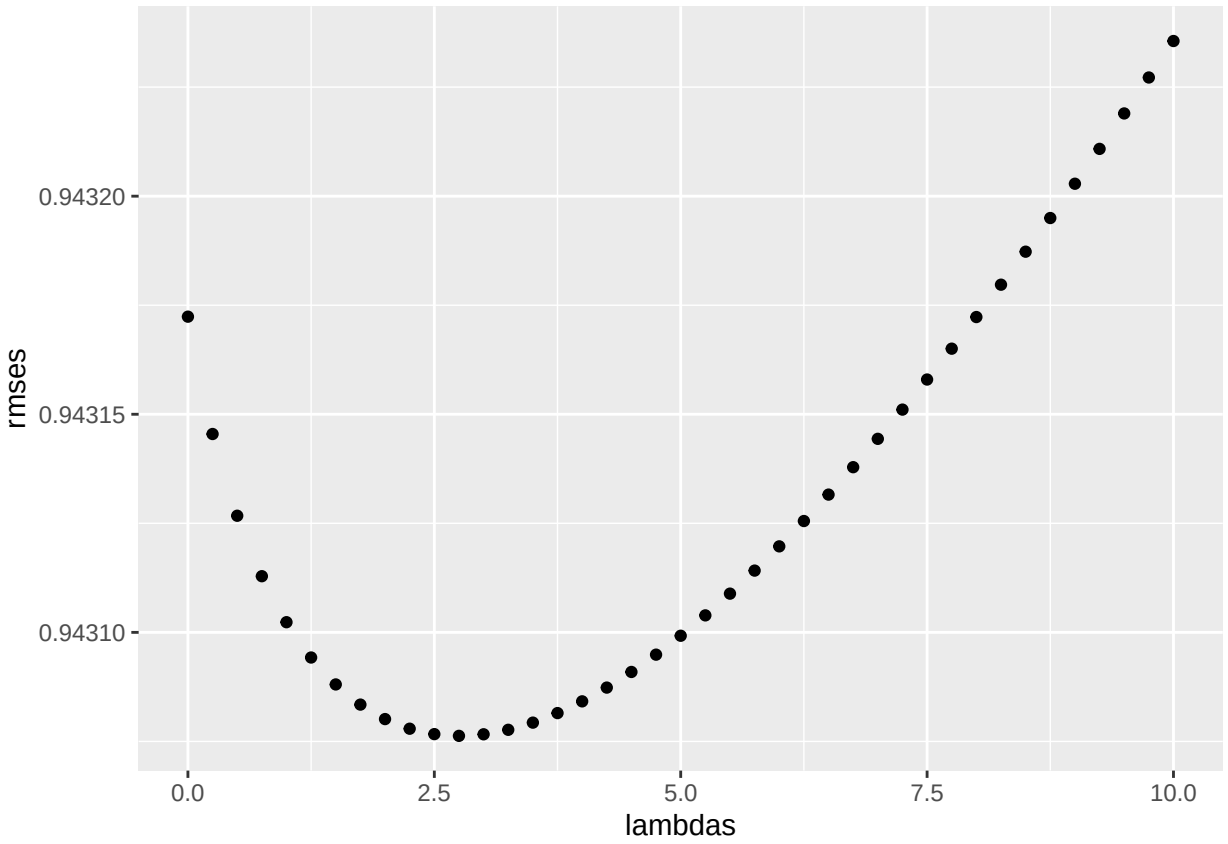
```
predicted_ratings <- test_set %>%
  left_join(movie_reg_avgs, by ="movieId") %>%
  mutate(pred = mu + b_i) %>%
```

```
  .$pred

mu_biReg <- RMSE(test_set_n$rating, predicted_ratings)

rmse_results <- rmse_results %>%
  add_row(method ="Average + Movie Effect Regularized", RMSE = mu_biReg)
rmse_results
```

```
## # A tibble: 6 x 2
##   method                              RMSE
##   <chr>                              <dbl>
## 1 Average                            1.06
## 2 Average + Movie Effect             0.943
## 3 + User Effect                      0.866
## 4 + Week Effect                      0.865
## 5 + Genres                           0.865
## 6 Average + Movie Effect Regularized 0.943
```

However, in regularization, there is the tuning parameter lambda. We want to find the best lambda by testing out a sequence of lambda and calculate the RMSE for each chosen lambda.

```
# check which lambda is the best for regularized movie effect
lambdas <- seq(0, 10, 0.25)
mu <- mean(train_set$rating)

just_the_sum <- train_set %>%
  group_by(movieId) %>%
  summarize(s = sum(rating-mu), n_i = n())


rmses <- sapply(lambdas, function(l){
  predicted_ratings <- test_set %>%
    left_join(just_the_sum, by ="movieId") %>%
    mutate(b_i = s/(n_i+l)) %>%
    mutate(pred = mu + b_i) %>%
    .$pred
  return(RMSE(predicted_ratings, test_set$rating))
})

qplot(lambdas, rmses)
```



The lowest lambda is 2.5.

Thus we enter that number again in our model.

```
lambda <- 2.5
mu <- mean(train_set$rating)
movie_reg_avgs <- train_set %>%
  group_by(movieId) %>%
  summarize(b_i = sum(rating- mu)/(n() + lambda), n_i = n())

predicted_ratings <- test_set %>%
  left_join(movie_reg_avgs, by ="movieId") %>%
  mutate(pred = mu + b_i) %>%
  .$pred

mu_biReg <- RMSE(test_set_n$rating, predicted_ratings)

rmse_results <- rmse_results %>%
  add_row(method ="Average + Movie Effect Regularized (new lambda)",
          RMSE = mu_biReg)

rmse_results
```

```
## # A tibble: 7 x 2
##   method                                          RMSE
##   <chr>                                          <dbl>
## 1 Average                                        1.06
## 2 Average + Movie Effect                         0.943
```

```
## 3 + User Effect                                    0.866
## 4 + Week Effect                                    0.865
## 5 + Genres                                         0.865
## 6 Average + Movie Effect Regularized               0.943
## 7 Average + Movie Effect Regularized (new lambda) 0.943
```

User Effect Regularization

We do this again for the user effects.

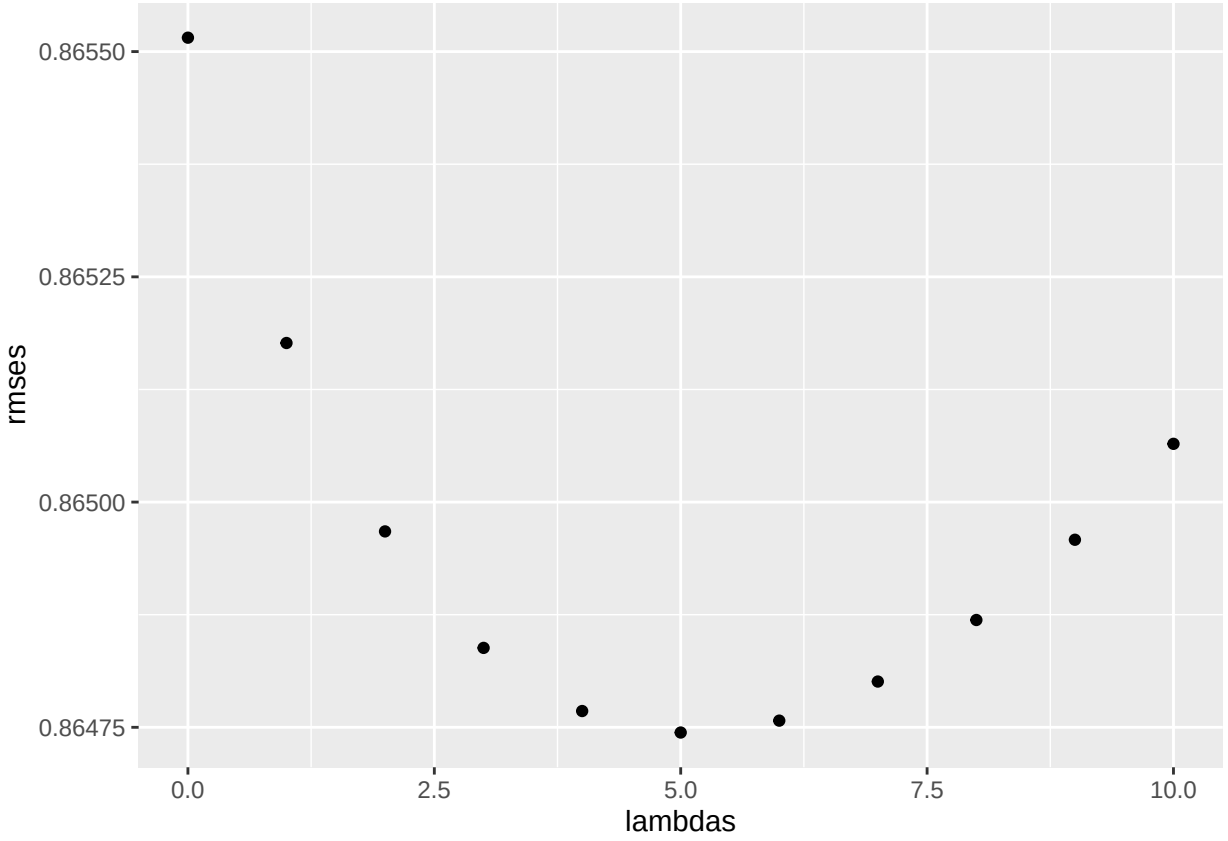
```
lambda <- 5
user_reg_avgs <- train_set %>%
  left_join(movie_reg_avgs, by ="movieId") %>%
  group_by(userId) %>%
  summarize(b_u = sum(rating - mu - b_i)/(n() + lambda))

# predict with regularized b_i and b_u
predicted_ratings <- test_set %>%
  left_join(movie_reg_avgs, by ="movieId") %>%
  left_join(user_reg_avgs, by="userId") %>%
  mutate(pred = mu + b_i + b_u) %>%
  .$pred


mu_biReg_buReg <- RMSE(test_set_n$rating, predicted_ratings)

rmse_results <- rmse_results %>%
  add_row(method ="+ User Effects Regularized",
          RMSE = mu_biReg_buReg)
rmse_results
```

```
## # A tibble: 8 x 2
##   method                                            RMSE
##   <chr>                                            <dbl>
## 1 Average                                          1.06
## 2 Average + Movie Effect                           0.943
## 3 + User Effect                                    0.866
## 4 + Week Effect                                    0.865
## 5 + Genres                                         0.865
## 6 Average + Movie Effect Regularized               0.943
## 7 Average + Movie Effect Regularized (new lambda) 0.943
## 8 + User Effects Regularized                       0.865
```

Checking again for the best lambda.

```
# one step seq as 0.25 requires too much computation power
# -> best lambda is 5

lambdas <- seq(0,10,1)
rmses <- sapply(lambdas, function(l){
  mu <- mean(train_set$rating)
  b_i <- train_set %>%
    group_by(movieId) %>%
    summarize(b_i = sum(rating - mu)/(n()+l))
```

```
  b_u <- train_set %>%
    left_join(b_i, by="movieId") %>%
    group_by(userId) %>%
    summarize(b_u = sum(rating - b_i - mu)/(n()+l))
  predicted_ratings <- test_set %>%
    left_join(b_i, by = "movieId") %>%
    left_join(b_u, by = "userId") %>%
    mutate(pred = mu + b_i + b_u) %>%
    .$pred
  return(RMSE(predicted_ratings, test_set$rating))
})

qplot(lambdas, rmses)
```



Lambda equals to 5 was already the best lambda for this model.

Adding back the other effects

Time effects.

```
test_set_n <- test_set %>%
  mutate(date = round_date(as_datetime(timestamp), unit ="week"))

# time effects
predicted_ratings <- test_set_n %>%
  left_join(movie_reg_avgs, by ="movieId") %>%
  left_join(user_reg_avgs, by ="userId") %>%
```

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


mu_biReg_buReg_bt_rmse <- RMSE(test_set_n$rating, predicted_ratings)

rmse_results <- rmse_results %>%
  add_row(method ="+ Week Effects (new lambda)",
          RMSE = mu_biReg_buReg_bt_rmse)
rmse_results
```

```
## # A tibble: 9 x 2
##   method                                          RMSE
##   <chr>                                          <dbl>
## 1 Average                                        1.06
## 2 Average + Movie Effect                         0.943
## 3 + User Effect                                  0.866
## 4 + Week Effect                                  0.865
## 5 + Genres                                       0.865
## 6 Average + Movie Effect Regularized             0.943
## 7 Average + Movie Effect Regularized (new lambda) 0.943
## 8 + User Effects Regularized                     0.865
## 9 + Week Effects (new lambda)                    0.865
```

Genres Effects.

```
# with genres
predicted_ratings <- test_set_n %>%
  left_join(movie_reg_avgs, by ="movieId") %>%
  left_join(user_reg_avgs, by ="userId") %>%
  left_join(time_avgs, by ="date") %>%
  left_join(genre_avgs, by ="genres") %>%
  mutate(pred = mu + b_i + b_u + b_t + b_g) %>%
  .$pred


mu_biReg_buReg_bt_rmse <- RMSE(test_set_n$rating, predicted_ratings)

rmse_results <- rmse_results %>%
  add_row(method ="+ Genre Effects", RMSE = mu_biReg_buReg_bt_rmse)
rmse_results
```

```
## # A tibble: 10 x 2
##    method                                         RMSE
##    <chr>                                         <dbl>
##  1 Average                                       1.06
##  2 Average + Movie Effect                        0.943
##  3 + User Effect                                 0.866
##  4 + Week Effect                                 0.865
##  5 + Genres                                      0.865
##  6 Average + Movie Effect Regularized            0.943
##  7 Average + Movie Effect Regularized (new lambda) 0.943
```

```
##  8 + User Effects Regularized                      0.865
##  9 + Week Effects (new lambda)                     0.865
## 10 + Genre Effects                                 0.864
```

4. Results

We can conclude that every step we took could decrease our RMSE. Our final model consists of a regularized movie and user effect as well as the normal week and genre effect.

```
rmse_results
```

```
## # A tibble: 10 x 2
##    method                                          RMSE
##    <chr>                                          <dbl>
##  1 Average                                        1.06
##  2 Average + Movie Effect                         0.943
##  3 + User Effect                                  0.866
##  4 + Week Effect                                  0.865
##  5 + Genres                                       0.865
##  6 Average + Movie Effect Regularized             0.943
##  7 Average + Movie Effect Regularized (new lambda) 0.943
##  8 + User Effects Regularized                     0.865
##  9 + Week Effects (new lambda)                    0.865
## 10 + Genre Effects                                0.864
```

Finally we verify our final model on the validation set.

```
# first train our edx data as we splitted it before

# movie effect regularized
lambda <- 2.5
movie_reg_avgs <- edx %>%
  group_by(movieId) %>%
  summarize(b_i = sum(rating- mu)/(n() + lambda), n_i = n())

# user effect regularized
lambda <- 5
user_reg_avgs <- edx %>%
  left_join(movie_reg_avgs, by ="movieId") %>%
  group_by(userId) %>%
  summarize(b_u = sum(rating - mu - b_i)/(n() + lambda))


# predict on validation set
predicted_ratings <- validation %>%
  mutate(date = round_date(as_datetime(timestamp), unit ="week")) %>%
  left_join(movie_reg_avgs, by ="movieId") %>%
  left_join(user_reg_avgs, by ="userId") %>%
  left_join(time_avgs, by ="date") %>%
  left_join(genre_avgs, by ="genres") %>%
  mutate(pred = mu + b_i + b_u + b_t + b_g) %>%
  .$pred
```

```r
RMSE(predicted_ratings, validation$rating)
```

```
## [1] 0.86463
```

5. Conclusion

The widely known movielens data set is a great first machine learning project. In this project we first analyzed the data set by various visualization techniques in order to gain more insights. Indeed, the plots helped us to understand which factors could have influenced the ratings of the users for the movies. With that knowledge we build our model. First we included the movie effect, user effect, week effect and genres effect without regularization. We saw, that many high rated movies were not well known and thus we regularized the movie and user effect. The final model, which included the regularized and unregulated effects could achieve the best RMSE. This model was then applied to the verification set and we achieved a RMSE of 0.86438.

However, the effect of genres was not tidied up as this would have taken too much computing power. In addition, the effects of the age of the movie as well as the time frame from the movie's launch till the user has rated the movie could be further explored.

Nevertheless, as first machine learning project, I had a lot of fun exploring and building models out of the movielens dataset.