# Report on Movielens Data set Analysis for Data Science Capstone Project

Nihar Madkaiker

05 march 2022

## Introduction

This report has been written as a part of a project for the Data Science: Capstone HarvardX - PH125.9x course. The objective of the project is to analyse the Movie lens data set to create a recommendation system based on the previous ratings by viewers.

The movie lens data set is a huge data set with 9000055 observations of 6 variables, the data set covers 10677 unique movies, their ratings and classification by genre. Ratings by 69878 unique viewers are logged.

A lot of the analysis around this data set was popularized by the Netflix prize, which was a challenge to build a better movie recommendation engine for netflix.

The MovieLens Data set is collected by GroupLens Research and can be found on the MovieLens web site (http://movielens.org).

You can read more about the netflix prize here (https://en.wikipedia.org/wiki/Netflix_Prize)

Building upon this, my analysis looks at first the exploratory analysis of the data, identifying the trends in the data set, we then look at what are the potential biases and create models to address the biases, based on which Root mean squared error is calculated to evaluate the effectiveness of each model.

At the outset, (given the data provided), we can look at the user wise effect (how the data varies user to user), the time effect (how ratings have evolved over the years), we can look at the genre effect ( how the data varies by genre), and the movie effect itsel (some movies are popular and hence have more ratings). The models would be evolved around these factors.

## Importing the Data and Creating Partitions

The data is imported through the code provided in the capstone course itself. The code after importing the data set from, http://files.grouplens.org/datasets/movielens/ml-10m.zip

```
##########################################################
# Create edx set, validation set (final hold-out test set)
##########################################################


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

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

```
## -- Attaching packages --------------------------------------- tidyverse 1.3.1 --
```

```
## v ggplot2 3.3.5     v purrr   0.3.4
## v tibble  3.1.6     v dplyr   1.0.7
## v tidyr   1.1.4     v stringr 1.4.0
## v readr   2.1.1     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
```

```
library(tidyverse)
library(caret)
library(data.table)

# 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 <- 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")


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")

# Validation set will be 10% of MovieLens data
set.seed(1, sample.kind="Rounding") # if using R 3.5 or earlier, use `set.seed(1)`
```

```
## Warning in set.seed(1, sample.kind = "Rounding"): non-uniform 'Rounding' sampler
## used
```

```
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)
```

This code above, gives us the partitioned data sets, below we are also adding any additional libraries that may be needed in this project.

```
# extra libraries

library(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
```

```
library(ggplot2)
library(dplyr)
```

We will now conduct exploratory analysis of the data set.

### Summary Statistics and Exploratory Data Analysis

Running some summary statistics to get a better idea of the data sets.

```
str(edx)
```

```
## Classes 'data.table' and '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|Ad
##  - attr(*, ".internal.selfref")=<externalptr>
```

```
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
## 1:                Comedy|Romance
## 2:         Action|Crime|Thriller
## 3:  Action|Drama|Sci-Fi|Thriller
## 4:       Action|Adventure|Sci-Fi
## 5: Action|Adventure|Drama|Sci-Fi
## 6:       Children|Comedy|Fantasy
```

```
summary(edx)
```

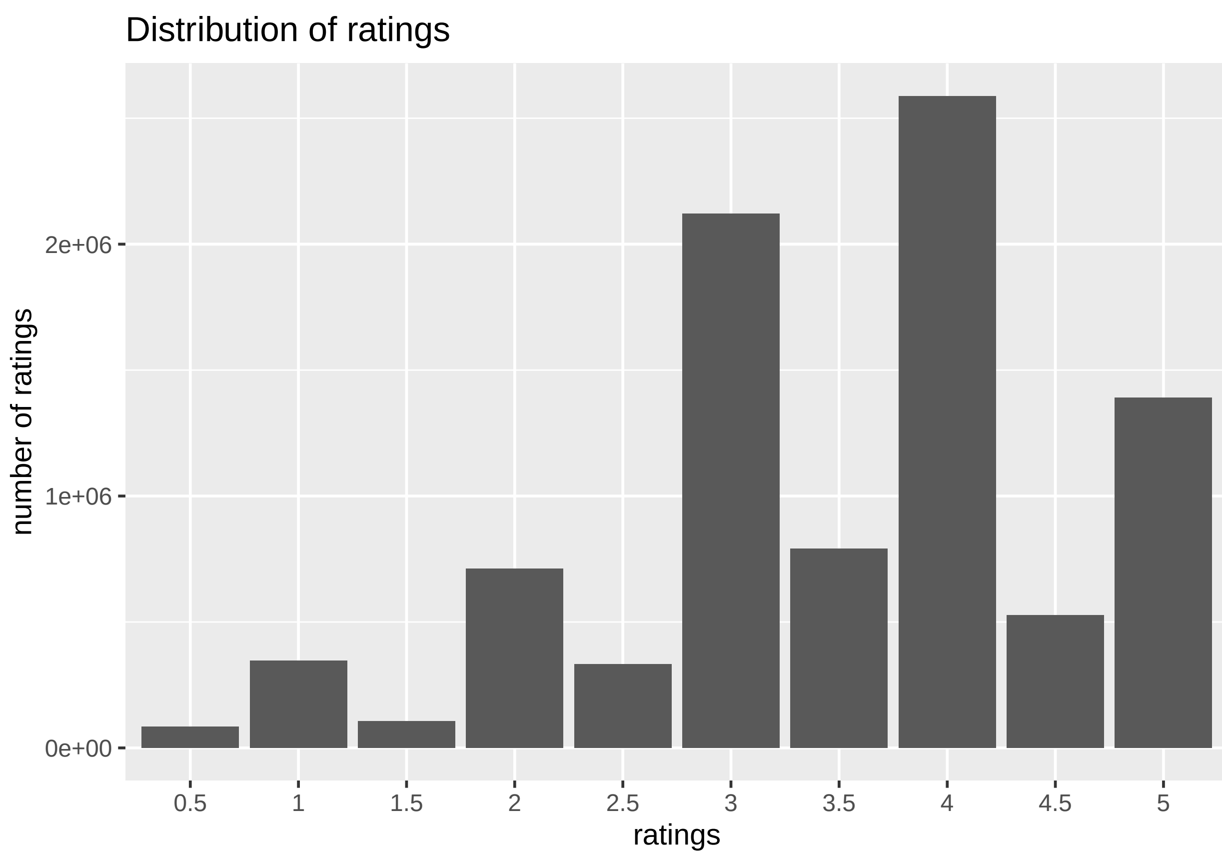
```
##      userId         movieId          rating        timestamp
##  Min.   :    1   Min.   :    1   Min.   :0.500   Min.   :7.897e+08
##  1st Qu.:18124   1st Qu.:  648   1st Qu.:3.000   1st Qu.:9.468e+08
##  Median :35738   Median : 1834   Median :4.000   Median :1.035e+09
##  Mean   :35870   Mean   : 4122   Mean   :3.512   Mean   :1.033e+09
##  3rd Qu.:53607   3rd Qu.: 3626   3rd Qu.:4.000   3rd Qu.:1.127e+09
##  Max.   :71567   Max.   :65133   Max.   :5.000   Max.   :1.231e+09
```

```
##      title              genres
##  Length:9000055     Length:9000055
##  Class :character   Class :character
##  Mode  :character   Mode  :character
##
##
##
```

**conducting exploratory analysis:**

1. We will first look at how the ratings are distributed,

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

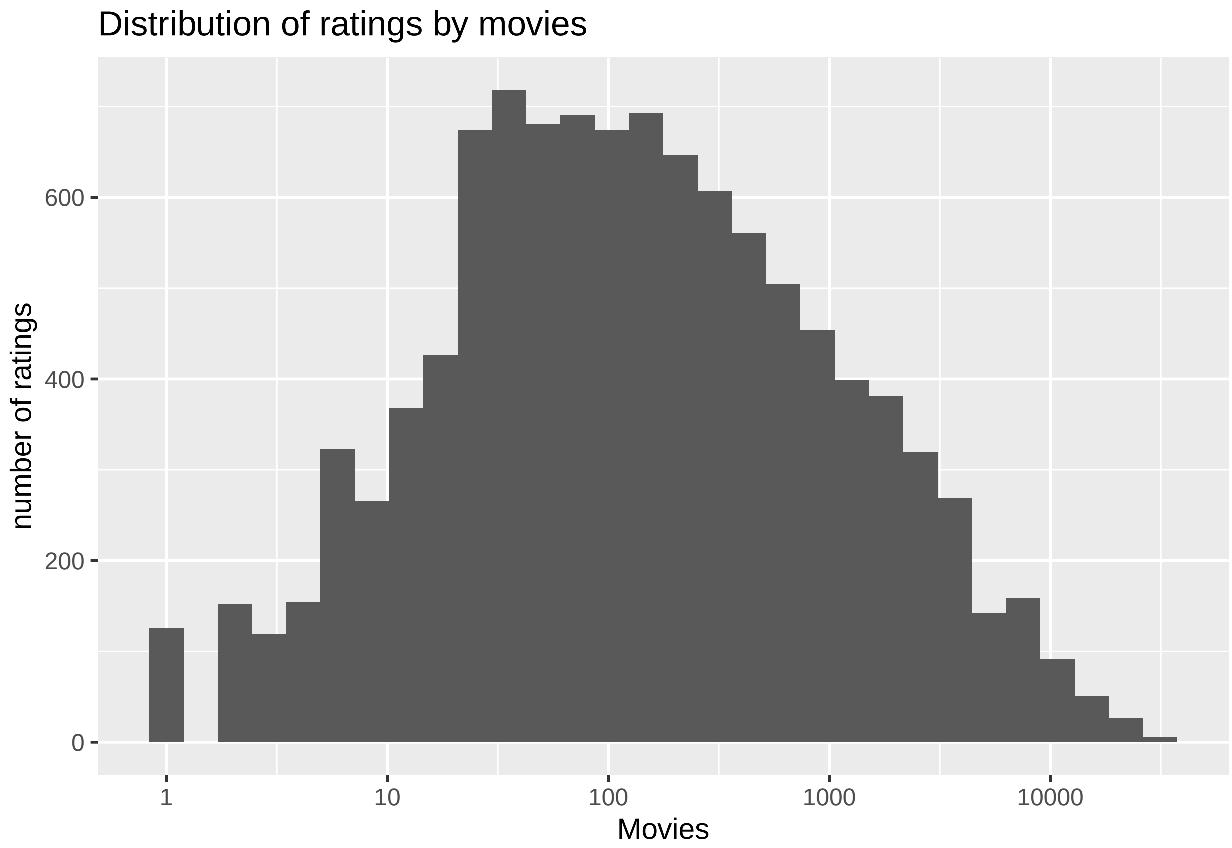



We can see that most of the ratings are between 3 and 4, with 4 being the mode. One of the observations is that positive ratings are more frequently rated as against the negative rating, it might be a case that more poular movies, are acted upon, ie. more people are likely to rate on the popular movies.

2. If we check how ratings are awarded movie wise,

```
movie_sum <- edx %>% group_by(movieId) %>% summarize(n_rating_of_movie = n(), mu_movie = mean(rating),
qplot(movie_sum$n_rating_of_movie, log="x", xlab = "Movies", ylab = "number of ratings")+ ggtitle("Dist
```

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

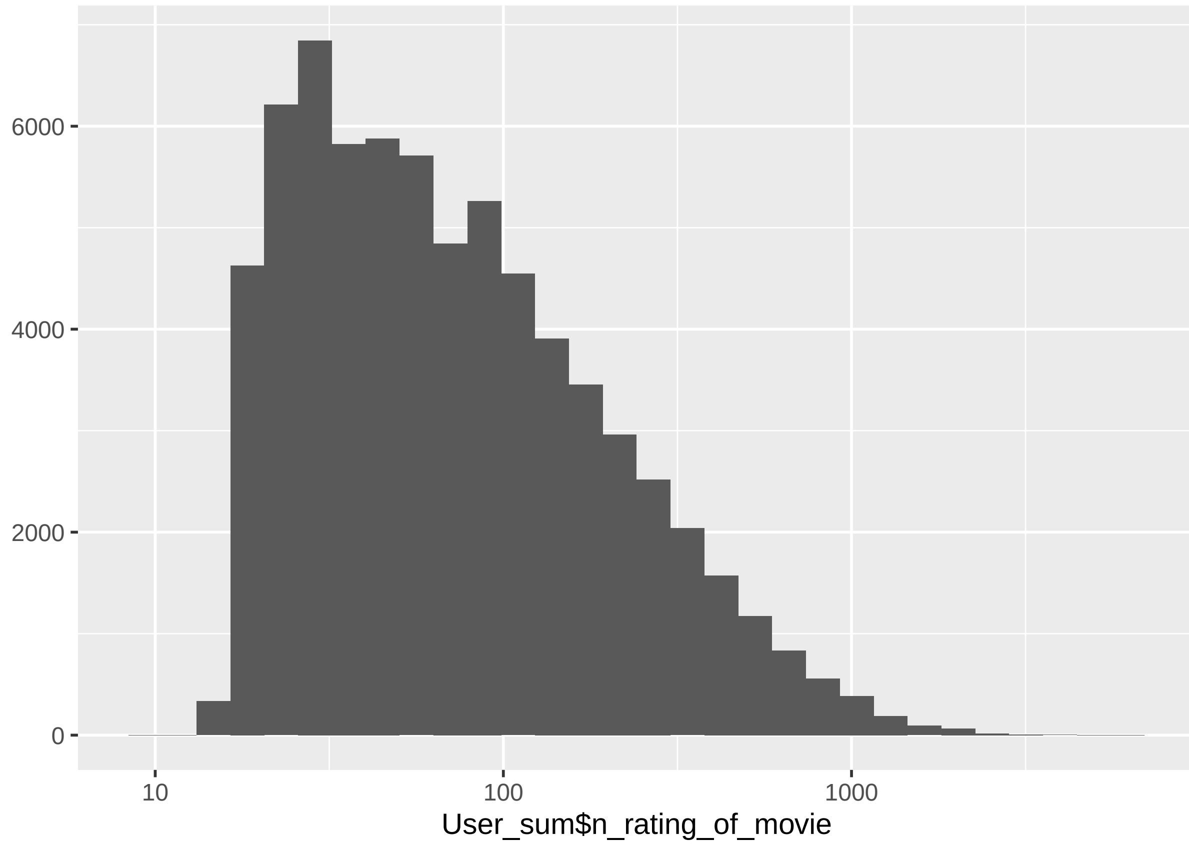



We see that most of the movies are not equally frequently rated, and this would be a case of bias.

3. Checking how many ratings are awarded across movies by users,

```
User_sum <- edx %>% group_by(userId) %>% summarize(n_rating_of_movie = n(), mu_movie = mean(rating), s
qplot(User_sum$n_rating_of_movie, log="x")
```

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




We see that very few users have rated many movies, some of the users are more active in rating than other users.

4. We can further evaluate the effect of genre on the ratings.

```
genres_ind <- str_replace(edx$genres,"\\|.*","")
genres_ind <- genres_ind[!duplicated(genres_ind)]
genres_ind
```

```
##  [1] "Comedy"             "Action"             "Children"
##  [4] "Adventure"          "Animation"          "Drama"
##  [7] "Crime"              "Sci-Fi"             "Horror"
## [10] "Thriller"           "Film-Noir"          "Mystery"
## [13] "Western"            "Documentary"        "Romance"
## [16] "Fantasy"            "Musical"            "War"
## [19] "IMAX"               "(no genres listed)"
```

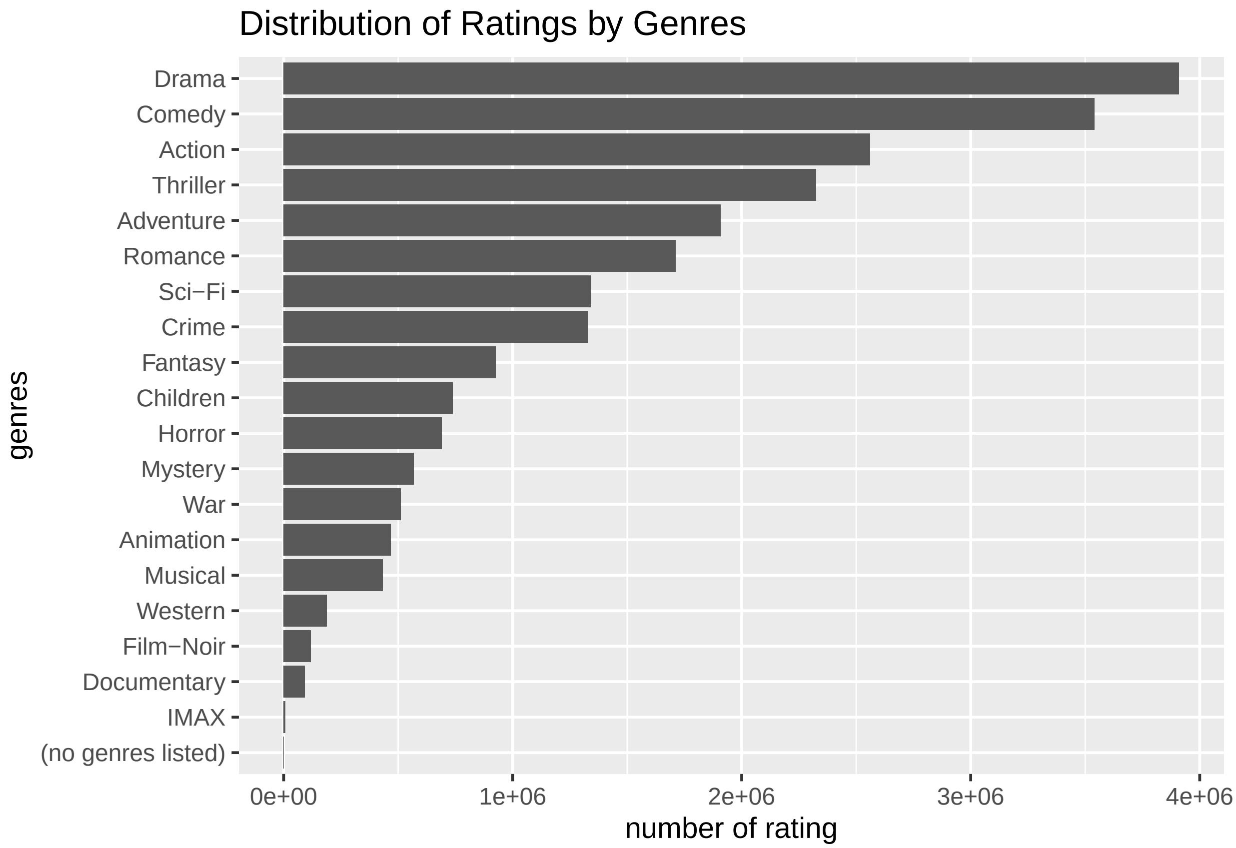
```
 ## this helps us identify the unique genres present in the data set.

n_genres <- sapply(genres_ind, function(Genre_match){
  index <- str_which(edx$genres, Genre_match)
length(edx$rating[index])
})
## this gets the count of ratings by genre.
```

```
genres_rating <- sapply(genres_ind, function(Genre_match){
  index <- str_which(edx$genres, Genre_match)
  mean(edx$rating[index], na.rm = T)
})


##this gets the mean of the ratings against each genre

##combining both of these below:

genres_sum <- data.frame(genres = genres_ind,
                         n_genres = n_genres,
                         average_rating = genres_rating)
genres_sum <- genres_sum %>% arrange(desc(n_genres))

genres_sum
```

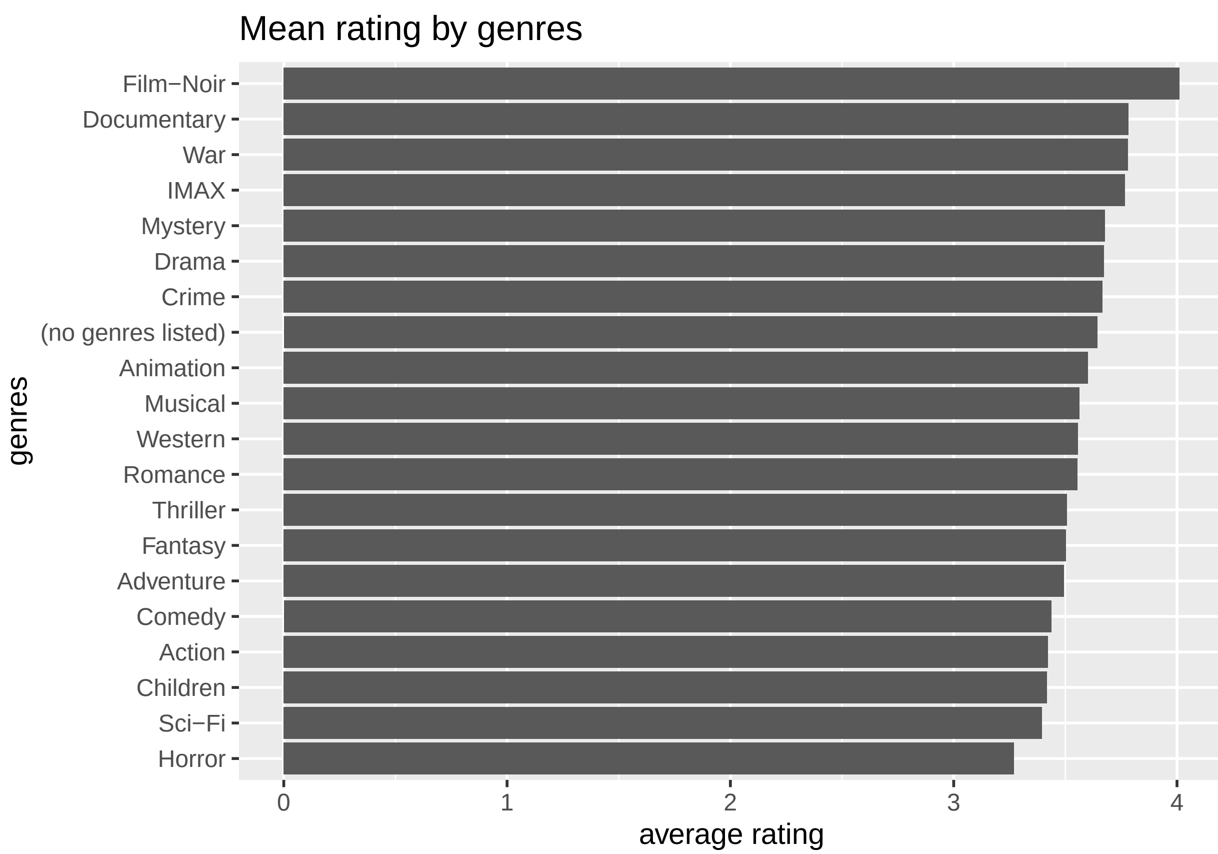
```
##                                genres n_genres average_rating
## Drama                           Drama  3910127       3.673131
## Comedy                         Comedy  3540930       3.436908
## Action                         Action  2560545       3.421405
## Thriller                     Thriller  2325899       3.507676
## Adventure                   Adventure  1908892       3.493544
## Romance                       Romance  1712100       3.553813
## Sci-Fi                         Sci-Fi  1341183       3.395743
## Crime                           Crime  1327715       3.665925
## Fantasy                       Fantasy   925637       3.501946
## Children                     Children   737994       3.418715
## Horror                         Horror   691485       3.269815
## Mystery                       Mystery   568332       3.677001
## War                               War   511147       3.780813
## Animation                   Animation   467168       3.600644
## Musical                       Musical   433080       3.563305
## Western                       Western   189394       3.555918
## Film-Noir                   Film-Noir   118541       4.011625
## Documentary               Documentary    93066       3.783487
## IMAX                             IMAX     8181       3.767693
## (no genres listed) (no genres listed)        7       3.642857
```

a. Plotting the effect of genres on the ratings:

```
## ploting genre wise number of ratings

genres_sum %>% ggplot(aes(x=reorder(genres,n_genres), n_genres)) +
               geom_col() +

               coord_flip() +
               labs(title = "Distribution of Ratings by Genres",
                    y = "number of rating",
                    x = "genres")
```




```
##plotting the genre wise average rating

ggplot(genres_sum, aes(x = reorder(genres, average_rating), average_rating)) +
               geom_col() +
  coord_flip() +
               labs(title = "Mean rating by genres",
                    y = "average rating", x = "genres",
                    )
```




We can see that film noir genre on average has the highest rating and horror has the lowest rating on average, this may be caused due to the low number of ratings in film noir as well. A possible bias due to genre exists.

5. We will now explore the effect of release year on the ratings.

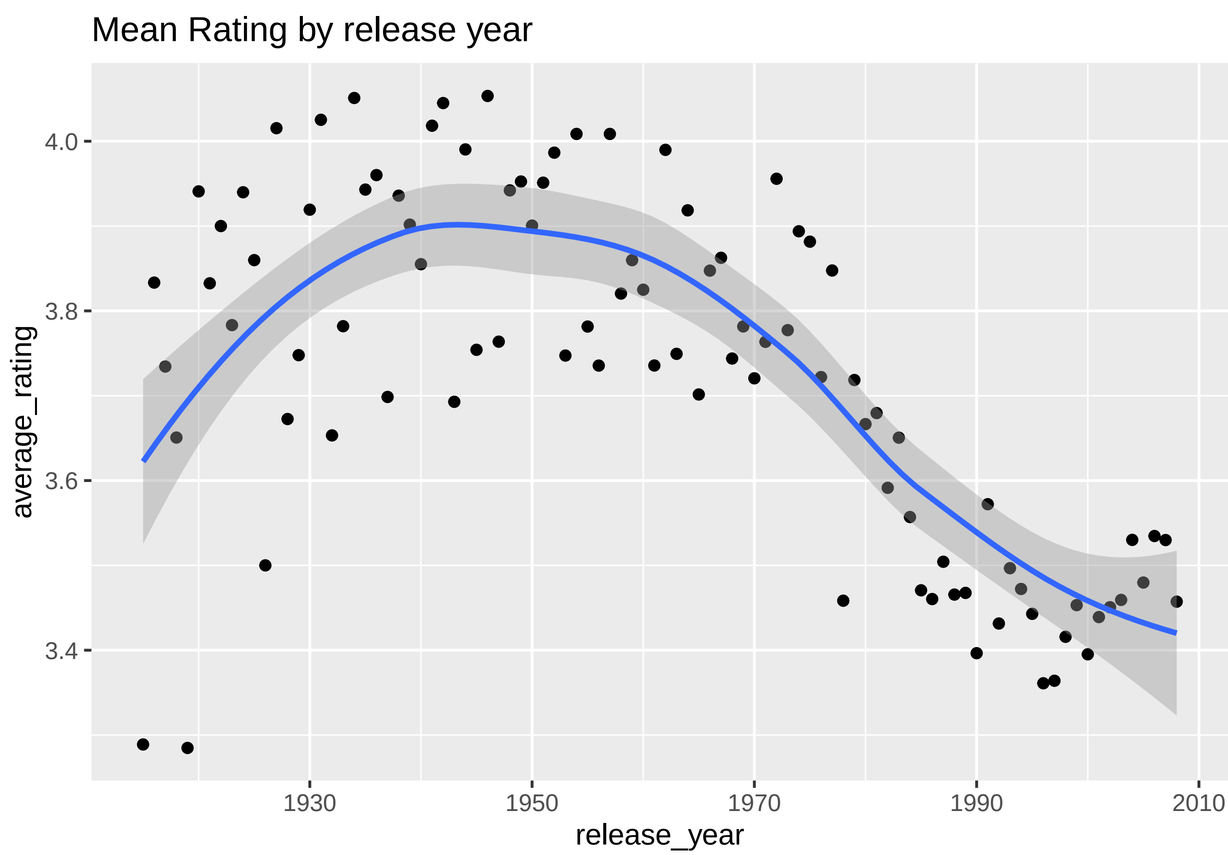
```
edx <- edx %>%
  mutate(rating_time = as.Date(as.POSIXct(timestamp, origin = "1970-01-01"))) %>%
  mutate(rating_year = year(rating_time))

#Adding the release year of each movie.

edx <- edx %>%  mutate(release_year = as.integer(substr(title, str_length(title) - 4,
                              str_length(title) - 1)))
```

Grouping by the release year,

```
release_year_sum <- edx %>% group_by(release_year) %>%
  summarize(n = n(), average_rating = mean(rating))
```

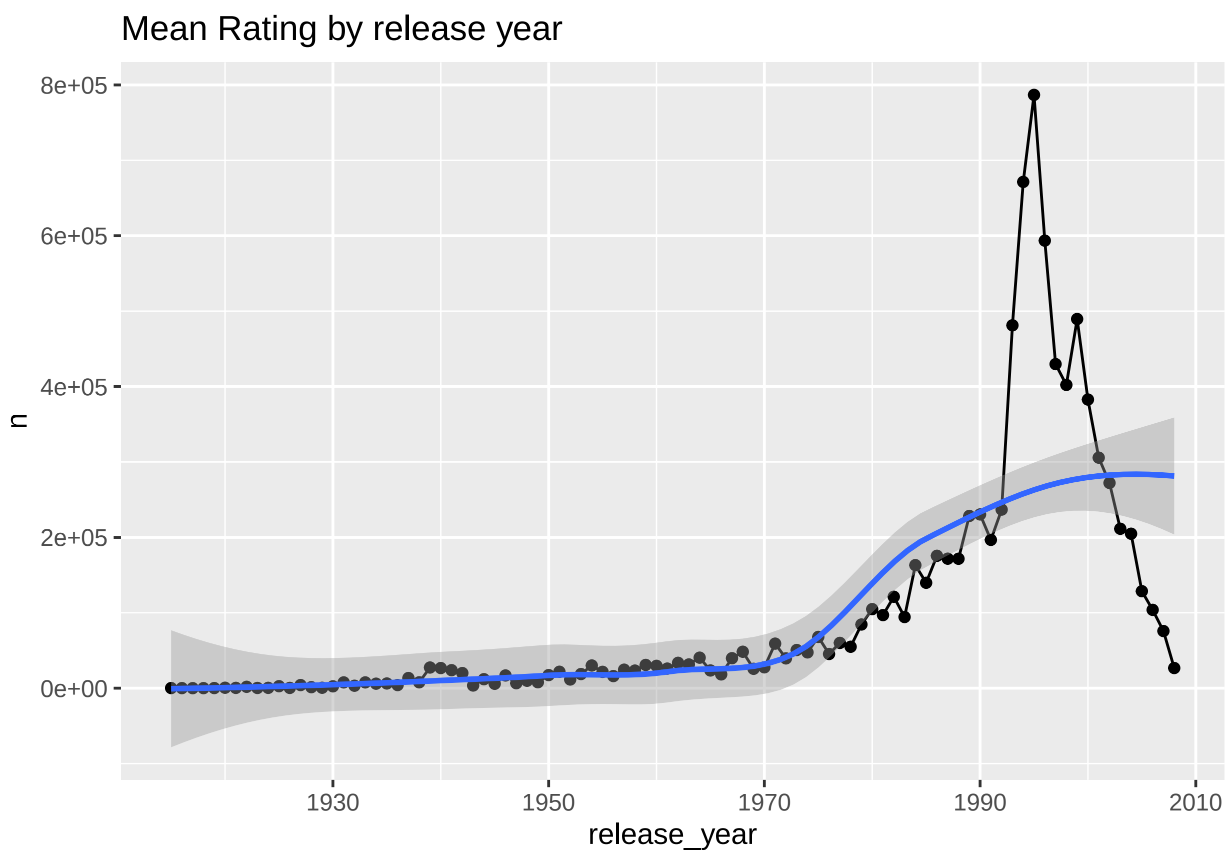
```
release_year_sum %>% ggplot(aes(release_year, average_rating)) +
  geom_point() +
    geom_smooth() +
    labs(title = "Mean Rating by release year",
         )
```

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




```r
release_year_sum %>% ggplot(aes(release_year, n)) +
  geom_point() +
  geom_line()+
    geom_smooth() +
    labs(title = "Mean Rating by release year",
         )
```

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




Another effect we notice is the recency effect, more recent movies are rated more, which could possibly skew the results.

##Building the Model

From the exploratory models we have noticed that the various factors in consideration affect the model. we have observed affects due to,

- The Genre of the movie.

- The Number of users voting

- The release year

- The specific movie in consideration

### Model 1: The Naive model approach

The naive approach would be to assign the mean value of the ratings to all the movies.

By that approach all movies will have the rating:

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

##the rating on average would be :
Model_1 <- mean_rating

Model_1
```

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

The RMSE for this would be:

```
RMSE_model_1 <- RMSE(Model_1, edx$rating)

RMSE_Table <- tibble(method="Model 1: Naive Method",  RMSE = RMSE_model_1)

RMSE_Table
```

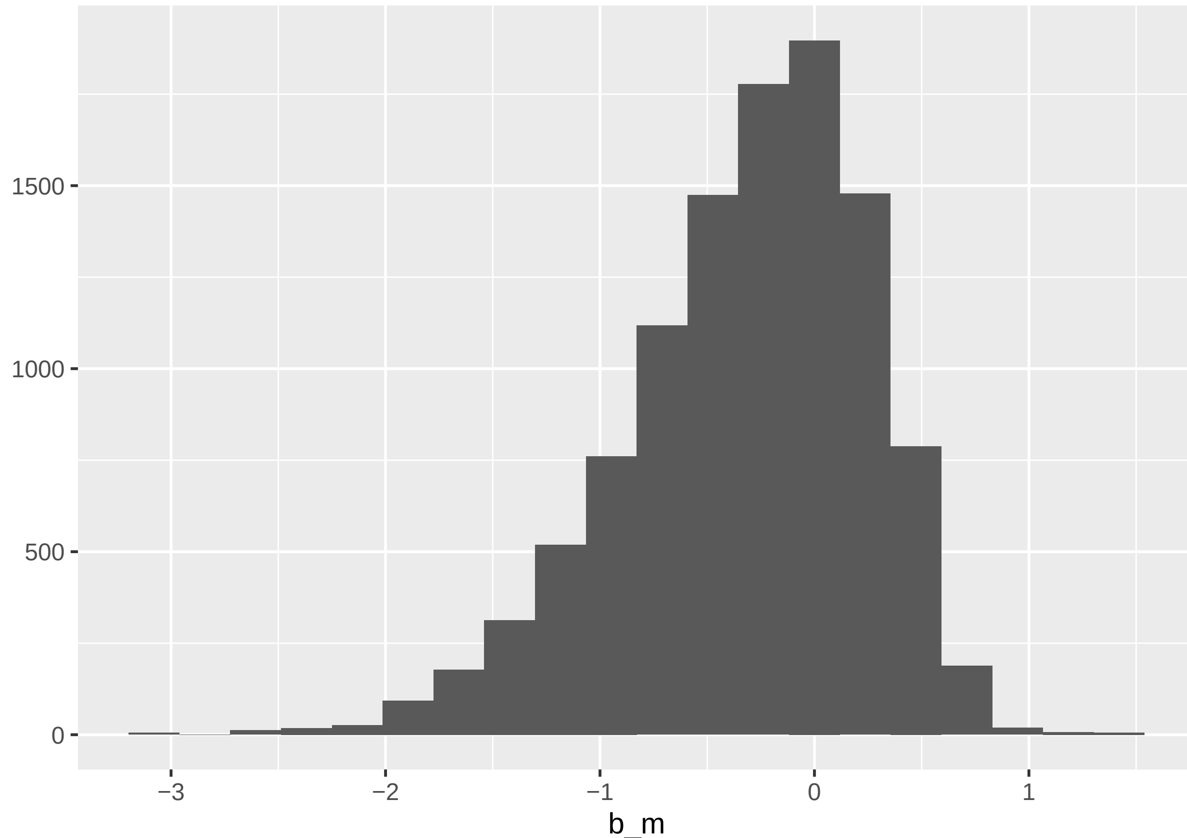
```
## # A tibble: 1 x 2
##   method                 RMSE
##   <chr>                 <dbl>
## 1 Model 1: Naive Method  1.06
```

The RMSE of 1.0603 shows that our estimate is off by more than 1 whole rating point.

### Model 2: Movie effect

We have noted that some movies are rated more than other movies, which can bring in some bias. This can be adjusted for by bringing in a penalty term to consider the effect of this bias.

```
##Calculating the penalty term for the movie effect.

movie_effect <- edx %>% group_by(movieId) %>% summarize(b_m = mean(rating - mean_rating))

movie_effect %>% qplot(b_m, geom = "histogram", bins =20, data =.)
```




The penalty term is visualised above.

```
##The movie effect model is as below,

edx <- edx %>% left_join(movie_effect, by = "movieId") %>%  mutate(Mean_m2 = mean_rating + b_m)
```

```
##RMSE calculation for model 2

RMSE_model_2 <- RMSE(edx$rating, edx$Mean_m2)

RMSE_Table <- bind_rows(RMSE_Table, data_frame(method="Model 2: Movie effect", RMSE =RMSE_model_2 ))
```

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

```
RMSE_Table
```

```
## # A tibble: 2 x 2
##   method                 RMSE
##   <chr>                 <dbl>
## 1 Model 1: Naive Method 1.06
## 2 Model 2: Movie effect 0.942
```

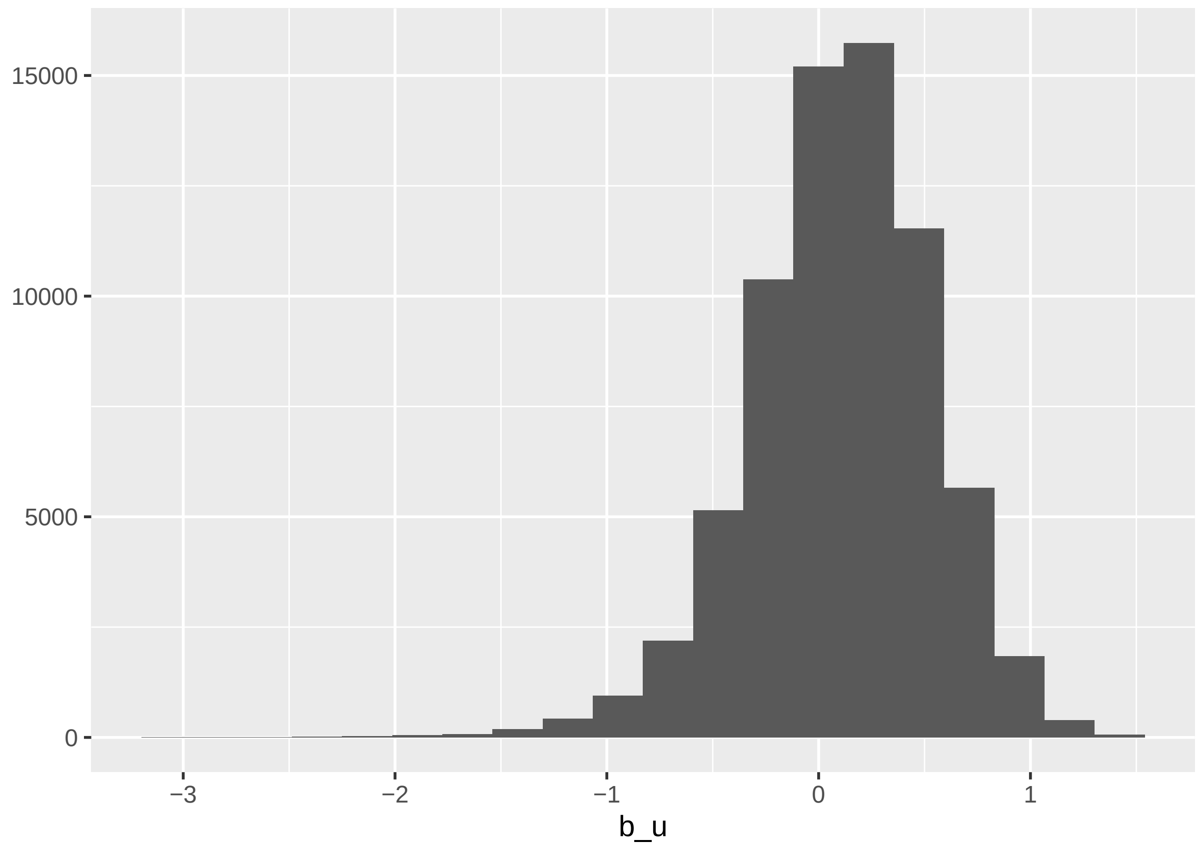
We can see that on considering the movie effect the RMSE has improved from 1.0603 to 0.9423, which is a 11% Reduction in the RMSE. This indicates we are headed in the right direction.

### Model 3: Movie effect + User effect

We have also seen that the user effect is another significant factor affecting the movies.

This can be adjusted for by bringing in a penalty term to consider the effect of this bias.

```
##Calculating the penalty term for the movie effect.

user_effect <- edx %>% group_by(userId) %>% summarize(b_u = mean(rating - mean_rating))

user_effect %>% qplot(b_u, geom = "histogram", bins =20, data =.)
```




The penalty term for user effect is visualised above.

```
##The movie effect model is as below,

edx <- edx %>% left_join(user_effect, by = "userId") %>%  mutate(Mean_m3 = Mean_m2 + b_u)
```

```
##RMSE calculation for model 2

RMSE_model_3 <- RMSE(edx$rating, edx$Mean_m3)

RMSE_Table <- bind_rows(RMSE_Table, data_frame(method="Model 3: Movie + user effect", RMSE =RMSE_model_3

RMSE_Table
```

```
## # A tibble: 3 x 2
##   method                      RMSE
##   <chr>                      <dbl>
## 1 Model 1: Naive Method       1.06
## 2 Model 2: Movie effect       0.942
## 3 Model 3: Movie + user effect 0.877
```

We can see that on considering the movie effect the RMSE has improved from 0.9423 to 0.8767, which is a 6.9% Reduction in the RMSE. This indicates we have improved the model by considering the user effect term.

**NOTE: I have parallely tried incorporating the "genre" effect but got no improvement in the RMSE, in fact it worsened the RMSE hence we wont be considering the Genre effect.**

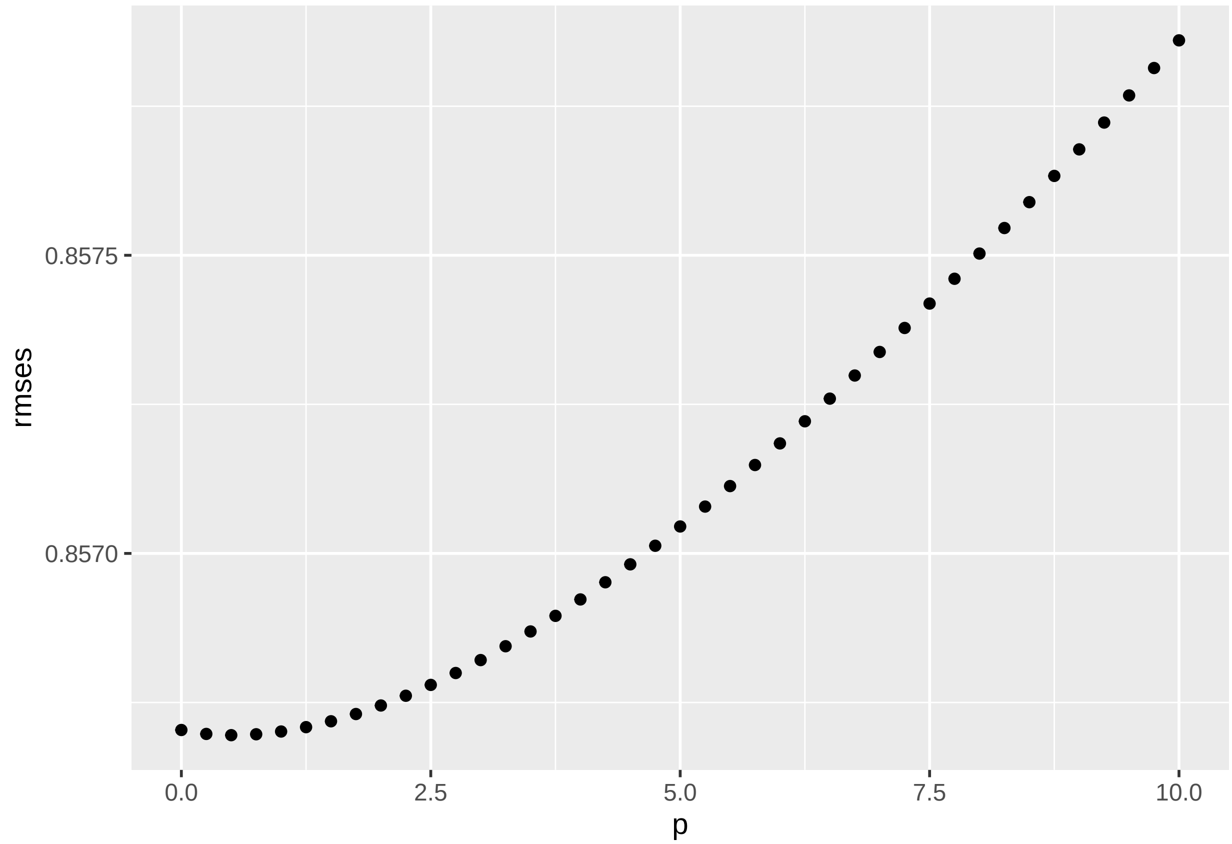
##Model 4: The Regularization model - self learning model

The data set in consideration is noisy, for example, some movies have a single rating, some users may have extremely low number of ratings. All of these factors affect the RMSE adversely. To improve this we can work on a machine learning styled model.

The model would test out multiple cases and identify the best ones.

We choose a tuning parameter - p. p is set to vary from 0 to 10 in intervals of 0.25 this gives us 40 test conditions.

```
##Model 4: Regularisation Approach (MOvie + users)

# p is the tuning parameter, we will cross validate it to chose the best value.

p <- seq(0, 10, 0.25) ##p is set to vary from 0 to 10 in intervals of 0.25 this gives us 40 test condit

# For each p, we will find b_m & b_u, and then run a prediction and test it against the data set.


rmses <- sapply(p, function(A){

mean_r <- mean(edx$rating)

b_movie <- edx %>% group_by(movieId) %>% summarize(b_movie = sum(rating - mean_r)/(n()+ A))
b_user <- edx %>% left_join(b_movie, by="movieId") %>% group_by(userId) %>% summarize(b_user = sum(ratin

predicted_ratings <- edx %>% left_join(b_movie, by = "movieId") %>% left_join(b_user, by = "userId") %>%

return(RMSE(edx$rating,predicted_ratings))

})

# Plotting the rmses vs p to select the optimal tuning factor

qplot(p, rmses)
```




We can check which p value is lowest,

```
p_min <- p[which.min(rmses)]

p_min ## the rmse is lowest for this value
```

```
## [1] 0.5
```

```
##regularising the factors.

# Compute regularized estimates of b_movie using p_min
mean_r <- mean(edx$rating)

Movie_effect_r <- edx %>% group_by(movieId) %>% summarize (b_movie = sum(rating - mean_r)/(n()+p_min), n

# Compute regularized estimates of b_user using p_min
User_effect_r <- edx %>% left_join(Movie_effect_r, by='movieId') %>% group_by(userId) %>% summarize(b_us

# Predicting the ratings
predicted_ratings_r <- edx %>% left_join(Movie_effect_r, by='movieId') %>% left_join(User_effect_r, by=
mutate(prediction = mean_r + b_movie + b_user) %>% .$prediction

# Test and save results
RMSE_model_4 <- RMSE(edx$rating, predicted_ratings_r)

RMSE_Table <- bind_rows(RMSE_Table, data_frame(method="Model 4: Regularized Movie and User Effect Model
```

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

| method | RMSE |
|---|---|
| Model 1: Naive Method | 1.0603313 |
| Model 2: Movie effect | 0.9423475 |
| Model 3: Movie + user effect | 0.8767534 |
| Model 4: Regularized Movie and User Effect Model | 0.8566952 |

We can see that the regularization model considering User and movie factor gives us an RMSE of 0.8566 vs an RMSE of 0.877 in the model no. 3. this is a 2% improvement in RMSE value. We could further improve this using more factors such as genre and release year.

### Model 5: Regularisation model with Movie, User, Genre, and year

```
# b_year and b_genre represent the year & genre effects, respectively

p_hat <- seq(0, 10, 1)


rmses <- sapply(p_hat, function(B){

  mean_r <- mean(edx$rating)

  b_movie <- edx %>% group_by(movieId) %>% summarize(b_movie = sum(rating - mean_r)/(n()+B))

  b_user <- edx %>% left_join(b_movie, by="movieId") %>% group_by(userId) %>% summarize(b_user = sum(ra

  b_year <- edx %>% left_join(b_movie, by='movieId') %>% left_join(b_user, by='userId') %>% group_by(re

  b_genre <- edx %>% left_join(b_movie, by='movieId') %>% left_join(b_user, by='userId') %>% left_join(

predicted_ratings <- edx %>% left_join(b_movie, by='movieId') %>% left_join(b_user, by='userId') %>% le
mutate(pred = mean_r + b_movie + b_user + b_year + b_genre) %>% .$pred

return(RMSE(edx$rating,predicted_ratings))
})
# Compute new predictions using the optimal p_hat

# Test and save results

qplot(p_hat, rmses)
```

##THe above model could not be executed due to memory limitations, and hence is not considered in the further analysis. ##

### Validation

To validate the data we would run the best model (i.e Model 4) through the validation data set.

```
##regularising the factors.

# Compute regularized estimates of b_movie using p_min
mean_r <- mean(edx$rating)

Movie_effect_r <- edx %>% group_by(movieId) %>% summarize (b_movie = sum(rating - mean_r)/(n()+p_min), n

# Compute regularized estimates of b_user using p_min
User_effect_r <- edx %>% left_join(Movie_effect_r, by='movieId') %>% group_by(userId) %>% summarize(b_us

# Predicting the ratings
predicted_ratings_final <- validation %>% left_join(Movie_effect_r, by='movieId') %>% left_join(User_eff
mutate(prediction = mean_r + b_movie + b_user) %>% .$prediction

# Test and save results
RMSE_model_Final <- RMSE(validation$rating, predicted_ratings_final)

RMSE_Table <- bind_rows(RMSE_Table, data_frame(method="Validation Set", RMSE = RMSE_model_Final ))

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

| method | RMSE |
|---|---|
| Model 1: Naive Method | 1.0603313 |
| Model 2: Movie effect | 0.9423475 |
| Model 3: Movie + user effect | 0.8767534 |
| Model 4: Regularized Movie and User Effect Model | 0.8566952 |
| Validation Set | 0.8652226 |

As we can see in the above table, the final model provides an RMSE of 0.865.

## Conclusion

After running multiple analysis we observed that different factors effect the model. The factors such as movie effect, user effect and release year impact the ratings. After considering the various factors we improved the RMSE from 1.06 for the naive model to 0.857 for the regularisation based model.

THe final hold out set RMSE was calculated on the validation set was 0.865.

## References

1] http://movielens.org

2] https://en.wikipedia.org/wiki/Netflix_Prize

3] https://en.wikipedia.org/wiki/MovieLens

4] https://rafalab.github.io/dsbook/