Movie Ratings Prediction Report

Capstone Project

Data Science Professional Certificate Program from HarvardX

Valmeti Srinivas

10/10/2019

Acknowledgements:

I have started the journey of learning R programming language and becoming a Data Scientist in April, 2019 by registering myself for HarvardX Data Science Professional Certificate Program through edX online portal.

My learning journey in this program had been very exhilarating and kept me flushed with enthusiasm all the time. All the 9 courses in the program, culminating in this project are very informative and educational.

I believe personally the program is well designed and apt for data science enthusiasts to start learning. I thank the program instructor Mr.Rafael Irizarry for his superb course delivery and for sharing his data science knowledge and skills.

I also thank the course staff for their excellent technical support through the blog posts and administrative staff for their email assistance.

It is also an opportunity for me to thank the R community, R Studio community, all programmers, developers and all others for their wonderful and dedicated contribution for the development of various packages and libraries, without which this data science project could have not been undertaken.

I would also like to thank Netflix for initiating the challenge and Grouplens for providing the datasets.

Lastly, I express my sincere gratitude to my fellow learners who kept me motivated with their participation in the course, interesting questions, feedback, suggestions and discussions.

Thanks everyone !!

Contents

1. Executive Summary:

In October 2006, Netflix had offered a challenge to the data science community: improve our recommendation algorithm by 10% and win a million dollars. The Netflix data which was offered to the community is not publicly available, but the GroupLens research lab had generated their own database with over 20 million ratings for over 27,000 movies by more than 138,000 users. This project makes use of Grouplens database as further explained below.

HarvardX Data Scientist Certificate Program (through edX) requires doing a Capstone project based on the data (datasets) and guidelines provided in the Capstone Project course, which is the 9th & final course in the certification program.

The essential goal of the project is to

- build a model based on the edx dataset (which is different from the model validation dataset)
- to predict movie ratings such that the RMSE (Root Mean Squared Error) calculated on the validation dataset is equal to or below 0.8649.

RMSE value is a measure of how close the predicted values are to the actual values. The lower the RMSE value the better the model as the same indicates that the difference between the predicted and actual values is not much and thus the model can predict future ratings that much accurately.

Formula for RMSE:

RMSE = Squareroot(Average((Actual ratings - Predicted ratings)^2))

This project report is a summary of the approach to develop an effective model and produce the desired results. Readers shall note that the model does not recommend movies, rather predicts movie ratings.

For this project, I have used the Grouplens dataset as instructed by the course administration team. As guided in the instructions, I have used the 10M version of the Grouplens dataset provided at http://files.grouplens.org/datasets/movielens/ml-10m.zip.

As one might notice, the project work includes application of various tools and techniques that I have learnt in the program courses including application of the knowledge base and skills in R - data analysis, data visualization, inference, data wrangling, data organization, regression, machine learning and modeling.

Finally, apart from creating the model that produces the target RMSE, the project work includes submitting the report in Rmd and PDF formats along with the R Script file.

The key steps that were undertaken in this project have been summarized as below.

- Data set-up step: Here we install/ load the required libraries, download the data files from Grouplens website and create the required datasets.
- Methods & Analysis:
    a. Data Exploration & results: We do data exploration using various methods including visualization & perform various analysis to understand our data.
    b. Model Building & results: We will build our 1st model, visualize & analyse the results, progressively build better models and continue our analysis of the results.
    c. Testing of Final model & RMSE calculation: We will apply the final model on the validation dataset, calculate the RMSE and check if we have achieved the target.
- Results: We will briefly discuss the performance of the models that we have built.
- Conclusion: We will highlight how the final model can be used, limitations of the project and what could be the future enhancements.

2. Data Set-up:

Let us start with the datasets creation as per the code provided in the course. The R script is reproduced here. The below chunk of code essentially loads the required R packages for the project

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

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

```
## -- Attaching packages ---------------------- tidyverse 1.2.1 --
```

```
## v ggplot2 3.2.1     v purrr   0.3.2
## v tibble  2.1.3     v dplyr   0.8.3
## v tidyr   0.8.3     v stringr 1.4.0
## v readr   1.3.1     v forcats 0.4.0
```

```
## -- 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
```

The next step is to create the required datasets.

- Essentially we create 2 datasets - the first "edx" dataset is to train and test the model that we attempt to create and the second dataset "validation" is for final validation of our model and calculating the target RMSE value.
- We start with applying a data partition factor of p = 0.1 to create the required edx & a temporary dataset.
- We make sure "edx" dataset contains all userIds and movieIds of "validation" dataset by removing from "temporary" dataset those movieIds and userIds which are in it but not in "edx" dataset.
- The remaining of the "temporary" dataset will be the required "validation" dataset.
- The removed records from "temporary" dataset will be then added back to our "edx" dataset.
- The above process will create the final "edx" dataset version with approximately 90% data and "validation" dataset with approximately 10% of data.

The below data wrangling process takes sometime, so please bear with it.

```
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(levels(movieId))[movieId],
                                           title = as.character(title),
                                           genres = as.character(genres))

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

# if using R 3.5 or earlier, use `set.seed(1)` instead
set.seed(1, sample.kind="Rounding")
```

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

```
# Validation set will be 10% of MovieLens data
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 temp set back into edx set
removed <- anti_join(temp, validation)
```

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

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

# Create Year column for later analysis
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 object is masked from 'package:base':
##
##     date
```

```
edx <- edx %>% mutate(year=year(as_datetime(timestamp)))

# remove objects that are no longer necessary
rm(dl, ratings, movies, test_index, temp, movielens, removed)
```

3. Methods & Analysis - Data Exploration & Results:

Now that the datasets are created, it is time to do some exploration of the dataset to understand the data. Please note that we have created above an additional variable (column), "year" in the "edx" dataset for later analysis.

Some of the initial data wrangling and exploration described below are the quiz queries in the course as they offer some good insight into our "edx" dataset.

The primary idea of data exploration and analysis is to check which variables have significant effect on ratings and reduce uncertainty caused by anomalies.

Let us first see how many variables and how many records (observations) we have in edx dataset. We will also see how many given ratings are zero (0) rating and how many given are a rating of 3. Note we are finding out the overall number of ratings and not ratings for a specific movie.

```
dim(edx)
```

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

```
glimpse(edx)
```

```
## Observations: 9,000,055
## Variables: 7
## $ userId    <int> 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1...
## $ movieId   <dbl> 122, 185, 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...
## $ timestamp <int> 838985046, 838983525, 838983421, 838983392, 83898339...
## $ title     <chr> "Boomerang (1992)", "Net, The (1995)", "Outbreak (19...
## $ genres    <chr> "Comedy|Romance", "Action|Crime|Thriller", "Action|D...
## $ year      <dbl> 1996, 1996, 1996, 1996, 1996, 1996, 1996, 1996, 1996...
```

```
edx %>% filter(rating==0) %>% count()
```

```
## # A tibble: 1 x 1
##       n
##   <int>
## 1     0
```

```
edx %>% filter(rating==3) %>% count()
```

```
## # A tibble: 1 x 1
##         n
##     <int>
## 1 2121240
```

As we can see there are over 9 Million ratings and 7 variables - userId, movieId, rating, timestamp, title, genre & year. Also note that there are some genre types that are combination of two or more genres.

We can say "rating" is the predicted/ response variable and all others are predictor variables in our dataset. Predicting "rating" using other variables is the problem at hand.

We can also see there are no movies with '0' rating and rating '3' is a common rating.

The next step in our data exploration is to check how many unique movies, how many unique users and how many unique genre types are there in the dataset for a better understanding of the data.

```
n_distinct(edx$movieId)
```

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

```
n_distinct(edx$userId)
```

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

```
n_distinct (edx$genres)
```

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

As mentioned earlier, genre types include combination of genres . For example a movie might belong to thriller, horror and action combination genre.

So let us find out how many movie ratings are there under the below mentioned sample genres - Drama, Comedy, Thriller & Romance. This involves matching and counting ratings that fall under individual genres (ex: Drama) and combination genres (ex: Drama | Romance).

```
edx %>% filter(grepl("Drama", genres)) %>% count()
```

```
## # A tibble: 1 x 1
##         n
##     <int>
## 1 3910127
```

```
edx %>% filter(grepl("Comedy", genres)) %>% count()
```

```
## # A tibble: 1 x 1
##         n
##     <int>
## 1 3540930
```

```
edx %>% filter(grepl("Thriller", genres)) %>% count()
```

```
## # A tibble: 1 x 1
##         n
##     <int>
## 1 2325899
```

```
edx %>% filter(grepl("Romance", genres)) %>% count()
```

```
## # A tibble: 1 x 1
##         n
##     <int>
## 1 1712100
```

Next, we will see

- the top 10 movies which have received the most number of ratings,
- the top 10 users who have given most number of ratings,
- the top 10 genres which have got most ratings and
- which ratings have highest rate.

We will also see how common are half star ratings compared to full star ratings.

```
# Top 10 movies with maximum ratings
edx %>% group_by(movieId,title) %>% summarize(count=n()) %>%
  arrange(desc(count)) %>% top_n(10)
```

```
## Selecting by count
```

```
## # A tibble: 10,677 x 3
## # Groups:   movieId [10,677]
##    movieId title                                                 count
##      <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) (19~ 25672
## 10     150 Apollo 13 (1995)                                      24284
## # ... with 10,667 more rows
```

```
# Top 10 users who gave maximum number of ratings
edx %>% group_by(userId) %>% summarize(count=n()) %>%
  arrange(desc(count)) %>% top_n(10)
```

```
## Selecting by count
```

```
## # A tibble: 10 x 2
##    userId count
##     <int> <int>
##  1  59269  6616
##  2  67385  6360
##  3  14463  4648
##  4  68259  4036
##  5  27468  4023
##  6  19635  3771
##  7   3817  3733
##  8  63134  3371
##  9  58357  3361
## 10  27584  3142
```

```r
# Top 10 genres with maximum number of ratings
edx %>% group_by(genres) %>% summarize(count=n()) %>%
  arrange(desc(count)) %>% top_n(10)
```

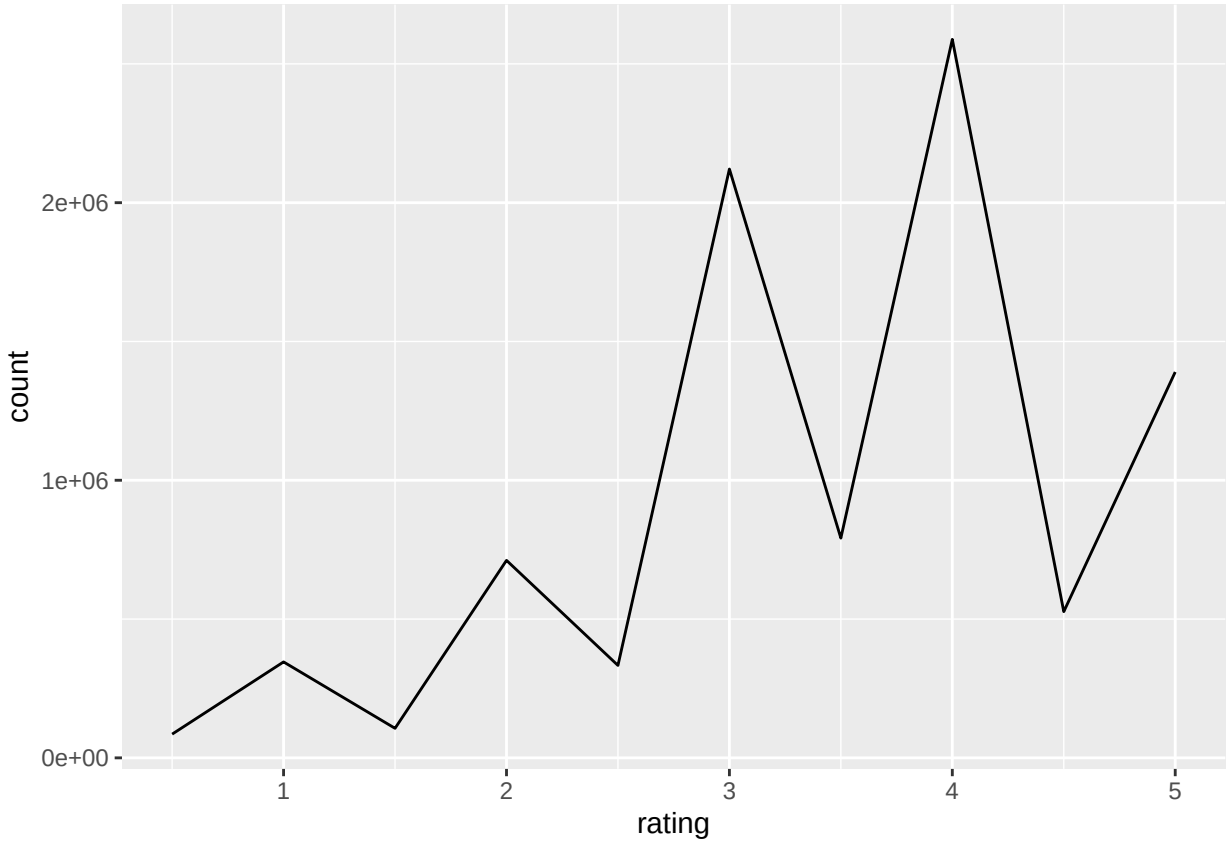
```
## Selecting by count
```

```
## # A tibble: 10 x 2
##    genres                      count
##    <chr>                       <int>
##  1 Drama                      733296
##  2 Comedy                     700889
##  3 Comedy|Romance             365468
##  4 Comedy|Drama               323637
##  5 Comedy|Drama|Romance       261425
##  6 Drama|Romance              259355
##  7 Action|Adventure|Sci-Fi    219938
##  8 Action|Adventure|Thriller 149091
##  9 Drama|Thriller             145373
## 10 Crime|Drama                137387
```

```r
# Ratings in the order of maximum rate
edx %>% group_by(rating) %>% summarize(count=n()) %>%
  arrange(desc(count))
```

```
## # A tibble: 10 x 2
##    rating   count
##     <dbl>   <int>
##  1    4   2588430
##  2    3   2121240
##  3    5   1390114
##  4    3.5  791624
##  5    2    711422
##  6    4.5  526736
##  7    1    345679
##  8    2.5  333010
##  9    1.5  106426
## 10    0.5   85374
```

```r
# Half star ratings Vs full star ratings
edx %>% group_by(rating) %>%
  summarize(count = n()) %>%
  ggplot(aes(x = rating, y = count)) +
  geom_line()
```



From the above first 3 tables, we see the top 10 movies, top 10 genres that have received maximum ratings and top 10 users who have given maximum ratings.

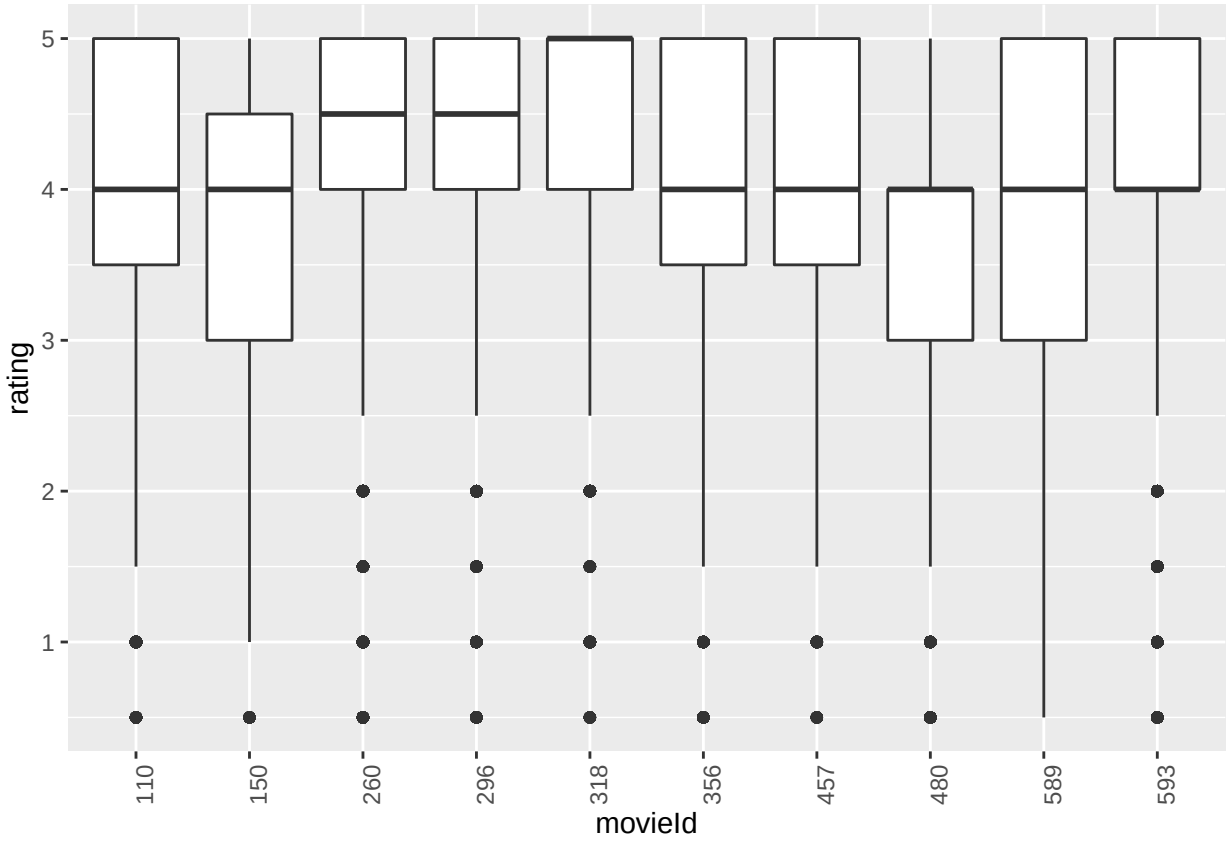
We can also see generally users tend to give higher ratings than lower ratings. Ratings 3, 3.5, 4 & 5 account for about 6.9 Million out of the 9 million ratings. From the plot, we also could see that users tend to give more whole star ratings than the half star ratings

Next, we will plot these top 10 movies and top 10 genres for the ratings they have received and the top 25 users for the ratings they have given using slightly different data wrangling code and ggplot.

```
# Store top 10 Movies by the number of ratings
store_top_10ms <- edx %>%
  dplyr::count(movieId) %>%
  top_n(10) %>%
  pull(movieId)
```

```
## Selecting by n
```

```
# Plot ratings for top 10 movies
edx %>%
  filter(movieId %in% store_top_10ms) %>%   mutate (movieId=as.character(movieId)) %>%
  select(movieId, rating) %>%
  ggplot(aes(movieId,rating)) +
  geom_boxplot() +
  theme(axis.text.x = element_text(angle = 90))
```

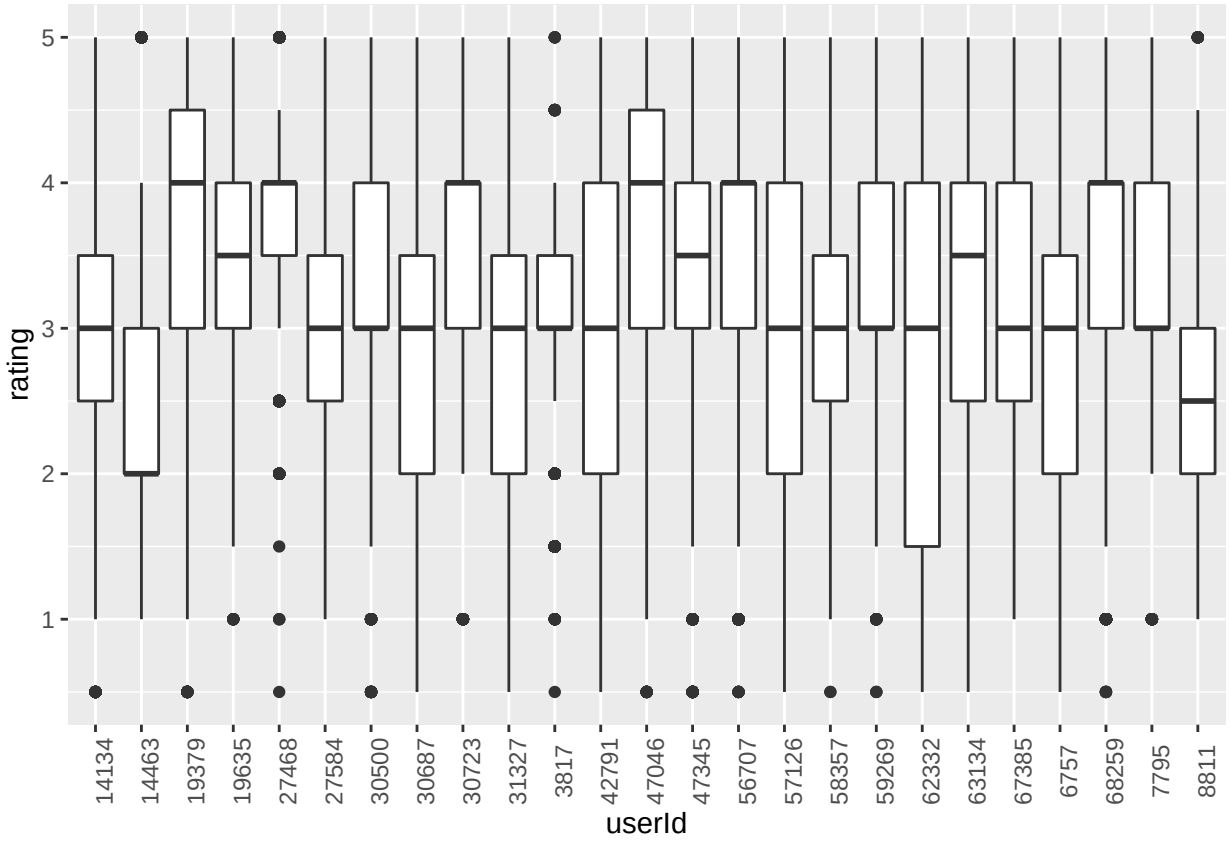


From the above plot, we could infer that the top 10 Movies that have received maximum ratings have generally been given a median rating of 4 or above. This is true given the fact that popular movies tend to get higher ratings. The Inter Quartile Range (IQR) for these movies is generally within 1 to 1.5 ratings. This certainly tells us that every individual movie has an effect on ratings it gets.

```
# Store top 25 users by the number of ratings
store_top_25us <- edx %>%
  dplyr::count(userId) %>%
  top_n(25) %>%
  pull(userId)
```

```
## Selecting by n
```

```
# Plot Top 25 users ratings
edx %>%
  filter(userId %in% store_top_25us) %>%
  mutate (userId=as.character(userId)) %>%
  select(userId, rating) %>%
  ggplot(aes(userId,rating)) +
  geom_boxplot() +
  theme(axis.text.x = element_text(angle = 90))
```

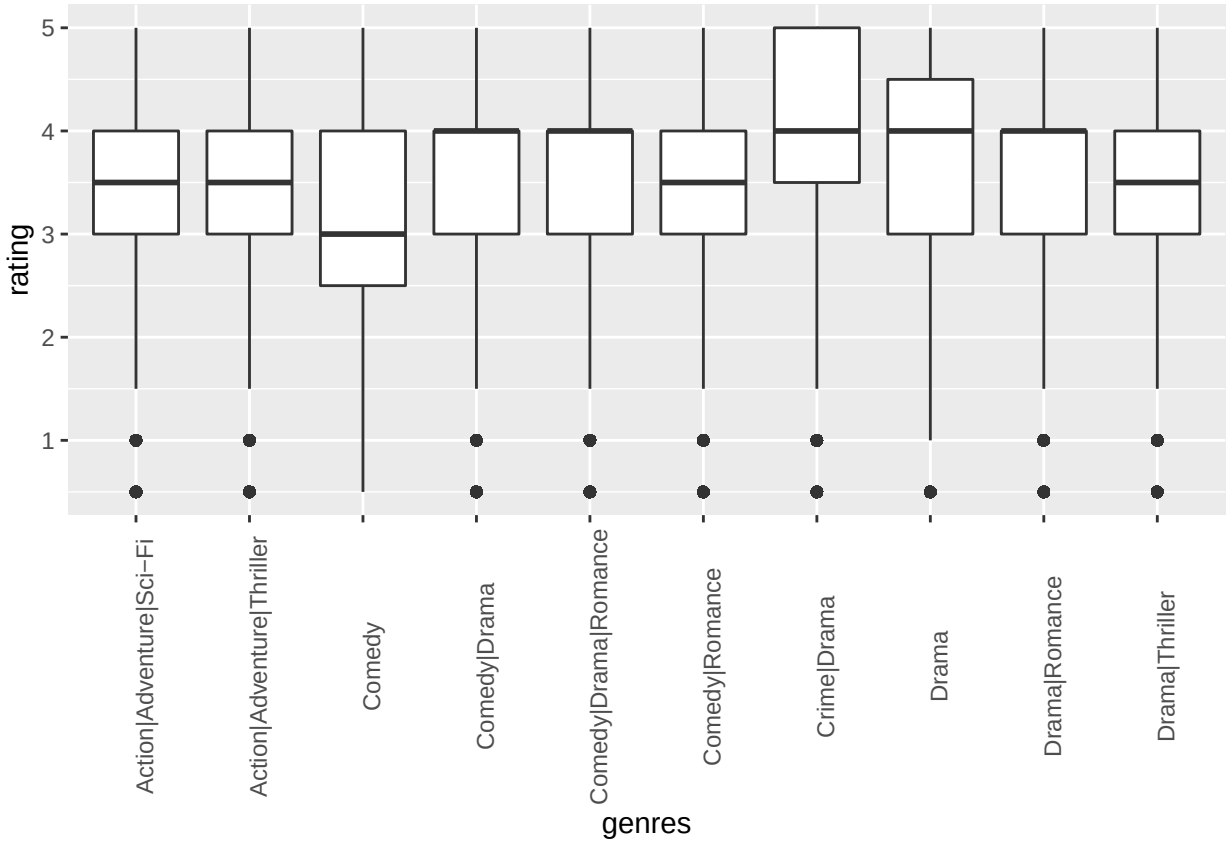


We can see that, excepting userIds 14463 & 8811, the top 25 Users who have given maximum number of ratings have generally given a median rating of 3 or above. This is also true because as we have seen earlier, the general tendency is to give more higher ratings and the more a user rates, the more higher ratings he gives. Also, we can note that the IQR is about 1 to 1.5 ratings for most users. This confirms that each individual user too have an effect on movie ratings.

```
# Store top 10 genres by the number of ratings
store_top_10gs <- edx %>%
  dplyr::count(genres) %>%
  top_n(10) %>%
  pull(genres)
```

```
## Selecting by n
```

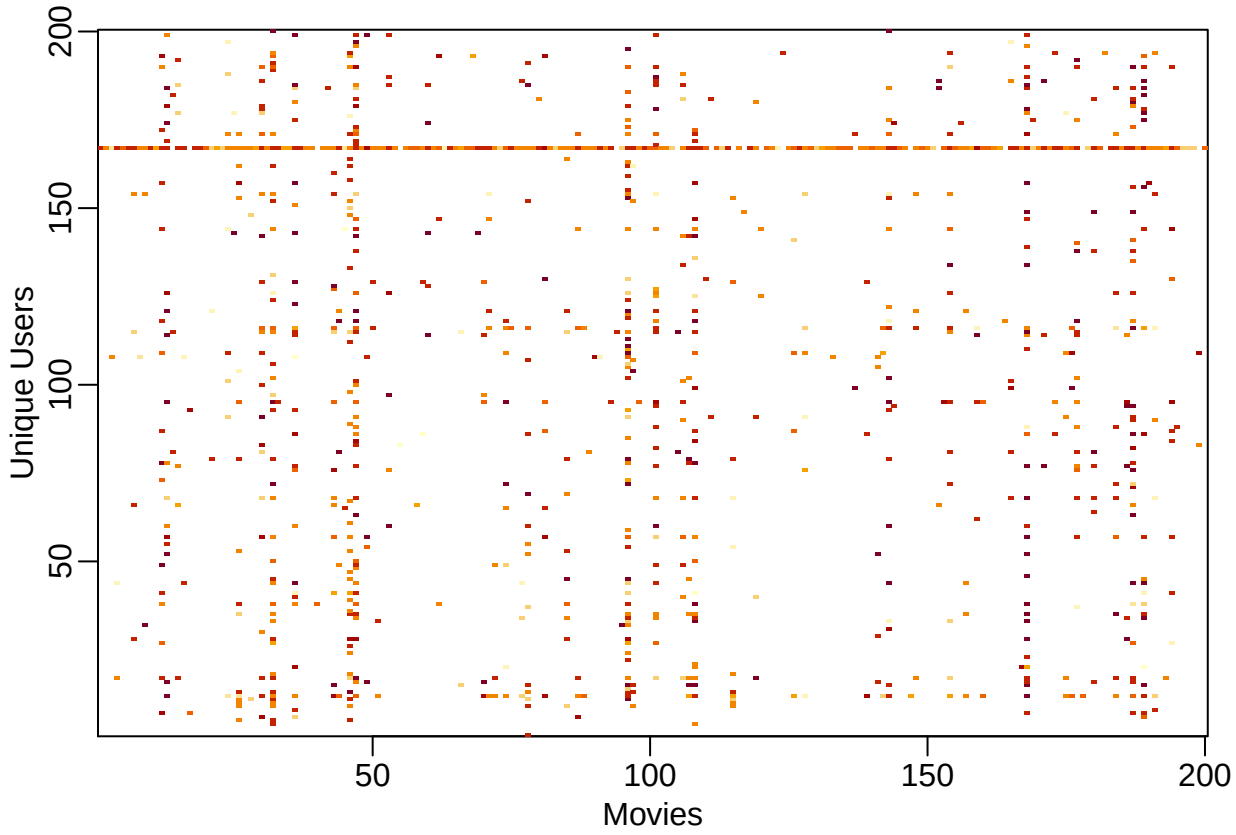
```
# Plot Top 10 genres ratings
edx %>%
  filter(genres %in% store_top_10gs) %>%
  select(genres,movieId, rating) %>%
  ggplot(aes(genres,rating)) +
  geom_boxplot() +
  theme(axis.text.x = element_text(angle = 90))
```



The top 10 genres all have relatively higher ratings and generally between 3 and 4. The IQR for top 10 genres is almost within 1 rating. This indicates genres too have an effect on movie ratings.

Next, not every user would have rated every movie. In fact, we can see from the below matrix for a random sample of 200 movies and 200 users, how sparse the rating is, with colored cells indicating a user/movie combination for which we have some rating.

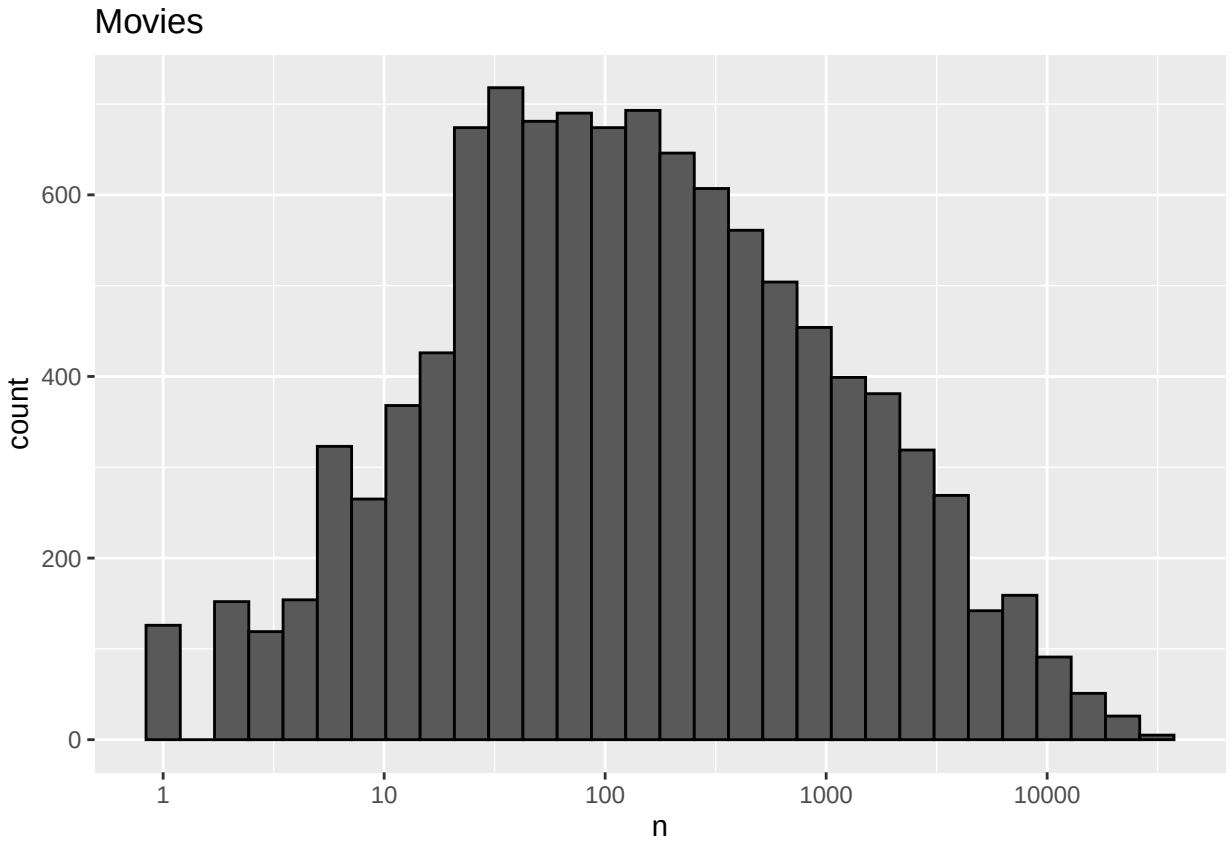
```
# all users might not rate all movies
uni_200users <- sample(unique(edx$userId), 200)
rafalib::mypar()
edx %>% filter(userId %in% uni_200users) %>%
  select(userId, movieId, rating) %>%
  spread(movieId, rating) %>% select(sample(ncol(.), 200)) %>%
  as.matrix() %>% t(.)%>%
  image(1:200, 1:200,. , xlab="Movies", ylab="Unique Users")
```



Curiously, one user (approximately between 160th - 170th of the 200 random users) seems to have rated almost all 200 random movies we picked and has given almost the same rating (same color) for all.

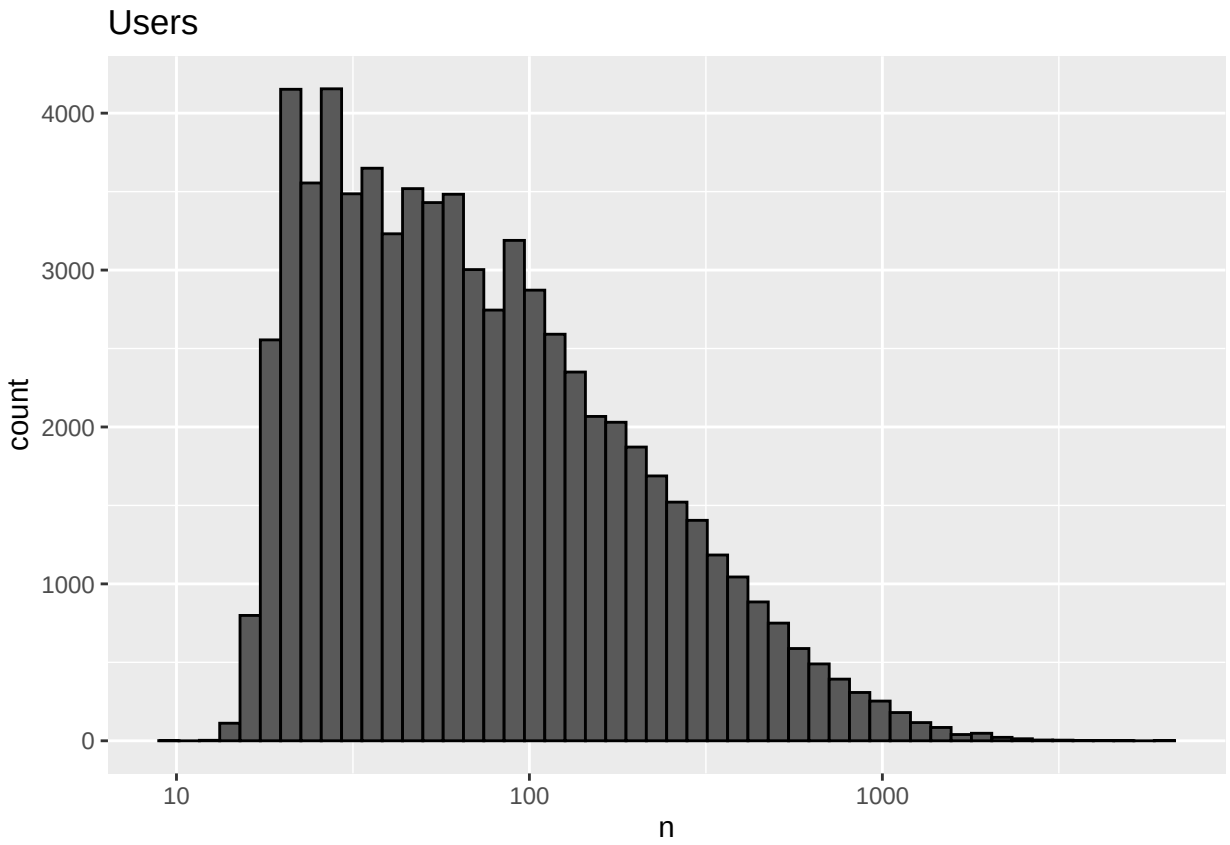
The below plot shows some movies get rated more than others. This is a fact because popular movies get rated more times.

```
# Distribution of movies based no.of ratings received
edx %>%
  dplyr::count(movieId) %>%
  ggplot(aes(n)) +
  geom_histogram(bins = 30, color = "black") +
  scale_x_log10() +
  ggtitle("Movies")
```



Similarly, some users are more active in rating, than others. The below plot proves the same.

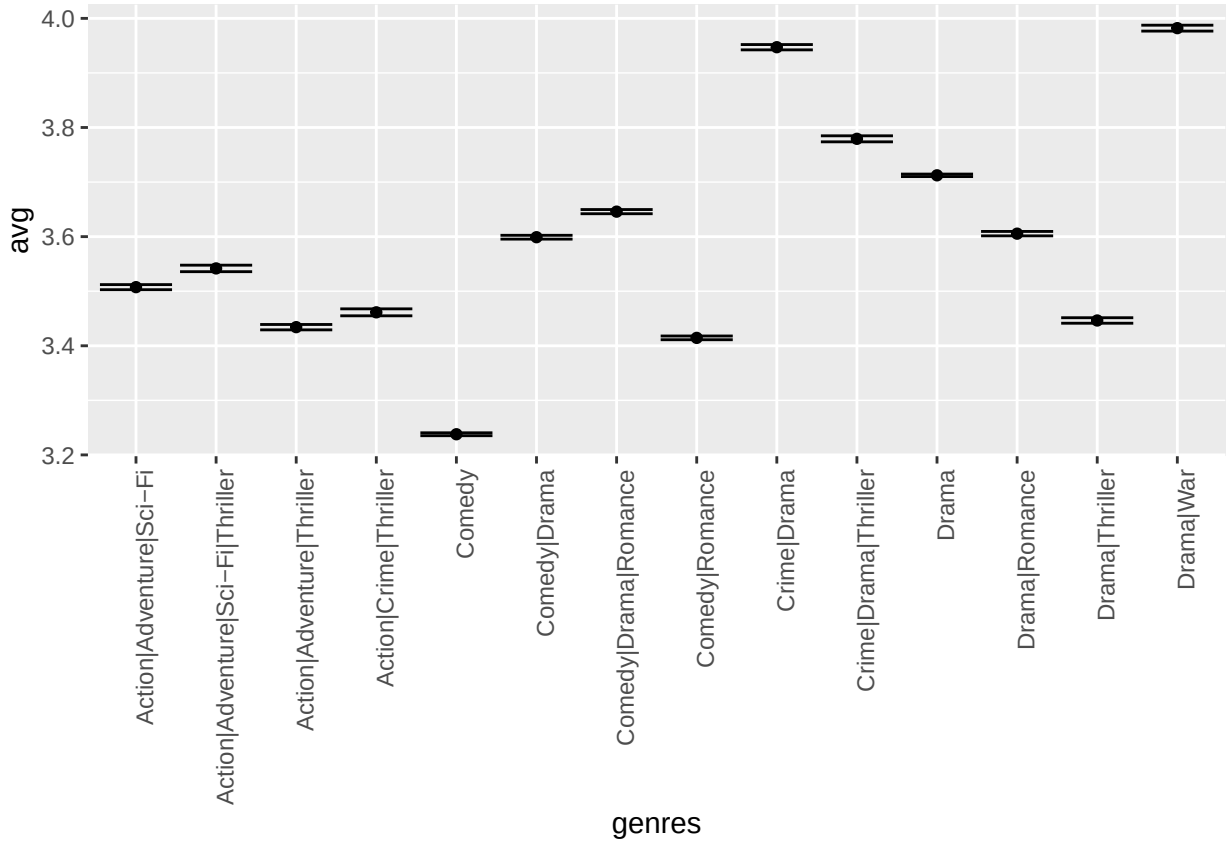
```
# Distribution of users based on no.of ratings ratings given
edx %>%
  dplyr::count(userId) %>%
  ggplot(aes(n)) +
  geom_histogram(bins = 50, color = "black") +
  scale_x_log10() +
  ggtitle("Users")
```



The above confirms that each individual user has a clear effect on the rating he/ she gives for a movie.

Let us consider genres with more than 100,000 ratings. Then compute the average and standard error for each category and plot these as error bar plots with 2 standard errors of upper and lower limits. The below plot shows clearly there is no overlap between any two genres suggesting a very strong evidence of a genre effect on ratings.

```
# Distribution of genres based on no.of ratings ratings under
edx %>%
  group_by(genres) %>%
  filter(n()>=100000) %>%
  summarize(n=n(),avg=mean(rating),se=sd(rating)/sqrt(n())) %>% arrange(desc(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))
```



Next, we will compute the number of ratings for each movie and then plot it against the year the rating is given. We will also check which are the earliest and latest years in our dataset.

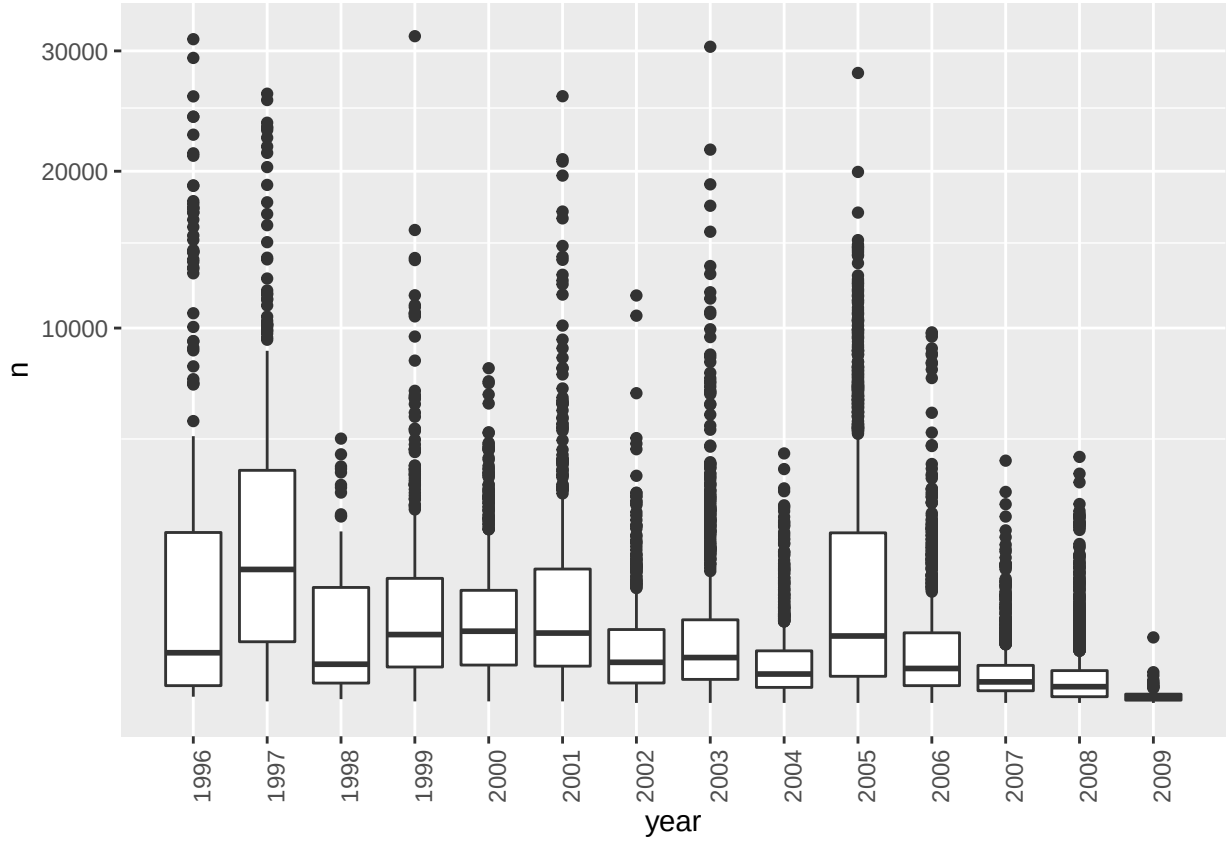
```
# Start year and end year in the dataset (of the ratings)

min(edx$year)
```

```
## [1] 1995
```

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

```
## [1] 2009
```

```
# Distribution of ratings over the years

edx %>% group_by(movieId) %>%
  summarize(n = n(), year = as.character(first(year)))%>%
  qplot(year, n, data = ., geom = "boxplot") +
  coord_trans(y = "sqrt")+
  theme(axis.text.x = element_text(angle = 90, hjust = 1))
```

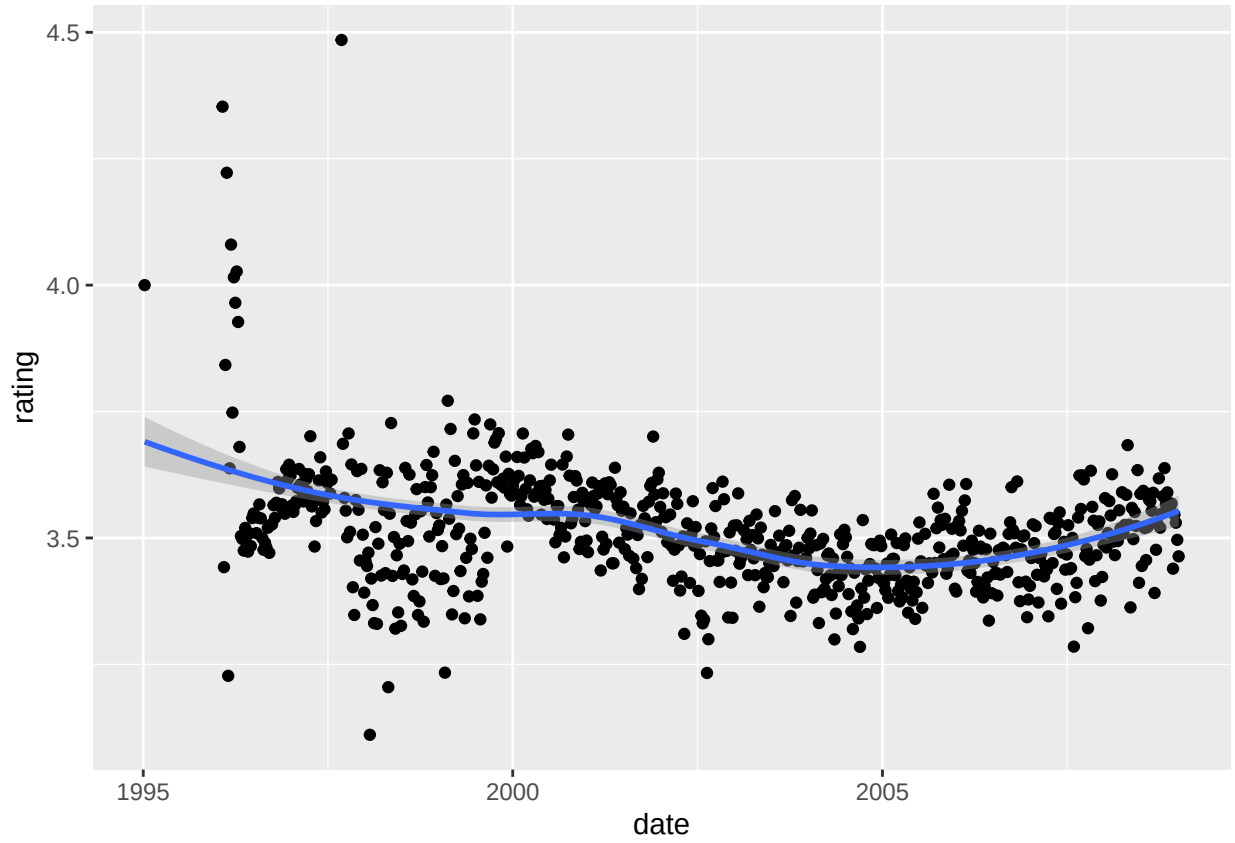


From the plot, we can see that the number of movies rated in a year is generally haphazard, excepting that there is a small, steady decrease in the number of ratings with each year from 2005. This decrease can be ascribed to the fact that the more recent a movie is, the less time users had to rate it. This indicates understandably that time of rating does not seem to have a major effect on ratings.

Let us create one more variable "date" indicating the date and time when a rating is given. Then, let us compute the average rating for each week and plot this average against the date.

```
# Distribution of no.of ratings week wise

edx <- mutate(edx, date = as_datetime(timestamp))

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



The plot above shows some effect of time over ratings but clearly not as much as the movie or the user or the genre has.

4. Methods & Analysis - Model Building & Results:

From the data exploration & analysis that we have conducted, we could clearly see that there is significant movie effect, user effect and genre effect on the ratings a movie gets while the time is not.

Hence, the approach we take to develop our model will be based on these effects, incorporating each effect one by one, in an iterative and progressive manner and calculate the RMSE at each stage to check if we have a significantly low RMSE so that we can be assured that the model produces the target RMSE when applied on "validation" dataset.

Let us start with creating the "train" dataset and "test" dataset out of the "edx" dataset, which was created in data setup section, earlier. We take the same approach that we have followed to create "edx" & "validation" datasets from "Movielens" dataset.

- Essentially we create 2 datasets - the first "train" dataset is to train the model that we attempt to create and the second dataset "test" is for testing of our model and calculating the RMSE value.
- We start with applying a data partition factor of p = 0.1 on "edx" dataset to create the required "train" & a temporary dataset.
- We make sure train dataset contains all userIds and movieIds of "test" dataset by removing from temporary dataset those movieIds and userIds which are in it but not in "train" dataset.
- The remaining of the temporary dataset will be the required "test" dataset.
- The removed records from temporary dataset will be then added back to our "train" dataset.
- The above process will create final "train" dataset version with approximately 90% data and "test" dataset with approximately 10% of data.

```
# Create test dataset and train dataset
set.seed(1, sample.kind="Rounding")
```

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

```
test_ind <- createDataPartition(y=edx$rating,times=1, p=.1, list=FALSE)
train_ds <- edx[-test_ind,]
test_ds_temp <- edx[test_ind,]

# Make sure userId and movieId in test dataset are also in train dataset
test_ds <- test_ds_temp %>%
  semi_join(train_ds, by = "movieId") %>%
  semi_join(train_ds, by = "userId")

# Add rows removed from test_ds_temp dataset back into train dataset
rmvd <- anti_join(test_ds_temp, test_ds)
```

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

```
train_ds <- rbind(train_ds, rmvd)
```

Now, we will create a formula for calculating RMSE for the predicted values generated using our model (based on the effects described earlier) as we move through the process of building and improving our model to achieve the target RMSE.

```
# Formula for RMSE
RMSE <- function(testval_ratings, pred_ratings){
  sqrt(mean((testval_ratings - pred_ratings)^2))
}
```

As mentioned earlier, predicting "rating" using other (predictor) variables is the problem at hand.

Let us start with our first model, which is a simple, straight forward model. Here, we assume that the average of all ratings in the "train" dataset will be the predicted value for a rating. This means we predict the average value as the rating for all "test" dataset items. With this criteria, let us check what RMSE value we get.

```
# Model.1: Computing predicted ratings based on the average of population

mu_simple <- mean(train_ds$rating)
mu_simple
```

```
## [1] 3.512456
```

```
mod1_rmse <- RMSE(test_ds$rating, mu_simple)

mod1_rmse
```

```
## [1] 1.060054
```

Let us also build a table to keep a note of all our RMSEs for different models as we go on improving and create our final model.

```
# RMSE table for different models
rmse_table <- tibble(Model = "Model.1", Method="Simple Population Mean Model",
                                     RMSE = mod1_rmse)
rmse_table %>% knitr::kable()
```

Model	Method	RMSE
Model.1	Simple Population Mean Model	1.060054

The population average of 3.51 as predicted rating is resulting in an RMSE of 1.06, which is not very good. This indicates that our predictions are at least 1 rating away from actual values.

We simply know that rating for movie to movie varies just because of movie, other things remaining the same. This is movie effect.

Similarly after accounting for movie effect, ratings for a movie still vary just because of the user, other things remaining the same. We can call this as user effect.

Same is the case with genre effect, which will reduce some more variation after accounting for movie and user effects.

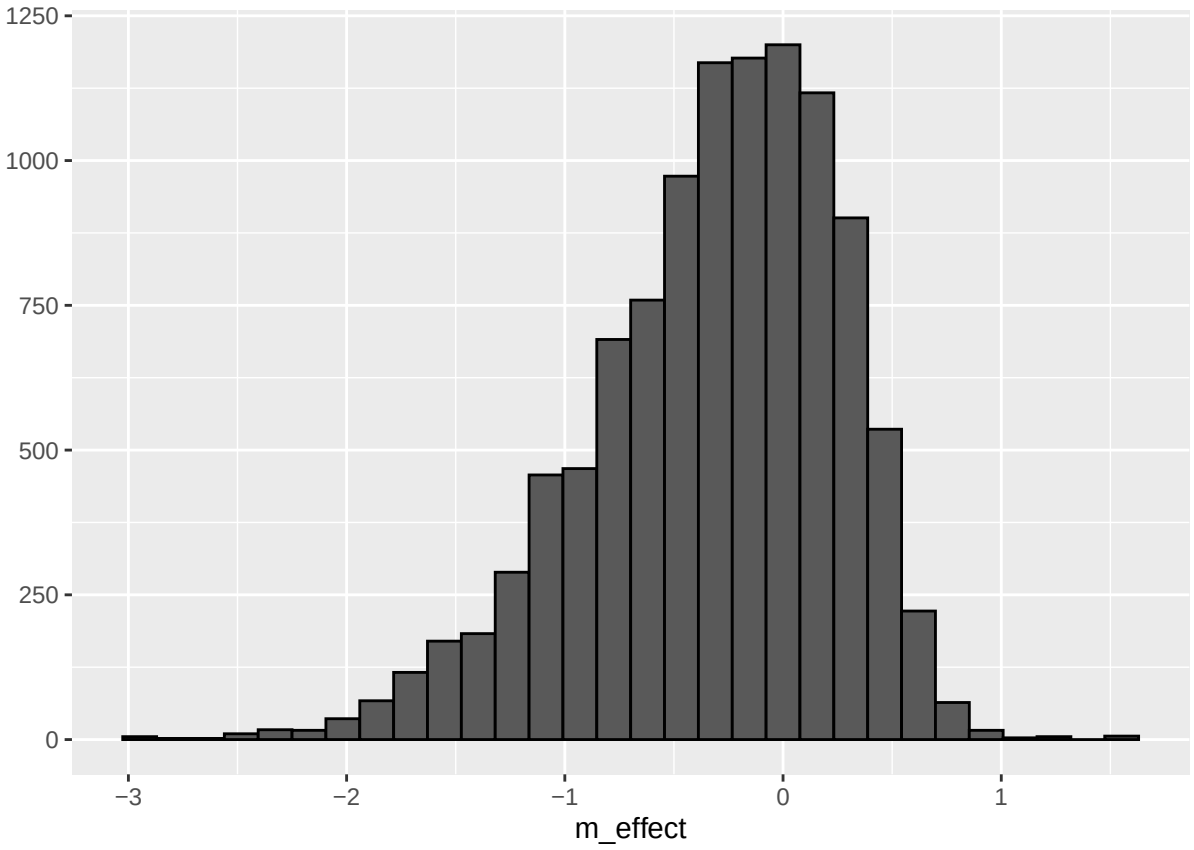
If the population mean is not the exact measure of actual values, then the gap can be ascribed to the above defined effects or some other parameters such as time effect, individual actor effect, movie director effect, movie budget effect, so on.

Since our dataset is limited to a few variables - users, movies, genres and time, we will make use of them to improve our model. As discussed, since time does not seem to have significant effect on ratings, we shall use only the user, movie and genre effects to build our model.

So, let us improve our first model by adding the movie effect. Adding the movie effect to the average primarily reduces that gap between the predicted values and actual ratings.

The below code apart from calculating the movie effects and plotting them, will also calculate theRMSE for our movie effect model.

```
# Model.2: Computing predicted ratings based on movie effects

mu <- mean(train_ds$rating)

# Add movie effect
movie_effects <- train_ds %>%
  group_by(movieId) %>%
  summarize(m_effect = mean(rating - mu))

movie_effects %>% qplot(m_effect, geom ="histogram", bins = 30, data = ., color = I("black"))
```



As we can see, the movie effects vary significantly. Naturally popular movies and unpopular movies will have significantly different effects.

```
# Predict ratings
pred_ratings <- mu + test_ds %>%
  left_join(movie_effects, by='movieId') %>%
```

```
  .$m_effect

mod2_rmse <- RMSE(test_ds$rating, pred_ratings)

rmse_table <- bind_rows(rmse_table,
                        tibble(Model = "Model.2", Method="Movie Effect Model",
                                   RMSE = mod2_rmse ))

mod2_rmse
```

```
## [1] 0.9429615
```

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

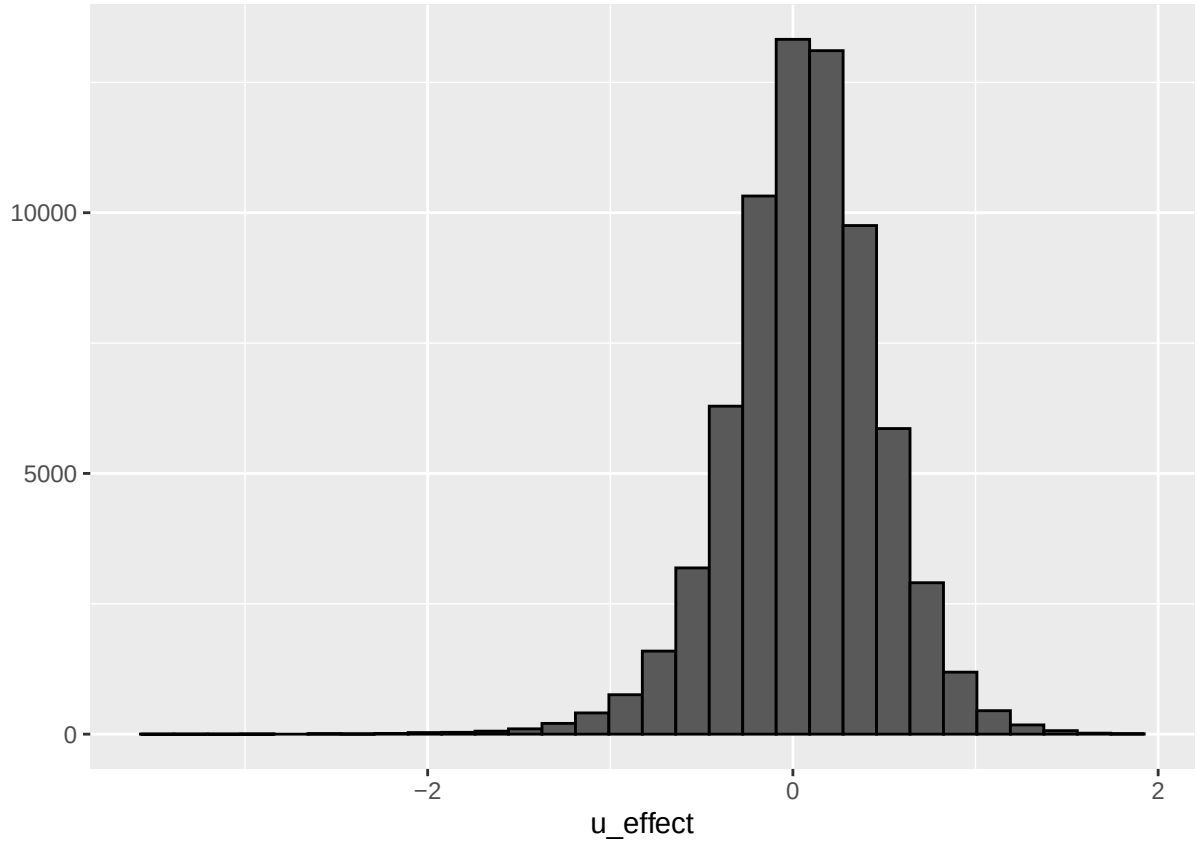
Model	Method	RMSE
Model.1	Simple Population Mean Model	1.0600537
Model.2	Movie Effect Model	0.9429615

We see our RMSE value has significantly improved from its initial 1.06 by adding movie effect. However, it is still far from our target RMSE.

We shall note at this juncture that "validation" set is significantly smaller than "train" dataset created out of "edx" dataset. Hence, in order to safely have the final RMSE with the "validation" dataset close enough to or better than the target RMSE, we may have to achieve an RMSE with "test" dataset smaller than target RMSE.

Let us further improve our model by adding user effects to the above model where we have incorporated movie effects. As earlier, we will calculate and plot the user effects too.

```
# Model.3: Computing predicted ratings based on movie effects and user effects

# Add user effect
user_effects <- train_ds %>%
  left_join(movie_effects, by='movieId') %>%
  group_by(userId) %>%
  summarize(u_effect = mean(rating - mu - m_effect))

user_effects %>% qplot(u_effect, geom ="histogram", bins = 30, data = ., color = I("black"))
```



We can see from the plot the user effects also significantly vary from user to user, which is a fact. Different users have different tastes and preferences. Some users might like every movie including bad movies and some may not like any movie including good movies.

```
# Predict ratings
pred_ratings <- test_ds %>%
  left_join(movie_effects, by='movieId') %>%
  left_join(user_effects, by='userId') %>%
  mutate(pred = mu + m_effect + u_effect) %>%
  .$pred

mod3_rmse <- RMSE(test_ds$rating, pred_ratings)
rmse_table <- bind_rows(rmse_table,
                        tibble(Model = "Model.3", Method="Movie + User Effect Model",
                                   RMSE = mod3_rmse ))


mod3_rmse
```

```
## [1] 0.8646843
```

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

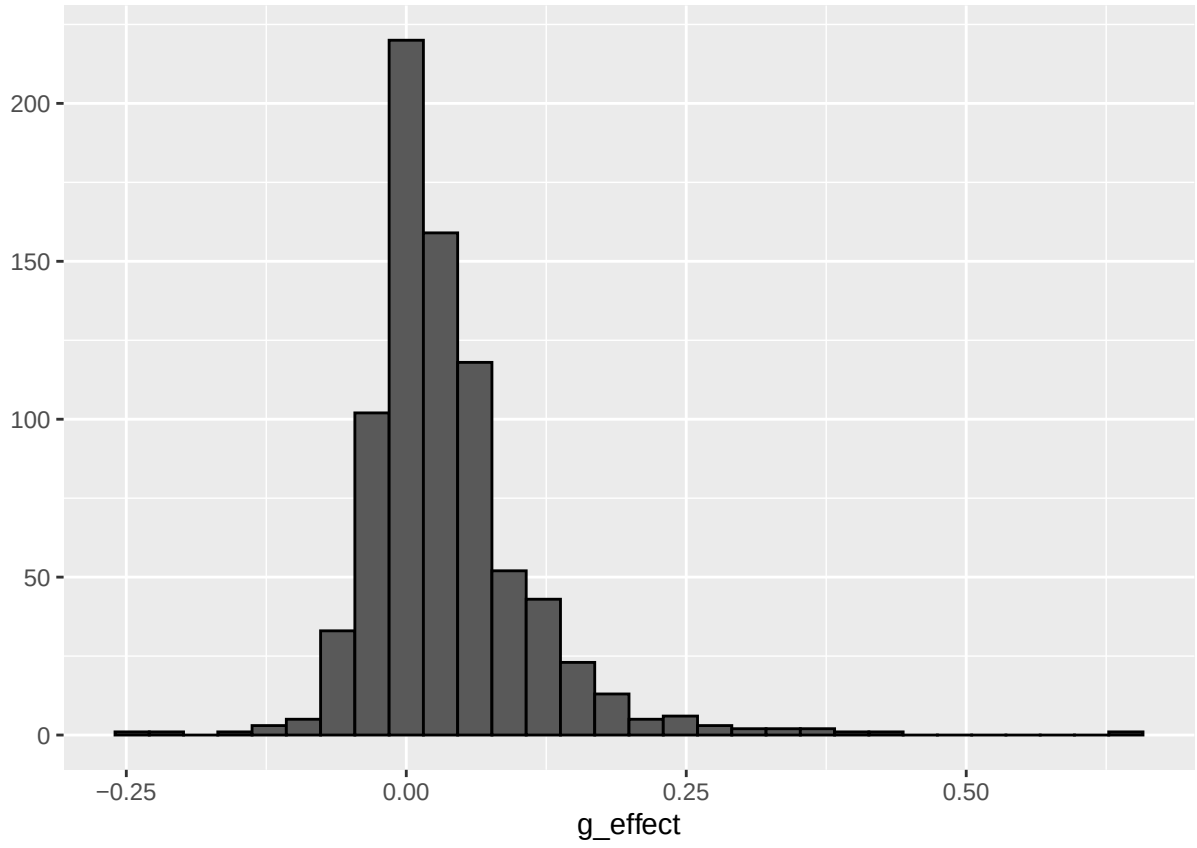
Model	Method	RMSE
Model.1	Simple Population Mean Model	1.0600537

Model	Method	RMSE
Model.2	Movie Effect Model	0.9429615
Model.3	Movie + User Effect Model	0.8646843

As we can see in the above model, wherein the movie and user effects have been taken into account, has reduced the variation some more. Clearly, our RMSE has come down further and indeed below the target RMSE. However, as we will see later, this level of low RMSE created out of "train" & "test" datasets is not sufficient to get lower RMSE than the target when we apply the model to the "validation" dataset, for the reasons explained earlier.

We can further improve our model by adding genre effects to the previous model, where we have incorporated move and user effects.

```
# Model.4: Computing predicted ratings based on movie, user & genre effects

# Add genre effect
genre_effects <- train_ds %>%
  left_join(movie_effects, by='movieId') %>%
  left_join(user_effects, by='userId') %>%
  group_by(genres) %>%
  summarize(g_effect = mean(rating - mu - m_effect - u_effect))

genre_effects %>% qplot(g_effect, geom ="histogram", bins = 30, data = ., color = I("black"))
```



We can see from the plot, the genre effect also varies from genre to genre, which is also a fact. It is common

sense that people, who like a particular genre of movies, watch more movies in that category and generally rate those movies higher than the movies in other genres.

```
# Predict ratings
pred_ratings <- test_ds %>%
  left_join(movie_effects, by='movieId') %>%
  left_join(user_effects, by='userId') %>%
  left_join(genre_effects, by='genres') %>%
  mutate(pred = mu + m_effect + u_effect + g_effect) %>%
  .$pred

mod4_rmse <- RMSE(test_ds$rating, pred_ratings)
rmse_table <- bind_rows(rmse_table,
                        tibble(Model = "Model.4", Method="Movie + User + Genre Effect Model",
                                   RMSE = mod4_rmse ))


mod4_rmse
```

```
## [1] 0.8643241
```

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

Model	Method	RMSE
Model.1	Simple Population Mean Model	1.0600537
Model.2	Movie Effect Model	0.9429615
Model.3	Movie + User Effect Model	0.8646843
Model.4	Movie + User + Genre Effect Model	0.8643241

As we are expecting, we were able to successfully further reduce the RMSE and improve our Model. Now, we have RMSE values for both Model.3 and Model.4 lower than the target RMSE. Let us see if our Model.4, which has the lowest of RMSEs produces a better result, a RMSE of 0.8649 or below, when applied to "validation" dataset too.

```
# Predict ratings on Validation Dataset for testing purpose
val_test_ratings <- validation %>%
  left_join(movie_effects, by='movieId') %>%
  left_join(user_effects, by='userId') %>%
  left_join(genre_effects, by='genres') %>%
  mutate(pred = mu + m_effect + u_effect + g_effect) %>%
  .$pred

val_test_rmse <- RMSE(validation$rating, val_test_ratings)

val_test_rmse
```

```
## [1] 0.865449
```

Unfortunately, for reasons as explained earlier, the RMSE resulted by applying Model.4 on the "validation" dataset, is above the target RMSE of 0.8649.

So, how can we further improve our movie + user + genre effect model (Model.4)? For that purpose, let us first take a step back and do a bit of analysis on our movie effect model (Model.2)

We start with exploring where we have big gaps in our predictions in our 2nd model, where we have used only movie effects.

```
# Retrospective analysis - 20 largest prediction gaps - Model.2:
test_ds %>%
  left_join(movie_effects, by='movieId') %>%
  mutate(error = rating - (mu + m_effect)) %>%
  arrange(desc(abs(error))) %>%
  select(title,  error) %>% slice(1:20)
```

```
##                              title     error
## 1       From Justin to Kelly (2003)  4.125683
## 2  Shawshank Redemption, The (1994) -3.956567
## 3  Shawshank Redemption, The (1994) -3.956567
## 4             Godfather, The (1972) -3.916651
## 5             Godfather, The (1972) -3.916651
## 6             Godfather, The (1972) -3.916651
## 7             Godfather, The (1972) -3.916651
## 8        Usual Suspects, The (1995) -3.866552
## 9           Schindler's List (1993) -3.864085
## 10          Schindler's List (1993) -3.864085
## 11          Schindler's List (1993) -3.864085
## 12          Schindler's List (1993) -3.864085
## 13          Schindler's List (1993) -3.864085
## 14          Schindler's List (1993) -3.864085
## 15  Barney's Great Adventure (1998)  3.860215
## 16               Rear Window (1954) -3.824947
## 17               Rear Window (1954) -3.824947
## 18                Casablanca (1942) -3.819643
## 19                Casablanca (1942) -3.819643
## 20                Casablanca (1942) -3.819643
```

These are primarily well known movies, but have large errors (difference between predicted and actual ratings). As these are well known movies, we can assume that these movies should have been rated by many users. Given this, our predictions should not vary much from actual values if the model is effective, which is not the case here.

Let us look at the top 20 best and 20 worst movies based on movie effects alone (Model.2), for which, first, let's create a database that connects movieId to movie title:

```
# create a database that connects movieId to movie title
mtitles <- edx %>%
  select(movieId, title) %>%
  distinct()
```

Given the fact that the larger the movie effect above the average rating, the better the movie is. Accordingly, below are the top 20 movies as predicted by our Model.2

```
# Best 20 movies as predicted by our Model.2
movie_effects %>% left_join(mtitles, by="movieId") %>%
  arrange(desc(m_effect)) %>%
```

```
  select(title, m_effect) %>%
  slice(1:20)
```

```
## # A tibble: 20 x 2
##    title                                                        m_effect
##    <chr>                                                           <dbl>
##  1 Hellhounds on My Trail (1999)                                    1.49
##  2 Satan's Tango (SÃ¡tÃ¡ntangÃ³) (1994)                              1.49
##  3 Shadows of Forgotten Ancestors (1964)                            1.49
##  4 Fighting Elegy (Kenka erejii) (1966)                             1.49
##  5 Sun Alley (Sonnenallee) (1999)                                   1.49
##  6 Blue Light, The (Das Blaue Licht) (1932)                         1.49
##  7 Who's Singin' Over There? (a.k.a. Who Sings Over There) (Ko to~    1.24
##  8 Life of Oharu, The (Saikaku ichidai onna) (1952)                 1.24
##  9 Human Condition II, The (Ningen no joken II) (1959)              1.24
## 10 Human Condition III, The (Ningen no joken III) (1961)            1.24
## 11 Constantine's Sword (2007)                                       1.24
## 12 More (1998)                                                      1.15
## 13 I'm Starting From Three (Ricomincio da Tre) (1981)               1.15
## 14 Class, The (Entre les Murs) (2008)                               1.15
## 15 Mickey (2003)                                                    0.988
## 16 Demon Lover Diary (1980)                                         0.988
## 17 Valerie and Her Week of Wonders (Valerie a tÃ½den divu) (1970)    0.988
## 18 Testament of Orpheus, The (Testament d'OrphÃ©e) (1960)           0.988
## 19 Power of Nightmares: The Rise of the Politics of Fear, The (20~   0.988
## 20 Kansas City Confidential (1952)                                  0.988
```

Let us see how often these movies are rated

```
train_ds %>% dplyr::count(movieId) %>%
  left_join(movie_effects) %>%
  left_join(mtitles, by="movieId") %>%
  arrange(desc(m_effect)) %>%
  select(title, m_effect, n) %>%
  slice(1:20)
```

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

```
## # A tibble: 20 x 3
##    title                                                  m_effect     n
##    <chr>                                                     <dbl> <int>
##  1 Hellhounds on My Trail (1999)                              1.49      1
##  2 Satan's Tango (SÃ¡tÃ¡ntangÃ³) (1994)                        1.49      1
##  3 Shadows of Forgotten Ancestors (1964)                      1.49      1
##  4 Fighting Elegy (Kenka erejii) (1966)                       1.49      1
##  5 Sun Alley (Sonnenallee) (1999)                             1.49      1
##  6 Blue Light, The (Das Blaue Licht) (1932)                   1.49      1
##  7 Who's Singin' Over There? (a.k.a. Who Sings Over There) ~    1.24      4
##  8 Life of Oharu, The (Saikaku ichidai onna) (1952)           1.24      2
##  9 Human Condition II, The (Ningen no joken II) (1959)        1.24      4
## 10 Human Condition III, The (Ningen no joken III) (1961)      1.24      4
## 11 Constantine's Sword (2007)                                 1.24      2
```

```
## 12 More (1998)                                                    1.15      6
## 13 I'm Starting From Three (Ricomincio da Tre) (1981)             1.15      3
## 14 Class, The (Entre les Murs) (2008)                             1.15      3
## 15 Mickey (2003)                                                  0.988     1
## 16 Demon Lover Diary (1980)                                       0.988     1
## 17 Valerie and Her Week of Wonders (Valerie a tÃ½den divu) ~       0.988     1
## 18 Testament of Orpheus, The (Testament d'OrphÃ©e) (1960)         0.988     1
## 19 Power of Nightmares: The Rise of the Politics of Fear, T~      0.988     4
## 20 Kansas City Confidential (1952)                                0.988     1
```

As we can see in the above table, these are all not so famous movies and are supposed to be not the best movies. The reason is that because they are rated very rarely, very few times, they actually create a big error in our model (gap between predicted and actual values).

Similarly, below are the 20 worst movies as predicted by our model.2, which are once again obscure movies and are rated a very few times.

```
# Worst 20 movies as predicted by our model.2
train_ds %>% dplyr::count(movieId) %>%
  left_join(movie_effects) %>%
  left_join(mtitles, by="movieId") %>%
  arrange(m_effect) %>%
  select(title, m_effect, n) %>%
  slice(1:20)
```

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

```
## # A tibble: 20 x 3
##    title                                                  m_effect     n
##    <chr>                                                     <dbl> <int>
##  1 Besotted (2001)                                           -3.01     1
##  2 Hi-Line, The (1999)                                       -3.01     1
##  3 Accused (Anklaget) (2005)                                 -3.01     1
##  4 Confessions of a Superhero (2007)                         -3.01     1
##  5 War of the Worlds 2: The Next Wave (2008)                 -3.01     2
##  6 SuperBabies: Baby Geniuses 2 (2004)                       -2.77    47
##  7 Disaster Movie (2008)                                     -2.75    30
##  8 From Justin to Kelly (2003)                               -2.64   183
##  9 Hip Hop Witch, Da (2000)                                  -2.60    11
## 10 Criminals (1996)                                          -2.51     1
## 11 Mountain Eagle, The (1926)                                -2.51     2
## 12 Stacy's Knights (1982)                                    -2.51     1
## 13 Dog Run (1996)                                            -2.51     1
## 14 Monkey's Tale, A (Les ChÃ¢teau des singes) (1999)         -2.51     1
## 15 When Time Ran Out... (a.k.a. The Day the World Ended) (1~  -2.51     1
## 16 Dischord (2001)                                           -2.51     1
## 17 Roller Boogie (1979)                                      -2.51    13
## 18 Relative Strangers (2006)                                 -2.51     1
## 19 PokÃ©mon Heroes (2003)                                    -2.47   124
## 20 Carnosaur 3: Primal Species (1996)                        -2.40    61
```

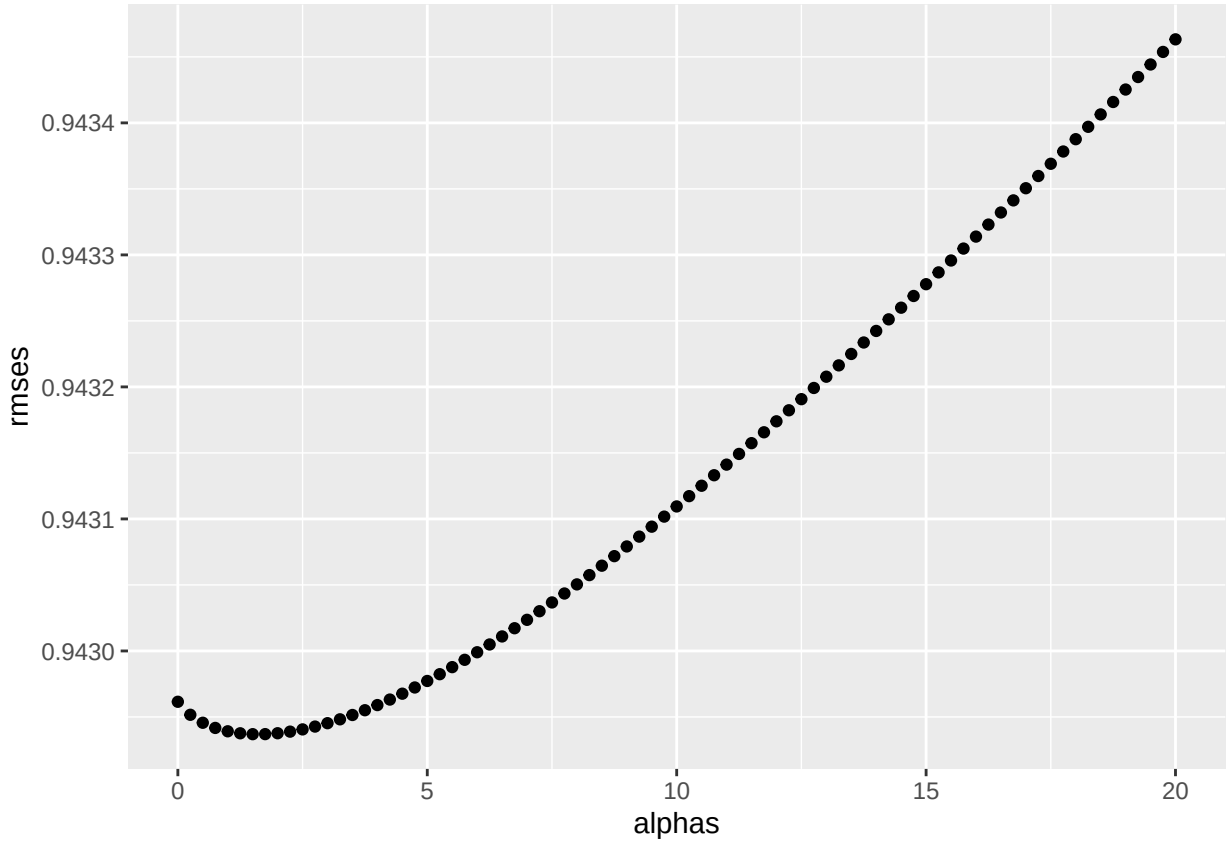
The supposed "best" and "worst" movies were rated by very few users, in most cases, just once. These movies are mostly unknown movies. With just a few users, the estimates are not precise and have more

uncertainty. Therefore, larger movie effects, negative or positive, are more likely. Large effects can increase our RMSE, so we should use the below described technique to regularize the large effects.

The technique is simple. We add a value, alpha, to the number of ratings, which if the number of ratings is high, will not have any significant effect, but if the number of ratings is less, will bring down the effect towards the average.

But, how do we know the alpha value! Is it 1 or 2 or 5 or 20? How do we know? For that purpose, we use the below code, to calculate multiple RMSEs for different alpha values and to pick the optimum alpha that produces the minimum RMSE.

```
# Finding optimal alpha - penalty term to reduce the effect of few ratings
alphas <- seq(0, 20, 0.25)
mu <- mean(train_ds$rating)
totals <- train_ds %>%
  group_by(movieId) %>%
  summarize(s = sum(rating - mu), n_i = n())
rmses <- sapply(alphas, function(a){
  pred_ratings <- test_ds %>%
    left_join(totals, by='movieId') %>%
    mutate(m_effect = s/(n_i+a)) %>%
    mutate(pred = mu + m_effect) %>%
    .$pred
  return(RMSE(test_ds$rating, pred_ratings))
})

qplot(alphas, rmses)
```

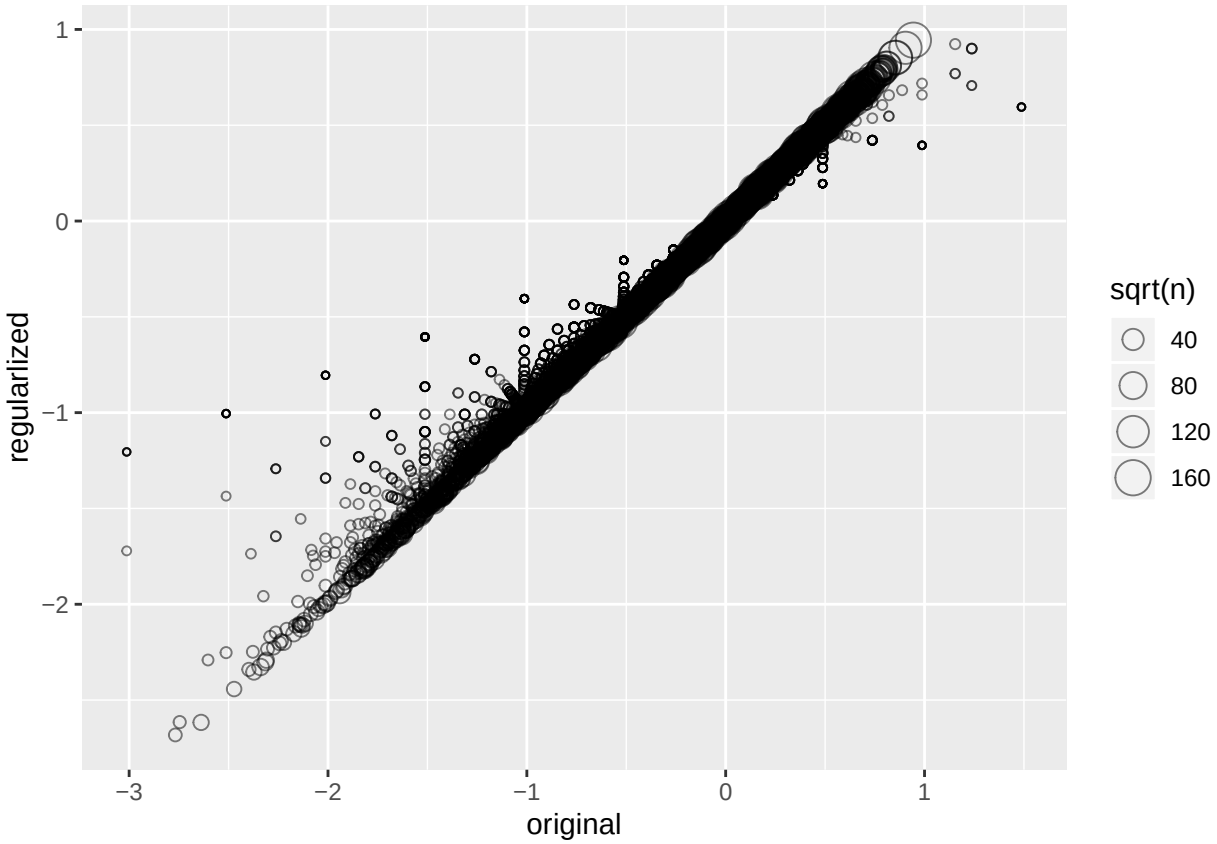


```
opt_alpha <- alphas[which.min(rmses)]

opt_alpha
```

```
## [1] 1.5
```

Now that we know the optimum alpha, let us recalculate the regularized movie effects and plot them against the initial movie effects.

```
# recalculate the regularized movie effects and plot against initial movie effects.
alpha <- opt_alpha
mu <- mean(train_ds$rating)
movie_reg_effects <- train_ds %>%
  group_by(movieId) %>%
  summarize(m_effect = sum(rating - mu)/(n()+alpha), n_i = n())

tibble(original = movie_effects$m_effect,
           regularlized = movie_reg_effects$m_effect,
           n = movie_reg_effects$n_i) %>%
  ggplot(aes(original, regularlized, size=sqrt(n))) +
  geom_point(shape=1, alpha=0.5)
```



As we can see from the above plot, the original extreme movie effects which were rated by a few users (indicated by the small circles in the plot) have shrunk towards the middle, thus reducing the errors and improving RMSEs.

Let us also recheck how our best movies and worst movies as they would have changed with the regularized effects.

```
# Best 20 movies predicted after regularisation
train_ds %>%
  dplyr::count(movieId) %>%
  left_join(movie_reg_effects) %>%
  left_join(mtitles, by="movieId") %>%
  arrange(desc(m_effect)) %>%
  select(title, m_effect, n) %>%
  slice(1:20)
```

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

```
## # A tibble: 20 x 3
##    title                                                m_effect     n
##    <chr>                                                   <dbl> <int>
##  1 Shawshank Redemption, The (1994)                        0.944 25188
##  2 More (1998)                                             0.923     6
##  3 Godfather, The (1972)                                   0.904 15975
##  4 Who's Singin' Over There? (a.k.a. Who Sings Over There) ~    0.900     4
##  5 Human Condition II, The (Ningen no joken II) (1959)     0.900     4
##  6 Human Condition III, The (Ningen no joken III) (1961)   0.900     4
##  7 Usual Suspects, The (1995)                              0.854 19457
##  8 Schindler's List (1993)                                 0.852 20877
##  9 Rear Window (1954)                                      0.812  7115
## 10 Casablanca (1942)                                       0.807 10141
## 11 Sunset Blvd. (a.k.a. Sunset Boulevard) (1950)           0.803  2633
## 12 Double Indemnity (1944)                                 0.794  1950
## 13 Seven Samurai (Shichinin no samurai) (1954)             0.793  4648
## 14 Paths of Glory (1957)                                   0.793  1410
## 15 Godfather: Part II, The (1974)                          0.790 10743
## 16 Third Man, The (1949)                                   0.787  2681
## 17 Dark Knight, The (2008)                                 0.787  2095
## 18 Lives of Others, The (Das Leben der Anderen) (2006)     0.785  1000
## 19 One Flew Over the Cuckoo's Nest (1975)                  0.783 11644
## 20 Dr. Strangelove or: How I Learned to Stop Worrying and L~    0.783  9623
```

```
# Worst 20 movies predicted after regularisation
train_ds %>%
dplyr::count(movieId) %>%
  left_join(movie_reg_effects) %>%
  left_join(mtitles, by="movieId") %>%
  arrange(m_effect) %>%
  select(title, m_effect, n) %>%
  slice(1:20)
```

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

```
## # A tibble: 20 x 3
##    title                                                m_effect     n
##    <chr>                                                   <dbl> <int>
##  1 SuperBabies: Baby Geniuses 2 (2004)                     -2.68    47
```

```
##  2 From Justin to Kelly (2003)                                    -2.62   183
##  3 Disaster Movie (2008)                                          -2.62    30
##  4 PokÃ©mon Heroes (2003)                                         -2.44   124
##  5 Barney's Great Adventure (1998)                                -2.35   186
##  6 Carnosaur 3: Primal Species (1996)                             -2.34    61
##  7 Glitter (2001)                                                 -2.33   311
##  8 Gigli (2003)                                                   -2.30   281
##  9 Pokemon 4 Ever (a.k.a. PokÃ©mon 4: The Movie) (2002)           -2.29   188
## 10 Hip Hop Witch, Da (2000)                                       -2.29    11
## 11 Roller Boogie (1979)                                           -2.25    13
## 12 Slaughterhouse 2 (1988)                                        -2.25    26
## 13 Faces of Death: Fact or Fiction? (1999)                        -2.23    48
## 14 Faces of Death 6 (1996)                                        -2.23    73
## 15 Carnosaur 2 (1995)                                             -2.20    82
## 16 Son of the Mask (2005)                                         -2.20   145
## 17 Yu-Gi-Oh! (2004)                                               -2.19    74
## 18 Horrors of Spider Island (Ein Toter Hing im Netz) (1960)       -2.17    27
## 19 House of the Dead, The (2003)                                  -2.15   182
## 20 Puma Man, The (a.k.a. The Pumaman) (L'Uomo Puma) (1980)         -2.15    28
```

The best and worst movies tables produced now make better sense than it was earlier.

We have seen that when movies are rated by few users it will increase movie effects (positive or negative), thus the errors in the predicted values. In order to reduce these effects, we penalize the low number of ratings for a movie by adding regularization (penalty) term alpha.

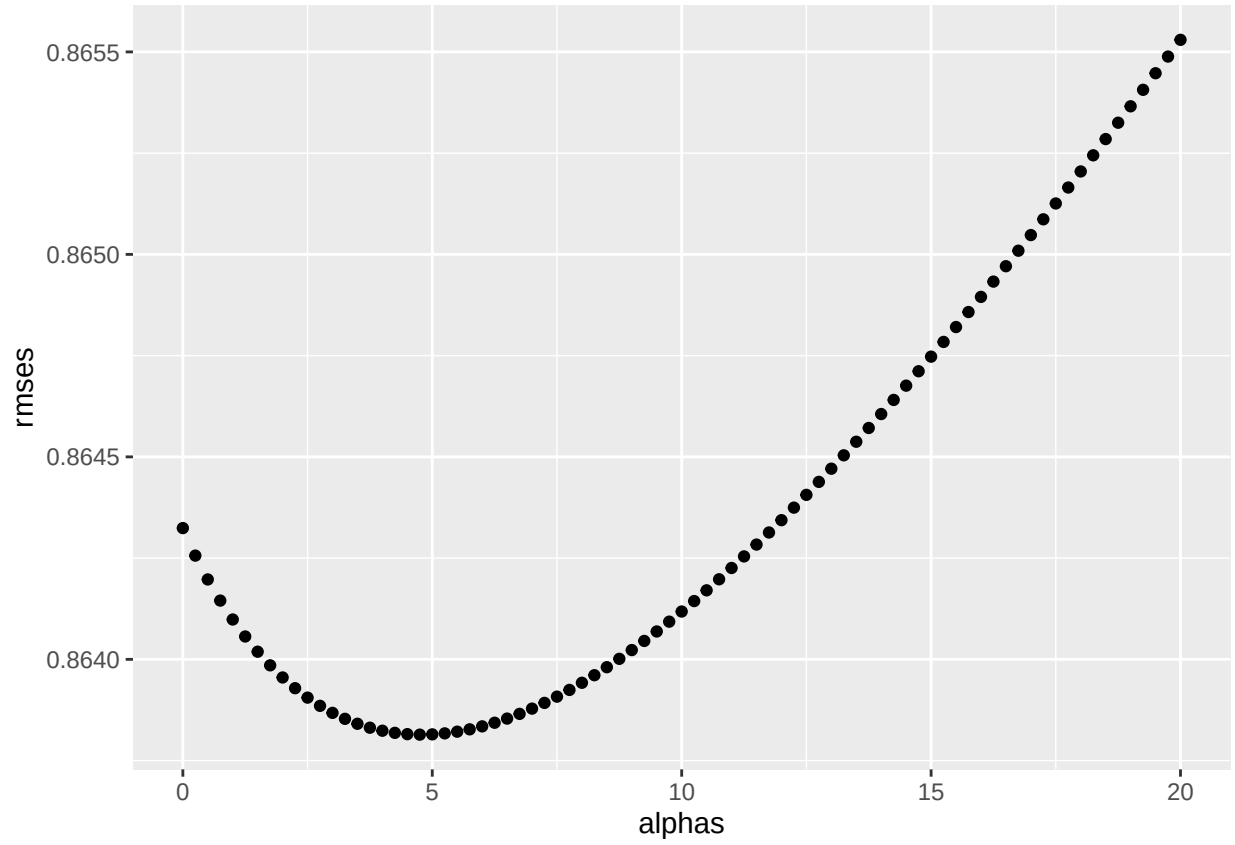
Similarly, when a user rates a few times only, it will increase user effects (positive or negative) also, thus the errors in the predicted values. Same is the case with genres, which have a few ratings covered under them.

Hence, we can apply regularization (adding the penalty term 'alpha') technique explained above to the user effects and genre effects also.

Instead of doing it for movie effects, user effects & genre effects separately, let us build our next model (Model.5), by regularization, incorporating all effects together in a single step. This way of regularizing all the effects will help in reducing the uncertainty and errors in our predictions and improve our model (lower RMSE).

```
# Model.5: Computing predicted ratings based on regularised movie, user & genre effects

alphas <- seq(0, 20, 0.25)

rmses <- sapply(alphas, function(a){
  mu <- mean(train_ds$rating)

  # Add movie effect with regularisation
  m_effect <- train_ds %>%
    group_by(movieId) %>%
    summarize(m_effect = sum(rating - mu)/(n()+a))

  # Add user effect with regularisation
  u_effect <- train_ds %>%
    left_join(m_effect, by="movieId") %>%
    group_by(userId) %>%
    summarize(u_effect = sum(rating - m_effect - mu)/(n()+a))

  # Add genre effect with regularisation
```

```
  g_effect <- train_ds %>%
    left_join(m_effect, by="movieId") %>%
    left_join(u_effect, by="userId") %>%
    group_by(genres) %>%
    summarize(g_effect = sum(rating - m_effect - u_effect - mu)/(n()+a))

  # Predict ratings
  pred_ratings <-
    test_ds %>%
    left_join(m_effect, by = "movieId") %>%
    left_join(u_effect, by = "userId") %>%
    left_join(g_effect, by = "genres") %>%
    mutate(pred = mu + m_effect + u_effect + g_effect) %>%
    .$pred
  return(RMSE(test_ds$rating, pred_ratings))
})

qplot(alphas, rmses)
```




```
alphas[which.min(rmses)]
```

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

```
mod5_rmse <- min(rmses)

rmse_table <- bind_rows(rmse_table,
                        tibble(Model = "Model.5",
                               Method="Regularized (Movie + User + Genre) Effect Model",
                                   RMSE = mod5_rmse ))


mod5_rmse
```

```
## [1] 0.8638141
```

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

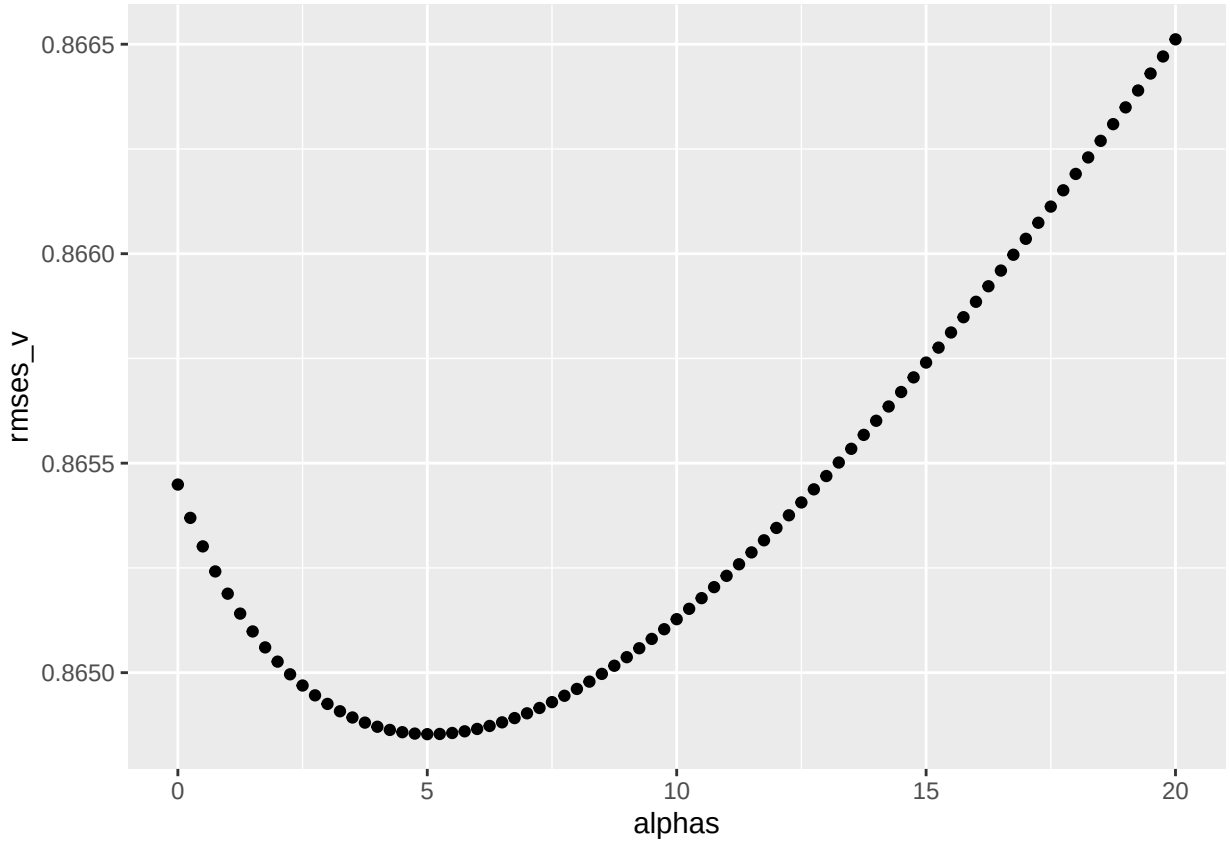
Model	Method	RMSE
Model.1	Simple Population Mean Model	1.0600537
Model.2	Movie Effect Model	0.9429615
Model.3	Movie + User Effect Model	0.8646843
Model.4	Movie + User + Genre Effect Model	0.8643241
Model.5	Regularized (Movie + User + Genre) Effect Model	0.8638141

As we can see our Model.5, which is a based on population average and regularized movie, user and genre effects has substantially brought down the RMSE value from Model.4.

Now, it is time to cross validate if our final model (Model.5) indeed produces a RMSE value less than 0.8649 for the validation dataset too.

5. Methods & Analysis - Final Model Testing on Validation Dataset:

```
# Testing the final model on the Validation dataset & Computing final RMSE
#(Regularised Movie + User + Genre Effect model)

alphas <- seq(0, 20, 0.25)
rmses_v <- sapply(alphas, function(a){
  mu <- mean(train_ds$rating)

  # Add regularised movie effect
  m_effect <- train_ds %>%
    group_by(movieId) %>%
    summarize(m_effect = sum(rating - mu)/(n()+a))

  # Add regularised user effect
  u_effect <- train_ds %>%
    left_join(m_effect, by="movieId") %>%
    group_by(userId) %>%
    summarize(u_effect = sum(rating - m_effect - mu)/(n()+a))

  # Add regularised genre effect
  g_effect <- train_ds %>%
    left_join(m_effect, by="movieId") %>%
    left_join(u_effect, by="userId") %>%
    group_by(genres) %>%
    summarize(g_effect = sum(rating - m_effect - u_effect - mu)/(n()+a))

  # Predict ratings for validation dataset
  pred_ratings <-
    validation %>%
    left_join(m_effect, by = "movieId") %>%
    left_join(u_effect, by = "userId") %>%
    left_join(g_effect, by = "genres") %>%
    mutate(pred = mu + m_effect + u_effect + g_effect) %>%
    .$pred
  return(RMSE(validation$rating, pred_ratings))
})

qplot(alphas, rmses_v)
```




```
alphas[which.min(rmses_v)]
```

```
## [1] 5
```

```
val_rmse <- min(rmses_v)


val_rmse
```

```
## [1] 0.8648532
```

As we can see, with our Model.5, the RMSE value has been clearly brought down below the target RMSE. **Eureka...**

The RMSE value that our final model achieved on "validation" dataset is 0.8648532

6. Results

We have analysed and visualized the edx dataset to better understand its structure, composition and content.

We have understood that each of movies, users and genres have significant effect on ratings, while the time of rating is not.

Hence, we decided that we will incorporate these effects in building our model in a step-by-step manner.

Below is the summary table of all models and the results that they have produced for us.

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

Model	Method	RMSE
Model.1	Simple Population Mean Model	1.0600537
Model.2	Movie Effect Model	0.9429615
Model.3	Movie + User Effect Model	0.8646843
Model.4	Movie + User + Genre Effect Model	0.8643241
Model.5	Regularized (Movie + User + Genre) Effect Model	0.8638141
The RMSE	value that Model.5 had achieved on “validation” d	ataset is 0.8648532**

We have started with our first model which basically assumes the mean of the ratings in the dataset (train dataset) as the ratings for the test dataset. Though this is very simple approach, it does not produce accurate predictions. As we have seen our RMSE resulted was 1.06, which indicates our predictions are 1 rating away from actual ratings on an average.

Then, in our second model, we have added the movie effect to the mean in order to reduce the gap between the actual ratings and our predictions. This indeed brought down the RMSE, thus indicating we have improved upon our predictions.

In our 3rd model, we have added user effects to the mean and movie effects, thus further improving our model. We can see from the table the RMSE has come down below the target RMSE of 0.8649.

In our attempt to further improve the model, in the next step, we have added genre effects too to our model. This further improved our predictions as indicated by the lowered RMSE value.

As we know, as the models are built on a significantly larger dataset (“train” dataset), we may not get the same low RMSE when our model is applied to the smaller “validation” dataset. We have confirmed this fact, when we have applied our Model.4 on “validation” dataset. Model.4 had produced a lower RMSE of 0.8643241 on “test dataset and a higher RMSE of 0.865499 on”validation" dataset, compared to target RMSE of 0.8649.

Then we checked on where we are making mistakes using Model.2 (for simplicity) and found out that few ratings for certain movies have resulted in large errors in our predictions (meaning higher RMSEs).

Hence, we applied a penalty term (regularization) on ratings on the movie effects so that we shrunk the large effects/ errors towards the mean. It worked.

In our final model, we have applied this regularization technique (adding penalty) to all the three - movie, user and genre effects. This has brought down the RMSE significantly, indicating our predictions are more accurate.

We have reconfirmed the above fact by applying our final model, with regularization on movie, user and genre effects on validation dataset. Indeed we have achieved a lower RMSE of 0.8648532, smaller than the target RMSE of 0.8649.

6. Conclusion & RMSE Achieved

Having successfully achieved the required RMSE on the validation dataset, we can conclude a few points.

- First in order to build prediction models, we need to first explore, understand and analyse the data. We have done this extensively from different perspectives.
- Building models, testing them for the accuracy of predicted values is an iterative process, based on our understanding of the data, problem at hand and intuition.

  In fact, all the stages of data science project are iterative and include data visualization, analysis, model building & testing repeated, several times. In the process we use various techniques like data wrangling, plotting, data organization, regression, machine learning and modeling.

  Accordingly, the approach we took was step-by-step, iterative and employing various analysis and model building techniques.
- There are several factors that might influence ratings for movies. We have used only user effects, movie effects and genre effects in this project because our data is limited to the same.

  However, there can be more better predictors such as movie budget, director, marketing, user groups, social trends, etc., provided data is available on them.
- The final model that we have developed can be used to predict relatively accurately the "rating" for a specific "userId", specific "movieId" & and specific "genre".
- The next logical step for this project is to gather additional data on user profiles and further work on the model and develop an actual movie recommendation system, which recommends movies to users (for specific userIds).

Once again, to highlight, the RMSE achieved is 0.8648532 on validation dataset in this project.

——————END——————