Harvardx PH125.9x Capstone MovieLens Project

Mayank Gaur

10/10/2020

1. Executive summary

This report is prepared for Harvardx Data Science Professional Certificate course. It is part of the Capstone project module. The purpose of this project is to build recommendation system for the movielens dataset provided in the instructions by using machine learning algorithms.

The movielens data set was as split into edx and validation data set. The edx data set contained 90% of the movielens data and validation data set contained 10% of the movielens data. Validation data set was used only at the end when the prediction model had been finalized. edx data set was used to build the model. edx data set was split into train and test data sets, train set was used to train the model and the performance of the model was evaluated on test set.

RMSE(Root Mean Square Error) was used as an indicator to evaluate the performance of the model. The goal of the project was to create a recommendation system that gives an RMSE less than 0.8649

The edx data set was analyzed and effects for various attributes such as movieId, userId, release year and genres on ratings were visualized. RMSE was calculated for different models. Regularization was carried out using k fold cross validation using k=5 to find the optimum lambda that gives the lowest RMSE. Regularization model was finally used with optimum lambda obtained to train the edx data set and predict the ratings using validation set. Final RMSE of 0.8642542 was obtained on the validation set.

2. Introduction

Recommendation system play an important role in today's commercial business environment. The system is being used by online streaming services such as Netflix and Hotstar for predicting ratings for their movies and online shows and also by e-commerce companies to predict customer ratings for their products. Usually, the recommendation system is based on a rating scale of 1 to 5 with 1 indicating lowest satisfaction and 5 indicating highest satisfaction.

The aim of a recommendation system is to help users find what they want on the basis of user preferences and hence help to predict rating for a new product or item thereby leading to increased revenue and profit.

3. Overview

10M version of the MovieLens data set was used for this project. The movieLens data set was split into edx and validation data set. The edx data set contained 90% of the movielens data and validation data set contained 10% of the movielens data. Validation data set was used only at the end when the prediction model had been finalized. edx data set was used to build the model. edx data set was split into train and test data sets, train set was used to train the model and the performance of the model was evaluated on test set.

The edx and validation data sets were automatically downloaded with the code provided in the project instructions.

4. Initial Data Exploration

The first six rows of the edx data set are

```
head(edx)%>%knitr::kable("simple")
```

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

```
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>
```

There are 9000055 observations and 6 variables in edx data set. The variables and the information contained in the variables are:-

userId: Integer variable describes the Unique Identification number for each user

movieId: Numeric variables describes the Unique Identification no given to each movie

rating: Numeric variable from 0 to 5 in steps of 0.5 describes the rating given to a movie.

timestamp: Integer variable that contains date and time when rating was given to a particular movie by a user

title: Character variable describes Title of the movie. Title of the movie also includes the year in which movie was released

genres: Character variable that defines the genre associated with a movie

The edx data set did not contain any NA values

```
any(is.na(edx))
```

```
## [1] FALSE
```

```
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
##
##
##
```

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

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

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

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

There are 69878 unique users and 10677 unique movies in the edx data set. The data is in tidy format as each row as one observation and column names are features. If we multiply the number of distinct users that rated the movies and number of distinct movies that were rated, we will have about 750 million user movie combinations but we have 9000055 rows in edx data set. This is because not every user rated every movie.

4.1 Feature Engineering

From the initial data exploration, it is was incurred that the release year of movie is contained in the title column of the data set. Similarly the timestamp column of the data set contains the date when the movie was rated by the particular user. We can extract the year in which the movie was rated from the timestamp column. The genres column contains the genre combination which includes every genre that applies to the particular movie. Some movies fall under several genres. For our analysis, we will consider the genre combination that appears in the genres column as a unique category. The following feature engineering was performed on the edx data set.

Extract release year from the title column

Extract rating year in which movie was rated from the timestamp column

Add a column which tells the age of the movie when it was rated. (Age of movie at time of rating is calculated as rating year-release year)

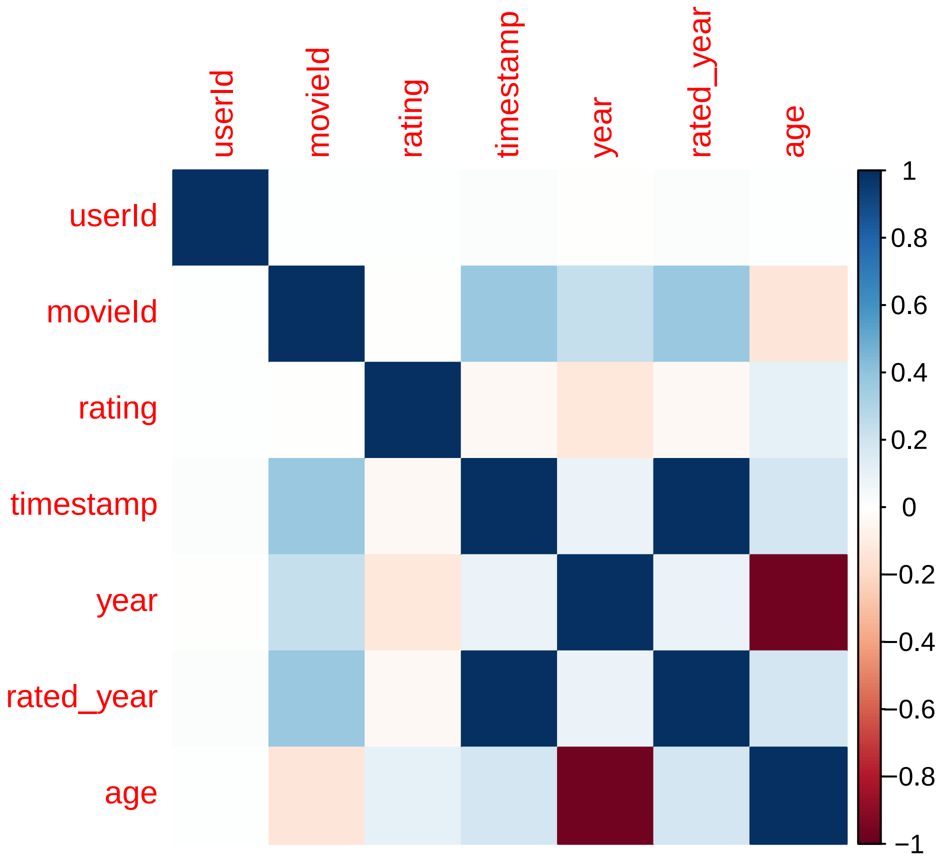
```r
edx<-edx%>%mutate(year=as.numeric(str_sub(title,-5,-2)))%>%
  mutate(rated_year=year(as_datetime(timestamp)))%>%
  mutate(age=rated_year-year)
```

We have used the function str_sub() to extract the movie release year from the title column and as_datetime() function to extract the year in which movie was rated from the timestamp column.

5. Exploratory Data Analysis

The correlation between the numeric variables was visualized.

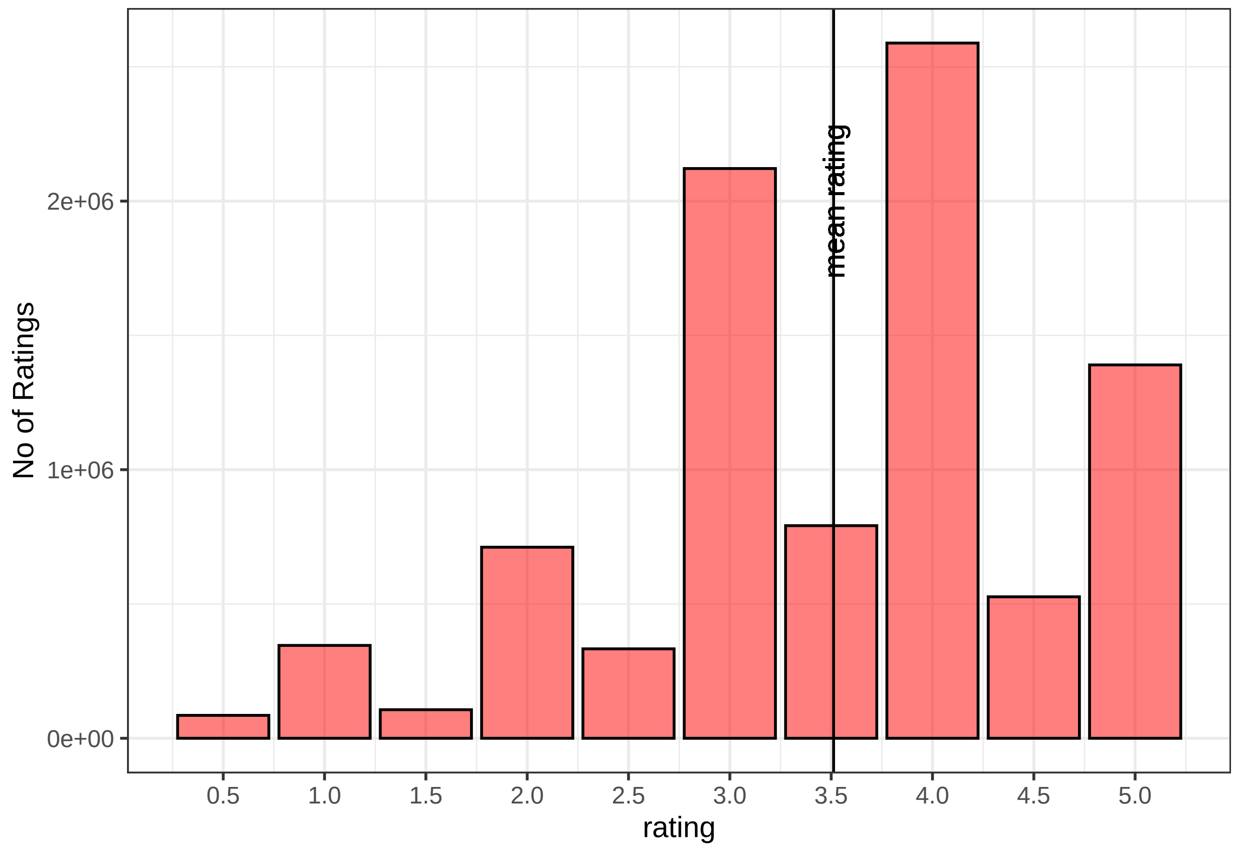
```r
cor.data<-cor(edx[,c("userId","movieId","rating",
"timestamp","year","rated_year","age")])
corrplot(cor.data,method='color')
```



The graph shows correlation between different numeric columns of the data set. The diagonal correlation between a variable with itself which is 1. Different shades of the color represent the correlation between the different numeric variables.

5.1 Visualization of Ratings distribution

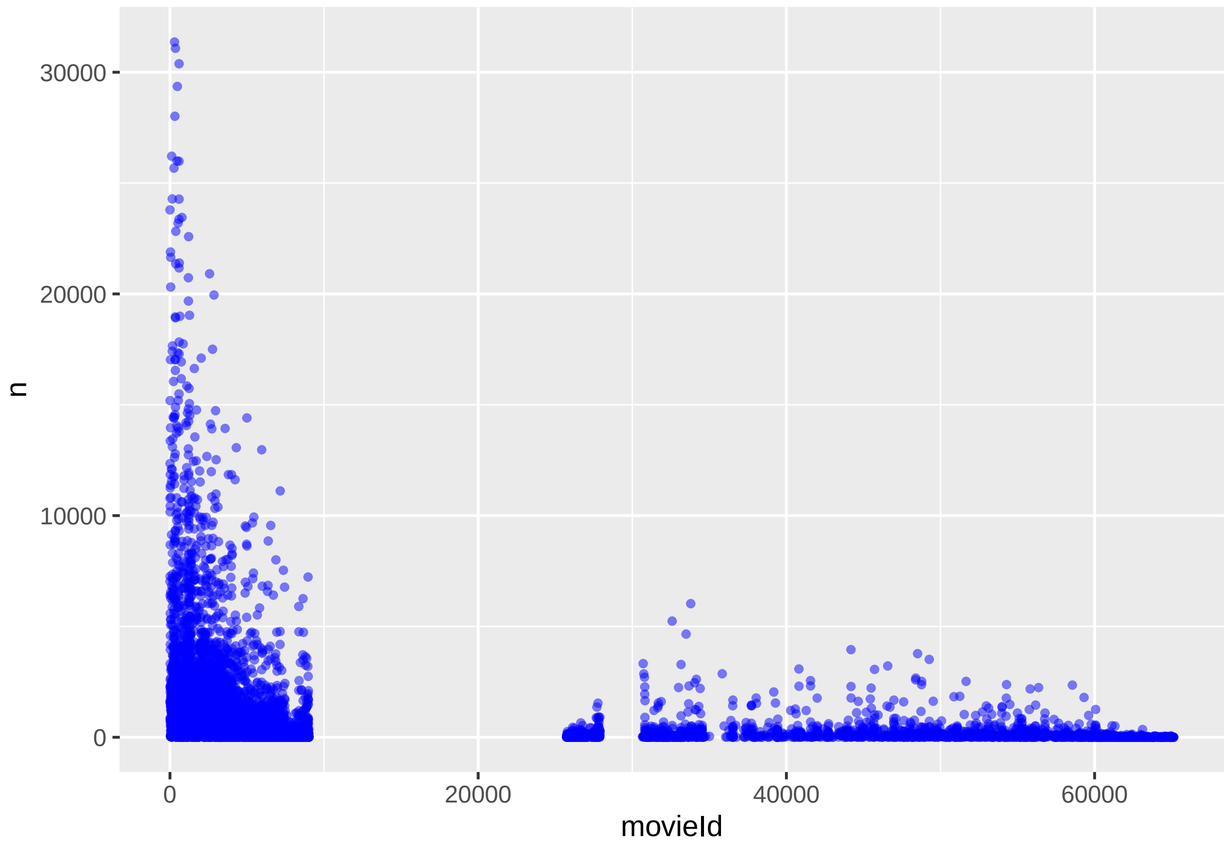
```
edx%>%group_by(rating)%>%summarize(n=n())%>%
  ggplot(aes(rating,n))+
  geom_bar(stat="identity",alpha=.5, color="black",fill="red")+
  theme_bw()+
  scale_x_continuous(breaks=seq(min(0),max(5),by=0.5))+
  ylab("No of Ratings")+geom_vline(xintercept = 3.512)+
  geom_text(aes(x=3.512,label="mean rating", y=2000000),angle=90)
```



From the ratings distribution, we find that users give more ratings in round values than in half values. Rating of 4 is more preferred rating given by users. The vertical line on the X axis represents the mean rating

5.2 Visualization of movies distribution

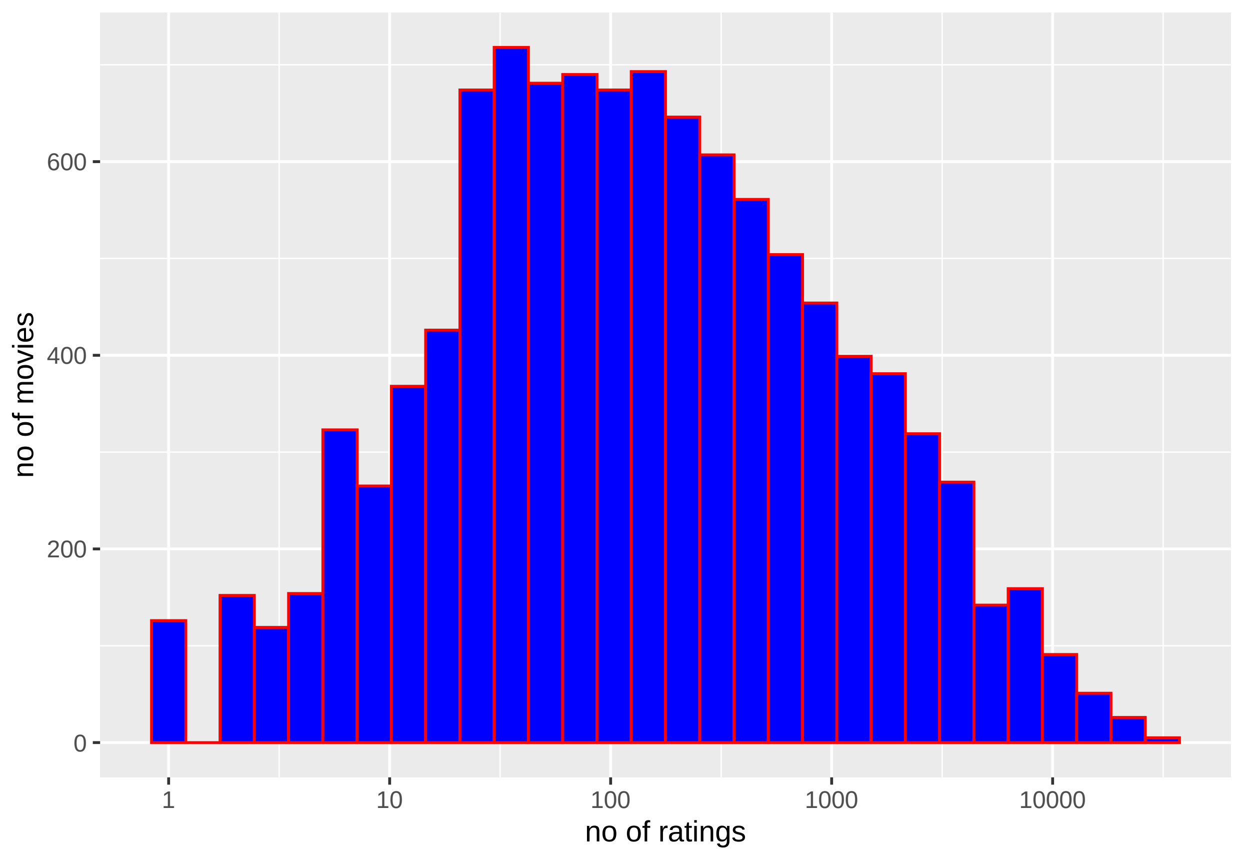
```r
edx%>%group_by(movieId)%>%
  summarize(n=n())%>%ggplot(aes(movieId,n))+
  geom_point(alpha=.5,size=1,color="blue")
```



if we visualize the scatter plot of the movieId appearing in the data set, we can see some movies are appearing more in the data set indicating that some movies were rated more than others.

We now group the data by movieId and plot the histogram.

```r
edx%>%group_by(movieId)%>%
  summarize(n=n())%>%ggplot(aes(n))+
  geom_histogram(bins = 30,color="red",fill="blue")+
  scale_x_log10()+xlab("no of ratings")+ylab('no of movies')
```



On analyzing the distribution of movies that are rated, we find that some movies have been rated more than other movies. This could be due to the fact that some movies are more popular than other movies and hence are rated by more people. Some movies have been rated very few no of times and hence should be given less importance while predicting ratings. The graph also shows us that some movies have been rated only once. There are 126 movies that have been rated just once. Some of these movies are:-

```
edx%>%group_by(movieId,title)%>%
  summarize(n=n())%>%filter(n==1)%>%
  head(10)%>%
  knitr::kable("pipe")
```

movieId	title	n
3191	Quarry, The (1998)	1
3226	Hellhounds on My Trail (1999)	1
3234	Train Ride to Hollywood (1978)	1
3356	Condo Painting (2000)	1
3383	Big Fella (1937)	1
3561	Stacy's Knights (1982)	1
3583	Black Tights (1-2-3-4 ou Les Collants noirs) (1960)	1
4071	Dog Run (1996)	1
4075	Monkey's Tale, A (Les ChÃ¢teau des singes) (1999)	1
4820	Won't Anybody Listen? (2000)	1

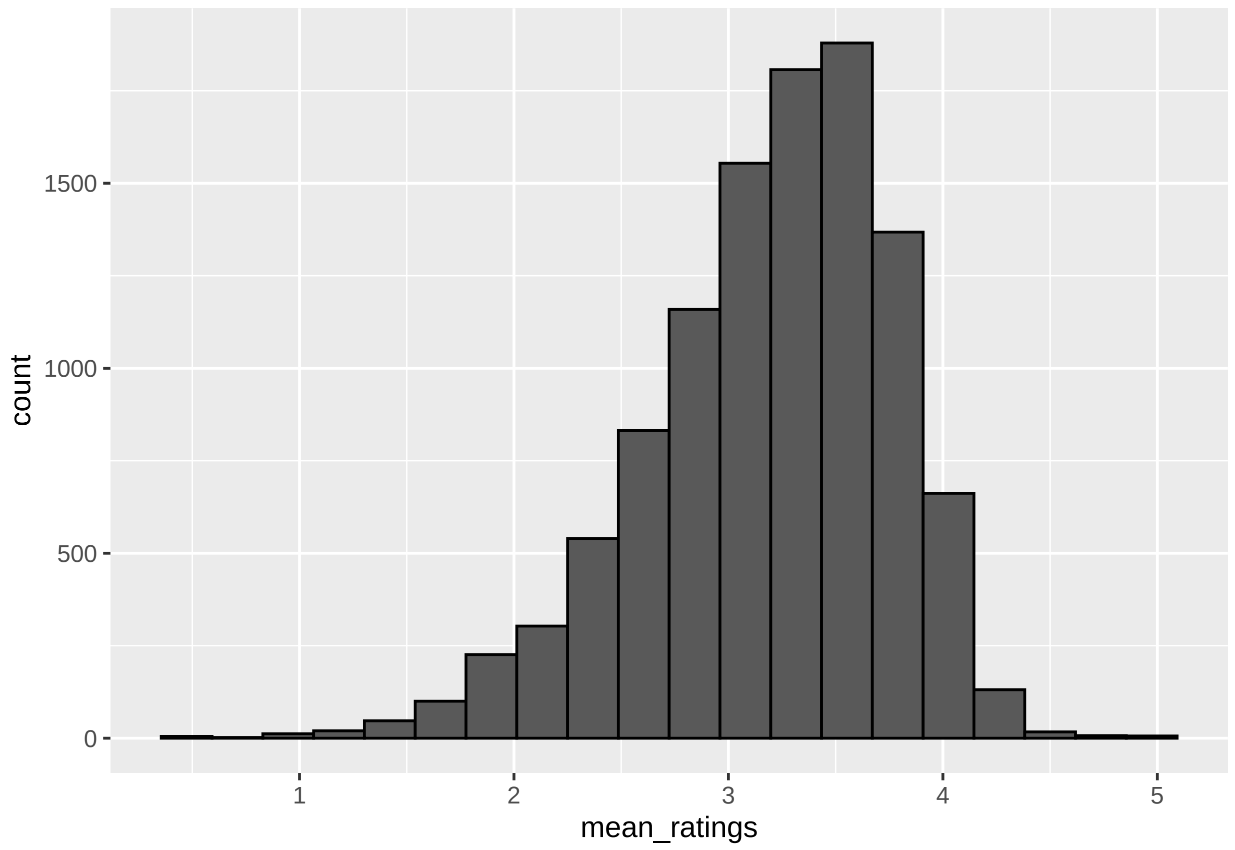
On the other hand the Top 10 rated movies are:-

```r
edx%>%group_by(movieId,title)%>%
  summarise(n=n())%>%
  arrange(desc(n))%>%
  head(10)%>%knitr::kable()
```

movieId	title	n
296	Pulp Fiction (1994)	31362
356	Forrest Gump (1994)	31079
593	Silence of the Lambs, The (1991)	30382
480	Jurassic Park (1993)	29360
318	Shawshank Redemption, The (1994)	28015
110	Braveheart (1995)	26212
457	Fugitive, The (1993)	25998
589	Terminator 2: Judgment Day (1991)	25984
260	Star Wars: Episode IV - A New Hope (a.k.a. Star Wars) (1977)	25672
150	Apollo 13 (1995)	24284

Below is the histogram plot for the average rating for movies.

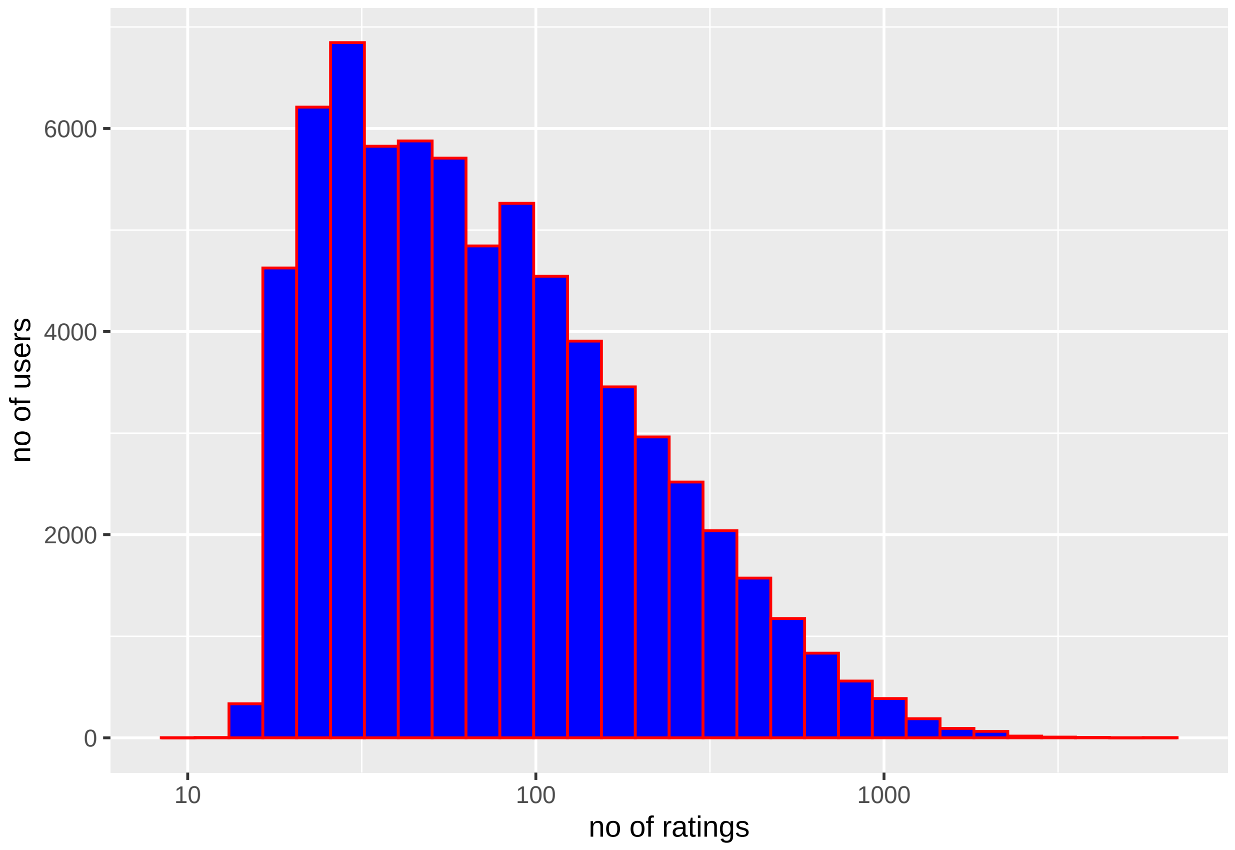
```r
edx%>%group_by(movieId)%>%
  summarize(mean_ratings=mean(rating))%>%
  ggplot(aes(mean_ratings))+
  geom_histogram(bins =  20, color="black")
```



Also on analyzing the mean ratings for the movies we find that some movies have received a higher rating while some other movies have received lower ratings. From the analysis of movies distribution we can conclude that the movie effect does affect the ratings.

5.3 Visualization of Users distribution

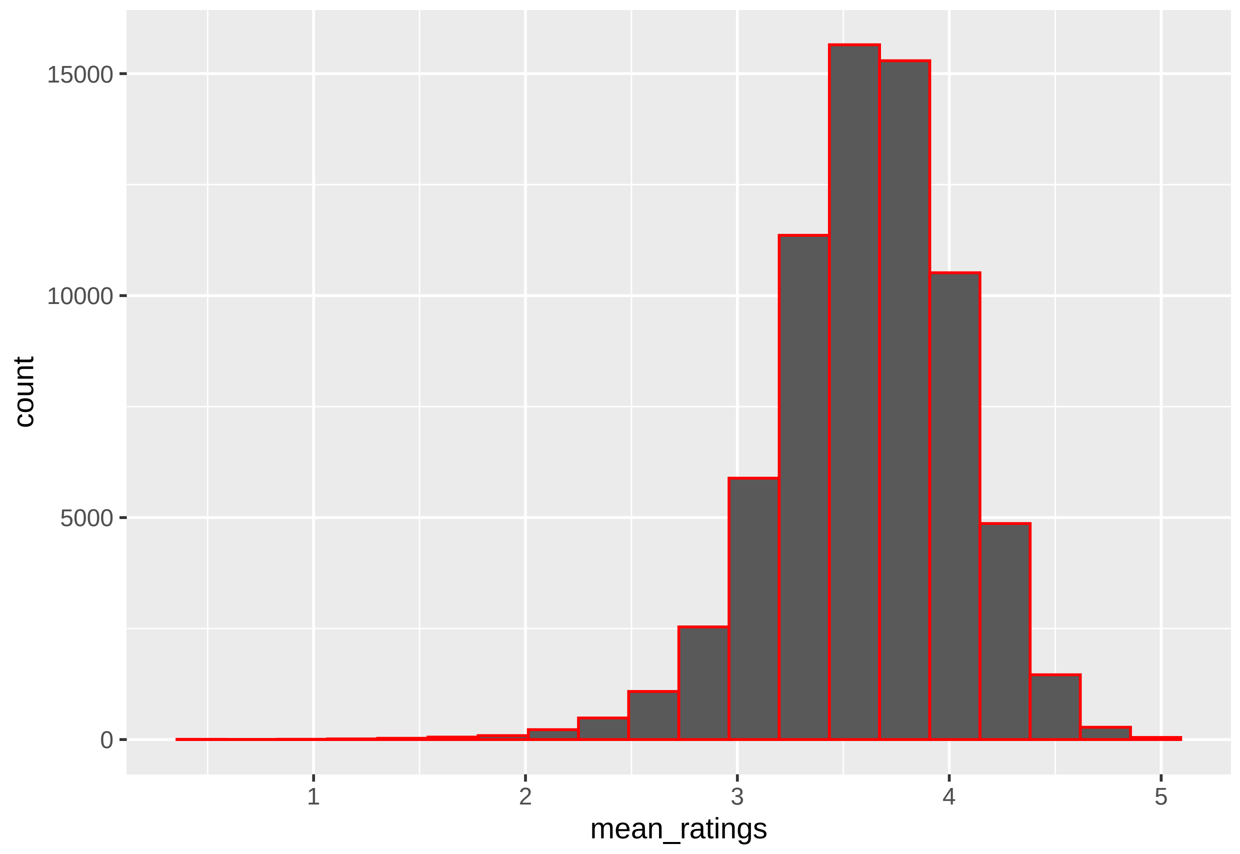
```r
edx%>%group_by(userId)%>%
  summarize(n=n())%>%ggplot(aes(n))+
  geom_histogram(bins = 30,color="red",fill="blue")+
  scale_x_log10()+xlab("no of ratings")+ylab("no of users")
```



There are total 69878 distinct users in the train set that rate movies. Only doing a basic analysis of the visualization of the no users and the no of ratings given by users, we find that some users have rated more movies than others. Most users have rated between 30 and 100 movies. This is further confirmed by the histogram visualization. Some users are less active and have rated fewer movies and their biases may impact prediction results.

Below is the histogram plot for mean ratings provided by the users.

```
edx%>%group_by(userId)%>%
  summarize(mean_ratings=mean(rating))%>%
  ggplot(aes(mean_ratings))+
  geom_histogram(bins=20,color="red")
```

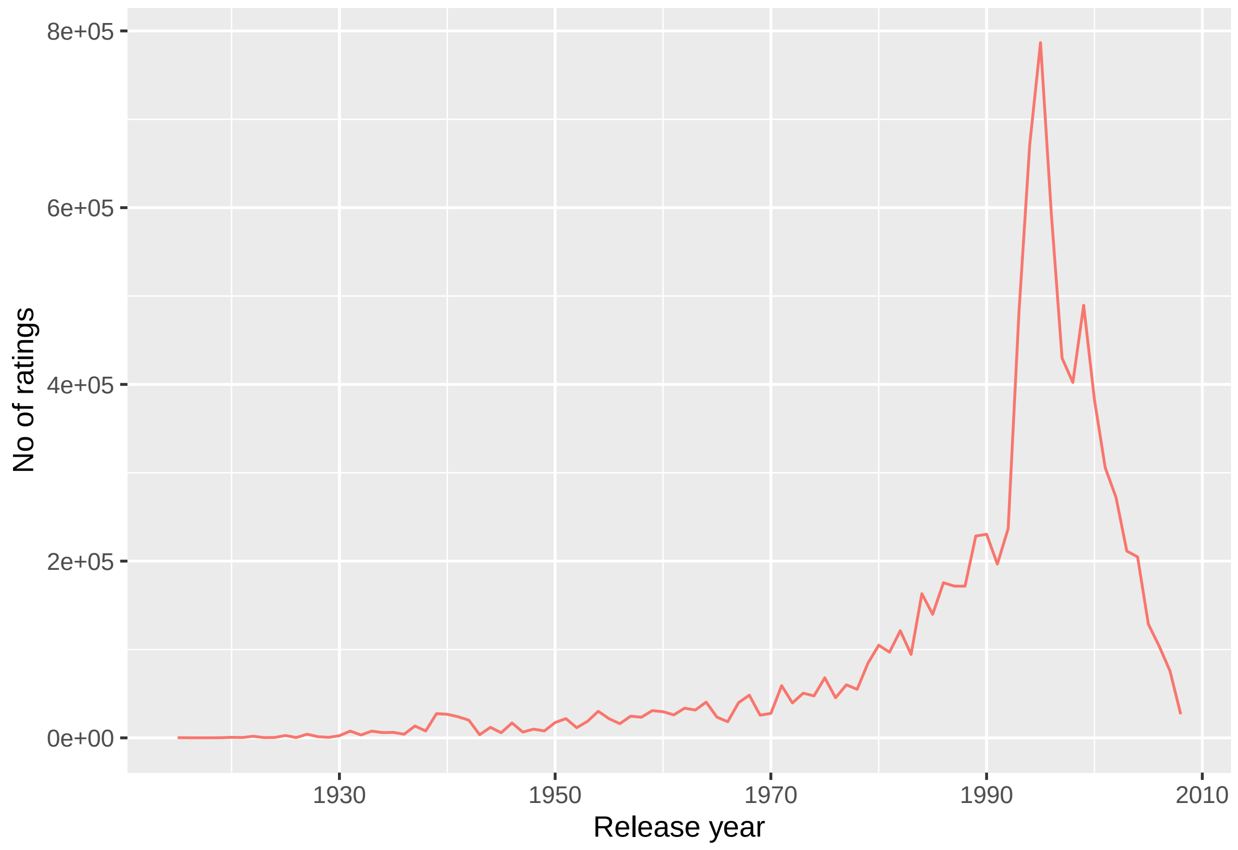


The analysis of mean rating for users distribution suggests that Some users tend to give higher ratings than other users and the preferences of the user or the user bias should be included in our model. This implies that some users are harsh while rating movies while other users are liberal in rating and tend to like every movie they rate.

5.4 Visualization of release year distribution

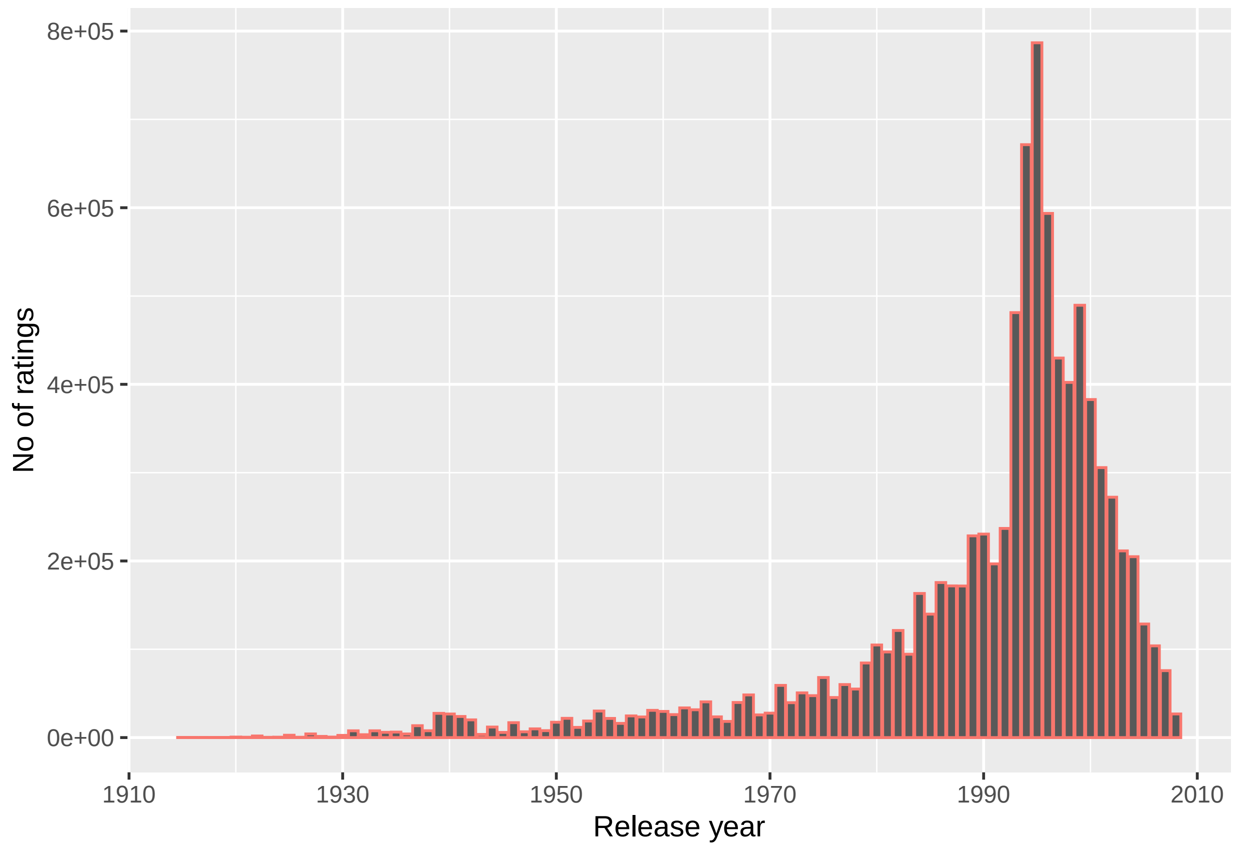
Below is the line graph for the trend for release years of movies appearing in the data set

```r
edx%>%group_by(year)%>%
  count()%>%
  ggplot(aes(year,n,color="red"))+
  geom_line(show.legend = FALSE)+ ylab("No of ratings")+
  xlab("Release year")
```



Below is the bar graph plot for movie release year distribution

```
edx%>%group_by(year)%>%
  count()%>%
  ggplot(aes(year,n,color="red"))+
  geom_bar(stat="identity",show.legend = FALSE)+
  ylab("No of ratings")+xlab("Release year")
```



We have used the line graph to visualize the trend for the number of ratings provided over the years. We also use a bar graph to visualize movies released in which years received most ratings. The visualization and analysis of release year of movies year tell us that movies released in the 90s received more ratings. Movies released in 1994-1996 were rated by more no of users.

The six release years of which most movies are provided ratings are:-

```
edx%>%group_by(year)%>%summarise(n=n())%>%
  arrange(desc(n))%>%
  head()%>%
  knitr::kable("pipe")
```

year	n
1995	786762
1994	671376
1996	593518
1999	489537

year	n
1993	481184
1997	429751

The six release years of which least movies are provided ratings are:-

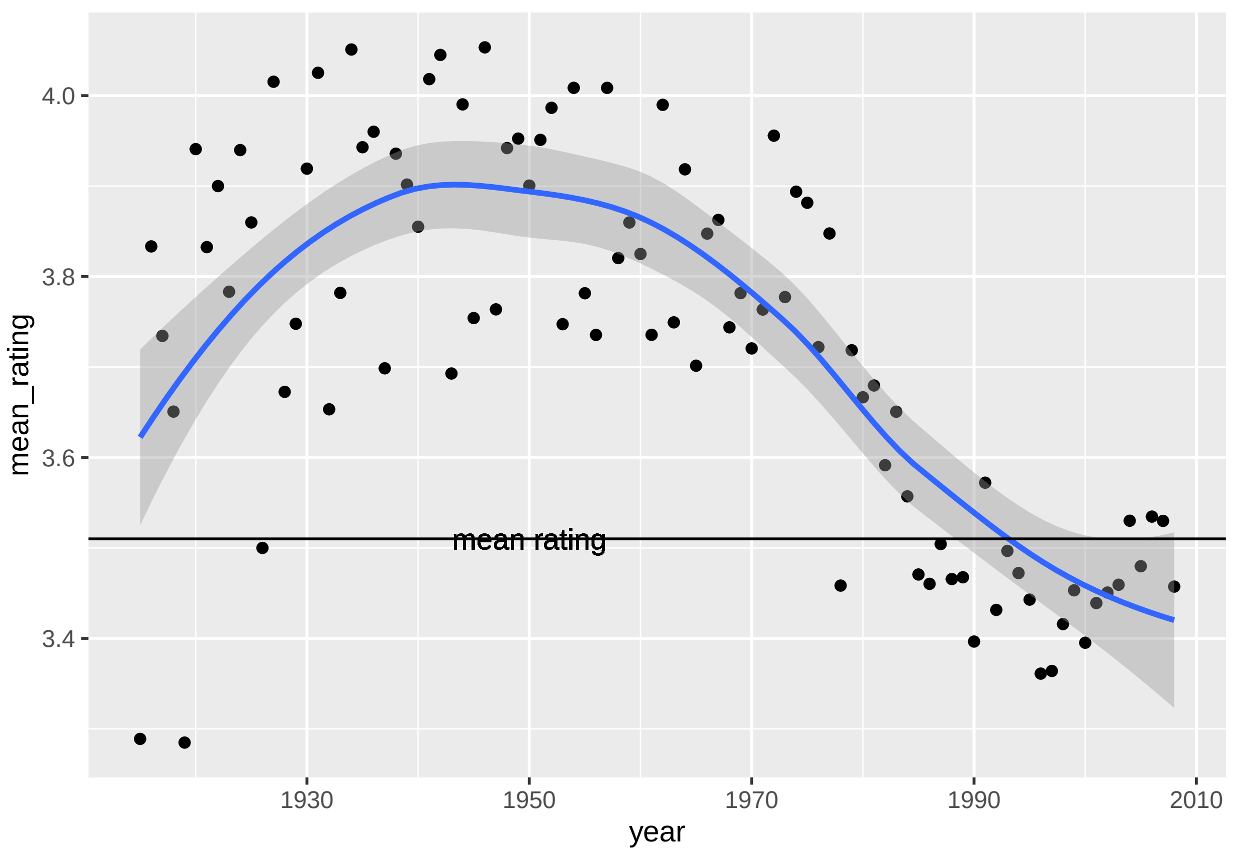
```r
edx%>%group_by(year)%>%summarise(n=n())%>%
  arrange(n)%>%
  head()%>%
  knitr::kable("pipe")
```

year	n
1917	32
1918	73
1916	84
1919	158
1915	180
1923	316

Highest no of ratings (7,86,762) were provided for movies released in the year 1996 and least no of ratings(32) were provided for movies released in the year 1917.

Below is the plot to visualize how the mean ratings have varied for various release years.

```r
edx%>%group_by(year)%>%
  summarize(mean_rating=mean(rating))%>%
  ggplot(aes(year,mean_rating))+
  geom_point()+geom_smooth()+
  geom_hline(yintercept = 3.51)+
  geom_text(aes(x=1950,y=3.51,label="mean rating"))
```

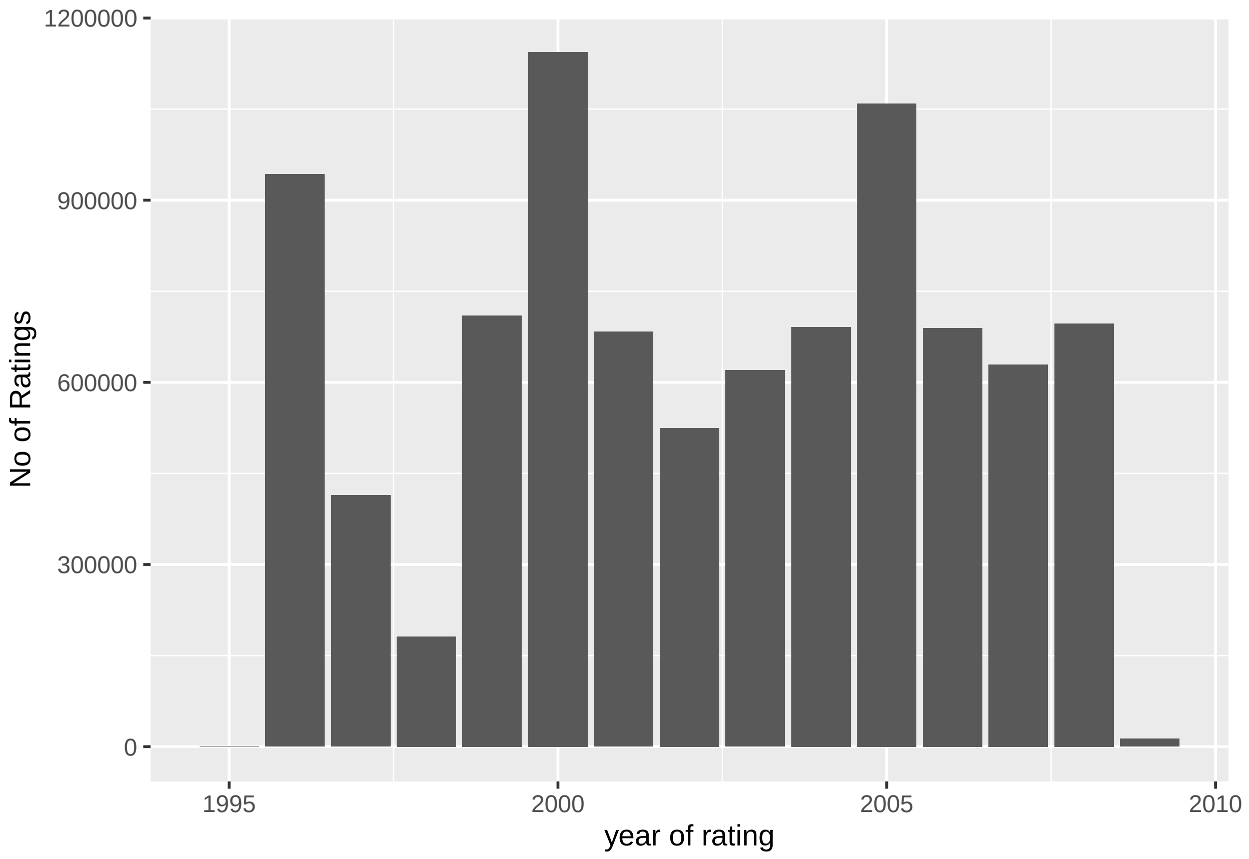


If the we analyze the trend of average ratings provided to the movies grouping the data set by release year, we find that the movies released recently are rated lower than average, Whereas movies released from 1930s to 1970s were on average rated higher. But on the other hand the movies released in the earlier decades were rated by less no of users and may be their bias resulted in higher average ratings for these movies.

5.5 Visualization of Rating year

Below is the bar graph for the total no of ratings provided by users in different years.

```
edx%>%group_by(rated_year)%>%
  count()%>%
  ggplot(aes(rated_year,n))+
  geom_bar(stat = "identity",show.legend = FALSE)+
  xlab("year of rating")+
  ylab("No of Ratings")
```



As per the data set the user started rating the movies from 1995 onward. Earlier we had seen that movies released in 1996 were rated by maximum no users. More no of Users tended to rate recently rated movies. Specifically the three years in which most users rated the movies are 2000, 2005 and 1996. The no of ratings given in these years are

```
edx%>%group_by(rated_year)%>%
 summarize(n=n())%>%
 arrange(desc(n))%>%
 head(3)%>%knitr::kable("pipe")
```

rated_year	n
2000	1144349
2005	1059277
1996	942772

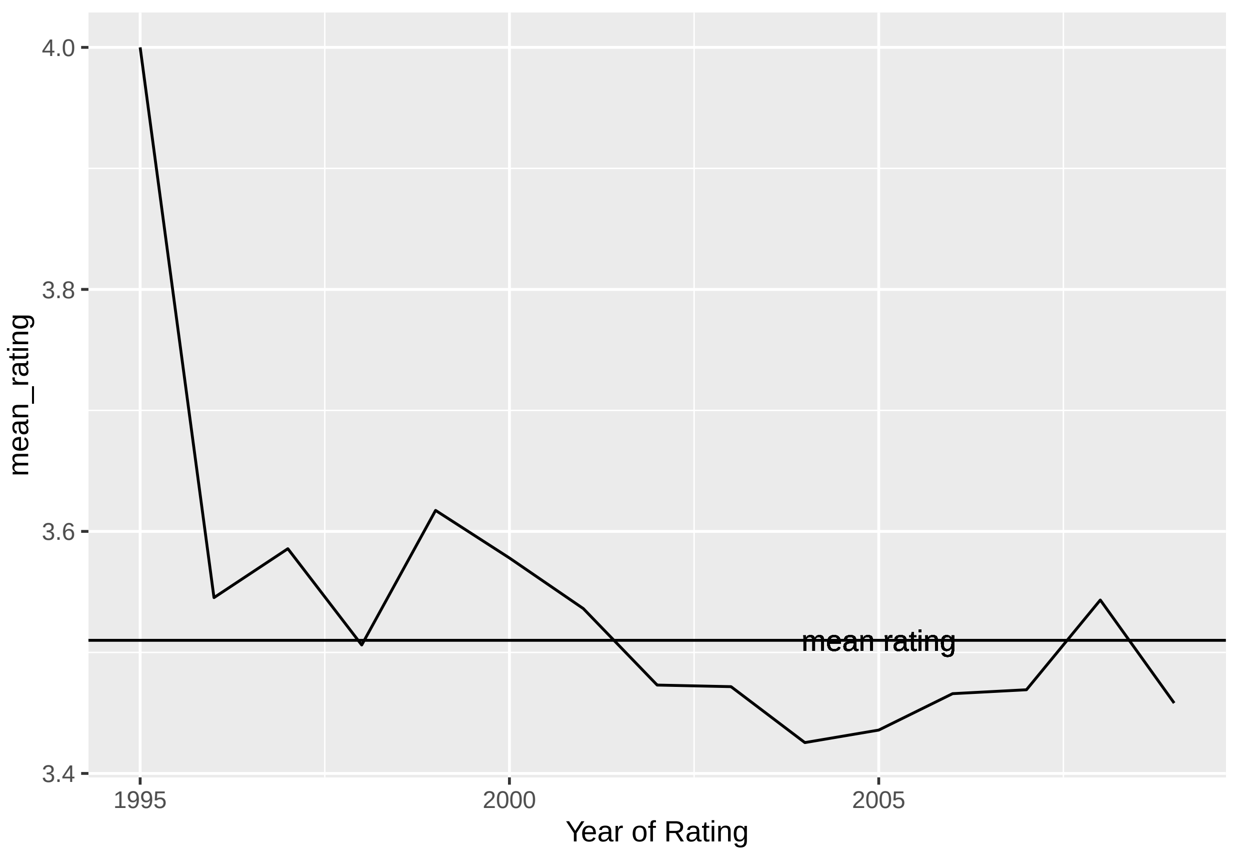
The average ratings given by users in different years in which the movie was rated are:-

```
edx%>%group_by(rated_year)%>%
  summarize(mean_rating=mean(rating))%>%
  knitr::kable("pipe")
```

rated_year	mean_rating
1995	4.000000
1996	3.545286
1997	3.585703
1998	3.506144
1999	3.617354
2000	3.578044
2001	3.536229
2002	3.473012
2003	3.471691
2004	3.425479
2005	3.435830
2006	3.465925
2007	3.469147
2008	3.543312
2009	3.458165

Below is the basic line graph for the average ratings provided in different users in which the movies were rated.

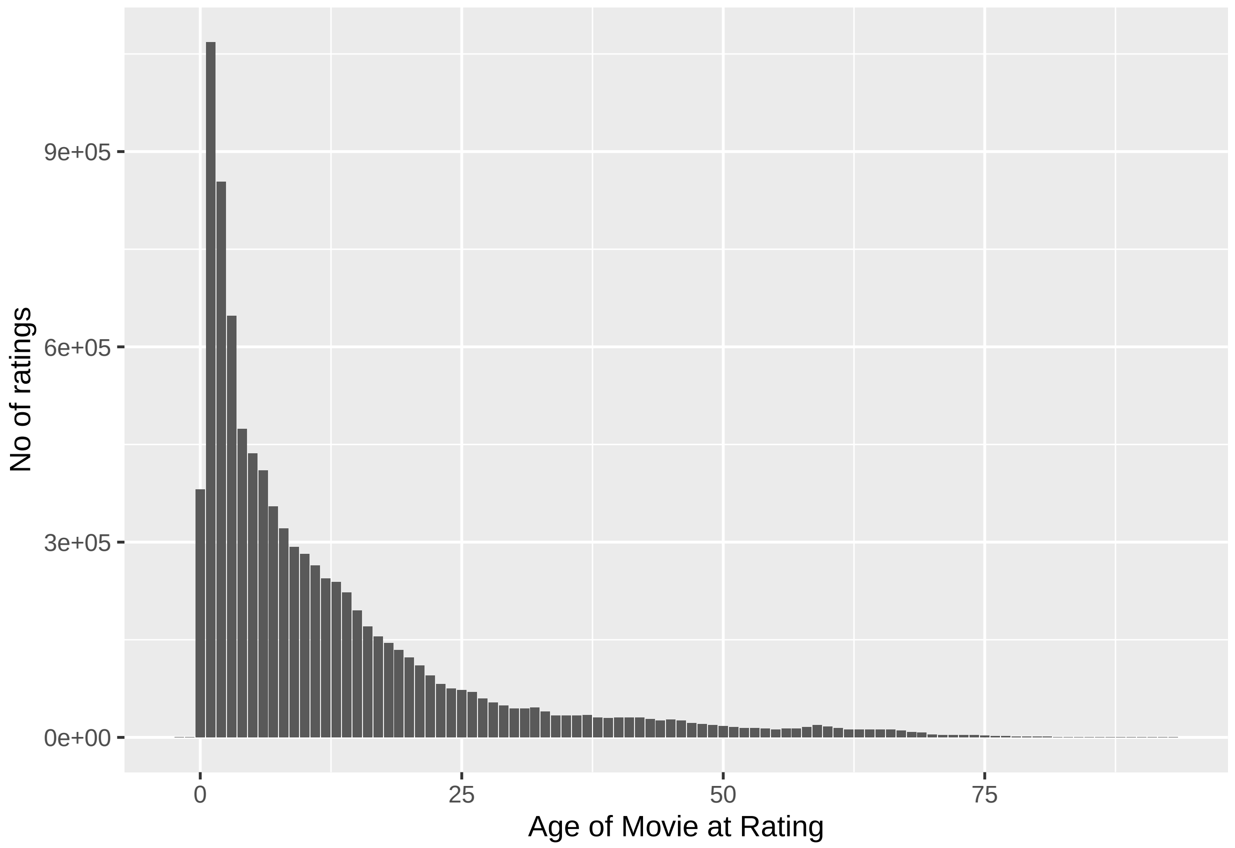
```r
edx%>%group_by(rated_year)%>%
  summarize(mean_rating=mean(rating))%>%
  ggplot(aes(rated_year,mean_rating))+
  geom_line()+
  geom_hline(yintercept = 3.51)+xlab("Year of Rating")+
  geom_text(aes(x=2005,y=3.51,label="mean rating"))
```



If we study the mean ratings that users have given over the years, we can analyze the table of mean ratings over the years in which ratings were provided and also the trend from the graph. Users have tended to give lower ratings to movies over the years. Since 2001, less than average ratings have been provided to movies.

5.6 Visualization of Age of Movie at time of rating

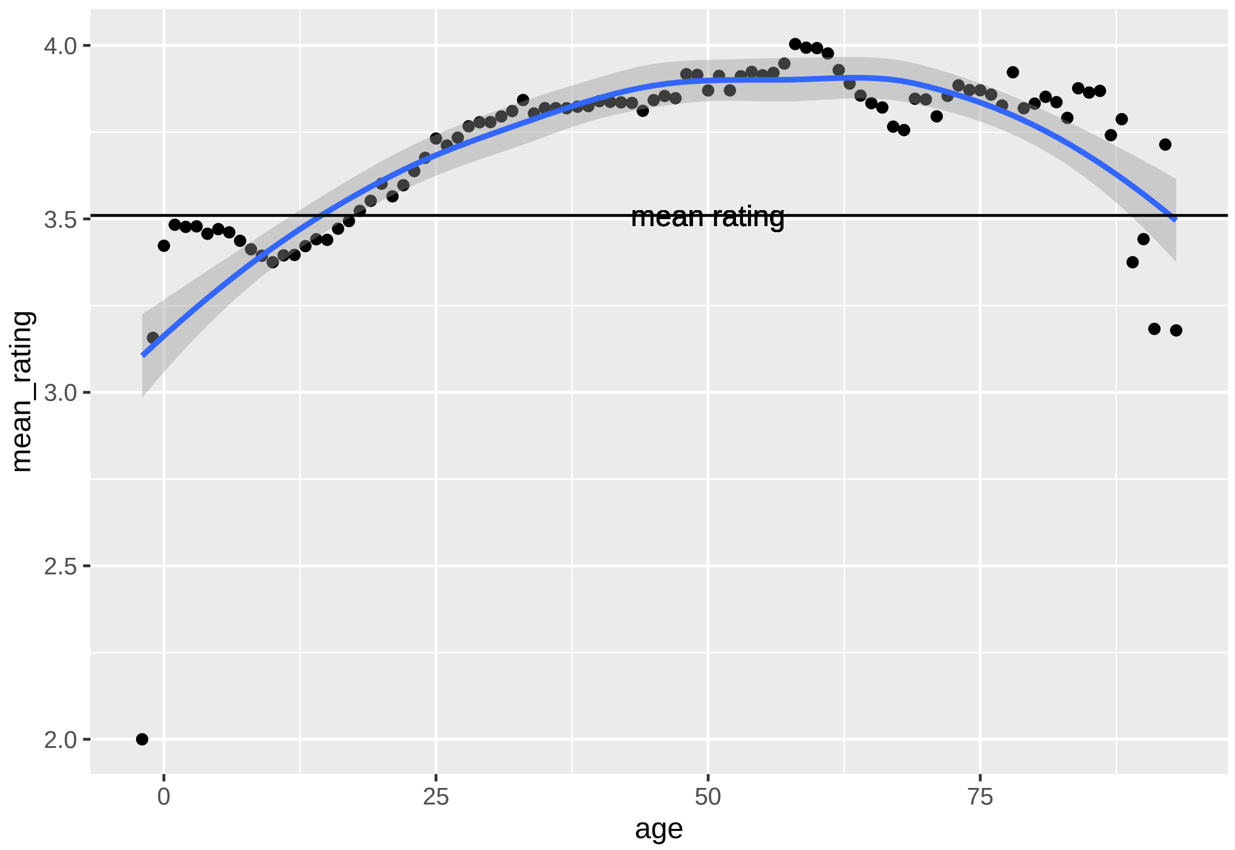
```r
edx%>%group_by(age)%>%
  count()%>%
  ggplot(aes(age,n))+
  geom_bar(stat="identity")+
  xlab("Age of Movie at Rating")+
  ylab("No of ratings")
```



A better estimate of the trend of year effect could be obtained by analyzing the age effect which was calculated as a column for the difference between the year in which movie was rated (rated_year) and the year in which it was released(year). Most ratings were given for newer movies(within 25 years of being released). It could also be that there are more recent movies in the data set that have been rated by users.

Below is the scatterplot and the trend for the average ratings for different ages of the movies at the time of rating

```
edx%>%group_by(age)%>%
  summarize(mean_rating=mean(rating))%>%
  ggplot(aes(age,mean_rating))+
  geom_point()+geom_smooth()+
  geom_hline(yintercept = 3.51)+
  geom_text(aes(x=50,y=3.51,label="mean rating"))
```



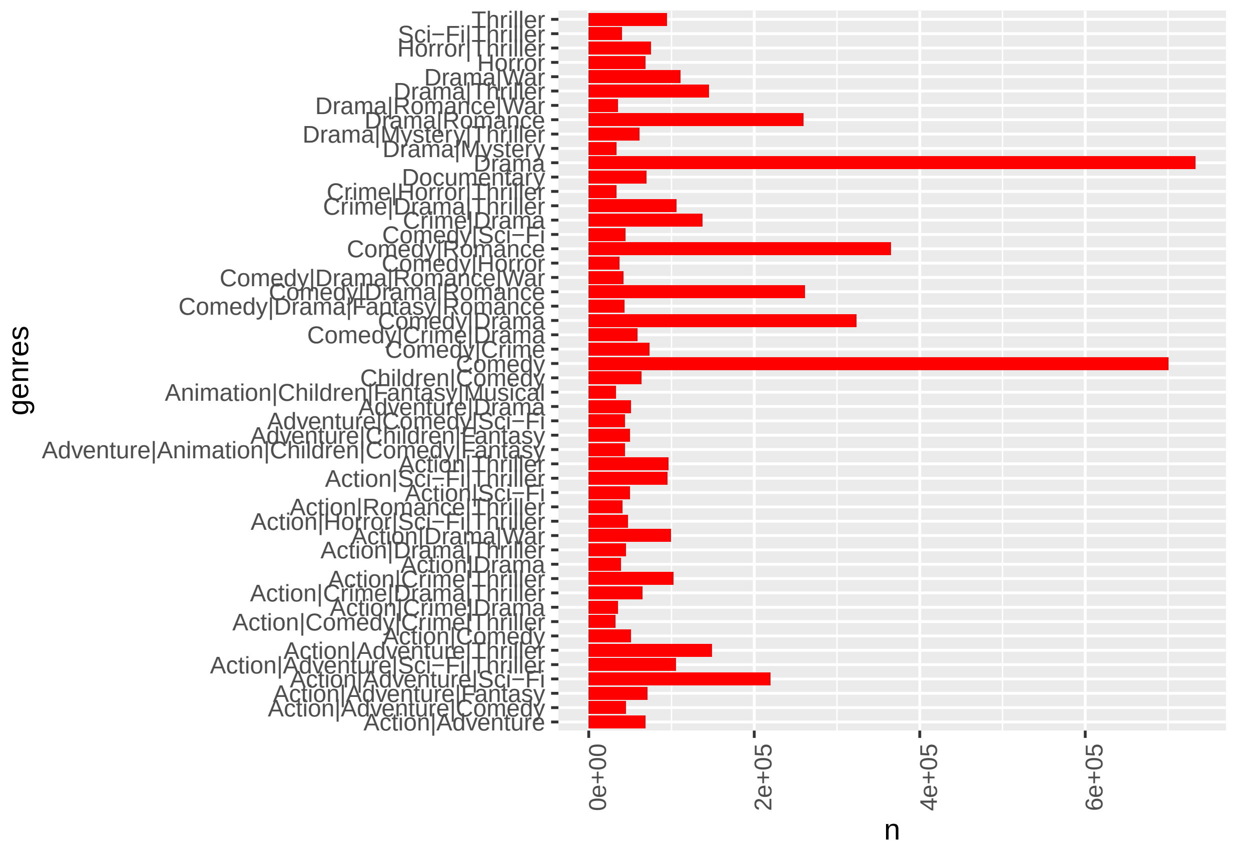
Movies with age 30 to 60 years at the time of rating have received higher than the average ratings and movies with age around 18-20 years at the time of rating tend to be around the average rating.

It can be interpreted together with the release year of movie and rating year in which the movie was rated that as a movie becomes older not only lesser no of users are rating it but also lower ratings are given for older movies. This could be due to the fact that users have lesser preferences for older movies.

5.7 Visualization of Distribution of genres

Below is the bar graph plot for the no of ratings provided for the 50 most popular genres or combination of genres.

```r
edx%>%group_by(genres)%>%
  summarize(n=n())%>%top_n(50,n)%>%
  ggplot(aes(n,genres))+
  geom_bar(stat="identity",fill="red")+
  theme(axis.text.x = element_text(angle = 90))
```



There a total of 797 distinct genres or genre combinations. The distribution for the 50 most popular genres or combination of genres in the data set were visualized and analyzed by treating the combination of genre as it appears in the genres column as a unique genre. Movies associated with some genres have been rated more than others. This could be due the the fact that certain genres are more popular than others.

Some of the most common genres/genres combinations that are associated with most movies are:-

```r
edx%>%group_by(genres)%>%
  summarize(n=n())%>%arrange(desc(n))%>%
  head(10)%>%knitr::kable("simple")
```

genres	n
Drama	733296
Comedy	700889
Comedy|Romance	365468

genres	n
Comedy|Drama	323637
Comedy|Drama|Romance	261425
Drama|Romance	259355
Action|Adventure|Sci-Fi	219938
Action|Adventure|Thriller	149091
Drama|Thriller	145373
Crime|Drama	137387

We can also see which are the ten genres that are appearing most in the data set either as individual genre or as part of combination of genre.

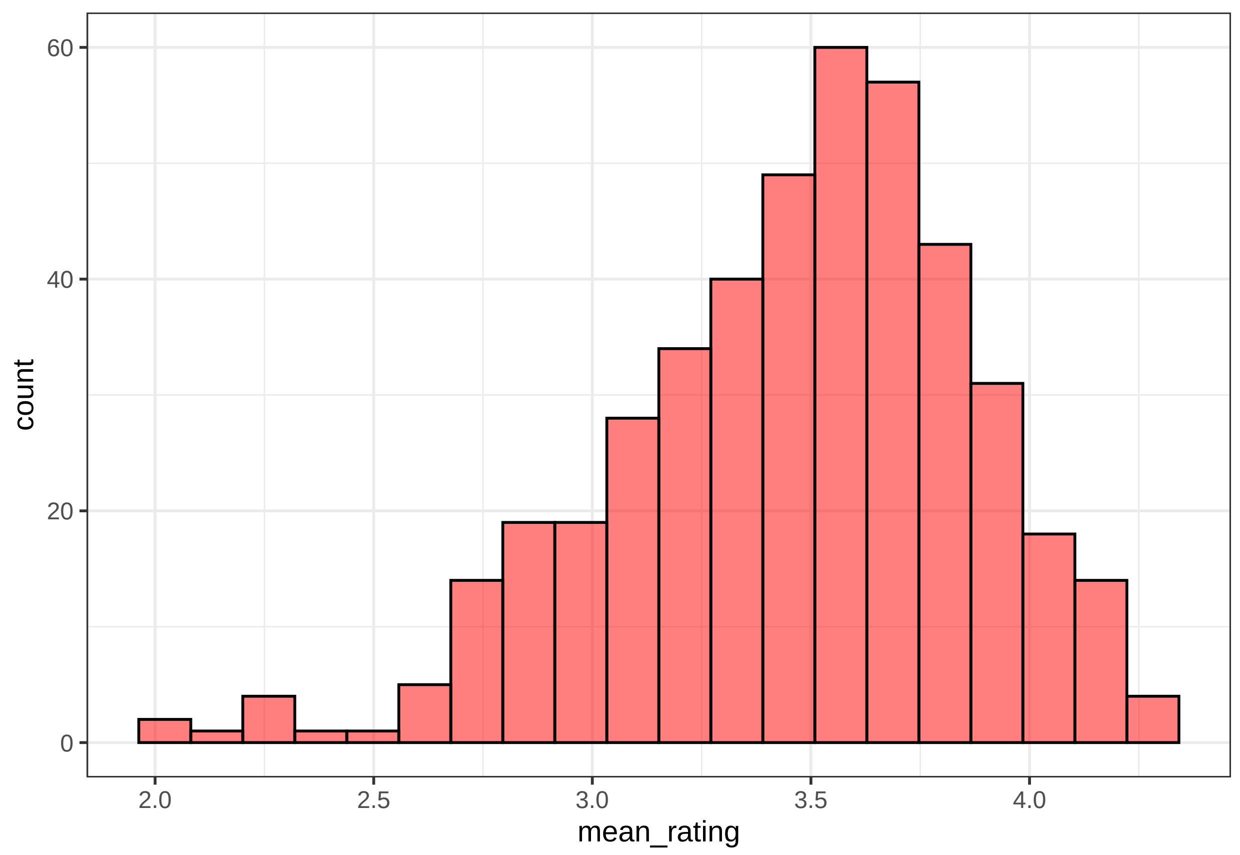
```r
genre = c("Drama", "Comedy", "Thriller",
          "Romance","Action","Adventure",
          "Sci-Fi","Horror","Fantasy","Children")

sapply(genre, function(g) {
  sum(str_detect(edx$genres, g))
})
```

```
##     Drama    Comedy  Thriller   Romance    Action Adventure    Sci-Fi    Horror
##   3910127   3540930   2325899   1712100   2560545   1908892   1341183    691485
##   Fantasy  Children
##    925637    737994
```

Drama,Comedy and Thriller are the three genres that appear the most either as unique genre or in combination with other genres.

Below is the histogram plot for the average ratings by grouping the data by genres by considering genres that have more than 1000 movies associated with them.

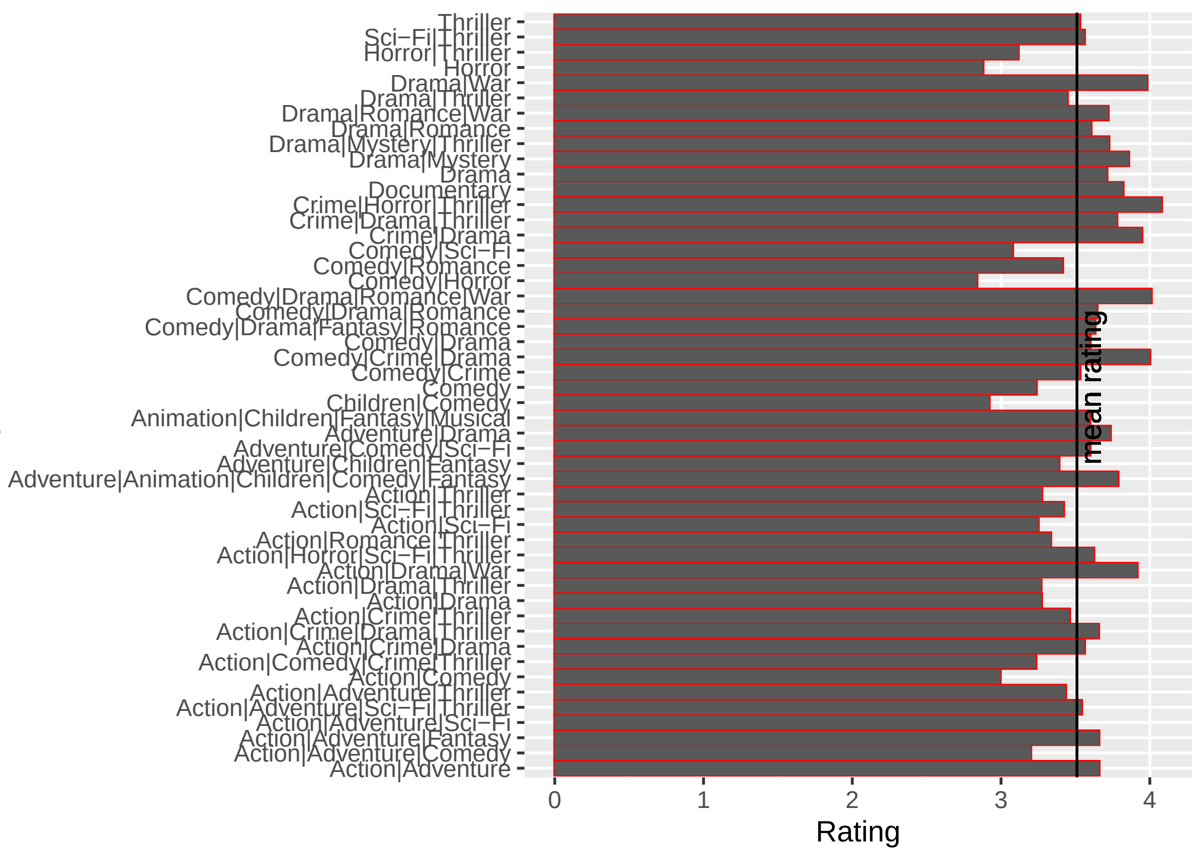
```r
edx%>%group_by(genres)%>%
  filter(n()>1000)%>%summarise(mean_rating=mean(rating))%>%
  ggplot(aes(mean_rating))+
  geom_histogram(bins=20,alpha=0.5,color="black",fill="red")+
  theme_bw()
```



If we plot the mean ratings for the genres by considering the genres that have more than 1000 movies associated with them it can be seen that movies associated with certain genres have received higher ratings than movies associated with other genres. Therefore in our model, we will include the genre effect.

Below is the bar graph for the average ratings for the 50 most pouplar genres or combination of genres.

```
edx%>%group_by(genres)%>%
  summarize(n=n(),mean_rating=mean(rating))%>%
  top_n(50,n)%>%
  ggplot(aes(mean_rating,genres))+
  geom_bar(stat="identity",color="red",fill="black",show.legend = FALSE)+
  geom_col()+xlab("Rating")+
  geom_vline(xintercept = 3.51)+
  geom_text(aes(x=3.6,y="Comedy",label="mean rating",angle=90))
```



We group the data set by genres and analyse the top 50 most popular and the average ratings that the movies associated with these genres received.

We can see that a substantial proportion of these these genres have received average or higher than average ratings. Not only yhe popular genres are being rated by more no of movie goers but also the movies associated with these popular genres are receiving higher ratings.

6. Methods and Analysis

6.1 Modeling Approach

The edx data set was split into train and test data sets. train set was exclusively used to train the model and the test set was used to evaluate the performance of the model. 10% of the edx data set was carved out as test set so that there is as much data as possible to train.

```r
set.seed(1, sample.kind = "Rounding")
test_index<-createDataPartition(edx$rating,times=1,p=0.1, list=FALSE)
train_set<-edx[-test_index,]
temp_1<-edx[test_index,]

#Ensure userId and MovieId in the test set are also in train set

test_set<-temp_1%>%semi_join(train_set,by="movieId")%>%
  semi_join(train_set,by="userId")

# Add rows removed from test set back into train set

removed<-anti_join(temp_1,test_set)
train_set<-rbind(train_set,removed)

rm(removed,temp_1,test_index)
```

Now we define the loss function. Root Mean Square Error(RMSE) is used to estimate the performance of the model that we build. RMSE details the standard deviation between the predicted and estimated values.

RMSE is defined as

$$RMSE = \sqrt{\frac{1}{N}\sum_{u,i}(\hat{y}_{u,i} - y_{u,i})^2}$$

$y_{u,i}$ is the rating for movie i by user u and $\hat{y}_{u,i}$ is the predicted value

N is the no of user movie combinations.

RMSE function is defined below that calculates the root Mean square error value given the true and predicted rating values

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

6.2 Baseline Average model

We begin our analysis by using the simplest model that predicts the same rating for all movies regardless of the user that rated movie. For this model,we calculate the mean rating of our train data set. The model could be written as

$$Y_{u,i} = \mu + \epsilon_{u,i}$$

μ is the mean rating and $\epsilon_{u,i}$ the independent error sampled from the same distribution centered at 0.

First, the mean rating for the train_set is calculated

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

Now the RMSE for the baseline model is calculated

```
basic_RMSE<-RMSE(mu,test_set$rating)
basic_RMSE
```

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

A Table is prepared depicting the RMSE values calculated with different models and effects.

```
rmse_calculations<-data.frame(method=
"baseline average model", RMSE=basic_RMSE)
rmse_calculations%>%knitr::kable("pipe")
```

method	RMSE
baseline average model	1.060054

We can see that the RMSE with just the average is greater than one so our model definitely needs improvement. We would now add attributes that we analyzed and see if we are able to lower our RMSE.

6.3 Age effect

Earlier we analyzed the distribution of ratings wrt age of movie. Now we try to see if inclusion of age effect lowers RMSE. If the age effect is considered the model would look like

$$Y_{u,i} = \mu + b\_a + \epsilon_{u,i}$$

here we have considered b_a as the bias due to the age of movie. We call it the age effect.

Now we predict the ratings using the age effect. We could use the lm() function in the following way:-

fit<-lm(rating~age,data=train_set)

But the function lm() is very slow in this case because of the size of the data set and due to the computational resources, it was not used. The least square estimate was calculated in the following way:-

```
age_avgs<-train_set%>%group_by(age)%>%summarize(b_a=mean(rating-mu))
predicted_ratings<-mu+test_set%>%left_join(age_avgs,by="age")%>%pull(b_a)
age_effect_rmse<-RMSE(predicted_ratings,test_set$rating)
rmse_calculations<-rbind(rmse_calculations,data.frame(method="Age effect",RMSE=age_effect_rmse))
rmse_calculations%>%knitr::kable("pipe")
```

method	RMSE
baseline average model	1.060054
Age effect	1.051399

The inclusion of age effect didnt lower the RMSE to below one, so going forward we will drop the age effect from the model.

6.4 Movie effect

From the visualization of distribution of ratings for movies we noted that different movies were rated differently and some movies received higher ratings than others.

If we include the movie bias than our model would look like :-

$$Y_{u,i} = \mu + b\_i + \epsilon_{u,i}$$

b_i is the movie effect or movie bias.

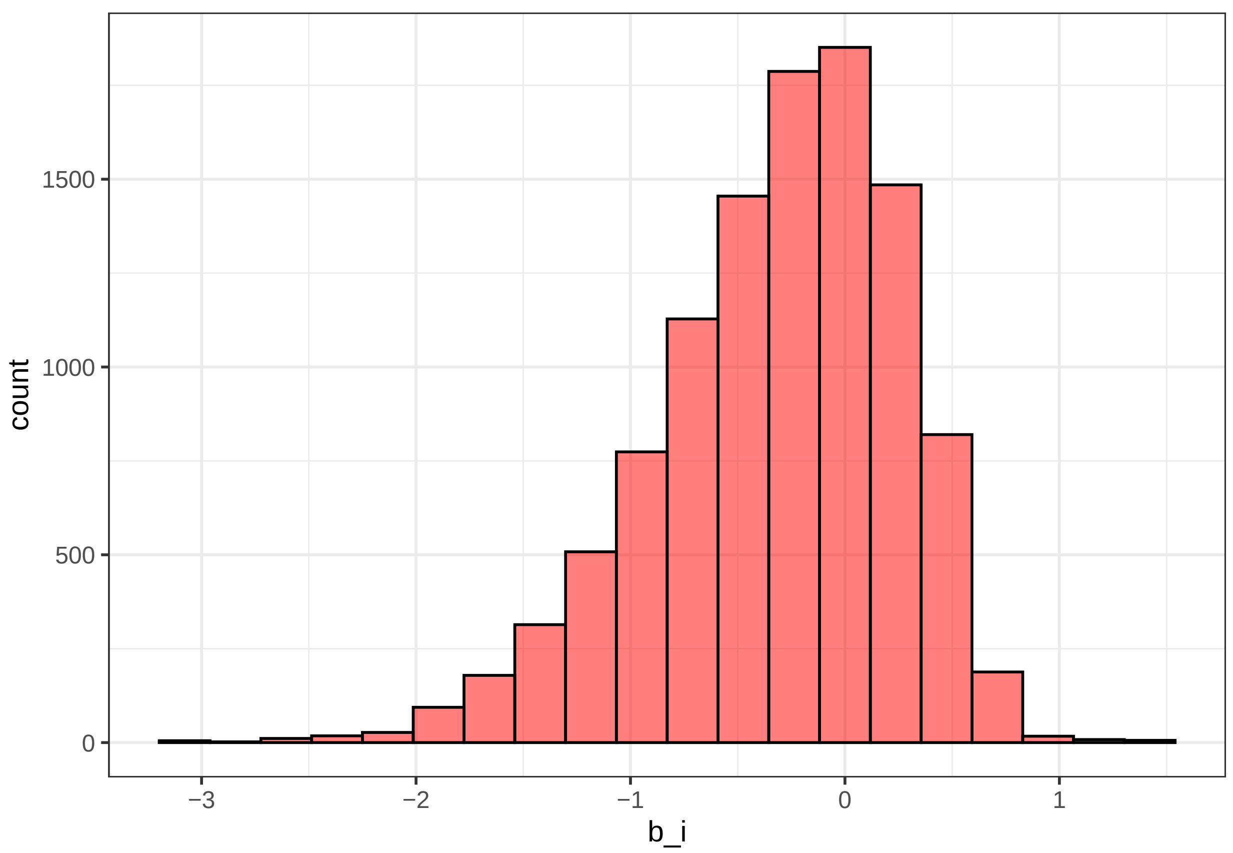
Again using the lm() function in the following way-

fit<-lm(rating~as.factor(movieId),data=train_Set) is very slow and hence the estimates were computed as:-

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

if we plot the distribution of estimates b_i, we find that they vary substantially

```
movie_averages%>%ggplot(aes(b_i))+
  geom_histogram(bins = 20,alpha=0.5,color="black",fill="red")+
  theme_bw()
```



Now we predict the ratings by incorporating the movie effect and calculate the RMSE

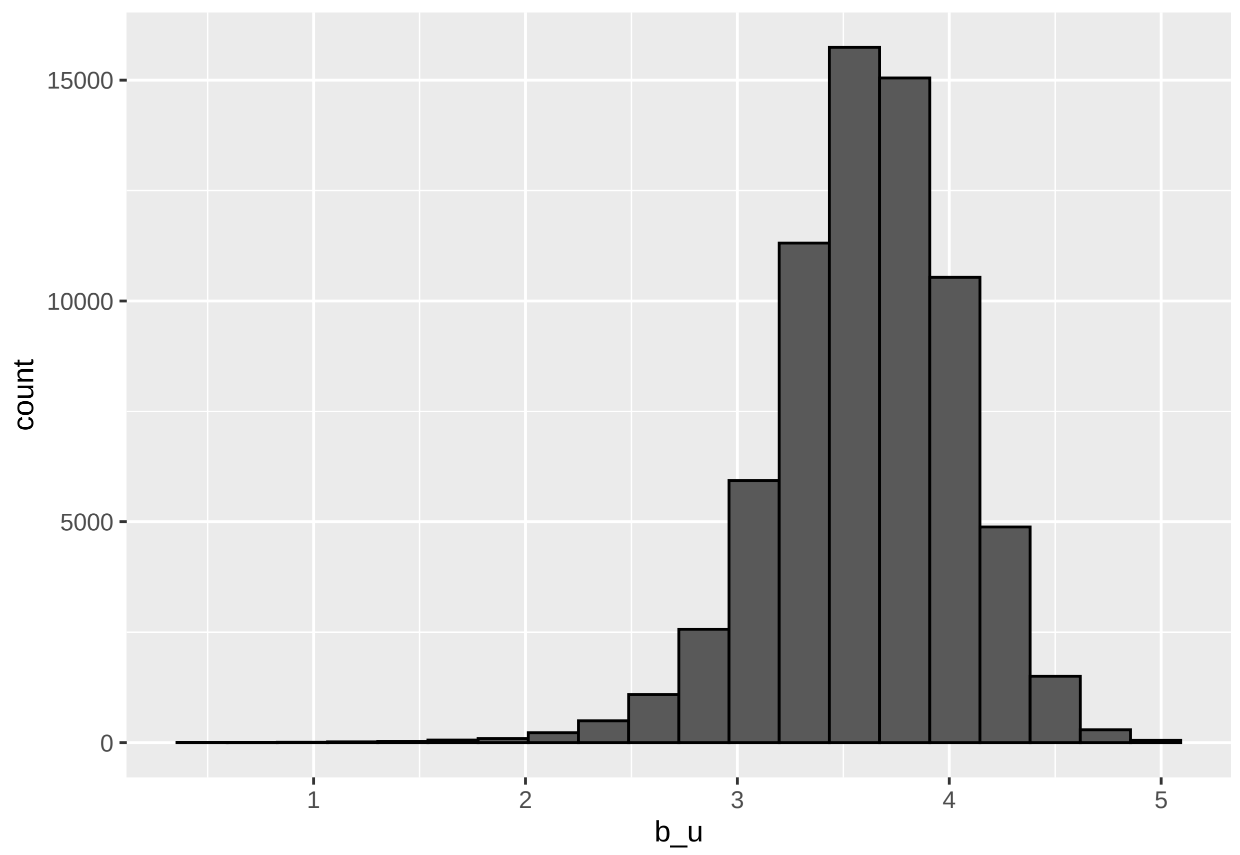
```
predicted_ratings<-mu+test_set%>%
  left_join(movie_averages,by="movieId")%>%
  pull(b_i)

movie_effect_rmse<-RMSE(predicted_ratings,test_set$rating)
rmse_calculations<-rbind(rmse_calculations,data.frame(method=
"Movie effect",RMSE=movie_effect_rmse))
rmse_calculations%>%knitr::kable("pipe")
```

method	RMSE
baseline average model	1.0600537
Age effect	1.0513991
Movie effect	0.9429615

The inclusion of movie effect lowered our RMSE. But we are still way off of our target RMSE as per the goal of project.Now we try to see if we can lower the RMSE by incorporating the other effects.

6.5 User Effect

```
train_set%>%group_by(userId)%>%
  summarize(b_u=mean(rating))%>%
  ggplot(aes(b_u))+geom_histogram(bins=20,color='black')
```



If we see the plot of average rating for users, we infer that there is substantial variability across users. Some users provide higher rating for movie while other users rate the same movie lower. Now if a user that generally gives low rating to movies rates an otherwise popular movie which received above average rating, the effects counter each other.

If we include the user effect to our model it would appear as:-

$$Y_{u,i} = \mu + b\_i + b\_u + \epsilon_{u,i}$$

Here b_u is the user bias or user effect.

lm() function not used because of the computational resources, it was very slow. We can calculate the user bias as

$y_{u,i} - \mu - b\_i$.

Now we predict the ratings and calculate RMSE using the test_set by incorporating user effect

```r
user_averages<-train_set%>%
  left_join(movie_averages,by="movieId")%>%
  group_by(userId)%>%summarize(b_u=mean(rating-mu-b_i))

predicted_ratings<-test_set%>%
  left_join(movie_averages,by="movieId")%>%
  left_join(user_averages,by="userId")%>%
  mutate(pred=mu+b_i+b_u)%>%pull(pred)

movie_user_effect_rmse<-RMSE(predicted_ratings,test_set$rating)
```

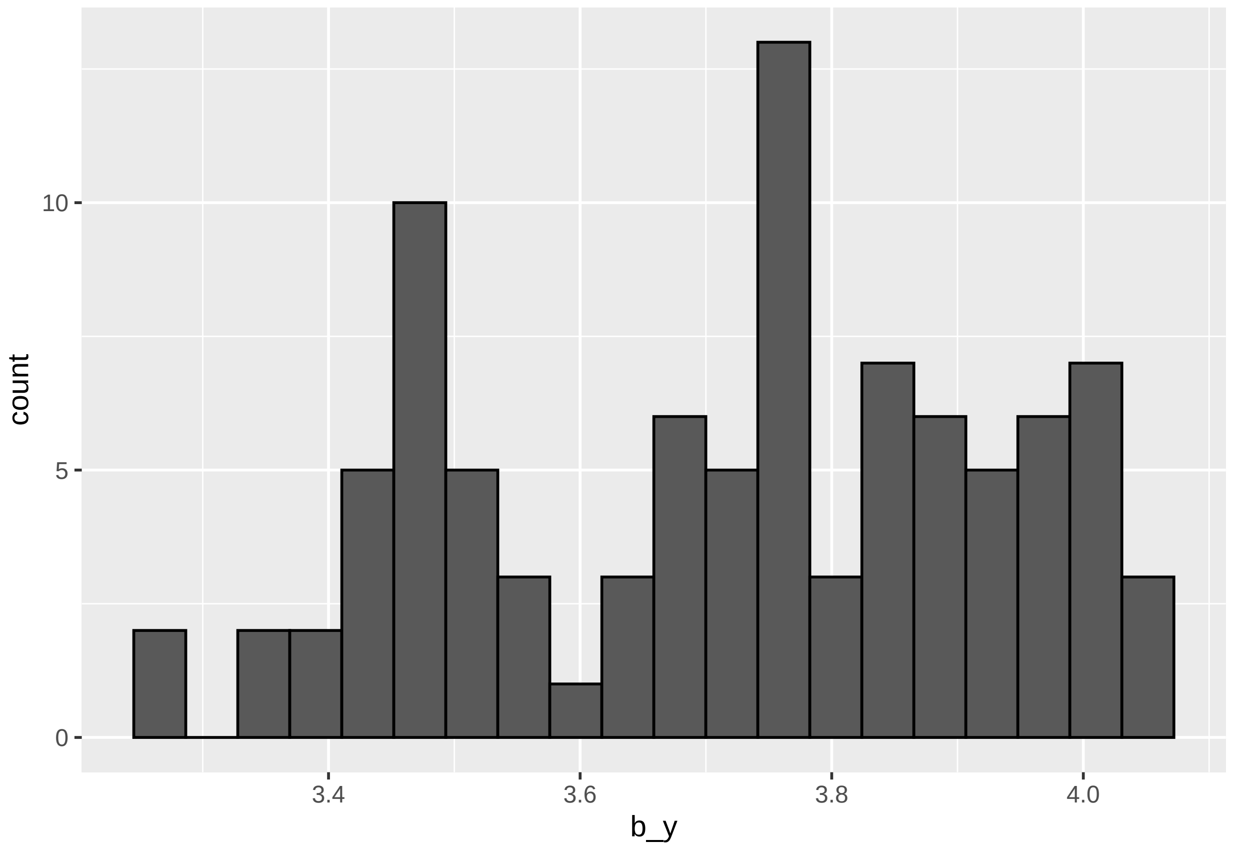
```r
rmse_calculations<-rbind(rmse_calculations,data.frame(method
="Movie+User effect",RMSE=movie_user_effect_rmse))

rmse_calculations%>%knitr::kable("pipe")
```

method	RMSE
baseline average model	1.0600537
Age effect	1.0513991
Movie effect	0.9429615
Movie+User effect	0.8646843

We are able to substantially lower our RMSE using the movie and user effect. But we are still away from the target RMSE. We will now try to see if effects of other attributes such as year(release year of movie) and genres lower RMSE.

6.5 Year Effect

```r
train_set%>%group_by(year)%>%
  summarize(b_y=mean(rating))%>%
  ggplot(aes(b_y))+geom_histogram(bins=20,color="black")
```



We had earlier visualized that movies released in recent years have received more ratings. If we also see the average ratings for release year of movies, we see that movies received in some years received higher ratings.

If we include the release year effect to our model it would appear as:-

$$Y_{u,i} = \mu + b\_i + b\_u + b\_y + \epsilon_{u,i}$$

Here b_y is the bias due to release year or year effect.

Now we predict the ratings using movie, user and year effect and calculate the RMSE.

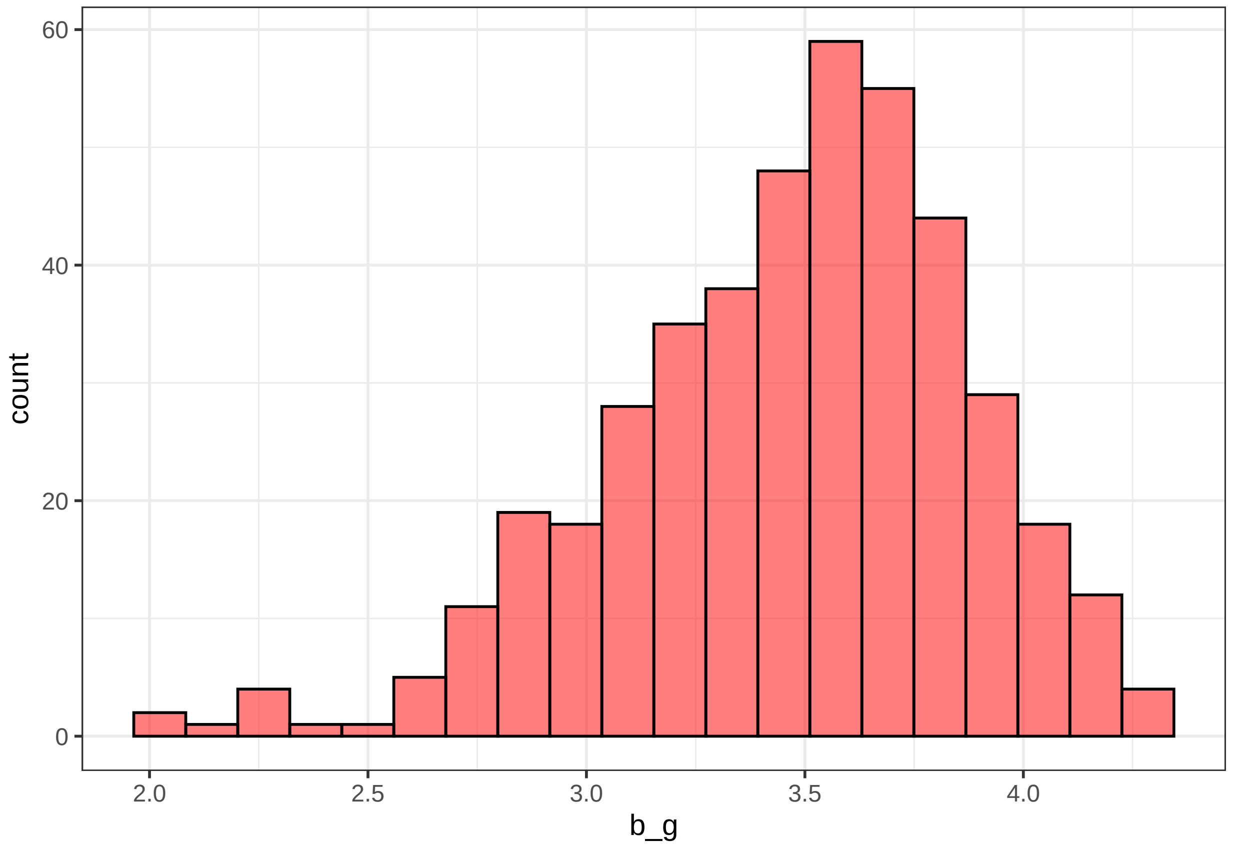
```r
year_averages<-train_set%>%
  left_join(movie_averages,by="movieId")%>%
  left_join(user_averages,by="userId")%>%
  group_by(year)%>%summarize(b_y=mean(rating-mu-b_i-b_u))

predicted_ratings<-test_set%>%
  left_join(movie_averages,by="movieId")%>%
  left_join(user_averages,by="userId")%>%
  left_join(year_averages,by="year")%>%
  mutate(pred_year=mu+b_i+b_u+b_y)%>%
  pull(pred_year)

movie_user_year_effect_rmse<-RMSE(predicted_ratings,test_set$rating)
rmse_calculations<-rbind(rmse_calculations,data.frame(method=
"Movie+User+Year effect",RMSE=movie_user_year_effect_rmse))
rmse_calculations%>%knitr::kable("pipe")
```

method	RMSE
baseline average model	1.0600537
Age effect	1.0513991
Movie effect	0.9429615
Movie+User effect	0.8646843
Movie+User+Year effect	0.8643301

We notice only a marginal reduction in RMSE. We will now include the genre effect in our model.

6.6 Genres Effect

```r
train_set%>%group_by(genres)%>%
  filter(n()>1000)%>%summarise(b_g=mean(rating))%>%
  ggplot(aes(b_g))+
  geom_histogram(bins=20,alpha=0.5,color="black",fill="red")+
  theme_bw()
```



The genres combination as it appears in the train set is considered as unique genre to which the movie is associated with. Analyzing the average ratings for the genres suggest that movies associated with some genres received higher ratings.

If we include the release year effect to our model it would appear as:-

$$Y_{u,i} = \mu + b\_i + b\_u + b\_y + b\_g + \epsilon_{u,i}$$

Here b_g is the bias due to release year or year effect.

Now we predict the ratings using movie, user,year and genres effect and calculate the RMSE.

```r
genres_averages<-train_set%>%
  left_join(movie_averages,by="movieId")%>%
  left_join(user_averages,by="userId")%>%
  left_join(year_averages,by="year")%>%
  group_by(genres)%>%
  summarize(b_g=mean(rating-mu-b_i-b_u-b_y))

predicted_ratings<-test_set%>%left_join(movie_averages,by="movieId")%>%
  left_join(user_averages,by="userId")%>%
  left_join(year_averages,by="year")%>%
  left_join(genres_averages,by="genres")%>%
  mutate(pred_genre=mu+b_i+b_u+b_y+b_g)%>%
  pull(pred_genre)

movie_user_year_genre_effect_rmse<-RMSE(predicted_ratings,test_set$rating)
rmse_calculations<-rbind(rmse_calculations,data.frame(method=
"Movie+User+Year+Genre effect",RMSE=movie_user_year_genre_effect_rmse))
rmse_calculations%>%knitr::kable("pipe")
```

method	RMSE
baseline average model	1.0600537
Age effect	1.0513991
Movie effect	0.9429615
Movie+User effect	0.8646843
Movie+User+Year effect	0.8643301
Movie+User+Year+Genre effect	0.8640801

The addition of genre effect marginally reduced the RMSE further.

6.7 Regularization effect

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

If we revisit the movie effect, we can see that the best movies are rated by very few users

```
train_set%>%count(movieId)%>%
  left_join(movie_averages,by="movieId")%>%
  left_join(movie_titles,by="movieId")%>%
  arrange(desc(b_i))%>%
  slice(1:10)%>%
  pull(n)
```

```
##  [1] 1 1 1 1 1 1 4 2 4 4
```

Similarly the supposed worst movies are rated by very users.

```
train_set%>%count(movieId)%>%
  left_join(movie_averages,by="movieId")%>%
  left_join(movie_titles,by="movieId")%>%
  arrange(b_i)%>%
  slice(1:10)%>%
  pull(n)
```

```
##  [1]   1   1   1   1   2  47  30 183  11   1
```

The estimates obtained with a small sample size users rating the movies could lead to uncertainty and incorrect predictions. Therefore we use the concept of Regularization to penalize large estimates obtained with small sample sizes.

In the Regularization model, instead of minimizing the least squares equation as was done earlier, equation containing a penalty term will be minimized.

If we just consider the movie effect the equation with the penalty term would be:-

$$\frac{1}{N}\sum_{u,i}(y_{u,i} - \mu - b\_i^2 + \lambda\sum_{i} b_i^2$$

If we include the other attributes whose effects have been considered than the equation to minimize would be :-

$$\frac{1}{N}\sum_{u,i}(y_{u,i} - \mu - b\_i - b\_u - b\_y - b\_g)^2 + \lambda(\sum_{i} b\_i^2 + \sum_{u} b\_u^2 + \sum_{y} b\_y^2 + \sum_{g} b\_g^2)$$

6.8 k fold cross validation to tune lambda parameter

We would optimize the parameter λ. To prevent overtraining which can occur if the same train set is used to optimize and evaluate. Therefor we will use k fold cross validation and split train set into k non overlapping sets.

The other task is to pick the value of k. Although large values of k are desirable as it improves the optimization of the parameter and accuracy of the model. But given the large size of the train set, larger values of k will take more computational time. Popular choices of k are 5 and 10 and in our k fold cross validation we use k=5.

We split our train set into 5 non overlapping sets or parts. Every time, four parts of the train set would be combined as the training set(train_set_cv) and the fifth part(test_set_cv) would be used as test set to calculate the RMSE. For a range of lambda values, model will be built by using train_set_cv and the test_set_cv will be used to calculate RMSE. We will obtain a set of RMSEs for each value of lambda.

There will 5 different combinations of train_set_cv and test_set_cv obtained from cross validation on the original train set. For each lambda value there will be 5 RMSEs and mean of these 5 RMSE values will be taken. Then the minimal RMSE will be taken and the corresponding λ that minimizes the RMSE will be selected as optimum lambda.

Once the optimum lambda has been obtained using cross validation and Regularization, model would be evaluated on the test_set.

For cross validation, we have used the createFolds function of the caret package. The createFolds function splits the data into k groups.

The function is written as createFolds(y,k,list=TRUE,returnTrain=TRUE)

The arguments of the function are:-

y: A vector of outcomes. In our case it would train_set$rating

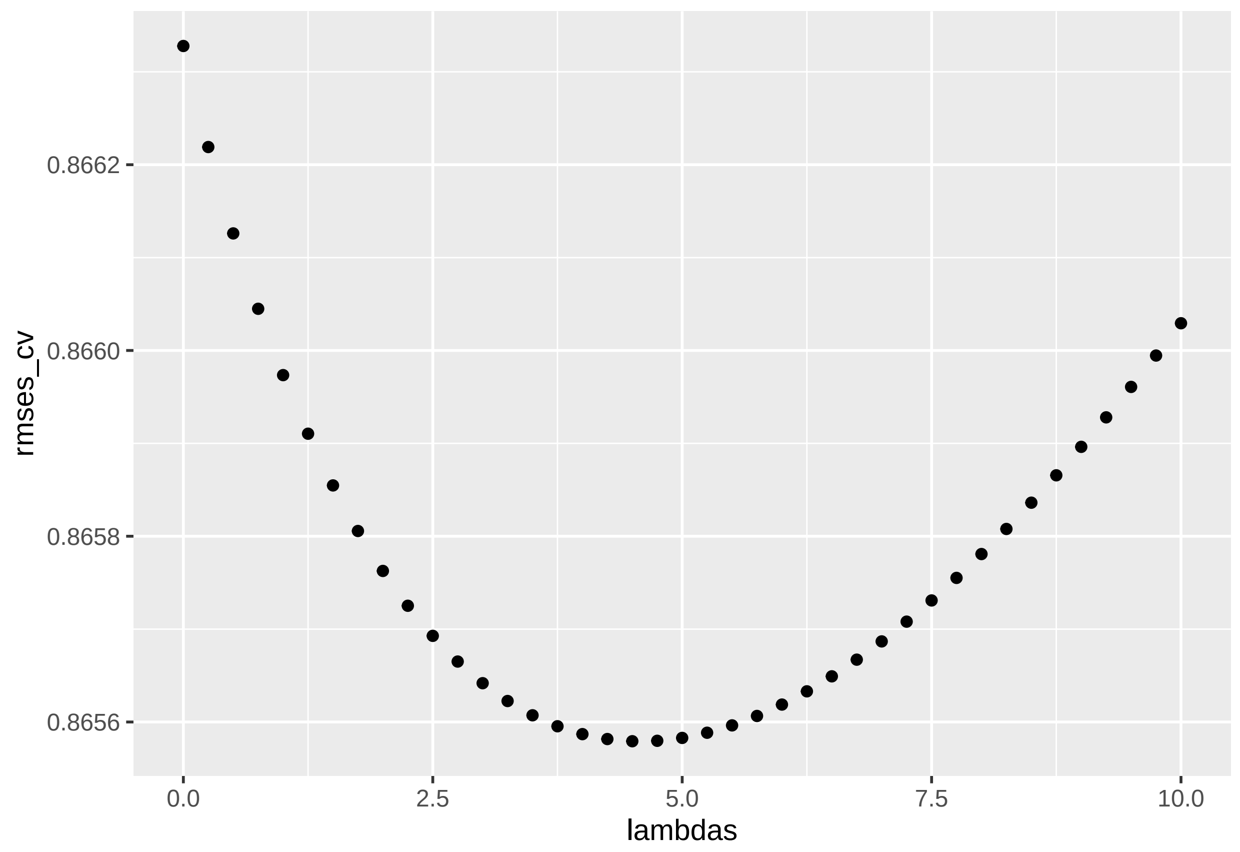
k: An integer for the number of folds. In our case its k=5

returnTrain: A Logical. When TRUE, the values returned are sample positions corresponding to the data used during training. This argument only works in conjunction with list=TRUE

```r
set.seed(1,sample.kind = "Rounding")
lambdas<-seq(0,10,.25)

###Split the train_set into 5 parts##

folds<-createFolds(train_set$rating,k=5,returnTrain = T)

###A matrix is defined to store the results of cross validation##

rmses<- matrix(nrow=5, ncol=length(lambdas))

ks<- seq(1,5,1)

### Perform cross validation for the valuees of k###
for(k in ks){
  train_set_cv<-train_set[folds[[k]],]
  temp<-train_set[-folds[[k]],]


#make sure userId and movieId in the test set are also in train set
  test_set_cv<-temp%>%
    semi_join(train_set_cv,by="movieId")%>%
    semi_join(train_set_cv,by="userId")

#Add rows removed from test set to train set

  removed<-anti_join(temp,test_set_cv)
  train_set_cv<-rbind(train_set_cv,removed)

  mu<-mean(train_set_cv$rating)

  rmses[k,]<-sapply(lambdas,function(l){

    b_i<-train_set_cv%>%
      group_by(movieId)%>%
      summarize(b_i=sum(rating-mu)/(n()+l))

    b_u<-train_set_cv%>%
     left_join(b_i,by="movieId")%>%
      group_by(userId)%>%
      summarize(b_u=sum(rating-mu-b_i)/(n()+l))

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

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

    predicted_ratings<-test_set_cv%>%
      left_join(b_i,by="movieId")%>%
      left_join(b_u,by="userId")%>%
      left_join(b_y,by="year")%>%
      left_join(b_g,by="genres")%>%
      mutate(y_hat=mu+b_i+b_u+b_y+b_g)%>%
      pull(y_hat)

    return(RMSE(predicted_ratings,test_set_cv$rating))

  })

}


rmses_cv<-colMeans(rmses)
```

We plot the range of different values of lambdas and the corresponding RMSE. (Mean RMSE of the set of RMSEs obtained for each λ)

```r
qplot(lambdas,rmses_cv)
```



The optimum lambda would the value the minimizes RMSE

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

```
## [1] 4.5
```

Now the lambda_opt will be used to evaluate the performance of Regularization model on the test_set

```r
#####Evaluation with test set using optimal lambda#####
mu<-mean(train_set$rating)
mu
```

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

```r
movie_reg_averages<-train_set%>%
  group_by(movieId)%>%
  summarize(b_i=sum(rating-mu)/(n()+lambda_opt))

user_reg_averages<-train_set%>%
  left_join(movie_reg_averages,by="movieId")%>%
  group_by(userId)%>%
  summarize(b_u=sum(rating-mu-b_i)/(n()+lambda_opt))

year_reg_averages<-train_set%>%
  left_join(movie_reg_averages,by="movieId")%>%
  left_join(user_reg_averages,by="userId")%>%
  group_by(year)%>%
  summarize(b_y=sum(rating-mu-b_i-b_u)/(n()+lambda_opt))


genre_reg_averages<-train_set%>%
  left_join(movie_reg_averages,by="movieId")%>%
  left_join(user_reg_averages,by="userId")%>%
  left_join(year_reg_averages,by="year")%>%
  group_by(genres)%>%
  summarize(b_g=sum(rating-mu-b_i-b_u-b_y)/(n()+lambda_opt))

predicted_ratings<-test_set%>%
  left_join(movie_reg_averages, by="movieId")%>%
  left_join(user_reg_averages,by="userId")%>%
  left_join(year_reg_averages,by="year")%>%
  left_join(genre_reg_averages,by="genres")%>%
  mutate(pred=mu+b_i+b_u+b_y+b_g)%>%
  pull(pred)
```

Now, the RMSE value is calculated using the predictions and test_set$rating.

```
Reg_cv_rmse<-RMSE(predicted_ratings,test_set$rating)
rmse_calculations<-rbind(rmse_calculations,data.frame(method=
"Regularisation with 5-fold CV-Movie+User+Year+Genre",RMSE=Reg_cv_rmse))
rmse_calculations%>%knitr::kable("pipe")
```

method	RMSE
baseline average model	1.0600537
Age effect	1.0513991
Movie effect	0.9429615
Movie+User effect	0.8646843
Movie+User+Year effect	0.8643301
Movie+User+Year+Genre effect	0.8640801
Regularisation with 5-fold CV-Movie+User+Year+Genre	0.8636099

We can see that the RMSE is lowered.

7. Model Finalization and Prediction on Validation set.

The Regularization model using k-fold cross validation to optimize lambda minimized the RMSE. The Regularization model will be used as final model to predict ratings on the validation set using the the edx set for training the regularization model. We will use the four attributes whose effects were previously considered- Movie effect, User effect, Year effect and Genres effect.

Adding year, Rating year and Age columns to Validation set

```
validation<-validation%>%mutate(year=as.numeric(str_sub(title,-5,-2)))%>%
  mutate(rated_year=year(as_datetime(timestamp)))%>%
  mutate(age=rated_year-year)
```

Defining the mean rating for edx data set

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

We will ise edx data set to train the Regularization model using lamda_opt previously obtained.

```
movie_reg_averages<-edx%>%
  group_by(movieId)%>%
  summarize(b_i=sum(rating-mu_edx)/(n()+lambda_opt))

user_reg_averages<-edx%>%
  left_join(movie_reg_averages,by="movieId")%>%
  group_by(userId)%>%
  summarize(b_u=sum(rating-mu_edx-b_i)/(n()+lambda_opt))

year_reg_averages<-edx%>%
  left_join(movie_reg_averages,by="movieId")%>%
  left_join(user_reg_averages,by="userId")%>%
  group_by(year)%>%
  summarize(b_y=sum(rating-mu_edx-b_i-b_u)/(n()+lambda_opt))

genre_reg_averages<-edx%>%
  left_join(movie_reg_averages,by="movieId")%>%
  left_join(user_reg_averages,by="userId")%>%
  left_join(year_reg_averages,by="year")%>%
  group_by(genres)%>%
  summarize(b_g=sum(rating-mu_edx-b_i-b_u-b_y)/(n()+lambda_opt))
```

Now we make predictions on the validation set

```
predicted_ratings_val<-validation%>%
  left_join(movie_reg_averages, by="movieId")%>%
  left_join(user_reg_averages,by="userId")%>%
  left_join(year_reg_averages,by="year")%>%
  left_join(genre_reg_averages,by="genres")%>%
  mutate(pred_validation=mu+b_i+b_u+b_y+b_g)%>%
  pull(pred_validation)
```

Calculation the final RMSE using the validation$rating

```r
RMSE_valid<-RMSE(predicted_ratings_val,validation$rating)

rmse_calculations<-rbind(rmse_calculations,data.frame(method=
"Final Regularization on Validation with optimal Lambda",RMSE=RMSE_valid))
rmse_calculations%>%knitr::kable("pipe", caption = "RMSE Results Table")
```

Table 16: RMSE Results Table

method	RMSE
baseline average model	1.0600537
Age effect	1.0513991
Movie effect	0.9429615
Movie+User effect	0.8646843
Movie+User+Year effect	0.8643301
Movie+User+Year+Genre effect	0.8640801
Regularisation with 5-fold CV-Movie+User+Year+Genre	0.8636099
Final Regularization on Validation with optimal Lambda	0.8642542

The final RMSE calculated on validation set is

```r
print(RMSE_valid)
```

```
## [1] 0.8642542
```

8. Conclusion

Table 17: RMSE Results Table

method	RMSE
baseline average model	1.0600537
Age effect	1.0513991
Movie effect	0.9429615
Movie+User effect	0.8646843
Movie+User+Year effect	0.8643301
Movie+User+Year+Genre effect	0.8640801
Regularisation with 5-fold CV-Movie+User+Year+Genre	0.8636099
Final Regularization on Validation with optimal Lambda	0.8642542

We started our analysis by visualizing the distribution of ratings for various effects such as Age, MovieId, userId, Release year, Rating year and genres associated with movie.

We first calculated RMSE using the baseline model using just the mean ratings. There onward, we added movie effect, user effect, year (release of movie) and genres effect. We were able to substantially lower RMSE using movie and user effect and subsequent addition of year and genre effect provided marginal improvement in RMSE.

Then Regularization was performed using k fold cross validation with k=5 and parameter lambda was optimized. The Regularization model was evaluated with the optimized lambda value.

Regularization model was selected as the final model as it gave the lowest RMSE and was used to make predictions on validation data set and calculate the final RMSE-0.8642 which was lower than the target RMSE.

As expected RMSE with the validation set was less than the target RMSE but slightly more than the that obtained with the test set. We trust that model would be able to perform well when we predict for the unseen data.

9. Limitations

Some of the machine learning algorithms were computationally expensive(for eg. the lm() function, knn and random forest) and were taking a lot of time as the data set had about 10 million rows. More advanced machine Learning models could not be deployed due to the computational and technological limitations due the large size of the dataset. Also larger value of k such 10 or more could not be considered because it would have slowered the computational time and hence k=5 was considered.

10. References

a. https://rafalab.github.io/dsbook/large-datasets.html#a-first-model

b. https://rafalab.github.io/dsbook/cross-validation.html#k-fold-cross-validation

c. http://files.grouplens.org/datasets/movielens/ml-10m-README.html

d. https://rmarkdown.rstudio.com/articles_intro.html

e. https://www.vedantu.com/maths/root-mean-square

f. https://towardsdatascience.com/

g. https://cran.r-project.org/web/packages/caret/caret.pdf

h. https://www.rdocumentation.org/packages/stringr/versions/1.4.0/topics/str_sub

i. https://www.rdocumentation.org/packages/lubridate/versions/1.7.9/topics/as_date