MovieLens Report (Harvard PH125.9)

Humayun Akram

28 February 2019

1. **Introduction**

1.1 **Overview**

Recommendation systems make suggestions about artifacts to a user. For instance, they may predict whether a user would be interested in seeing a particular movie. Social recomendation methods collect ratings of artifacts from many individuals, and use different techniques to make recommendations to a user concerning new artifacts (Basu, Hirsh & Cohen, 1998).

Recommendation systems are currently being used in various areas areas like social tags, news, movies, boos etc. and many Fortune 500 companies are using recommendations systems to evaluate performance of differnet products. Generally, star ratings are used to rank any particular product which usually ranges from 0 to 5 star, where 0 indicates least liked while 5 indicating most loved item.

As part of first project in Harvard PH125.9 Capstone course, we will be building a movie recommendation system that will predict ratings for a sample of users based on trained data model. At the end, we will verify the performance of our predictions using RMSE as a metric.

1.2 **DataSet**

This project is based on 'MovieLens' dataset which will be used to create a recommender system. The dataset is available at below location:

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

1.3 **Target**

The target is to develop a machine learning algorithm that takes input from provided training subset and predict movie ratings on validation dataset.

The focus is on RMSE of the algorithm which will be used to evaluate as how close movie predictions are to the true values in the validation set.

1.4 **Key Steps**

In this project, we will use the **10M records** version of Movielens dataset. Major steps for this project is detailed as below:

- Load the data and do initial exploration
- Insight analysis and vizualization
- Build 4 models based on the `edx` dataset
- Validate the final model by computing RMSEs for `validation` dataset.

2. **Analysis**

2.1 **Data Loading & Exploration**

We will start off by loading the data code provided by the course page.

Once loaded, we can see that `edx` dataset has **9000055** rows and **6** columns.
While, `validation` dataset has **999999** rows and **6** columns.

2.1.1 **Data Check**

Let's have an overview of dataset and verify if it has any missing/null values

```
######################################################################
######################### Data Check #################################
######################################################################

#Rows and columns of edx
dim(edx)
```

```
## [1] 9000055       6
```

```
#Unique Movies & Unique Users
n_distinct(edx$userId)
```

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

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

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

```
# Checking edx missing values
any(is.na(edx))
```

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

It can be seen that there are no missing values in dataset. Similarly, there are **69878** different users and **10677** different movies in the edx dataset.

2.1.2 **Data Summarization**

Summarization shows that rating has uniform distribution with 50th percentile ranges between 3 & 4 rating. On the other hand, movieId appeared to be rightly skewed showing that some movies are rated more than others.

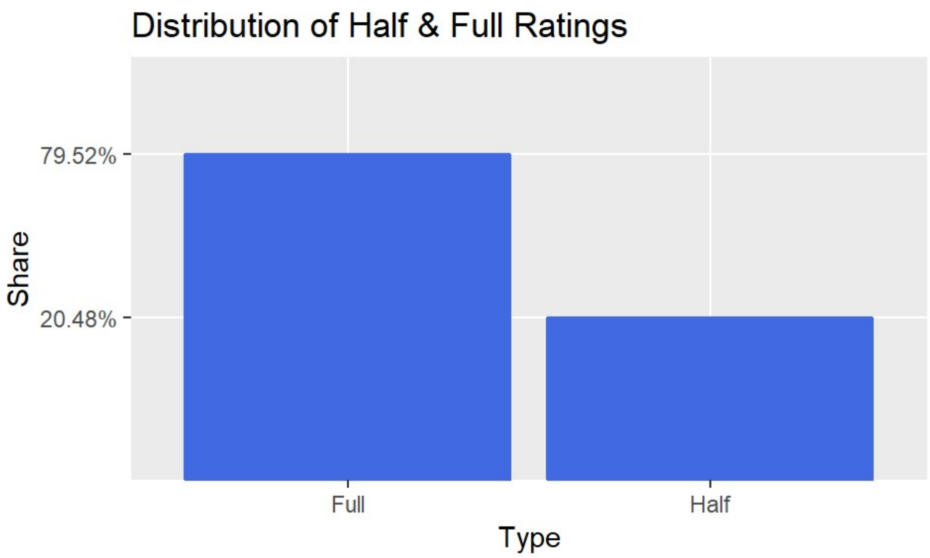
```
# Summary of the dataset
summary(edx)
```

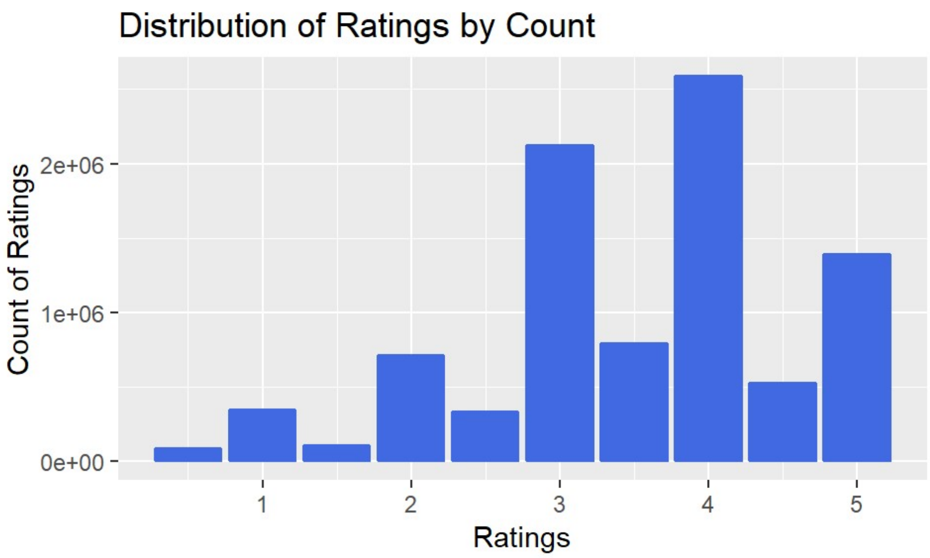
```
##      userId         movieId          rating        timestamp
##  Min.   :    1   Min.   :    1   Min.   :0.500   Min.   :7.897e+08
##  1st Qu.:18124   1st Qu.:  648   1st Qu.:3.000   1st Qu.:9.468e+08
##  Median :35738   Median : 1834   Median :4.000   Median :1.035e+09
##  Mean   :35870   Mean   : 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
##
##
##
```

2.2 **Insights Analysis & Vizualization**

2.2.1 **Ratings Distribution**

We can see that **79.5%** ratings are full-star ratings

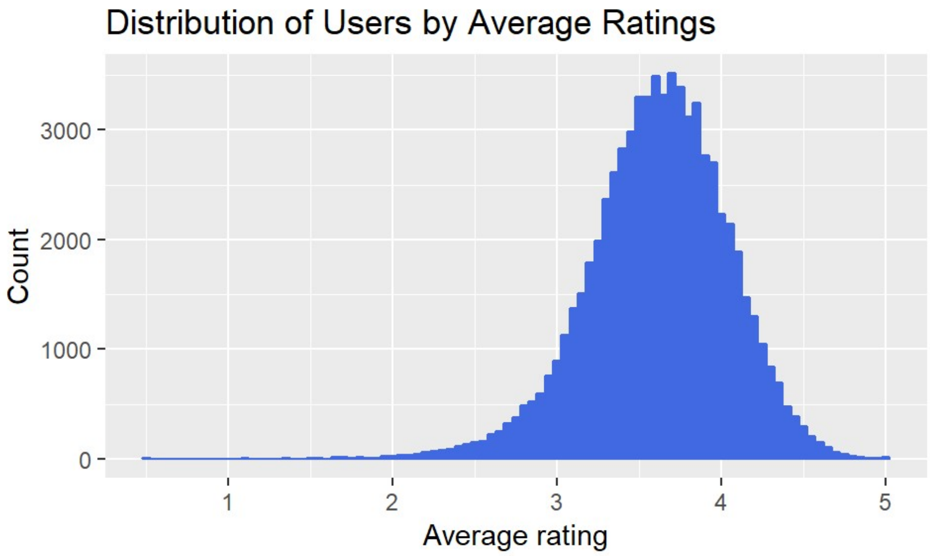


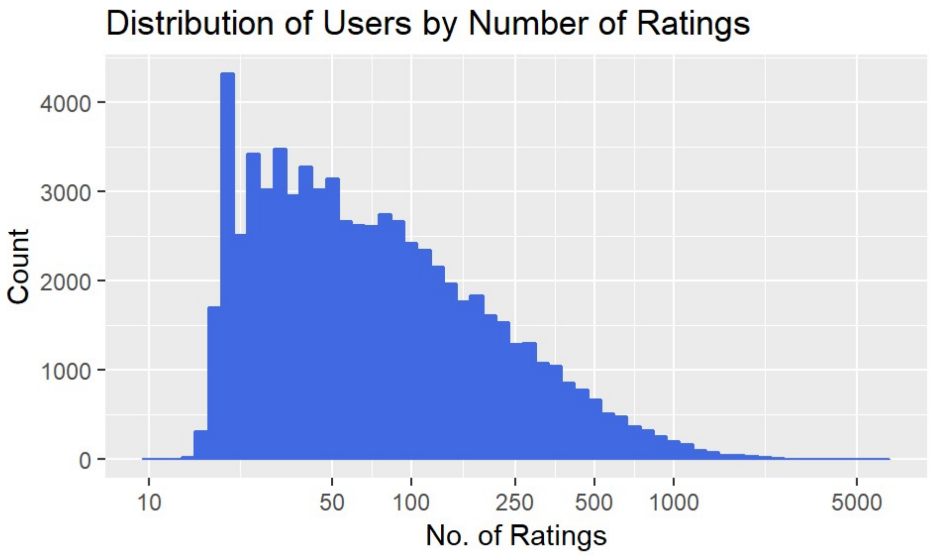


Rating distribution shows unbalance between whole & half star ratings as majority of ratings belong to **4** and **3** followed by **5**.

2.2.2 **Users Distribution**

For user distribution, we can see majority of users rate between 3 & 4 with an average of **3.6**

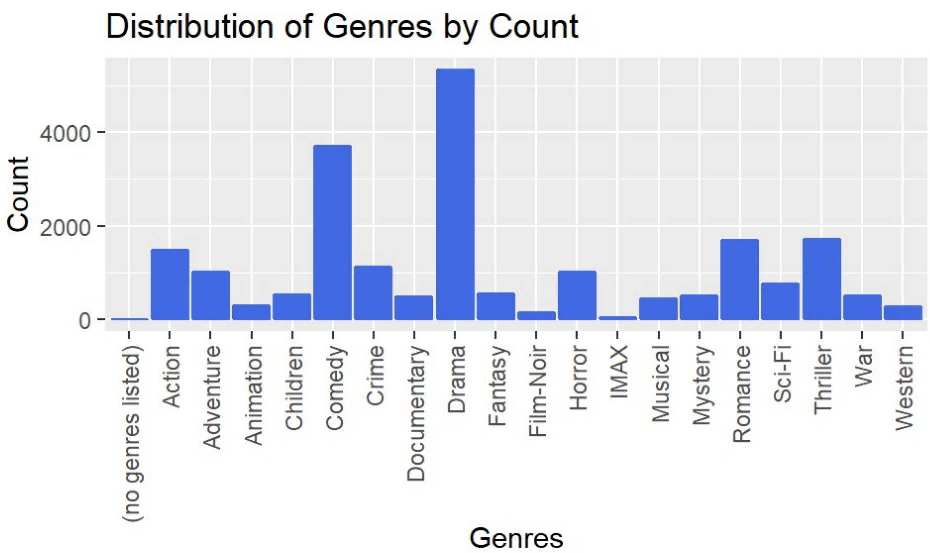




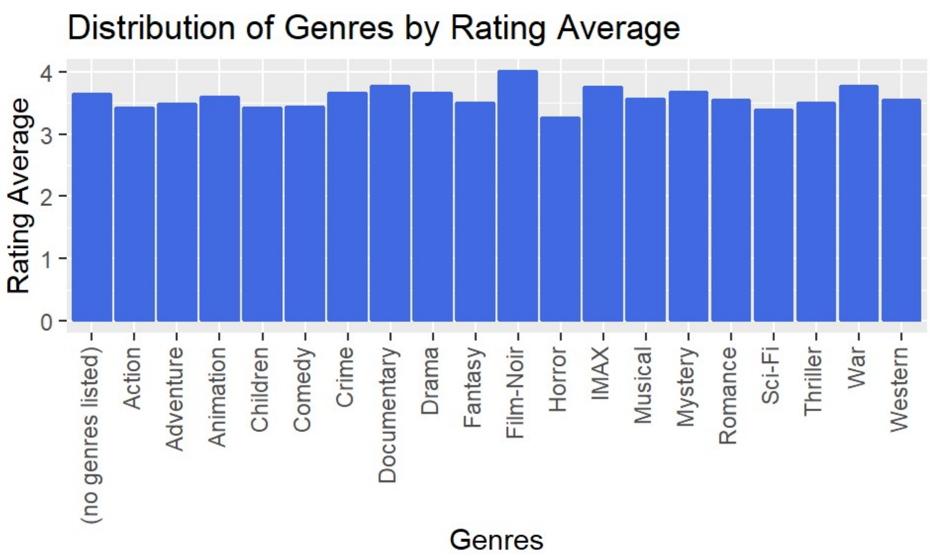
Number of ratings shows that few users rated more movies as compared to others

2.2.3 **Genres Distribution**

Genres distribution indicating that Drama and Comedy as most rated Genres as compared to others.

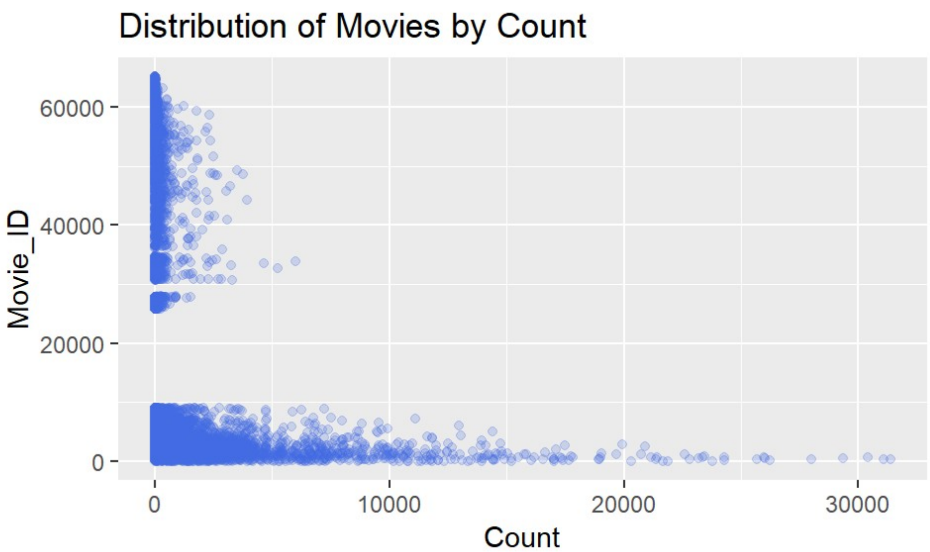


Further, genre effect can be seen by below variations in ratings where highly rated genres are not amongst highly reviewed.

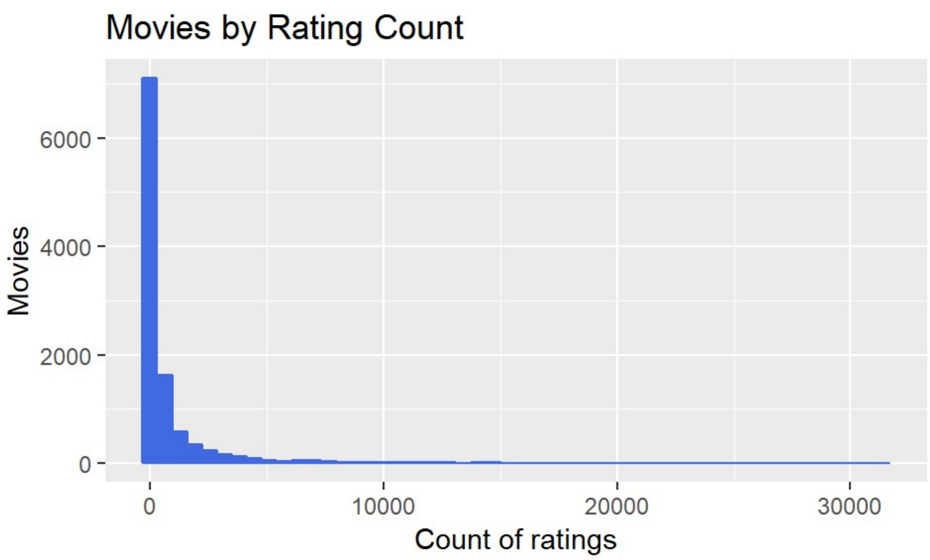


2.2.4 **Movies Distribution**

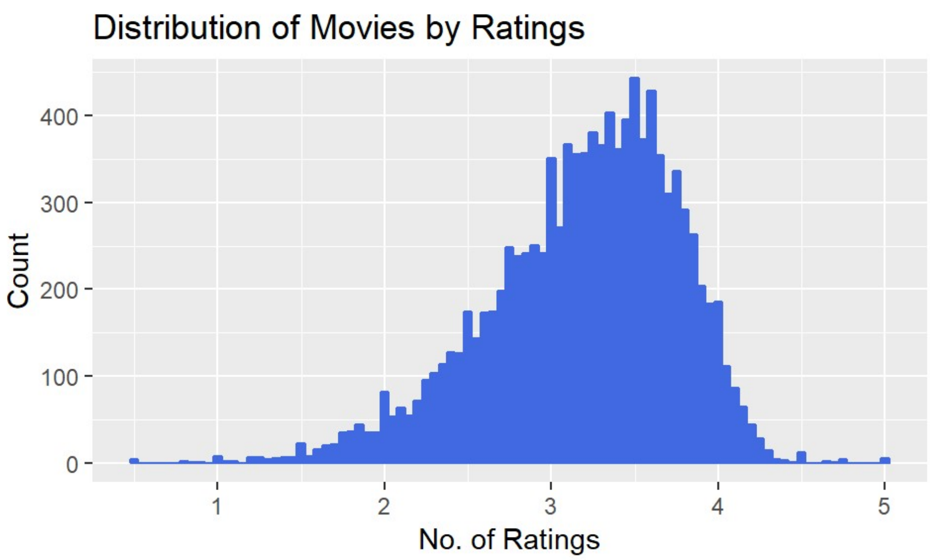
From movie distribution, it can be observed that some movies are rated more than others, which is normal behavior.



For **10677** movies, 75th percentile movies have less than ~550 ratings count



Similarly, movies that are reviewed more are tend to have better rating as well.

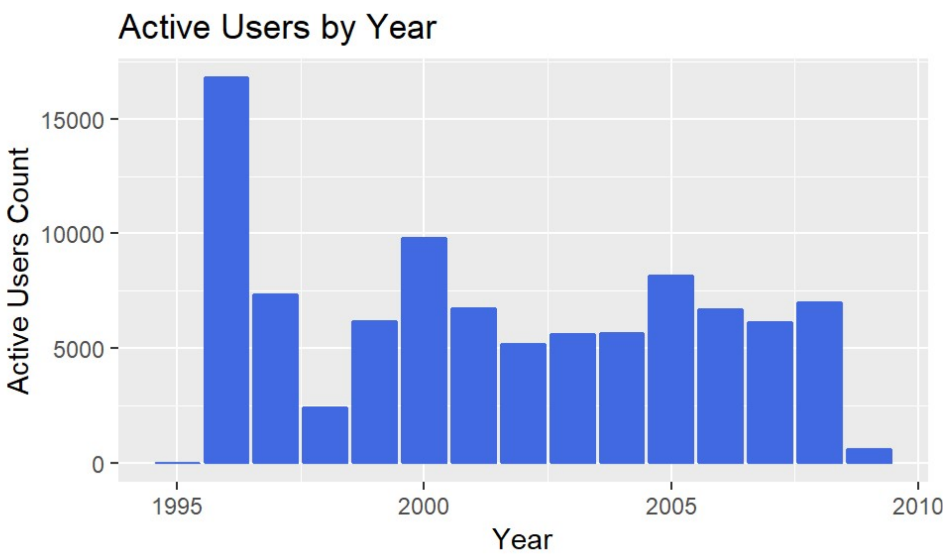


Pulp Fiction (1994) and Forrest Gump (1994) are the two most rated movies in the dataset

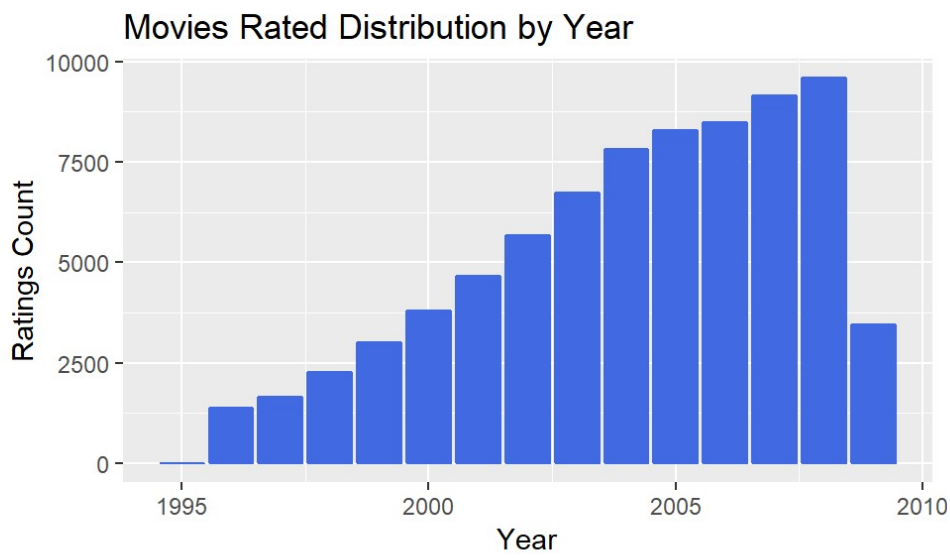
```
## # A tibble: 6 x 3
## # Groups:   movieId [6]
##   movieId title                              count
##     <dbl> <chr>                              <int>
## 1     296 Pulp Fiction (1994)                31362
## 2     356 Forrest Gump (1994)                31079
## 3     593 Silence of the Lambs, The (1991) 30382
## 4     480 Jurassic Park (1993)               29360
## 5     318 Shawshank Redemption, The (1994) 28015
## 6     110 Braveheart (1995)                  26212
```

2.2.5 **Years Distribution**

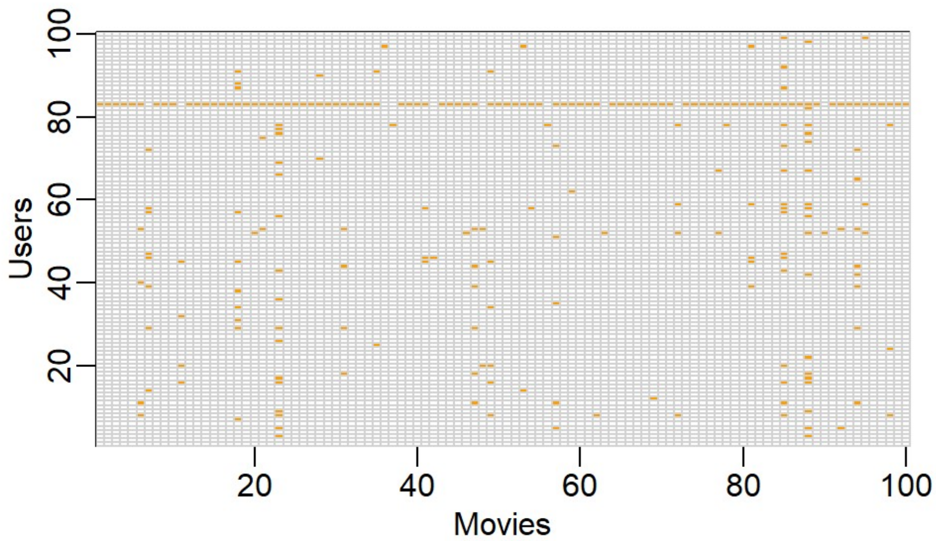
From year distribution, we can see that in 1996, active users count was maximum. On the other hand, lowest ones (i.e. in 1995 & 2009) are due to incomplete yearly data



It can also be observed that ratings for movies have grown, irrepective of number of active users, as more & more movies have become available for review/ratings.

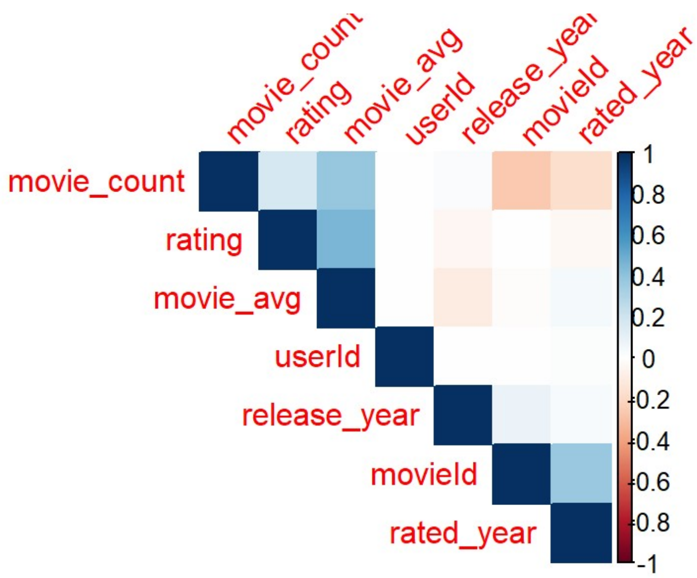


For sparsity, we can use course reference for given dataset to see sparsity for matrix of random data sample (100 movies x 100 users)



2.2.6 **Data Correlation**

Finally, the correlation matrix shows positive correlation between movie_avg against movie_count and rating. Similarly, rated_year also has correlation with movie being rated.



2.3 **Data Modeling**

For modeling, we will use randomly selected 20% of the **edx set** as test set.

2.3.1 **Simple Average Model**

Since RMSE would be used as typical error while predicting movie ratings, we start off by writing loss-function that computes RMSE.We define $y_{u,i}$ as the rating for movie i by user u and denote our prediction with $\hat{y}_{u,i}$ (Irizarry A. Rafael, 2018). The RMSE is then defined as:

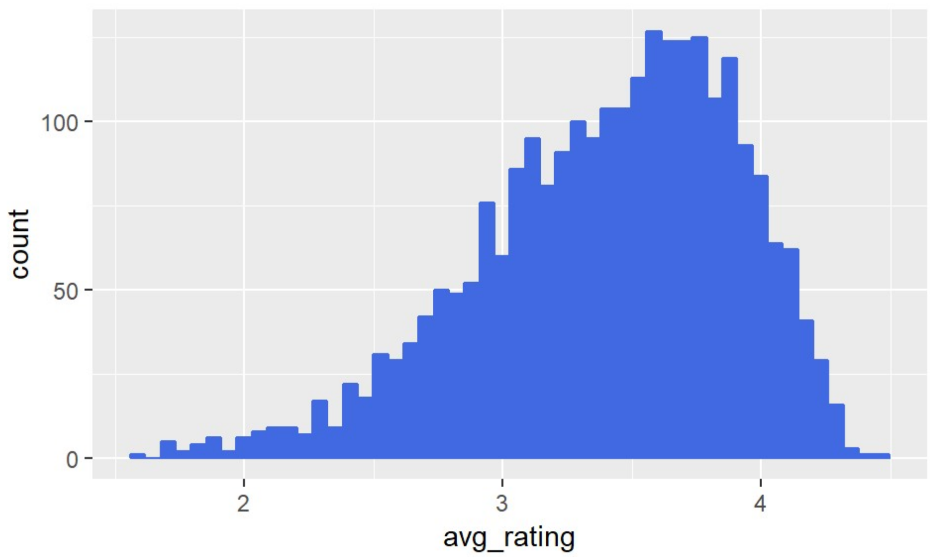
$$\mathrm{RMSE} = \sqrt{\frac{1}{N} \sum_{u,i} \left(\hat{y}_{u,i} - y_{u,i} \right)^2}$$

with N being the number of user/movie combinations and the sum occurring over all these combinations.

We can now use simple average as our baseline model to predict ratings on test set and evaluate the resulting RMSE, thus, we predict the same rating for all movies regardless of user. We can define a model that assumes the same rating for all movies and users with all the differences explained by random variation would look like this:

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

```
# Starting by simple possible recommendation, we predict average rating across all users fo
r all movies
mu <- mean(train_set$rating)


# Plotting average rating for movies rated at least 500 times.
train_set %>% group_by(movieId) %>%
  filter(n()>=500) %>%
  summarize(avg_rating = mean(rating)) %>%
  ggplot(aes(avg_rating)) +
  geom_histogram(bins = 50, colour = "royalblue", fill = "royalblue")
```




```
# Prediction on test set.
predictions <- rep(mu, nrow(test_set))

# RMSE for test set.
naive_rmse <- RMSE(test_set$rating, predictions)
```

The results will be stored in below table:

Method	Dataset	RMSE
Simple Average Model	**test_edx**	**1.059318**

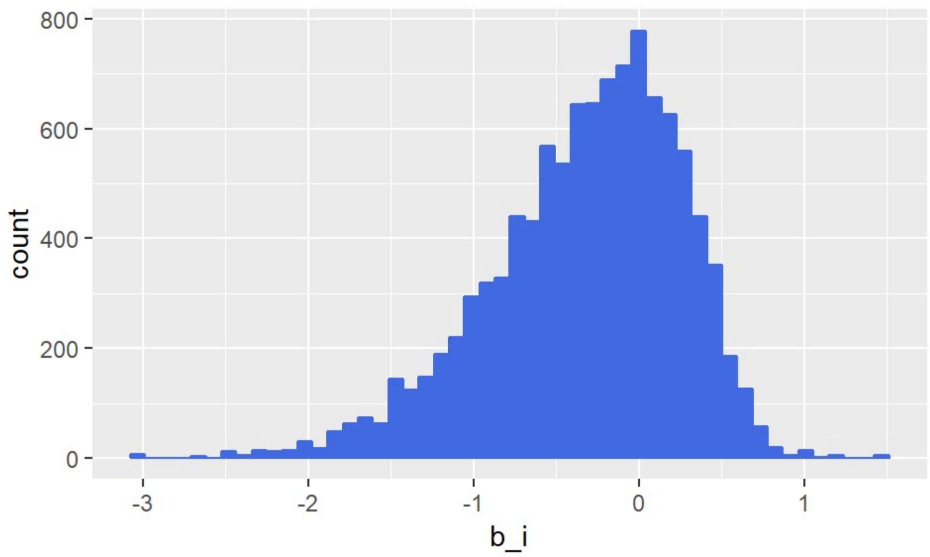
2.3.2 **Movie Effect Model**

As observed from insights, some movies are rated more than others. We can improve our model by adding this **movie_effect**. Thus, for each movie, the movie effect is calculated as the average of $Y_{u,i} - \hat{\mu}$ for each movie i .

So, We will add the term b_i to represent average ranking for movie i :

$$Y_{u,i} = \mu + b_i + \varepsilon_{u,i}$$

```
########################### MODELING MOVIE EFFECT ############################

#Now including movie effect,

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

ggplot(data = movie_means, aes(x = b_i)) +
  geom_histogram(bins = 50,colour = "royalblue", fill = "royalblue")
```




```
final <- test_set %>%
  left_join(movie_means, by='movieId')

prediction_ratings <- mu + final$b_i

model_movie_effect_rmse <- RMSE(prediction_ratings, test_set$rating)

rmse_results <- bind_rows(rmse_results,
                          data_frame(Method = "Movie Effect Model", Dataset="test_edx",
                                     RMSE = model_movie_effect_rmse ))
```

Some improvement can be observed in RMSE results, as shown below:

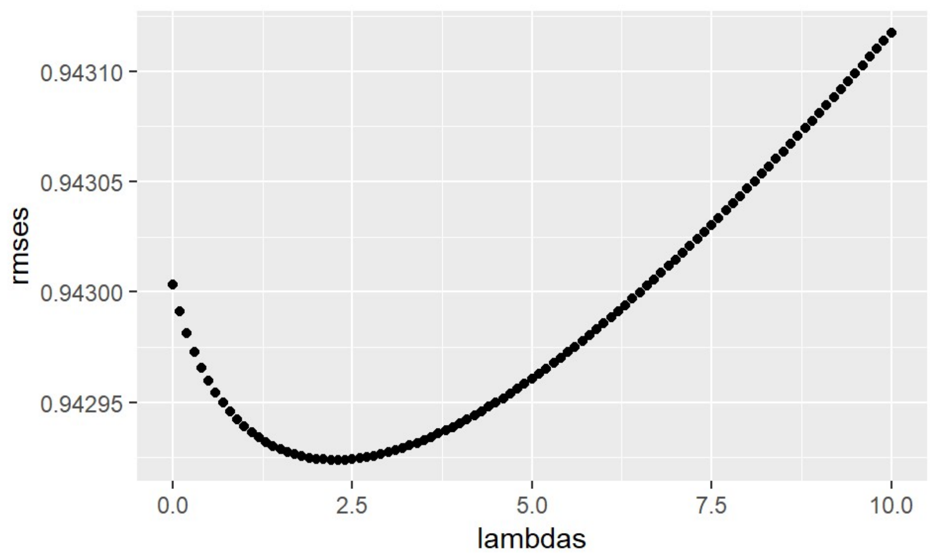
Method	Dataset	RMSE
Simple Average Model	**test_edx**	**1.0593180**
Movie Effect Model	**test_edx**	**0.9430034**

2.3.3 **Regularized Movie Effect Model**

On point to consider is that some movies considered to be “High Rated” or “Worst Rated”" are rated by very few viewers. These movies will contribute towards higher uncertainity and larger estimates of b_i . Thus, we would use regularization to remove these estimates.

With Regularization, we can penalize large estimates coming from small sample sizes anbd general concept is to minimize the sum of squares equation while penalizing for large values of b_i

```
####################### REGULARIZATION OF THE MOVIE EFFECT ####################


#In order to avoid impact of movies with few ratings, we would use regularization to nullif
y thier impact

# Thus lambda (tuning parameter) wil be calculated for regularized estimates of b_i.
lambdas <- seq(0, 10, 0.1)

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

rmses <- sapply(lambdas, function(l){
    final <- test_set %>%
    left_join(summation, by='movieId') %>%
    mutate(b_i = sum/(n_i+l))
    prediction_ratings <- mu + final$b_i
    return(RMSE(prediction_ratings, test_set$rating))
})

qplot(lambdas, rmses)
```



```
# Thus, 2.4 appeared to be the most optimized lambda value i.e. giving smallest RMSE

lambdas <- 2.4

movie_reg_means <- train_set %>%
                   group_by(movieId) %>%
                   summarize(b_i = sum(rating - mu)/(n()+lambdas), n_i = n())

final <- test_set %>%
         left_join(movie_reg_means, by='movieId') %>%
         replace_na(list(b_i=0))

prediction_ratings <- mu + final$b_i

model_movie_reg_rmse <- RMSE(prediction_ratings, test_set$rating)

rmse_results <- bind_rows(rmse_results,
                          data_frame(Method = "Regularized Movie Effect Model", Dataset="te
st_edx",
                                     RMSE = model_movie_reg_rmse ))

kable(rmse_results,align=rep("c",3)) %>%
  kable_styling(full_width = F) %>%
  column_spec(1:3,bold=T,border_right = T,color='royalblue')
```

Method	Dataset	RMSE
Simple Average Model	**test_edx**	**1.0593180**
Movie Effect Model	**test_edx**	**0.9430034**
Regularized Movie Effect Model	**test_edx**	**0.9429240**

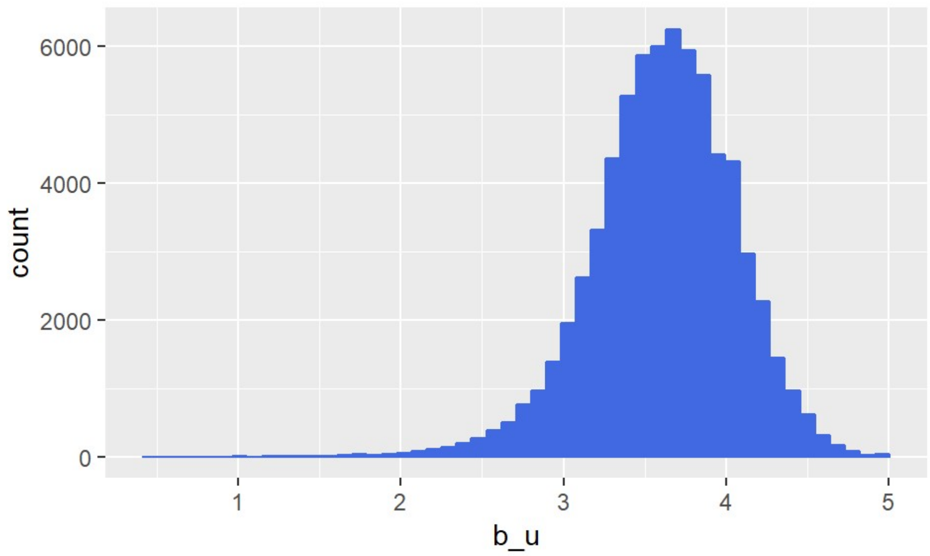
2.3.4 **Movie + User Effect Model**

Besides movie effect, we have seen some users rate more than others. So we will include the **user_effect** in addition to **regularized_movie_effect** which has already been accounted in the modeling.

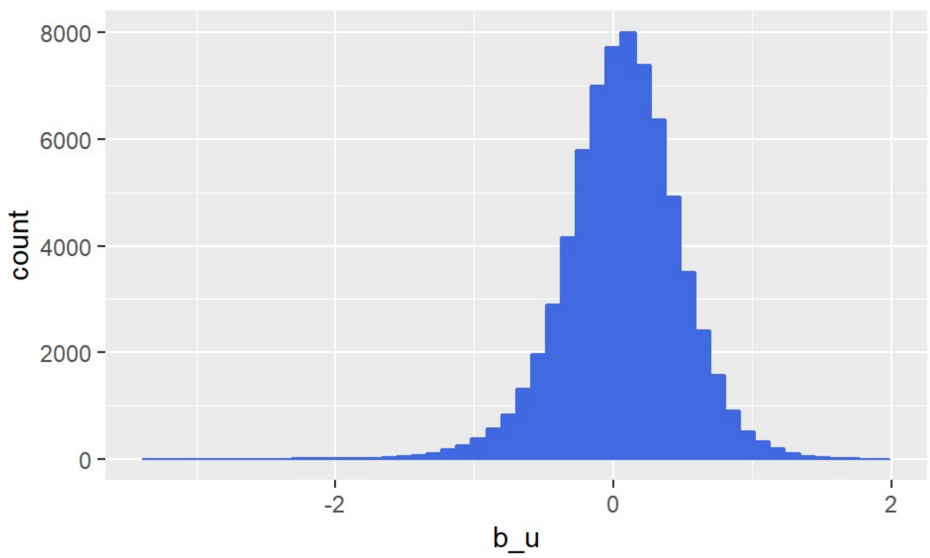
Thus, below is the model we are targeting in this section, where μ is the average rating, $\hat{b}_i(\lambda)$ is the regularized_movie_effect, b_u is the user-specific effect and $\varepsilon_{u,i}$ is the error term:

$$Y_{u,i} = \mu + \hat{b}_i(\lambda) + b_u + \varepsilon_{u,i}$$

```
############### Movie + User Effect Model ###########################

# Similar to movie effect, user effect also impacted by users who rate less movies, as can
be seen below
train_set %>%
  group_by(userId) %>%
  summarize(b_u = mean(rating)) %>%
  filter(n() >= 50) %>%
  ggplot(aes(b_u)) +
  geom_histogram(bins = 50, color  = "royalblue", fill  = "royalblue")
```




```
user_means <- train_set %>%
  left_join(movie_means, by='movieId') %>%
  group_by(userId) %>%
  summarize(b_u = mean(rating - mu - b_i))

ggplot(data = user_means, aes(x = b_u)) +
  geom_histogram(bins = 50,colour = "royalblue", fill = "royalblue")
```



```
final <- test_set %>%
  left_join(movie_means, by='movieId') %>%
  left_join(user_means, by='userId')

prediction_ratings <- mu + final$b_i + final$b_u

model_user_movie_rmse <- RMSE(prediction_ratings, test_set$rating)

rmse_results <- bind_rows(rmse_results,
                          data_frame(Method = "Movie + User Effect Model", Dataset="test_ed
x",
                                     RMSE = model_user_movie_rmse ))

kable(rmse_results,align=rep("c",3)) %>%
  kable_styling(full_width = F) %>%
  column_spec(1:3,bold=T,border_right = T,color='royalblue')
```

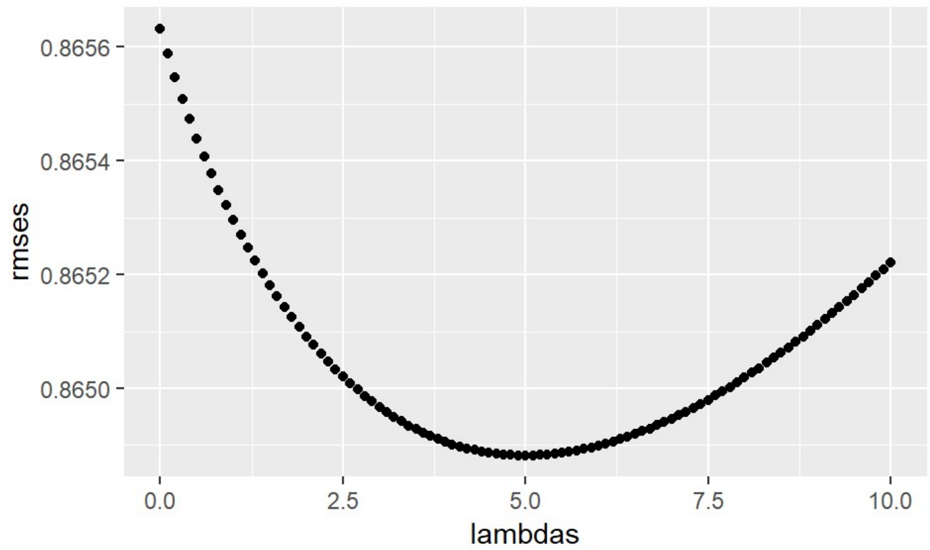
Method	Dataset	RMSE
Simple Average Model	**test_edx**	**1.0593180**
Movie Effect Model	**test_edx**	**0.9430034**
Regularized Movie Effect Model	**test_edx**	**0.9429240**
Movie + User Effect Model	**test_edx**	**0.8656322**

2.3.5 **Regularized Movie + User Effect Model**

Similar to **regularized movie effect**, we will also **regulariz** the user effect. Thus, we will use below code to find the best λ_u to use for the final model:

$$Y_{u,i} = \mu + \hat{b}_i(\lambda_i) + \hat{b}_u(\lambda_u) + \varepsilon_{u,i}$$

```r
# For regularization, lambda (tuning parameter) wil be calculated for regularized estimates of b_i & b_u.

lambdas <- seq(0, 10, 0.1)

rmses <- sapply(lambdas, function(l){

  mu <- mean(train_set$rating)

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

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

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

  return(RMSE(prediction_ratings, test_set$rating))
})

qplot(lambdas, rmses)
```



```
# It's clear that lambda=5 has the least value for RMSE

lambdas <- 5

user_reg_means <- train_set %>%
  left_join(movie_reg_means) %>%
  mutate(resids = rating - mu - b_i) %>%
  group_by(userId) %>%
  summarize(b_u = sum(resids)/(n()+lambdas))
```

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

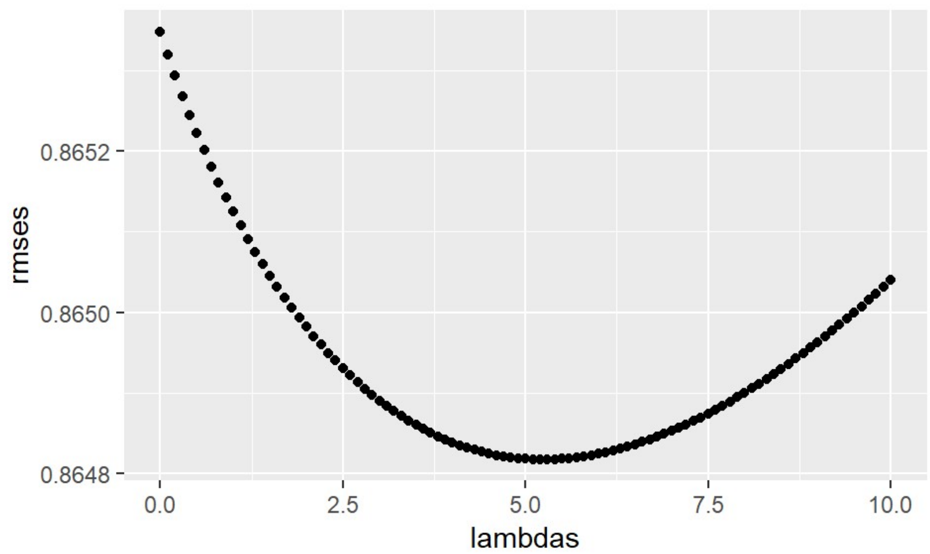
```
final <- test_set %>%
  left_join(movie_reg_means, by='movieId') %>%
  left_join(user_reg_means, by='userId') %>%
  replace_na(list(b_i=0, b_u=0))

prediction_ratings <- mu + final$b_i + final$b_u

model_user_movie_reg_rmse <- RMSE(prediction_ratings, test_set$rating)

rmse_results <- bind_rows(rmse_results,
                          data_frame(Method = "Regularized Movie + User Effect Model", Data
set="test_edx",
                                     RMSE = model_user_movie_reg_rmse ))

kable(rmse_results,align=rep("c",3)) %>%
  kable_styling(full_width = F) %>%
  column_spec(1:3,bold=T,border_right = T,color='royalblue')
```

Method	Dataset	RMSE
Simple Average Model	**test_edx**	**1.0593180**
Movie Effect Model	**test_edx**	**0.9430034**
Regularized Movie Effect Model	**test_edx**	**0.9429240**
Movie + User Effect Model	**test_edx**	**0.8656322**
Regularized Movie + User Effect Model	**test_edx**	**0.8649025**

3. **Results**

Finally, the model is tested on validation dataset to calculate RMSE

```r
################# Regularized Movie + User Effect Model (Validated) ###########

# Verifying the model for validation dataset
lambdas <- seq(0, 10, 0.1)

rmses <- sapply(lambdas, function(l){

  mu <- mean(edx$rating)

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

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

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

  return(RMSE(predicted_ratings, validation$rating))
})

qplot(lambdas, rmses)
```



```r
# It's clear that lambda=5 has the least value for RMSE

lambdas <- 5

user_reg_means <- edx %>%
  left_join(movie_reg_means) %>%
  mutate(resids = rating - mu - b_i) %>%
  group_by(userId) %>%
  summarize(b_u = sum(resids)/(n()+lambdas))
```

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

```r
final <- validation %>%
  left_join(movie_reg_means, by='movieId') %>%
  left_join(user_reg_means, by='userId') %>%
  replace_na(list(b_i=0, b_u=0))

prediction_ratings <- mu + final$b_i + final$b_u

model_user_movie_reg_rmse <- RMSE(prediction_ratings, validation$rating)
```

The RMSEs for the `validation` and `test_edx` dataset are summarized below:

Method	Dataset	RMSE
Simple Average Model	**test_edx**	**1.0593180**
Movie Effect Model	**test_edx**	**0.9430034**
Regularized Movie Effect Model	**test_edx**	**0.9429240**
Movie + User Effect Model	**test_edx**	**0.8656322**
Regularized Movie + User Effect Model	**test_edx**	**0.8649025**
Regularized Movie + User Effect Model	**Validation**	**0.8653595**

4. **Conclusion**

To conclude, below is the summary and highlights of this capstone project:

- After loading dataset, We started with an exploratory analysis and key insights about the data.
- Not much data wrangling is required as dataset was pretty much cleansed and in reasdy to use state.
- First observation was the unbalance distribution of full vs half star ratings.
- The ratings distribution was pretty much uniform with median rating of ~3.5.
- Genres impact is minimal while rated year has correlation with movies.
- Impact of users and movies for ratings can be clearly seen in the dataset.
- The fact that some users have rated more than others while other movies have been rated more than others prompt us to use regularized model
- The modeling approach satrts with simple baseline model and then improving upon by adding movie effetc, user effect & both movie + user effect in the model giving us the acceptable RMSE.

- Finally, we validate the model by testing it on validation dataset and summarizing the whole results as below:

Method	Dataset	RMSE
Simple Average Model	**test_edx**	**1.0593180**
Movie Effect Model	**test_edx**	**0.9430034**
Regularized Movie Effect Model	**test_edx**	**0.9429240**
Movie + User Effect Model	**test_edx**	**0.8656322**
Regularized Movie + User Effect Model	**test_edx**	**0.8649025**
Regularized Movie + User Effect Model	**Validation**	**0.8653595**

5. **References**

- Irizarry A. Rafael (2018) Introduction to Data Science: Data Analysis and Prediction Algorithms with R
- Basu, Hirsh, Cohen (1998) Recommendation as Classification:Using Social and Content-Based Information in Recommendation
- Ungar, L. H., and Foster, D. P. (1998) Clustering Methods for Collaborative Filtering. In Workshop on Recommender Systems at the 15th National Conference on Artificial Intelligence.