Movie Recommendation System

Report Submitted by

T Baskaran

(In partial fulfillment of the Data Science: Capstone course under the HarvardX Professional Certificate Program in Data Science)

April 10, 2019

Preface

As a Statistician, I have spent more than three decades of government service. On the eve of my retirement, I came across R as a tool for statistical analysis, which brought me to the field of Data Science. It has caught up my mind and I decided to become a data scientist, in spite of my age (I am now sixty five years young). edX and the HarvardX gave me the opportunity to pursue Data Science. I had completed all the previous eight courses. Joining the **Data Science: Capstone** which is the final course in the **HarvardX Professional Certificate Program in Data Science** has resulted in doing this project. This is the first ever project done by me in the field of Data Science.

In spite of starting at the ground level on Data Science, I have come to this level of doing a project. For this, I am indebted to Prof Rafael A. Irizarry, whose lectures had given me the necessary insights into Data Science and confidence to take up Data Science as my post-retirement carrier. I am thankful to the edX platform and the HarvardX for having made available all the facilities for completing all the courses and this project. I also thank all the fellow learners for their wonderful inputs through the discussion forum.

Abstract

The project is about conceiving a Movie Recommendation System with a reasonably low error. The targeted RMSE is 0.87750. This has been achieved by taking the MovieLens 10M data set and constructing various models and examining their RMSEs. As the Movie + User Effect model was developed with an RMSE less than the required, it was consider as the appropriate model. The model was also validated using a validation data set, earmarked for the purpose. It was found to have predicted the ratings appropriately. Therefore, the Movie + User Effect model was recommended to be used as the Movie Recommendation System.

Table of Contents

1. Introduction

1.1 Objective

The present project is a part of the **Data Science: Capstone** of the HarvardX Professional Certificate Program in Data Science. In this project, a movie recommendation system using the MovieLens dataset is created.

The objective of the project is to develop a movie recommendations system using the MovieLens 10M data set with the Root Mean Square Error (RMSE) less than or equal to 0.87750.

1.2 Overview of Recommendation System

A recommendation system attempts to make predictions on user preferences and make recommendations which should interest customers. It uses ratings that users have given to items, so as to make specific recommendations to users. The system is particularly useful in making the user experience more pleasant and in cutting the time spent on research. Recommendation systems find the maximum use on many e-commerce platforms. They are utilized in a variety of areas including movies, music, news, books, research articles, search queries, social tags, and products in general. Recommendation system is one of the most common and easily comprehendible applications of big data and machine learning.

1.3 Approach

While developing the movie recommendation system using the 10M version of the MovieLens dataset, it is proposed to divide the data set into two data sets namely “validation” and “edx” data sets in 10:90 ratio. While the “validation” data set is considered as an unknown data set and is used to validate the model, the “edx” data set is the data set available for all practical purposes of developing the model for the movie recommendation system. In this project, it is proposed to develop algorithms for a number of models and select the appropriate model based on the Residual Mean Square Error (RMSE). Each model is to be trained using the training data set, and evaluated using the test data set. The RMSE is then calculated. Models are improved when the RMSE is high. The objective as stated earlier is to obtain a model with RMSE less than 0.87750. When the RMSE reaches that level, the model is considered appropriate, and ready for applications on the external data sets. Accordingly, the model is finally validated using the validation data set.

1.4 Structure of the report

The present report has four sections. The **Section 1: Introduction**, introduces the project and its objectives. It also presents an overview of the "Recommendation System" in the machine learning context. The section also briefs about the approach followed in this project for developing the required model. In **Section 2: Data Set**, the report talks of the data set used in the project namely the MovieLens 10M Dataset. It introduces the reader to the different files available in the data set, the variables in the data sets and their nature of arrangement. The **Section 3: Analysis of the data**, presents the actual analysis performed in the project. Initially, an Exploratory Data Analysis has been attempted, including data visualization and brought out several insights into the data. This is followed by developing the model. While doing so, the report briefly elucidates the different models used in the process of reaching the final model. It also constructs the loss function so as to measure the error loss while testing the model. The section also provides the results of the data analysis. **Section 4: Conclusion** recommends the final model which can be adopted as the Movie Recommendation System.

2. Data Set

GroupLens Research has collected and made available rating data sets from the MovieLens web site (http://movielens.org). Among the data sets, we use the **MovieLens 10M Dataset.** This is a stable benchmark dataset with 10 million ratings and 100,000 tag applications applied to 10,000 movies by 72,000 users. The data set is available at https://grouplens.org/datasets/movielens/10m/ and is downloaded from the url:- http://files.grouplens.org/datasets/movielens/ml-10m.zip

On extraction using R packages like, tidyverse, caret etc, it is found that the data set contains three files: movies.dat, ratings.dat and tags.dat. On suitably arranging, the MovieLens 10M data set contains 10000054 ratings and 95580 tags applied to 10681 movies by 71567 users of the online movie recommender service MovieLens. Users have been selected at random for inclusion. All users selected had rated at least 20 movies. Each user is represented by an Id.

The ratings.dat file, on reading to "ratings", has four columns of data with names namely "userId", "movieId", "rating" and "timestamp". The object "ratings" now has 10000054 observations of 4 variables. It is also found that the mean rating is 3.5.

```
#Read the ratings.dat file
ratings <- read.table(text = gsub("::", "\t", readLines(unzip(dl, "ml-
10M100K/ratings.dat"))),
                      col.names = c("userId", "movieId", "rating",
"timestamp"))
class(ratings)

## [1] "data.frame"

ratings[1:5,]

##   userId movieId rating timestamp
## 1      1     122      5 838985046
## 2      1     185      5 838983525
## 3      1     231      5 838983392
## 4      1     292      5 838983421
## 5      1     316      5 838983392

dim(ratings)

## [1] 10000054        4

mean(ratings$rating)

## [1] 3.512422
```

Movie information is contained in the file movies.dat. Each line of this file represents one movie, and has the format: MovieID::Title::Genres. While the MovieID is the Id given to movies, Title gives the title of the movies, and Genres is a pipe-separated list of genres. The information is read to "movies". The "movies" is a dataframe with 10681 rows and 3 columns.

```
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))
class(movies)

## [1] "data.frame"

dim(movies)

## [1] 10681     3
```

A new data frame “movielens” is created by combining the ratings and movies files. The movielens dataframe now has 10000054 observations and 6 variables.

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

## [1] "data.frame"

str(movielens)

## 'data.frame':    10000054 obs. of  6 variables:
##  $ userId   : int  1 1 1 1 1 1 1 1 1 1 ...
##  $ movieId  : num  122 185 231 292 316 329 355 356 362 364 ...
##  $ rating   : num  5 5 5 5 5 5 5 5 5 5 ...
##  $ timestamp: int  838985046 838983525 838983392 838983421 838983392 
838983392 838984474 838983653 838984885 838983707 ...
##  $ title    : chr  "Boomerang (1992)" "Net, The (1995)" "Dumb & Dumber 
(1994)" "Outbreak (1995)" ...
##  $ genres   : chr  "Comedy|Romance" "Action|Crime|Thriller" "Comedy" 
"Action|Drama|Sci-Fi|Thriller" ...
```

3. Analysis of the Data

3.1 Residual Mean Square Error

While analyzing the data, it is essential to measure the bias or error loss during prediction. This project proposes to use the Residual Mean Square Error (RMSE) on the test set to measure the error loss. The RMSE is defined as:

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

where $y_{u,i}$ is the rating for movie i by user u with $\hat{y}_{u,i}$ denoting our predictions and N being the number of user/movie combinations and the sum occurring over all these combinations. It is the typical error we make when predicting a movie rating.

Let’s write a function that computes the RMSE for vectors of ratings and their corresponding predictors:

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

3.2 Separating data for validation.

As delineated earlier in the approach, the movielens data set is divided into “validation” and “edx” datasets. The “ëdx” data set is the data set that is considerd for developing the model. The edx data set has 9000055 observations and 6 variables, the validation data set has 999999 observations and 6 variables.

```
# Validation set will be 10% of MovieLens data
set.seed(1)
test_index <- createDataPartition(y = movielens$rating, times = 1, p = 0.1,
list = FALSE)
edx <- movielens[-test_index,]
temp <- movielens[test_index,]

# Make sure userId and movieId in validation set are also in edx set

validation <- temp %>%
     semi_join(edx, by = "movieId") %>%
     semi_join(edx, by = "userId")

# Add rows removed from validation set back into edx set

removed <- anti_join(temp, validation)

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

edx <- rbind(edx, removed)


dim(edx)

## [1] 9000055       6

dim(validation)

## [1] 999999      6
```

3.3 Exploratory Data Analysis

The edx data set is taken for Exploratory Data Analysis (EDA). There is no movie having a rating of 0, as the ratings start from 0.5. The number of movies having a rating of 3, for instance, is 2121240.

```
#Number of movies with 0 and 3 ratings
edx %>% filter(rating == 0) %>% tally()

##   n
## 1 0

edx %>% filter(rating == 3) %>% tally()

##         n
## 1 2121240
```

There are 10677 unique movies in the edx data set with 69878 unique users.

```
#Number of distinct movies and users
edx %>% summarise(users =n_distinct(userId), movies= n_distinct(movieId))

##   users movies
## 1 69878  10677
```

While the Drama genres has the maximum number of ratings (3910127), the Comedy follows suit with 3540930 ratings. While the movie Pulp Fiction (1994) has the maximum number of ratings (31362), Forrest Gump (1994) has 31079 ratings. Silence of the Lambs, The (1991) comes in the third position with 30382 ratings. Jurassic Park (1993) secures the fourth rank with 29360 rankings.

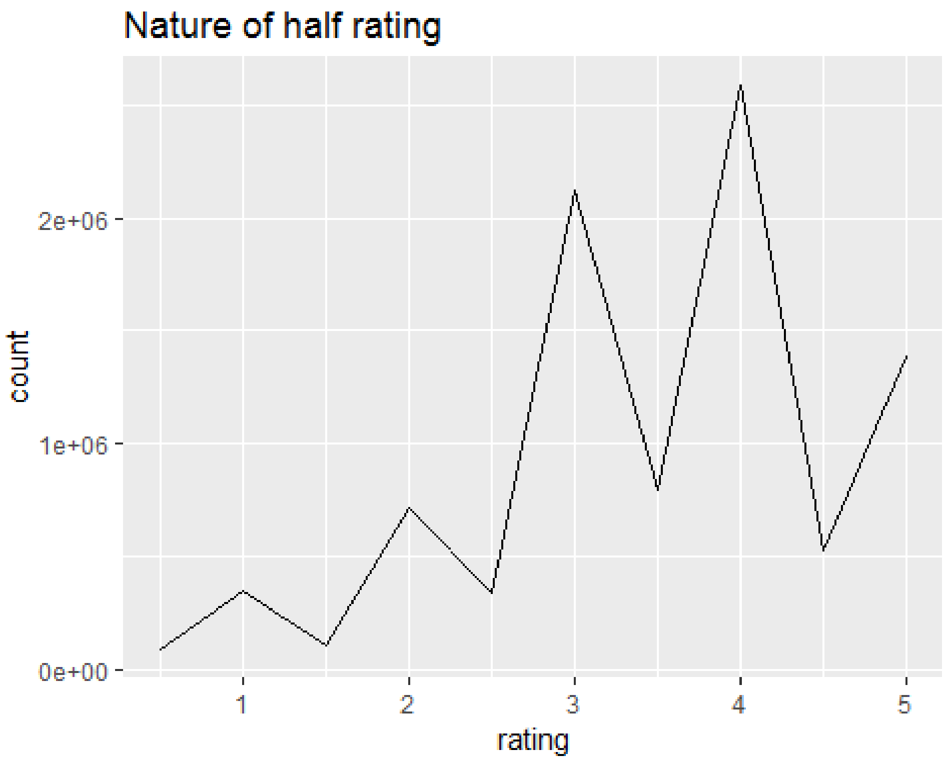
```
#Number of ratings in different genres
edx %>% separate_rows(genres, sep = "\\|") %>%
  group_by(genres) %>%
  summarize(count = n()) %>%
  arrange(desc(count)) %>%
  head()

## # A tibble: 6 x 2
##   genres      count
##   <chr>       <int>
## 1 Drama     3910127
## 2 Comedy    3540930
## 3 Action    2560545
## 4 Thriller  2325899
## 5 Adventure 1908892
## 6 Romance   1712100
```

```
#Number of ratings for different movies
edx %>% group_by(movieId, title) %>%
  summarize(count = n()) %>%
  arrange(desc(count)) %>%
  head()

## # 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
```

Among the different ratings, the rating 4 occurs the maximum number of times (2588430), followed by the rating 3 (2121240 times). The rating 0.5 occurs the minimum number of times (85374). It is also interesting to note that half star ratings are less common than whole star ratings (e.g., there are fewer ratings of 3.5 than there are ratings of 3 or 4, etc.).

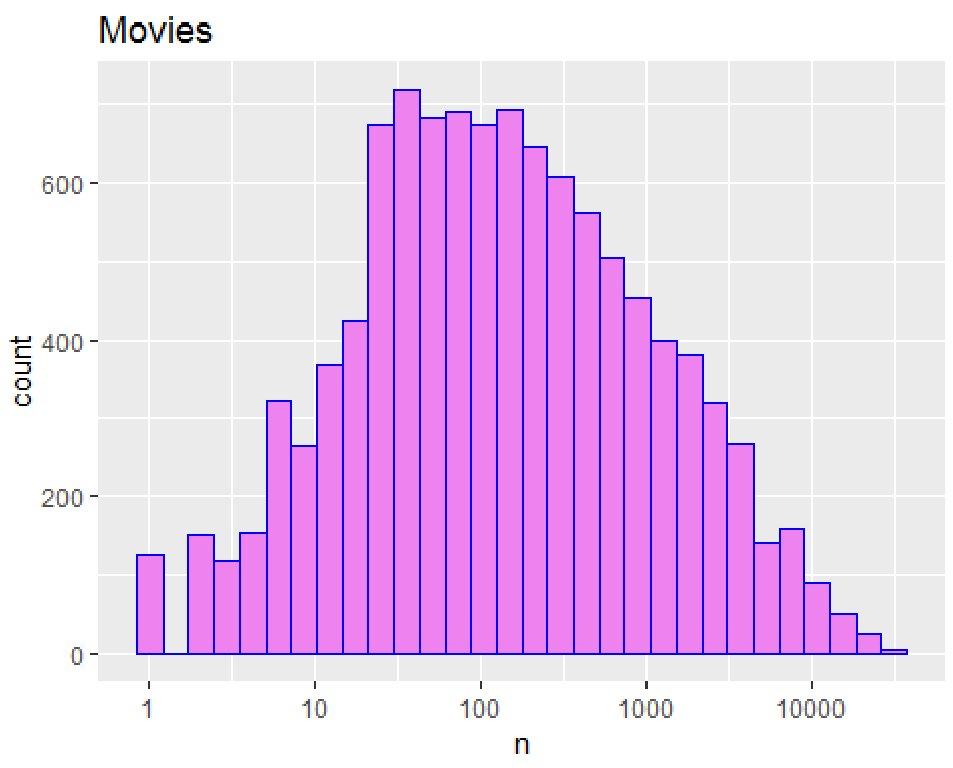
```
#Count of ratings
edx %>% group_by(rating) %>% summarize(count = n()) %>%
  arrange(desc(count))

## # 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

#Nature of half rating numerically
half_rating <- edx %>% group_by(rating) %>% summarize(count = n())
head(half_rating)

## # A tibble: 6 x 2
##   rating   count
##    <dbl>   <int>
## 1     0.5    85374
## 2     1     345679
## 3     1.5   106426
## 4     2     711422
## 5     2.5   333010
## 6     3    2121240
```

```
#Nature of half rating visually
half_rating %>%
  ggplot(aes(x = rating, y = count)) +
  geom_line()+
  ggtitle("Nature of half rating")
```

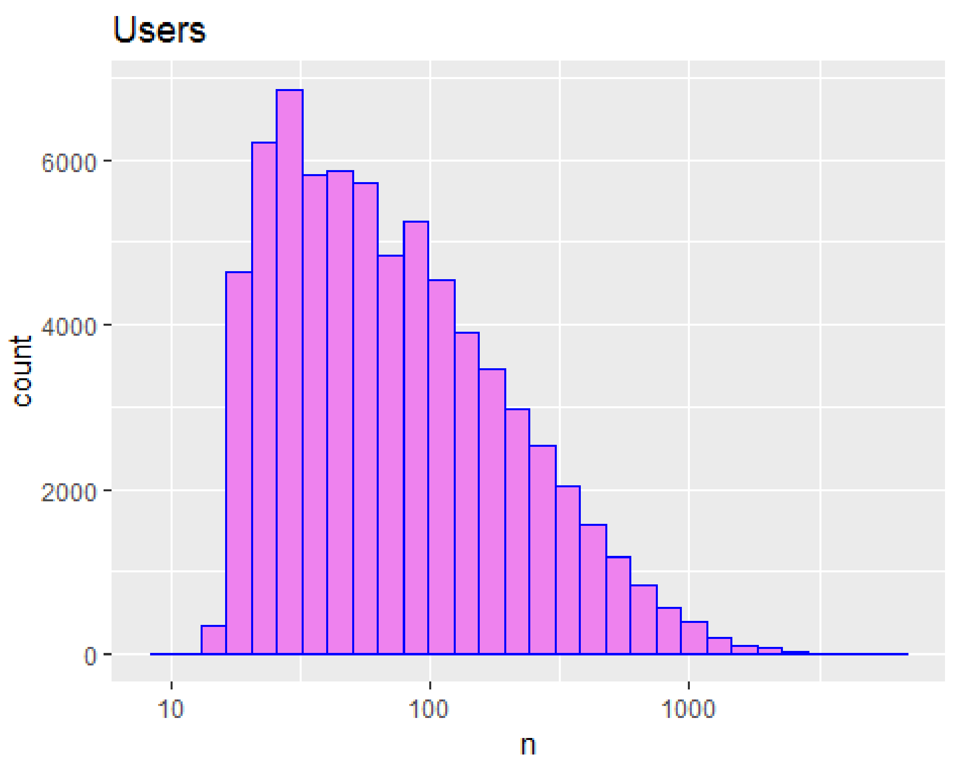


The distribution of movies rating can be seen from the following histogram:

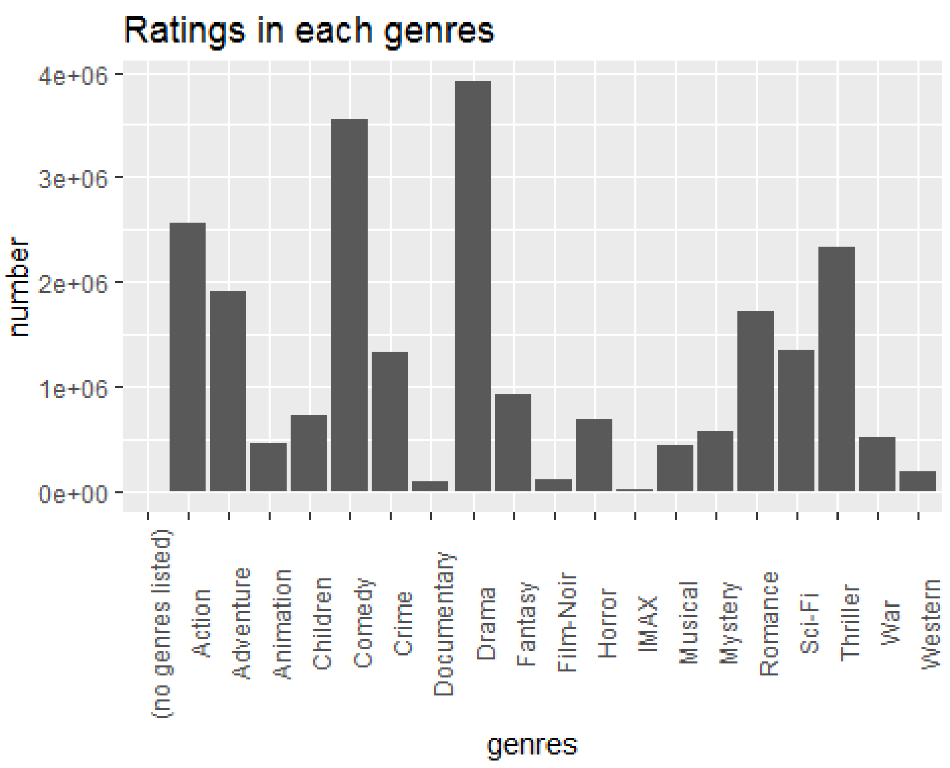
```
edx %>%
  dplyr::count(movieId) %>%
  ggplot(aes(n)) +
  geom_histogram(bins = 30, color = "blue", fill = "violet") +
  scale_x_log10() +
  ggtitle("Movies")
```



```r
# See the distribution of users rating activity
edx %>%
  dplyr::count(userId) %>%
  ggplot(aes(n)) +
  geom_histogram(bins = 30, color = "blue", fill = "violet") +
  scale_x_log10() +
  ggtitle("Users")
```



```
#Create a dataframe of genres
edx %>%
  separate_rows(genres, sep = "\\|") %>%
  group_by(genres) %>%
  summarise(number = n()) %>%
  arrange(desc(number)) %>%
  ggplot(aes(x=genres, y=number))+
  geom_bar(stat="Identity")+
  ggtitle("Ratings in each genres") +
  theme(axis.text.x = element_text( angle = 90))
```



3.4 Developing the model

As elucidated earlier, different models are being developed and the model which has the lowest RMSE considered appropriate and recommended for use. The recommended model then is adopted for application on the validation set.

3.4.1 Partitioning the edx data set

For the purpose of developing the model, we partition the edx data set into training and test data sets in the ratio of 80:20. We train the model using the training data set, then test it using the test set and then proceed to calculate the RMSE. The process continues till we get the desired RMSE. We also create a results table so that the different methods employed and the corresponding RMSEs are properly recorded.

```
###Partition edx data set into train and test sets

set.seed(500)
test_index <- createDataPartition(y = edx$rating, times = 1, p = 0.2, list =
FALSE)
train_set <- edx[-test_index,]
test_set <- edx[test_index,]
### Adjusting_for_rows
```

```
test_set1 <- test_set %>%
  semi_join(train_set, by = "movieId") %>%
  semi_join(train_set, by = "userId")

# Add rows removed from test_set set back into train_set set
removed <- anti_join(test_set, test_set1)

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

train_set <- rbind(train_set, removed)
test_set <- test_set1
```

3.4.2 Model 1: Guessing Model

Here we consider predicting the ratings simply by randomly assigning the ratings. We call this method as "Guessing the ratings". The RMSE for this method is obtained as 1.94.

```
#Prediction by gussing
mu_hat <- as.numeric(sample(c("0.5", "1",   "1.5", "2",  "2.5", "3",   "3.5",
"4",   "4.5", "5" ), length(test_set$rating), replace = TRUE))

Guess_RMSE <- RMSE(test_set$rating,mu_hat)
Guess_RMSE

## [1] 1.94088

###Create a results table
rmse_results <- data_frame(method = "Guessing Model", RMSE = Guess_RMSE)
```

3.4.3 Model 2: Proportional Model

In this this model, we consider predicting the unknown ratings by assigning the ratings with probabilities proportional to their occurrence in the test data set. We call this model as "Proportional ratings". The RMSE now is 1.5, which is a very minor decrease.

```
#Define a sequence of ratings
s <- seq(0.5,5,0.5)
#Define a vector of 10 zeros
p <- rep(0,10)
# Populate p
for (k in 1:10) {
  i <- s[k]
  p[k] <- mean(test_set$rating==i)
}
p
```

```
##  [1] 0.009494608 0.038454551 0.011945761 0.078958026 0.036970081
##  [6] 0.235632391 0.087937626 0.287622731 0.058421006 0.154563217

mu_hat <- as.numeric(sample(c("0.5", "1",   "1.5", "2",  "2.5", "3",   "3.5",
"4",   "4.5", "5" ), length(test_set$rating), replace = TRUE, prob=p))

Prop_RMSE <- RMSE(test_set$rating,mu_hat)
Prop_RMSE

## [1] 1.500101

rmse_results <- bind_rows(rmse_results,
                          data_frame(method="Proportional Model", RMSE =
Prop_RMSE))
```

3.4.4 Model 3: Same Mean rating Model

The present model uses the same rating for all the unknown ratings. The model can be thought of as

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

with $\varepsilon_{i,u}$ independent errors sampled from the same distribution centered at 0 and μ the "true" rating for all movies. We know that the estimate that minimizes the RMSE is the least squares estimate of μ and, in this case, is the average of all ratings. Let us denote it as $\hat{\mu}$. We call this model as "Same Mean ratings" model. The RMSE is calculated and is found to be 1.06. This model is found to be superior to the previous two models, yet it is not satisfactory.

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

## [1] 3.512496

same_rating_RMSE <- RMSE(test_set$rating, mu_hat)
same_rating_RMSE

## [1] 1.060684

rmse_results <- bind_rows(rmse_results,
                          data_frame(method="Same Mean rating Model", RMSE =
same_rating_RMSE))
```

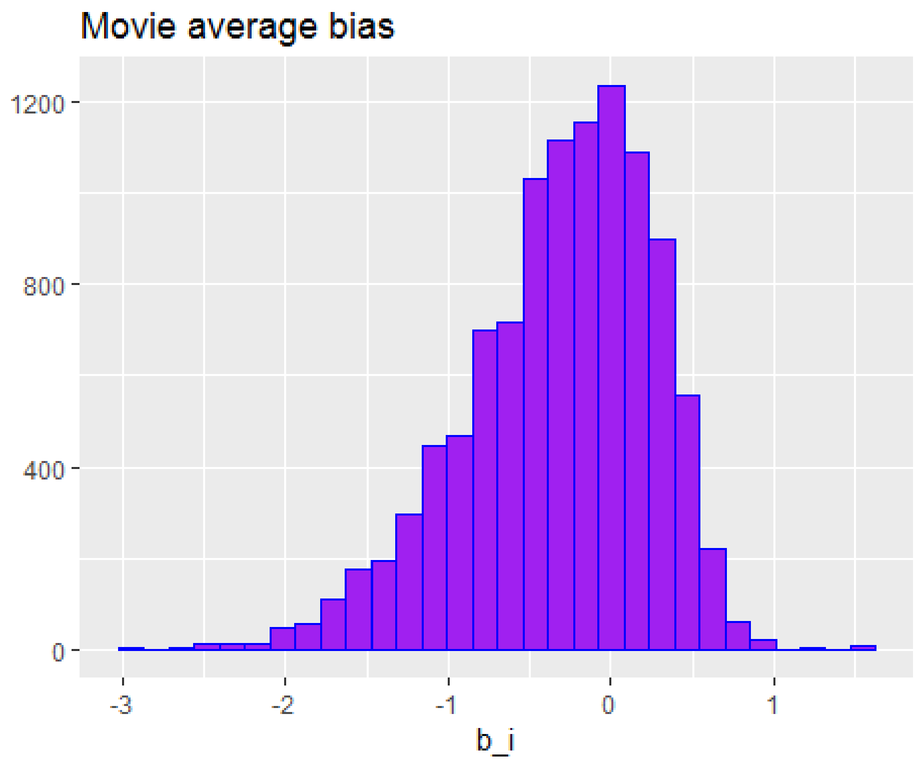
3.4.5 Model 4: Movie Effect Model

In this model, we take note of the fact that movies are rated differently by different users. In other words, there can be variation among the ratings for a single movie. We now improve the previous model by adding the average rating of a movie i. In other words, the model has now become

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

where b_i is the average rating given to a movie i. But the least square estimate $\hat{b}_i$ is just the average of $Y_{u,i} - \hat{\mu}$. The model is called as "Movie effect" model. It can be seen visually how far the estimates of this bias varies.

```
####Histogram
movie_avgs <- train_set %>%
  group_by(movieId) %>%
  summarize(b_i = mean(as.numeric(rating) - mu_hat))

movie_avgs %>%
      qplot(b_i, geom ="histogram", bins = 30,
      data = ., color = I("blue"), fill=I("purple"),
      main = "Movie average bias")
```



This RMSE is then calculated and it is now reduced to less than one (0.944). But still it is not satisfactory.

```
####Predictions
predicted_ratings <- mu_hat + test_set %>%
  left_join(movie_avgs, by='movieId') %>%
  pull(b_i)
movie_rmse <- RMSE(predicted_ratings, test_set$rating)
movie_rmse

## [1] 0.9438643

rmse_results <- bind_rows(rmse_results,data_frame(method="Movie Effect
Model",RMSE = movie_rmse))
```

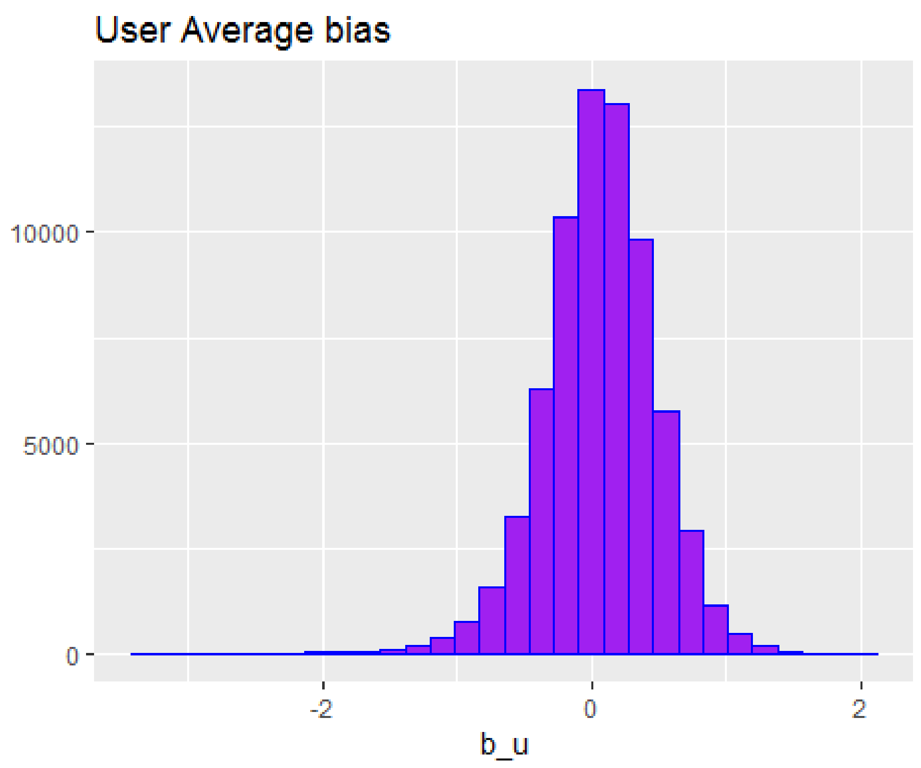
3.4.6 Model 5: Movie + User Effect Model

As an improvement over the previous model, we consider now the possible variations in the ratings given by a user u. Let's compute the average rating for user u. As we see substantial variations across users, we introduce the user effect also in the model. The model now can be written as

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

where b_u is a user-specific effect. Now if a cranky user (negative b_u) rates a great movie (positive b_i), the effects counter each other and we may be able to correctly predict that this user gave this great movie a 3 rather than a 5.

```
####Histogram
user_avgs <- train_set %>%
  left_join(movie_avgs, by='movieId') %>%
  group_by(userId) %>%
  summarize(b_u = mean(rating - mu_hat - b_i))

user_avgs %>%
      qplot(b_u, geom ="histogram", bins = 30, data = .,
      color = I("blue"), fill=I("purple"), main= "User Average bias")
```



On fitting the model the RMSE is calculated and found to be 0.866, which is a substantial reduction.

```
####Predictions
predicted_ratings <- test_set %>%
  left_join(movie_avgs, by='movieId') %>%
  left_join(user_avgs, by='userId') %>%
  mutate(pred = mu_hat + b_i + b_u) %>%
  pull(pred)

movie_user_rmse <- RMSE(predicted_ratings, as.numeric(test_set$rating))

movie_user_rmse

## [1] 0.8662211

rmse_results <- bind_rows(rmse_results,
data_frame(method="Movie + User Effects Model",  RMSE = movie_user_rmse))
```

3.4.7 Model 6: Regularized Movie Effect Model

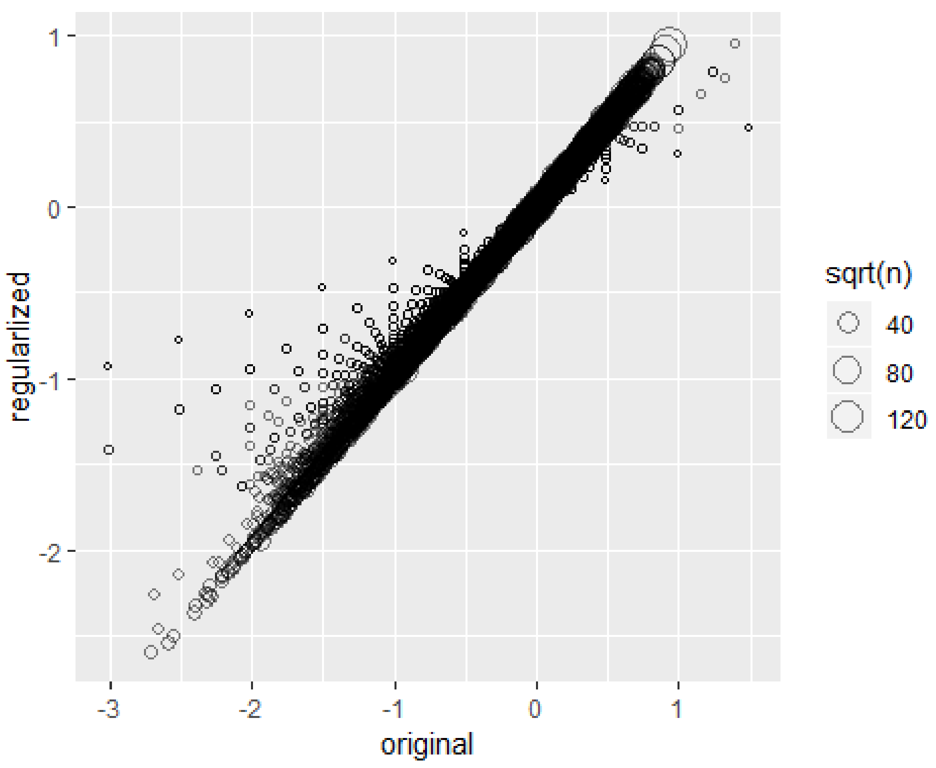
When some movies are rated by a few users, then the estimates becomes uncertain and $\hat{b}_i$ becomes noisy. It may not be appropriate to trust such estimates, especially when it comes to prediction. Large errors can increase our RMSE, so we would rather be conservative when unsure.

Regularization permits us to penalize large estimates that are formed using small sample sizes. In this model we use the regularization technique and attempt to predict the ratings. The general idea behind regularization is to constrain the total variability of the effect sizes. When our sample size n_i is very large, a case which will give us a stable estimate, then the penalty λ is effectively ignored since $n_i + \lambda \approx n_i$.

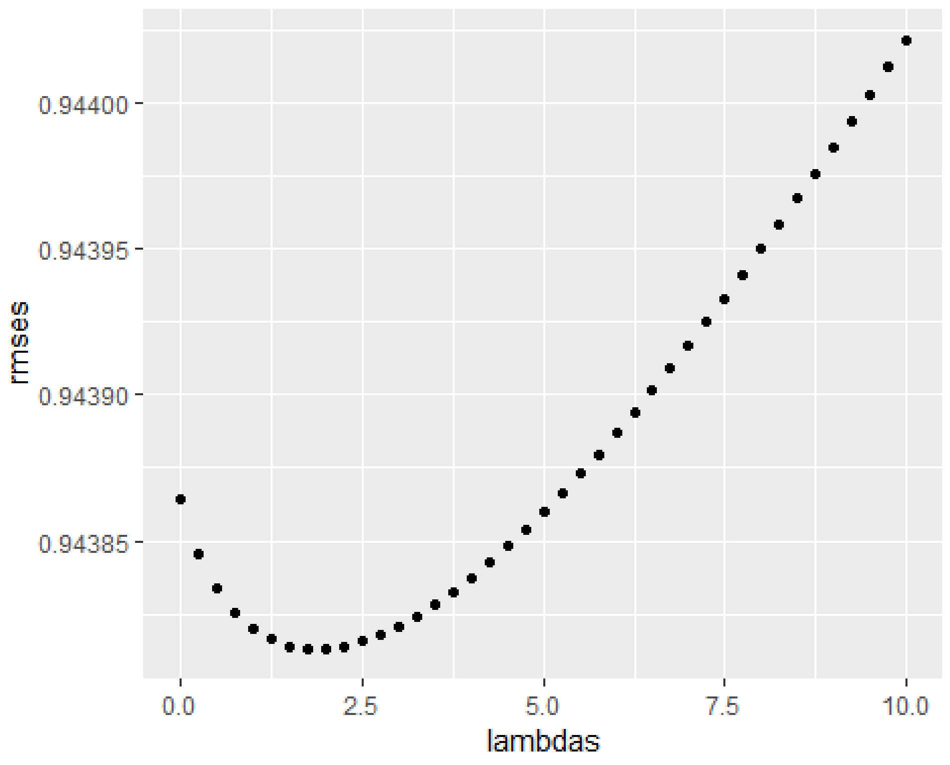
As λ is a tuning parameter, let us use cross validation and tune the parameter λ. This has resulted in the minimum RMSE of 0.944 for the $\lambda = 2.25$. Let us compute regularized estimates of b_i using $\lambda = 2.25$.

We can also see how the estimates shrink, by plotting the regularized estimates versus the least squares estimates. As the RMSE is higher than the previous model, the current model is not of use.

```
lambda <- 2.25
mu <- mean(train_set$rating)
movie_reg_avgs <- train_set %>%
  group_by(movieId) %>%
  summarize(b_i = sum(as.numeric(rating) - mu)/(n()+lambda), n_i = n())

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



```
###Cross validation for lambda for movie effect
lambdas <- seq(0, 10, 0.25)

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

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

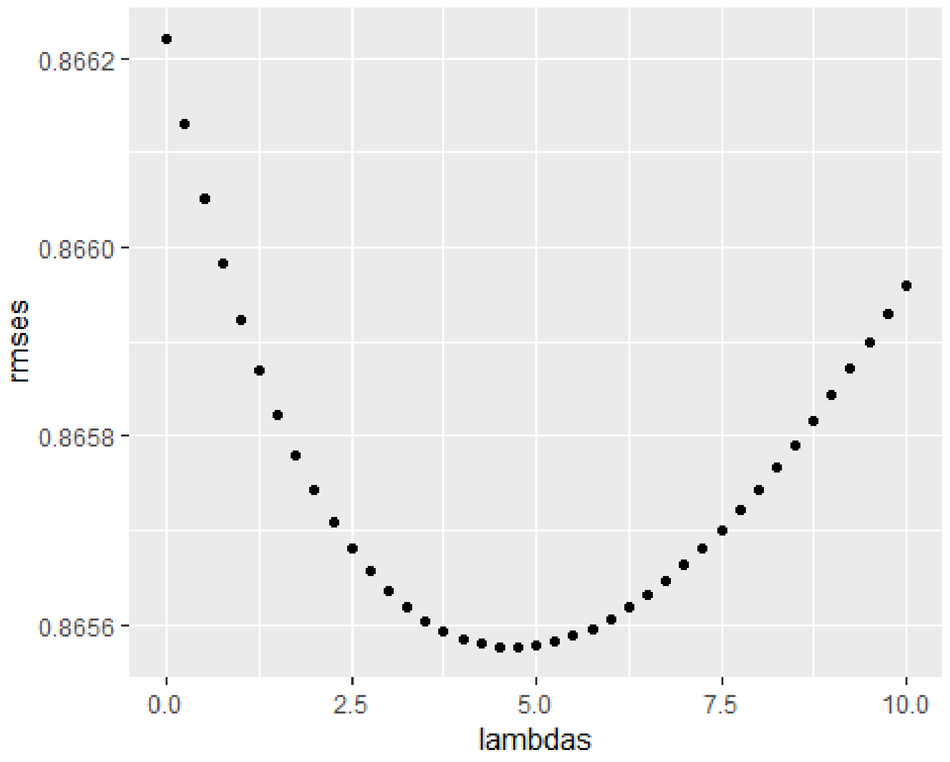


```
lambdas[which.min(rmses)]

## [1] 1.75

min(rmses)

## [1] 0.943813

####Predictions
rmse_results <- bind_rows(rmse_results,
                          data_frame(method="Regularized Movie Effect Model",
                                     RMSE = min(rmses)))
```

3.4.8 Model 7: Regularized Movie + Effect Model

As in the previous model, let us now take the Movie + User effect model and apply regularization technique with cross validation. This model yields an RMSE of 0.866 for the $\lambda = 4.75$.

```
###Cross validation for lambda for movie + user effect
lambdas <- seq(0, 10, 0.25)

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

  mu <- mean(as.numeric(train_set$rating))
  b_i <- train_set %>%
    group_by(movieId) %>%
    summarize(b_i = sum(as.numeric(rating) - mu)/(n()+l))

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

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

  return(RMSE(predicted_ratings, as.numeric(test_set$rating)))
})

qplot(lambdas, rmses)
```



```
lambda <- lambdas[which.min(rmses)]
lambda

## [1] 4.75

min(rmses)
```

```
## [1] 0.865576

rmse_results <- bind_rows(rmse_results,
                  data_frame(method="Regularized Movie + User Effect  Model",
RMSE = min(rmses)))
```

3.4.9 Results

We have developed seven models to prescribe a movie recommendation system. These models had different levels of error. The models along with their RMSEs are given below:

```
rmse_results

## # A tibble: 7 x 2
##   method                                RMSE
##   <chr>                                <dbl>
## 1 Guessing Model                       1.94
## 2 Proportional Model                   1.50
## 3 Same Mean rating Model               1.06
## 4 Movie Effect Model                   0.944
## 5 Movie + User Effects Model           0.866
## 6 Regularized Movie Effect Model       0.944
## 7 Regularized Movie + User Effect Model 0.866
```

Our objective in developing the model is to have such a model with RMSE less than 0.87750. As the Movie + User effect model and Regularized Movie + User effect model satisfy our objective in terms of their RMSEs, both being 0.866, we consider the Movie + User effect model, being simpler of the two, as the appropriate one for application on the validation set, which has been considered as external data set. On application of the Movie + User effect model on the validation set, it predicts the user ratings with the RMSE of 0.866.

```
####Predictions

movie_avgs <- train_set %>%
  group_by(movieId) %>%
  summarize(b_i = mean(as.numeric(rating) - mu_hat))

user_avgs <- train_set %>%
  left_join(movie_avgs, by='movieId') %>%
  group_by(userId) %>%
  summarize(b_u = mean(rating - mu_hat - b_i))

predicted_ratings <- validation %>%
  left_join(movie_avgs, by='movieId') %>%
  left_join(user_avgs, by='userId') %>%
  mutate(pred = mu_hat + b_i + b_u) %>%
  pull(pred)

movie_user_rmse_vali <- RMSE(predicted_ratings, validation$rating)
movie_user_rmse_vali

## [1] 0.8663736
```

4. Conclusion

The Movie + User Effect Model is the model with 0.866 RMSE. It is also validated with the validation data set and found to predict the ratings with the same RMSE. It is therefore concluded that the Movie + User effect Model is the appropriate movie recommendation system.

References

1. Recommender systems - Wikipedia https://en.wikipedia.org/wiki/Recommender_system
2. GroupLens Research https://grouplens.org/datasets/movielens/
3. Introduction to Data Science - Data Analysis and Prediction Algorithms with R by Prof. Rafael A. Irizarry (2019-03-27) https://rafalab.github.io/dsbook/
4. RecommenderSystem_MovieLens https://github.com/amboussetta/RecommenderSystem_MovieLens
5. https://github.com/AdityaRoongta/movie_recommender/blob/d93169ca9cb7440a24e5abf90e425d634d1babb4/RCode.R
6. https://github.com/VarunRaosr/MovieRecommenderSystem/blob/master/recommendation.R
7. https://github.com/lee-stats/movielens-project/blob/master/MovieLens%20Project%20Report.Rmd
8. https://rstudio-pubs-static.s3.amazonaws.com/288836_388ef70ec6374e348e32fde56f4b8f0e.html
9. https://github.com/amboussetta/RecommenderSystem_MovieLens/blob/master/RecoSystem%20models%20Analysis.R