MovieLens Project

John Downey

1/9/2020

```r
knitr::opts_chunk$set(echo = TRUE)
```

1. Introduction

The MovieLens dataset contains ratings for 10667 movies reviewed by 69878 users. There were broken down to 19 different genres or up to 797 where many movies had more than one genre.

-Objective

The objective of this assignment is to create a Movie Recommendation System based on MovieLens data set provided. Code is provided that will create and “edx” data set for training and a “validation set for testing. The edx set consisted of 9,000,055 observations and the validation set 999,999.

The database hads six(6) columns with the following labels.

- userId: Unique identification number given to each user. It is a Factor variable.
- movieId: Unique movie identification number
- rating: Movie Rating
- timestamp: Date-Time when movie was reviewed
- title: Movie title and Movie Year
- genres: Motion-picture category associated to the film. it is Factor variable.

- Measurement of Success The Residual Mean Squared Error less than 0.8775l will indicate success.

The structure used is listed below"

1. Create a training set (edx) and test set from the code provided.
2. Perform analysis on the test set
3. Create models There are several motels used. The initial model used used the n described in the book.
4. Test the Models
5. Conclusion

The key measurement is RMSE residual means loss means. This is the square root of the sum of the squares of differences between actual and predicted ratings divided by the total number of observations. The model prediction is less than 0.8775.

It should be noted several .R files were created for this project for ease of editin and debugging. Once one section was developed I did not want to go back recreate tje dataframe.

Executive Summary Using the MovieLens database a dateframe for R was created with the ratings of over 10,000 movies by over 70,000 users. This resulted in over 10,000,000 observations.

Using Penalized Least Squares model was optimized

Using liner

Methods

2. Training Set Creation.

2.1 Initial Data Sets.

Code was provided to generate the training and validation data sets. This code is included in the ".R" file but not indluded in this report. This secton of code a dataframe call edx.Rdm is the training set and validation.Rdm is the validation set. To ease developemnt the data sets edx.R and validation.R are stored.

2.2 Data Storage

The provide code was used to create the edx and validaion sets. The code below save the datframes

```
# Save training and testing data s
save(edx, file="edx.Rda")
save(validation, file="validation.Rda")
```

This section creates the test data set and training data set from the edx set.

```
##
## Attaching package: 'dplyr'

## The following objects are masked from 'package:stats':
##
##     filter, lag

## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union

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

## v tibble  2.1.3     v purrr   0.3.3
## v tidyr   1.0.0     v stringr 1.4.0
## v readr   1.3.1     v forcats 0.4.0
```

```
## -- Conflicts ------------------------------------------------------------
------------------- tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```
load("edx.Rda")
load("validation.Rda")
```

A review of the first six(6) rows of the edx and valication dataframe indicates saving and loading the dataframe worked. When running this file the "#" can be removed to

edX Dataframe

```
#head(edx)
```

As can be seen from the code below the edx dataframe is 10,000,000 obserbations

```
dim(edx)

## [1] 9000061       6
```

This is the section of code used to create the training sets and the test sets. This code is not run because it crashed R. When ran as "R" cost the creation of the test set works. For simplicity the code is listed.

```
#Create Train Sets and Test Sets
set.seed(755)
test_index <- createDataPartition(y = edx$rating, times = 1, p = 0.2, list =
FALSE)
train_set <- edx[-test_index,]
test_set <- edx[test_index,]

test_set <- test_set %>%
  semi_join(train_set, by = "movieId") %>%
  semi_join(train_set, by = "userId")
#Save the training set and test set so they will not need to be recalculated.
save(train_set, file="train_set.Rda")
save(test_set, file="test_set.Rda")
```

The code below shows there are 19 different genres and 6 movies with no genres. Movies can have more than one genre.

```
#Names oF Genres
genre_list <-  edx %>% separate_rows(genres, sep = "\\|") %>%
  group_by(genres) %>%
  summarize(count = n()) %>%
  arrange(desc(count))

genre_list

## # A tibble: 20 x 2
##    genres               count
```

```
##    <chr>                 <int>
##  1 Drama               3909401
##  2 Comedy              3541284
##  3 Action              2560649
##  4 Thriller            2325349
##  5 Adventure           1908692
##  6 Romance             1712232
##  7 Sci-Fi              1341750
##  8 Crime               1326917
##  9 Fantasy              925624
## 10 Children             737851
## 11 Horror               691407
## 12 Mystery              567865
## 13 War                  511330
## 14 Animation            467220
## 15 Musical              432960
## 16 Western              189234
## 17 Film-Noir            118394
## 18 Documentary           93252
## 19 IMAX                   8190
## 20 (no genres listed)        6

#Add seperate column for each genre
#Create a new Dataframe adding a seperate column for each genre
edx_genre_col <- edx %>% mutate(Drama=str_detect(genres, "Drama")) %>%
  mutate(Comedy=str_detect(genres, "Comedy")) %>%
  mutate(Action=str_detect(genres, "Action")) %>%
  mutate(Thriller=str_detect(genres, "Thriller")) %>%
  mutate(Adventure=str_detect(genres, "Adventure")) %>%
  mutate(Romance=str_detect(genres, "Romance")) %>%
  mutate(Sci_fi=str_detect(genres, "Sci-Fi")) %>%
  mutate(Crime=str_detect(genres, "Crime")) %>%
  mutate(Fantasy=str_detect(genres, "Fantasy")) %>%
  mutate(Children=str_detect(genres, "Children")) %>%
  mutate(Horror=str_detect(genres, "Horror")) %>%
  mutate(Mystery=str_detect(genres, "Mystery")) %>%
  mutate(War =str_detect(genres, "War")) %>%
  mutate(Animation=str_detect(genres, "Animation")) %>%
  mutate(Musical=str_detect(genres, "Musical")) %>%
  mutate(Western=str_detect(genres, "Western")) %>%
  mutate(Film_Noir=str_detect(genres, "Film-Noir")) %>%
  mutate(Documentary=str_detect(genres, "Documentary")) %>%
  mutate(IMAX=str_detect(genres, "IMAX")) %>%
  mutate(None=str_detect(genres, ""))
```

Because the edx file are so large using any caret package crashes the pc becasue of the length of time it takes to perform the analysis, therefore smaller datasets need to be created.

```r
#Save the dataframe with the genres broken out.
save(edx_genre_col, file = "edx_genre_col.Rda")
#remove all column accept movie title
edx_genre_logical <-  subset(edx_genre_col, select= -c(userId, movieId,
timestamp, title, genres))

#Create a 10,000 observation dataframe for testing
edx_genre_logical_10 <- edx_genre_logical[sample(nrow(edx_genre_logical),
10000),]
save(edx_genre_logical_10, file = "edx_genre_logical_10.Rda")
dim(edx_genre_logical_10)

## [1] 10000    21

#Create a 1,000 observation dataframe for testing
edx_genre_logical_1 <- edx_genre_logical[sample(nrow(edx_genre_logical),
1000),]
save(edx_genre_logical_10, file = "edx_genre_logical_1.Rda")
dim(edx_genre_logical_10)

## [1] 10000    21
```

Analysis This section is where the we analyze the data.

First we need to know the number of unique users, movies and genres that are in the dataset.

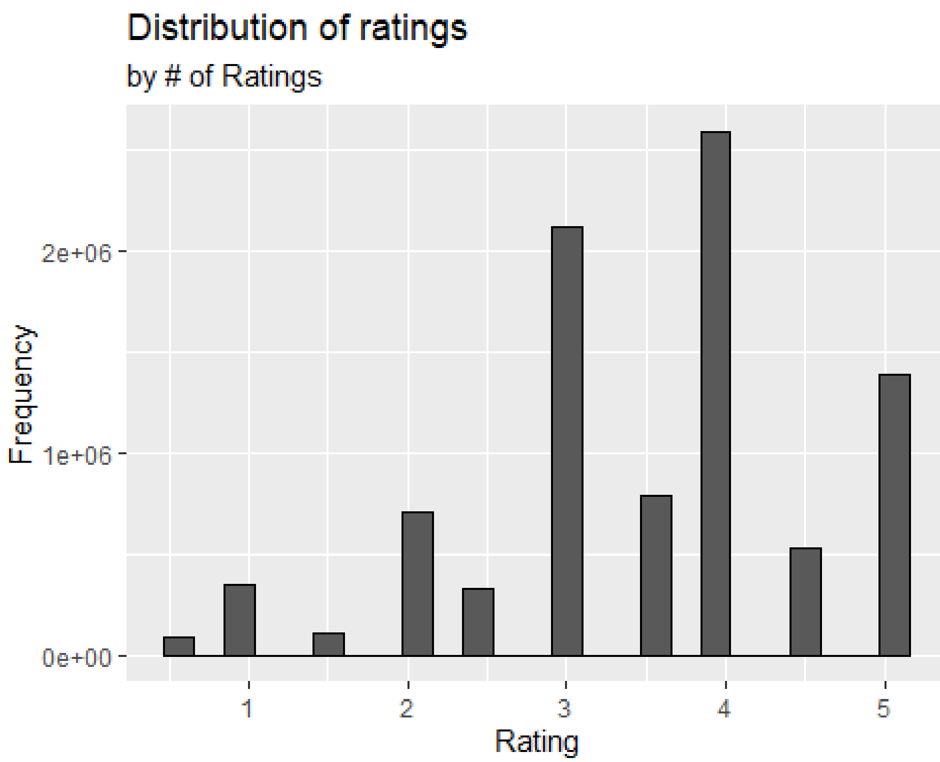
```r
# Summary of the number of users and movies.
edx %>%
  summarize(n_users = n_distinct(userId),
            n_movies = n_distinct(movieId),
            n_genres = n_distinct(genres))

##   n_users n_movies n_genres
## 1   69878    10677      797
```

This section show the mean rating 3.5 and the Median rating is 4.0 We can see over half the movies are good.

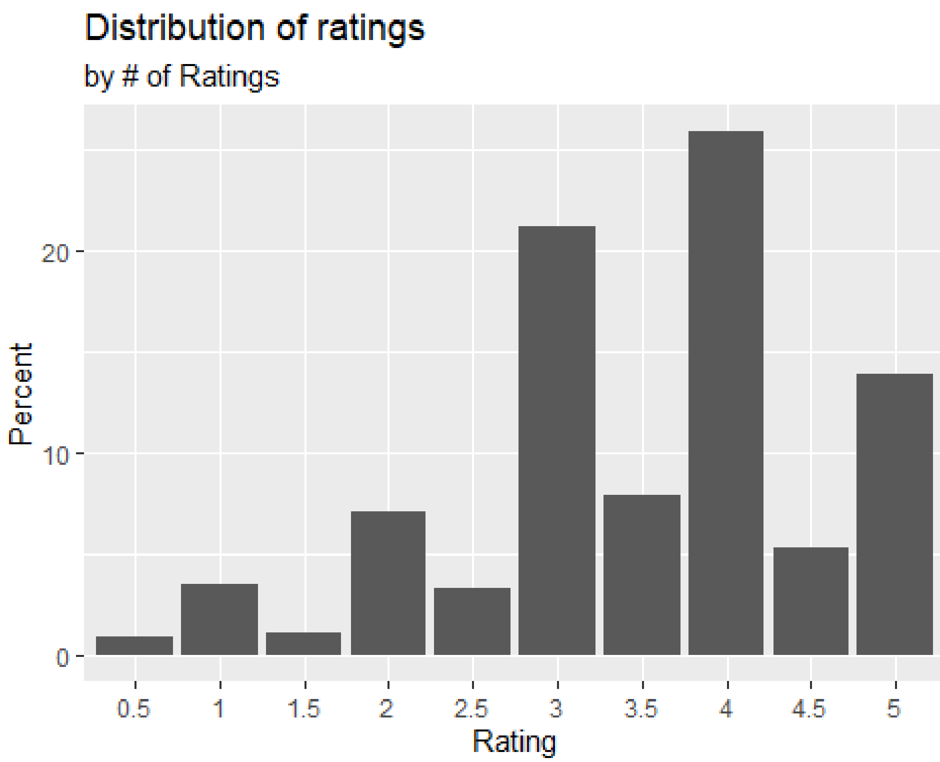
```r
summary(edx$rating)

##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   0.500   3.000   4.000   3.512   4.000   5.000
```

The plots below shows over half the movies are good.

```r
edx %>% ggplot(aes(rating)) +
  geom_histogram(bins = 25, color = "black") +
  labs(title = "Distribution of ratings",
       subtitle = "by # of Ratings",
       x = "Rating",
       y = "Frequency")
```




```
#Graph of percentage each rating recieved
x <- round(table(edx$rating)/10000000*100,1)

x1 <- as.data.frame(x)

ggplot(data = x1, aes(Var1, Freq)) +
  geom_bar( stat = "identity") +
  labs(title = "Distribution of ratings",
     subtitle = "by # of Ratings",
     x = "Rating",
     y = "Percent")
```




```
# This shows the movies that have move than 3000 reviews

movieId_count <- edx %>% count(userId) %>% arrange(desc(n))
print(filter(movieId_count, n > 3000))
```

```
## # A tibble: 11 x 2
##    userId     n
##     <int> <int>
##  1  59269  6637
##  2  67385  6376
##  3  14463  4637
##  4  68259  4056
##  5  27468  4018
##  6  19635  3740
##  7   3817  3736
##  8  63134  3390
##  9  58357  3318
## 10  27584  3139
## 11   6757  3086
```

Top Movies with more 20,000 users rating the movie.

```
movie_title <- edx %>% group_by(title) %>% count() %>% arrange(desc(n))
filter(movie_title, n > 20000)
```

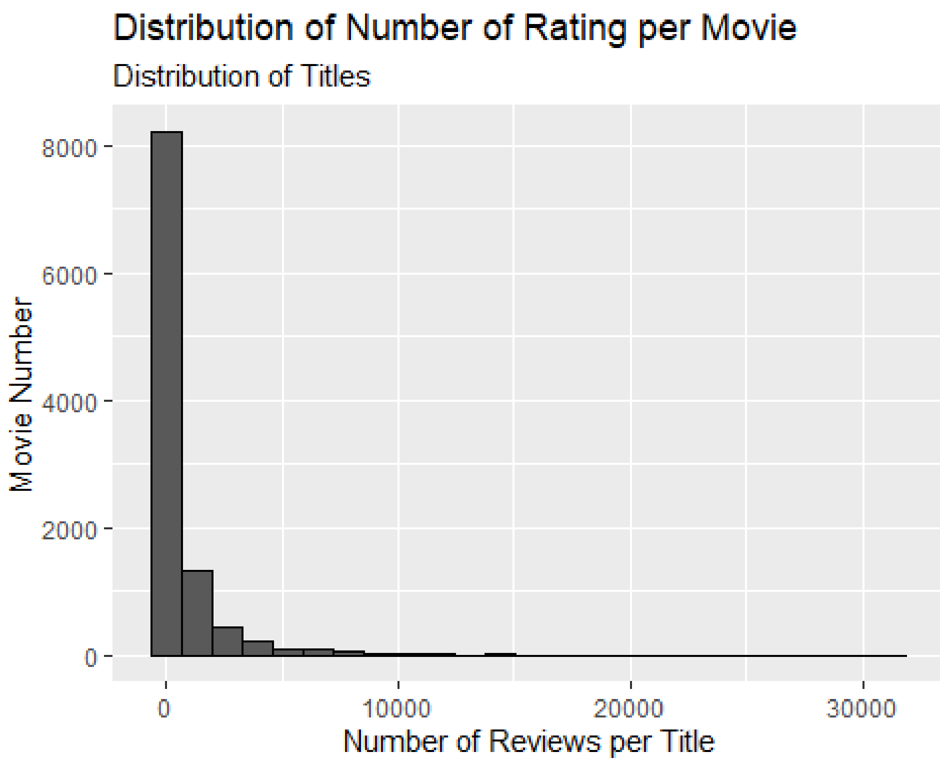
```
## # A tibble: 25 x 2
## # Groups:   title [25]
```

```
##    title                                                          n
##    <chr>                                                      <int>
##  1 Pulp Fiction (1994)                                        31336
##  2 Forrest Gump (1994)                                        31076
##  3 Silence of the Lambs, The (1991)                           30280
##  4 Jurassic Park (1993)                                       29291
##  5 Shawshank Redemption, The (1994)                           27988
##  6 Braveheart (1995)                                          26258
##  7 Terminator 2: Judgment Day (1991)                          26115
##  8 Fugitive, The (1993)                                       26050
##  9 Star Wars: Episode IV - A New Hope (a.k.a. Star Wars) (1977) 25809
## 10 Batman (1989)                                              24343
## # ... with 15 more rows
```

Historgram of number of movie ratings per movie.

We can see only 1907 out of 10067 were rated by more than 1,000 people.

```
dim(filter(movie_title, n > 1000))

## [1] 1907    2

movie_Id <- edx %>% group_by(movieId) %>% count() %>% arrange(desc(n))

movie_Id %>% ggplot(aes(n)) +
  geom_histogram(bins = 25, color = "black") +
  labs(title = "Distribution of Number of Rating per Movie",
       subtitle = "Distribution of Titles",
       x = "Number of Reviews per Title",
       y = "Movie Number")
```



Genre Analysis

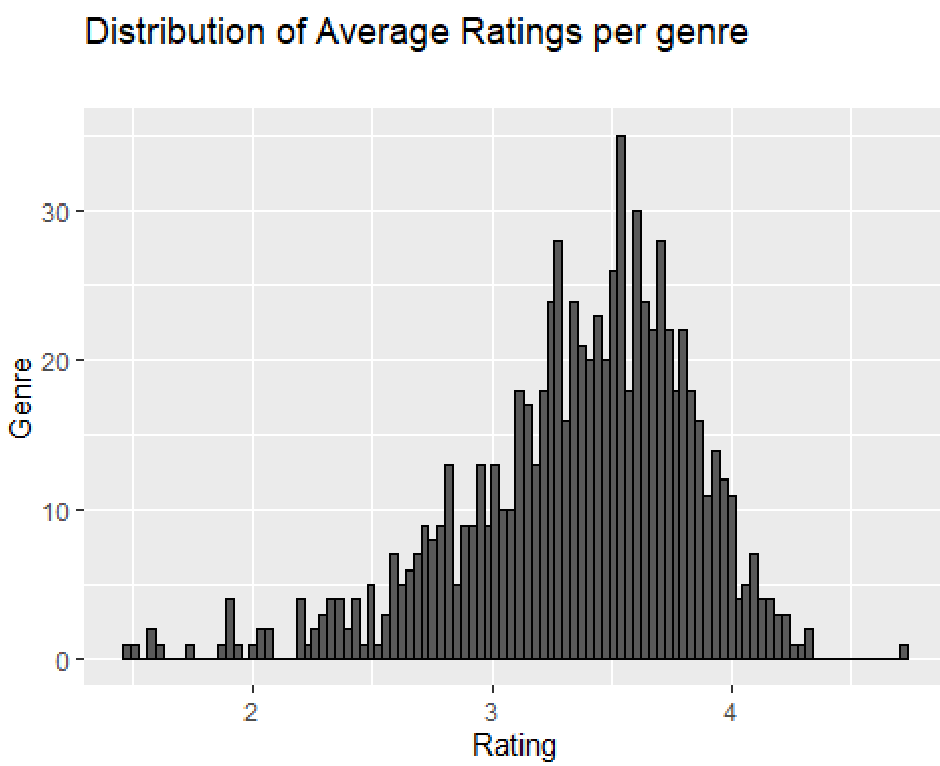
```
#This section provides the average rating and SD per genres.
#Group by genres
genres_grouped <- group_by(edx, genres)

genre_mean <- summarise(genres_grouped, Avg_genre = mean(rating), sd_genre =
sd(rating))
head(genre_mean)
```

```
## # A tibble: 6 x 3
##   genres                                          Avg_genre sd_genre
##   <chr>                                               <dbl>    <dbl>
## 1 (no genres listed)                                   3.5     1.14
## 2 Action                                               2.93    1.08
## 3 Action|Adventure                                     3.66    1.07
## 4 Action|Adventure|Animation|Children|Comedy           3.97    0.779
## 5 Action|Adventure|Animation|Children|Comedy|Fantasy   2.94    0.961
## 6 Action|Adventure|Animation|Children|Comedy|IMAX      3.27    0.944
```

Analysis of ratings based on genre In this section the we can see genre can indicate the movie rating. There were 43 genres with ratings of 4 or greater. From the data below genres can be a good indicator of a good movie.

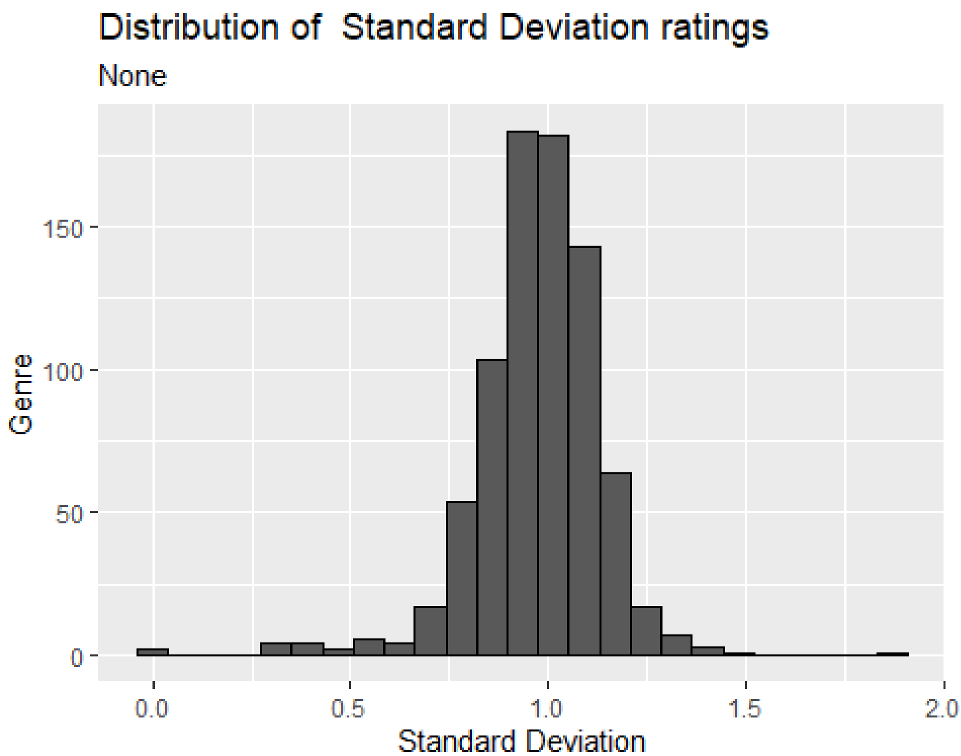
```
count(filter(genre_mean, Avg_genre >= 3 & Avg_genre < 4))
```

```
## # A tibble: 1 x 1
##       n
##   <int>
## 1   594

count(filter(genre_mean, Avg_genre >= 4))

## # A tibble: 1 x 1
##       n
##   <int>
## 1    43

#Plot to show distrubution  of the averge ratings per genre

genre_mean %>% ggplot(aes(Avg_genre)) +
  geom_histogram(bins = 100, color = "black") +
  labs(title = "Distribution of Average Ratings per genre",
       subtitle = "",
       x = "Rating",
       y = "Genre")
```

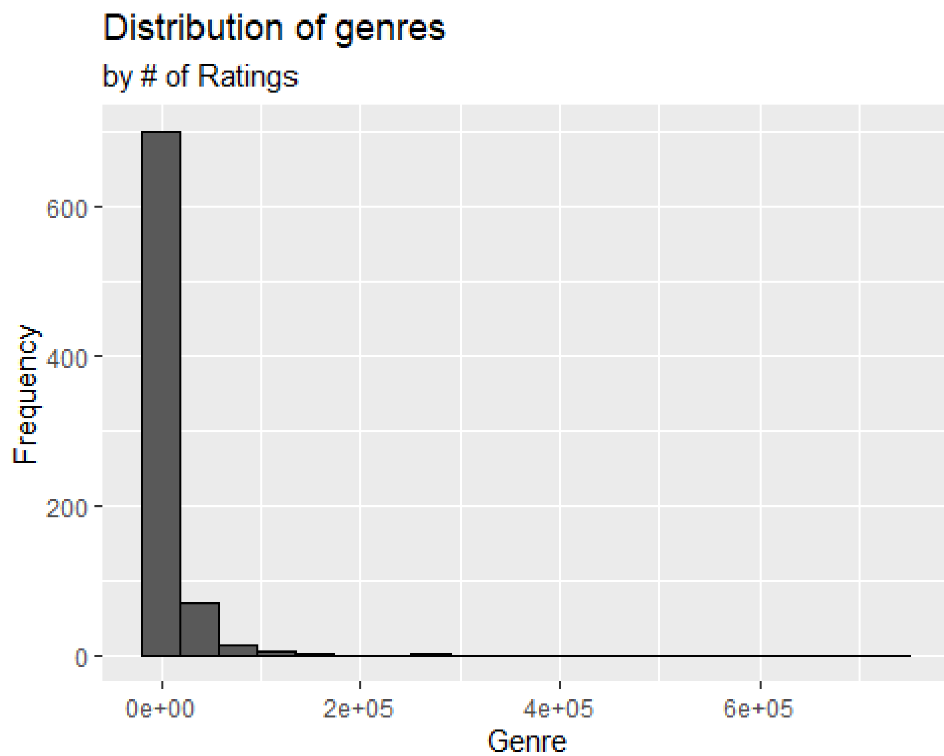



```
#This plot shows the distrubuion of the SD of the rating per genres.
genre_mean %>% ggplot(aes(sd_genre)) +
  geom_histogram(bins = 25, color = "black") +
  labs(title = "Distribution of  Standard Deviation ratings",
       subtitle = "None",
```

```
       x = "Standard Deviation",
       y = "Genre")
```




```
#Genarate a plot showing the
genres_list <- as.data.frame(table(edx$genres))

head(genres_list)

##                                                     Var1  Freq
## 1                                     (no genres listed)     6
## 2                                                 Action 24575
## 3                                       Action|Adventure 68611
## 4           Action|Adventure|Animation|Children|Comedy  7438
## 5 Action|Adventure|Animation|Children|Comedy|Fantasy   191
## 6    Action|Adventure|Animation|Children|Comedy|IMAX    62

genres_list %>% ggplot(aes(Freq)) +
  geom_histogram(bins = 20, color = "black") +
  labs(title = "Distribution of genres",
       subtitle = "by # of Ratings",
       x = "Genre",
       y = "Frequency")
```



3. Development of recommendation model

The code to generate the training sets and the test sets were provided with the problem. Therefore, it was not required to create test and validation sets.

An exploratory test I tried to use CARET LM function and and randomForest function to test the data set Error: cannot allocate vector of size 67.1 Gb. The lm function was also tested using genre factors only. This required

My first step was to try to attempt to run the code provided to create the two data sets. After running the code, the data was saved as a Rda file. An attempt to tried running the lm machine learning algorithm but I received an error message. In order to avoid the error message I took a subset of the dataset.
The success of the model is measured using RMSE. The formula for is located in ML_3.R line 23

Load test and training dataframes from previously

```
load("test_set.Rda")
load("train_set.Rda")
```

We can see the average for movie is 3.51

```
#Test to ensure table command exracted all the ratings

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

```
## [1] 3.512496
```

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

We will first use the average rating and compare it to the test set. We the RMSE is greater than the required RMSE for maximum points.

```
naive_rmse <- RMSE(test_set$rating, mu_hat)
naive_rmse
```

```
## [1] 1.060428
```

```
rmse_results <- data_frame(method = "Just the Average", RMSE = naive_rmse)
```

```
## Warning: `data_frame()` is deprecated, use `tibble()`.
## This warning is displayed once per session.
```

```
rmse_results
```

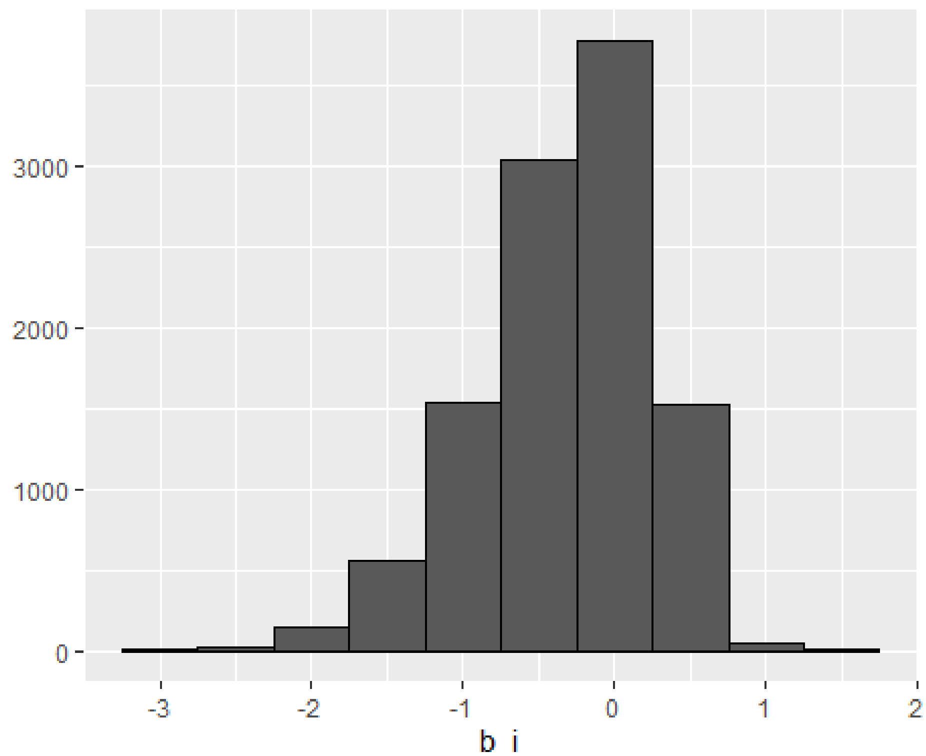
```
## # A tibble: 1 x 2
##   method            RMSE
##   <chr>            <dbl>
## 1 Just the Average  1.06
```

Next we add take into account the individual movie ID

```
#Mean Ration Calcuation
mu <- mean(train_set$rating)
# Add inidvidules movie ID to Model pg 650
movie_avgs <- edx %>%
  group_by(movieId) %>%
  summarize(b_i = mean(rating - mu_hat))
```

From the histogram we can see each movie has b value.

```
# pg 650
movie_avgs %>% qplot(b_i, geom ="histogram", bins = 10, data = ., color =
I("black"))
```




```
# pg 651
```

3.1. Model: Average Rating

This is the simplest model. This model is an average of all movies. This is not a good model because all movies are rated the same. (ML_3, l 30)

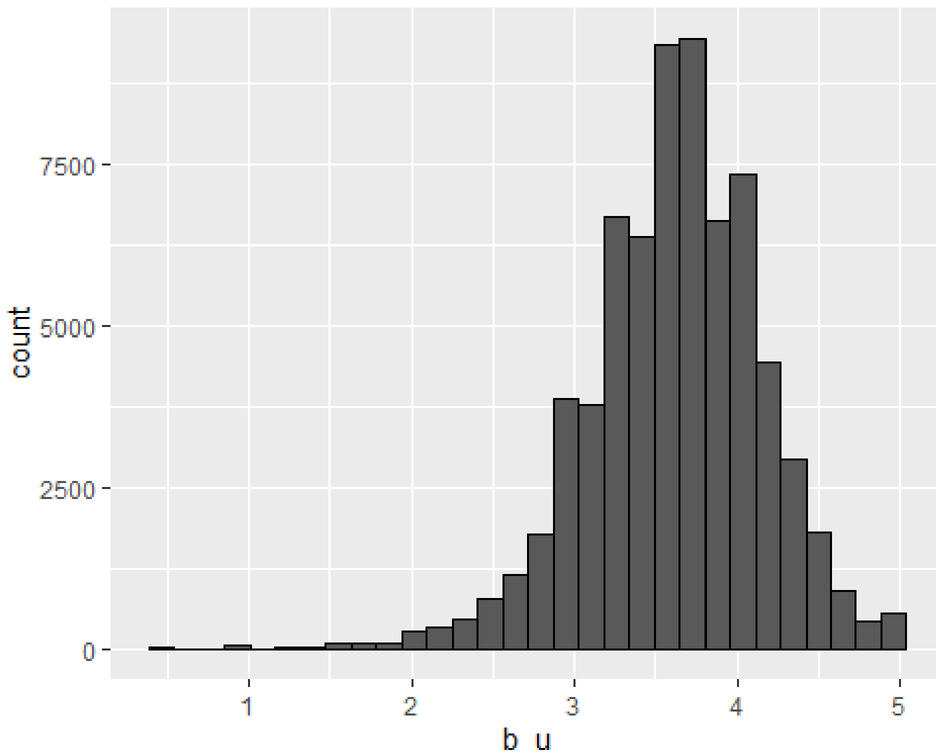
```
predicted_ratings <- mu + train_set %>%
  left_join(movie_avgs, by='movieId') %>%
  .$b_i

model_1_rmse <- RMSE(predicted_ratings, test_set$rating)

## Warning in true_ratings - predicted_ratings: longer object length is not a
## multiple of shorter object length

model_1_rmse

## [1] 1.166498

rmse_results <- bind_rows(rmse_results,
                          data_frame(method="Movie Effect Model",
                                     RMSE = model_1_rmse))
rmse_results %>% knitr::kable()
```

method	RMSE
Just the Average	1.060428

Movie Effect Model   1.166498

Histogram of the userId factor. We can see the user does effect the influence the outcome.

```
#User ID Effect Pg 650
#
test_set %>%
  group_by(userId) %>%
  summarize(b_u = mean(rating)) %>%
  filter(n()>=100) %>%
  ggplot(aes(b_u)) +
  geom_histogram(bins = 30, color = "black")
```



3.2 Add movie ID

```
#pg 652
user_avgs <- train_set %>%
  left_join(movie_avgs, by='movieId') %>%
  group_by(userId) %>%
  summarize(b_u = mean(rating - mu - b_i))

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

```r
model_2_rmse <- RMSE(predicted_ratings, validation$rating)
```

```
## Warning in true_ratings - predicted_ratings: longer object length is not a
## multiple of shorter object length
```

```r
model_2_rmse
```

```
## [1] 1.229485
```

```r
rmse_results <- bind_rows(rmse_results,
                          data_frame(method="Movie + Users Effect Model",
                                     RMSE = model_2_rmse))
rmse_results %>% knitr::kable()
```

method	RMSE
Just the Average	1.060428
Movie Effect Model	1.166498
Movie + Users Effect Model	1.229485

Finally we add regularization

```r
#pg 659
#Movie effect and User Effect with Regualization using the test_set data
lambdas <- seq(0, 10, 0.25)

rmses <- sapply(lambdas, function(l){
  mu <- mean(train_set$rating)
  b_i <-  train_set %>%
    group_by(movieId) %>%
    summarize(b_i = sum(rating - mu)/(n()+l))
  b_u <- train_set %>%
    left_join(b_i, by="movieId") %>%
    group_by(userId) %>%
    summarize(b_u = sum(rating - b_i - mu)/(n()+l))
  predicted_ratings <-
    test_set %>%
    left_join(b_i, by = "movieId") %>%
    left_join(b_u, by = "userId") %>%
    mutate(pred = mu + b_i + b_u) %>%
    pull(pred)
  return(RMSE(predicted_ratings, test_set$rating))
})
```

Several test were conducted using genres to predict the ratings. In order to run the code, only 10,000 obsevation could be used. The 19 genres were converted to 19 dummy variables. Random forests was used to model the data.

```r
#Using a 10,000
#train_10 <-  train(rating ~ ., data = edx_genre_logical_10, method="rf")
```

```
#train_10
#predict_10 <- predict(train_10, edx_genre_logical_1, type = "raw")
```

This section we are using genre to model ratings.

```
#Using a 10,000
train_10 <-  train(rating ~ ., data = edx_genre_logical_10, method="rf")
train_10
predict_10 <- predict(train_10, edx_genre_logical_1, type = "raw")
```

Below is the RMSE for using genre only. Genre is the variable independent of the userId and movie name. This can be used to predict ratings of new movies.

The RMSE was 1.017262. This code does select not to run because of the excessively large coumpution time.

```
RMSE(predict_10, edx_genre_logical_1$rating)
```

Other model from the caret package were attempted but crashed the PC or never stopped running. A example of r code to test several different models is listed below.

```
models <- c("lda",  "naive_bayes",  "svmLinear",
            "gamboost",  "gamLoess", "qda",
            "knn", "kknn", "loclda", "gam",
            "rf", "ranger",  "wsrf", "Rborist",
            "avNNet", "mlp", "monmlp",
            "adaboost", "gbm",
            "svmRadial", "svmRadialCost", "svmRadialSigma")

models <- c("lm", "knn", "kknn", "brnn", "randomGLM")
models <- c( "lm", "knn")
models <- c("rf", "lm")

str(models)

fits1 <- sapply(models, function(model){
  print(model)
  train(rating ~., method = models, data = edx_genre_logical_10)
  predict(models, data = edx_genre_logical_10)
  return(RMSE(predict_10, edx_genre_logical_10$rating))
})
```

Results

This sections shows the results of the above analysis.

Comparison of RMSES to the Grading Rubric

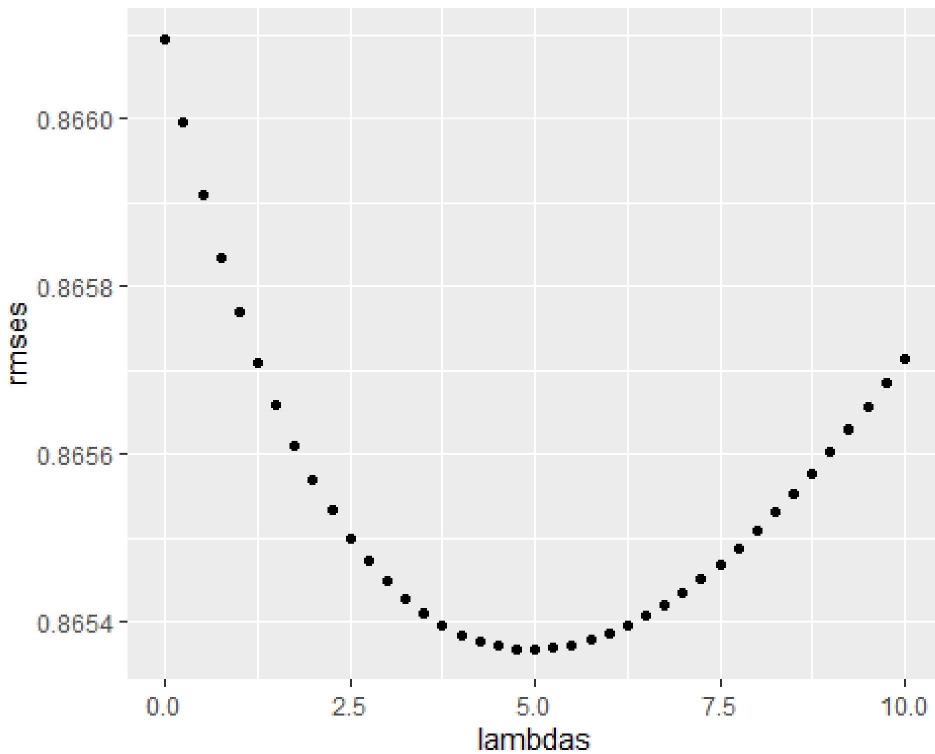
```
min(rmses)

## [1] 0.8653664
```

```r
min(rmses) <= 0.87750
```

```
## [1] TRUE
```

Plot showing lamba tuning parameter vs. RMSE

```r
qplot(lambdas,rmses)
```



Conclusion

Based on the RMSE results above the model was rerun using edx as the training set and validation as the test set. the minimum RSME was 0.8649587. This is 0.000587 greater the RMSE value that grades with the maximum points.

Use a genre only model where 20 dummy variables were created from genres column. The model was run with only 5000 obsevations. This models RMSE was 1.01.

Forward Looking

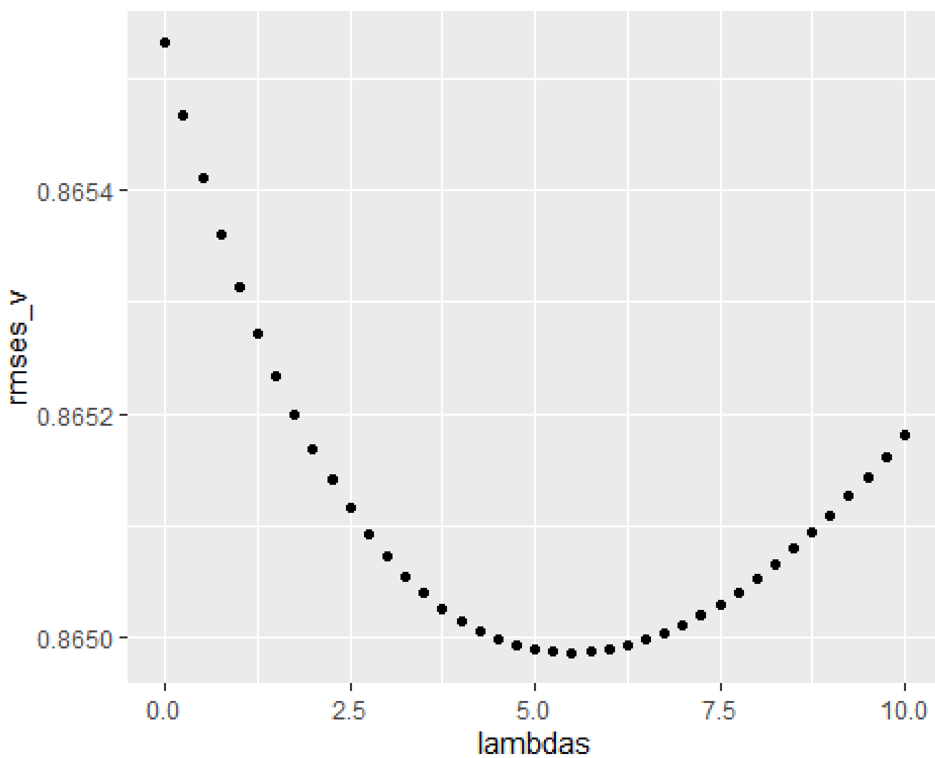
To use advanced modeling package such as caret requires a much more powerful computer.

```r
#Run the same function using the edx and validation datasets lamba calculated above.
rmses_v <- 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) %>%
    pull(pred)
  return(RMSE(predicted_ratings, validation$rating))
})
```

This is the plot that shows the minimum lamba.

```
qplot(lambdas,rmses_v)
```




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

## [1] 5.5

min(rmses_v)

## [1] 0.8649857

min(rmses_v) <= 0.8649
```

```
## [1] FALSE

min(rmses_v) <= 0.86499

## [1] TRUE
```