#title: “Capstone MovieLens Project”

#author: Maurice Rankin

#date: November 19, 2019

**Introduction**

The goal of Project the MovieLens project was to first train a machine learning algorithm using the inputs in the MovieLens 10M dataset to predict movie ratings in the validation set which consisted of 10% of the MovieLens data. My goal during this project was to become familiar with the MovieLens data and to try to find a way to create an algorithm that predicts movie ratings using the validation set using RMSE.

**Here are the key steps I performed during this project:**

1. Create the edx set and validation set

```
if(!require(tidyverse)) install.packages("tidyverse", repos = "http://cran.us.r-projec
t.org")
if(!require(caret)) install.packages("caret", repos = "http://cran.us.r-project.org")
if(!require(data.table)) install.packages("data.table", repos = "http://cran.us.r-proj
ect.org")

dl <- tempfile()
download.file("http://files.grouplens.org/datasets/movielens/ml-10m.zip", dl)

ratings <- fread(text = gsub("::", "\t", readLines(unzip(dl, "ml-10M100K/ratings.da
t"))),
                 col.names = c("userId", "movieId", "rating", "timestamp"))

movies <- str_split_fixed(readLines(unzip(dl, "ml-10M100K/movies.dat")), "\\::", 3)
colnames(movies) <- c("movieId", "title", "genres")
movies <- as.data.frame(movies) %>% mutate(movieId = as.numeric(levels(movieId))[movie
Id],
                                           title = as.character(title),
                                           genres = as.character(genres))

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

2. Ensure all data needed to create my projection is included in the validation set.

```
set.seed(1, sample.kind="Rounding")
test_index <- createDataPartition(y = movielens$rating, times = 1, p = 0.1, list = FAL
SE)
edx <- movielens[-test_index,]
temp <- movielens[test_index,]
validation <- temp %>%
     semi_join(edx, by = "movieId") %>%
     semi_join(edx, by = "userId")
```

3. Clean the data and validation sets and ensure all data is tidy.

```
removed <- anti_join(temp, validation)
edx <- rbind(edx, removed)

rm(dl, ratings, movies, test_index, temp, movielens, removed)

validation_CM <- validation
validation <- validation %>% select(-rating)
```

4. View data and validation sets to get a feel of what I am working with.

```
head(edx)
summary(edx)
```

5. Get total observations and define ratings frequency

```
total_observation <- length(edx$rating) + length(validation$rating)

RMSE <- function(true_ratings, predicted_ratings){
  sqrt(mean((true_ratings-predicted_ratings)^2,na.rm=T))
}
```

6. Modifying genres variable

```
edx <- edx %>% mutate(year = as.numeric(str_sub(title,-5,-2)))
validation <- validation %>% mutate(year = as.numeric(str_sub(title,-5,-2)))
validation_CM <- validation_CM %>% mutate(year = as.numeric(str_sub(title,-5,-2)))
```

7. Modifying genres variable

```
split_edx  <- edx  %>% separate_rows(genres, sep = "\\|")
split_valid <- validation   %>% separate_rows(genres, sep = "\\|")
split_valid_CM <- validation_CM  %>% separate_rows(genres, sep = "\\|")
```

8. Viewing data and getting summary of edx and split edx

```
head(edx)
head(split_edx)
summary(edx)
```

9. Finding unique users and movies in edx, then pulilng movies ratings by genre

```
edx %>% summarize(n_users = n_distinct(userId), n_movies = n_distinct(movieId))

genre_rating <- split_edx%>%
  group_by(genres) %>%
  summarize(count = n()) %>%
  arrange(desc(count))
```

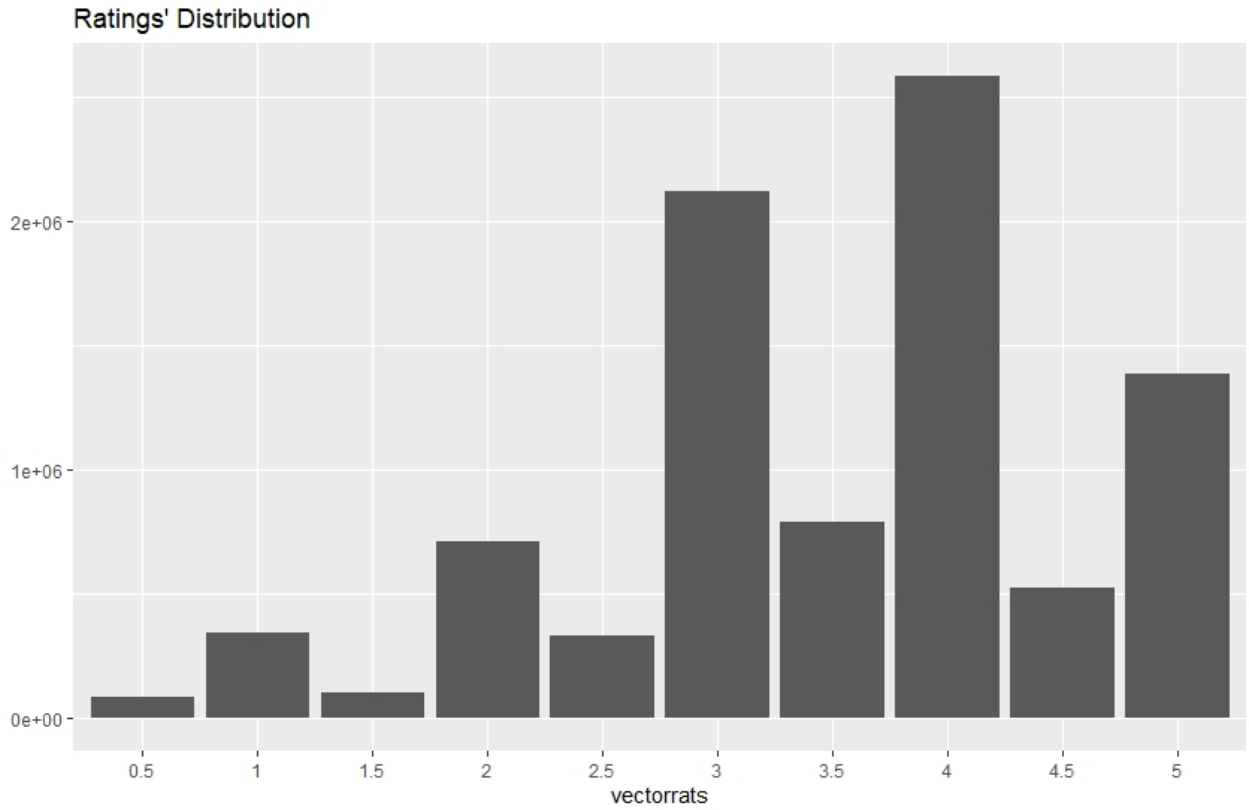
9. Adding extra libraires to help with visualization and analysis

```
library(ggplot2)
library(lubridate)
```

10. Creating a vector for unique ratings and plotting it

```
vectorrats <- as.vector(edx$rating)
unique(vectorrats)
vectorrats <- vectorrats[vectorrats != 0]
vectorrats <- factor(vectorrats)
qplot(vectorrats) +
  ggtitle("Ratings' Distribution")
```

**Ratings' Distribution vs Vector Ratings**



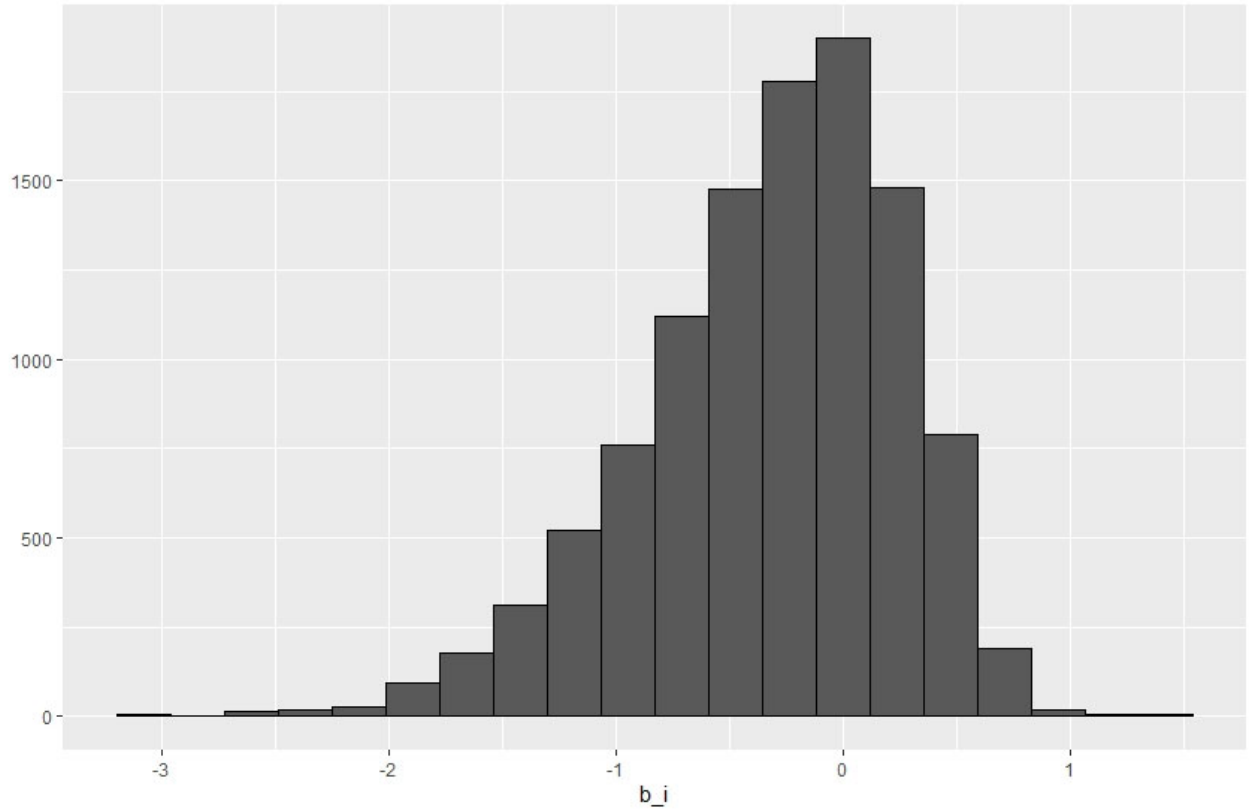
11. Creating a simple mode for MU

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

12. Summarizing the movie effect

```
movieavgnorm <- edx %>%
  group_by(movieId) %>%
  summarize(b_i = mean(rating - mu))
movieavgnorm %>% qplot(b_i, geom ="histogram", bins = 20, data = ., color = I("blac
k"))
```

**Movie Effect Distribution**




13. Creating a base model and testing the reuslts

```
bsl_rmse <- RMSE(validation_CM$rating,mu)

bsl_rmse
rsmresults <- data_frame(method = "Using mean only", RMSE = bsl_rmse)
rsmresults
```

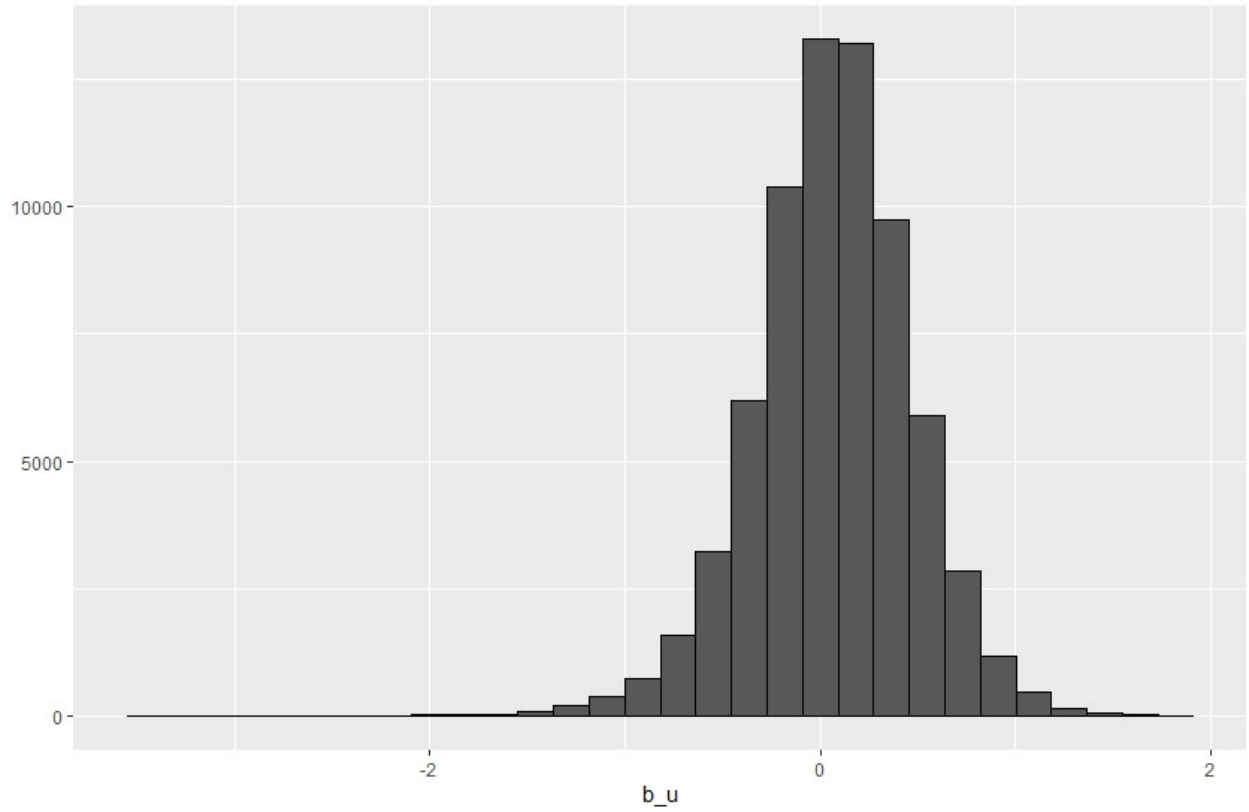
14. Showing my prediction with for my 1st Model showing the Movie Effect

```
predictedratingsmovienorm <- validation %>%
  left_join(movieavgnorm, by='movieId') %>%
  mutate(pred = mu + b_i)
Model1RMSE <- RMSE(validation_CM$rating,predictedratingsmovienorm$pred)
rsmresults <- bind_rows(rsmresults,
                          data_frame(method="Movie Effect Model",
                                     RMSE = Model1RMSE ))
rsmresults %>% knitr::kable()
```

15. Creating and Plotting a historgram to see how the user effect changes RSME of my prediction

```
useravgsnrm <- edx %>%
  left_join(movieavgnorm, by='movieId') %>%
  group_by(userId) %>%
  summarize(b_u = mean(rating - mu - b_i))
useravgsnrm %>% qplot(b_u, geom ="histogram", bins = 30, data = ., color = I("black"))
```

**User Effect Distribution**




16. Creating a model that accounts for User and Movie Effect

```
predictedratingsusernorm <- validation %>%
  left_join(movieavgnorm, by='movieId') %>%
  left_join(useravgsnrm, by='userId') %>%
  mutate(pred = mu + b_i + b_u)
```

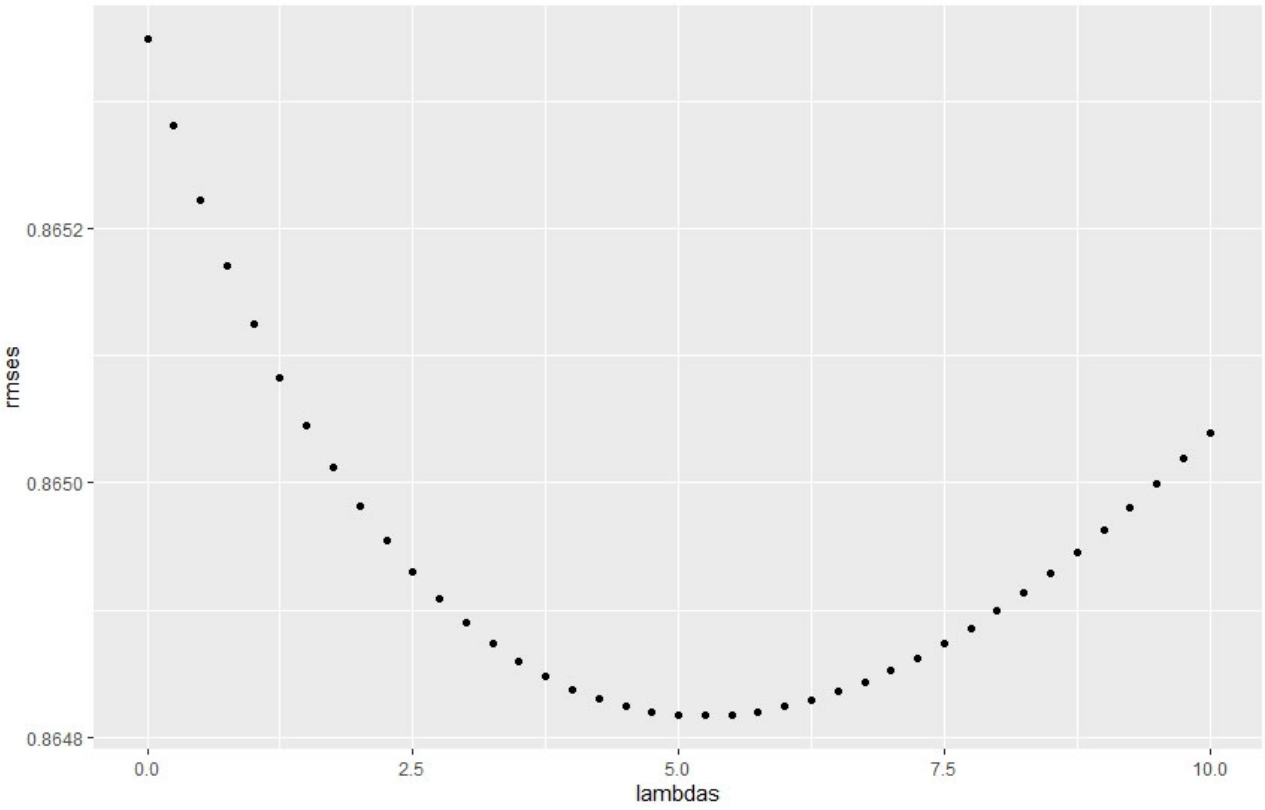
17. Testing result of 2nd Model showing the User and Movie Effect

```
Model2RMSE <- RMSE(validation_CM$rating,predictedratingsusernorm$pred)
rsmresults <- bind_rows(rsmresults,
                          data_frame(method="Movie and User Effect Model",
                                     RMSE = Model2RMSE ))
rsmresults %>% knitr::kable()
```

18. Creating a function to test lambda using User and movieID's accounting for bias/noise

```
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(validation_CM$rating,predicted_ratings))
})

qplot(lambdas, rmses)
```

**RMSE vs Lambdas Plot**




19. Using best lambda to normalize User effect and MOvie Effect

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

movieavgsr <- edx %>%
  group_by(movieId) %>%
  summarize(b_i = sum(rating - mu)/(n()+lambda), n_i = n())

useravgsr <- edx %>%
  left_join(movieavgsr, by='movieId') %>%
  group_by(userId) %>%
  summarize(b_u = sum(rating - mu - b_i)/(n()+lambda), n_u = n())
```

20. Predicting new ratings with noise adjustments

```
predicted_ratings_reg <- validation %>%
  left_join(movieavgsr, by='movieId') %>%
  left_join(useravgsr, by='userId') %>%
  mutate(pred = mu + b_i + b_u) %>%
  .$pred
```

21. Testing my 3rd model and saving results

```
Model3rmse <- RMSE(validation_CM$rating,predicted_ratings_reg)
rsmresults <- bind_rows(rsmresults,
                          data_frame(method="Regularized Movie and User Effect Model",
                                     RMSE = Model3rmse ))
rsmresults %>% knitr::kable()
```

**Final Results of My Movie Ratings Prediction for the MoveiLens Project**

**1st Model RMSE: 0.9439087**

**2nd Model RMSE: 0.8653488**

**3rd Model RMSE: 0.8653488**

**Analysis Method**

I chose to clean up the edX data set by removing the ratings column first and getting the total number of observations to ensure the data was tidy. Next, I used RMSE initially to get an idea of the frequency for the true ratings and predictions. I then modified the year and genres column to split the data into validation sets (which took some time to process). By creating summaries of unique users, I was able to create vectors for the ratings and use visualization to solve for a MU (simple model). By creating a histogram of the movie norm, I was able to see and adjust for what I call the Movie Effect. I created a base line model and tested the results using a prediction that added MU and the Movie Effect (b_i). This first model only gave me an RSME of 0.9439087 (which would not give me points in our grading rubric.I had to do better). I plotted a second histogram showing the userID and movie average norm. This showed there was another effect affecting the ratings system. I called this the User Effect. I created a 2nd RMSE model that accounted for both Movie and User Effect. The RMSE after adding both Movie and User Effect got the RMSE down to 0.8653488 (much better that the first model but still did not get me the points I desired.want to be as close to 25 points as possible). I did research online and found out about a Netflix challenge in where they were able to use cross validation to improve their ratings model to regularize my results. To apply regularization to my new model, I used lambdas and tested a few ranges for the sequence. This was a very painful process because it took 3 to 5 minutes to run each simulation. I finally settle on a sequence of (0, 10, 0.25). I create a function using sapply to add both my user bias, movie bias, and my ratings prediction to find the lambdas vs RMSE values that gave the least amount of RMSE (which was a lambda of 5.25). Finally, I was able to use the lambda to normalize my prediction and adjust my algorithm. I created a third model to test, which gave me an RMSE of 0.8653488 (best model and gets me at least 20 points according to our grading rubric).

**Conclusion and Results**

In conclusion, I found this project very challenging mostly because I had issues getting the PDF's to print in R Studio. The second challenge was spliting the data sets as to it took long periods of time and my computer was runing at 98% memory capacity. Finally, I also found it challenging to test for lambdas and building the RMSE models and predictions. It too some time via trial an error before I found an optimal sequence to give a smooth graph for the lambda vs RMSE plot. Building the models was the morst rewarding part of the project because I was able to plot my results and see how the user and movei effets changed the overal RMSE of my algorythm.