MovieLens Edx Capstone Project

Marina Yamasaki

1/8/2021

1. Introduction:

The MovieLens dataset is sourced from the online movie recommender service MovieLens. The dataset includes ~10M ratings for ~ 11K movies from ~72K users. The following report aims to evaluate movie rating behavior and build models to predict movie ratings. Understanding rating behavior enhances the accuracy and ability of the service to give users movie recommendations. Before building the models, the dataset will be reviewed to understand the variables and ensure that they are in the correct format. The dataset will be split into training and validation to prevent overfitting of the models. Exploratory data analysis will be performed to evaluate trends in the data, as well as, relationships between different variables and movie ratings. Models produced will be evaluated based on root mean squared error (RMSE), with the ultimate goal of producing a model with RMSE < 0.86490.

Building the Data Set Install and load relevant packages and libraries:

```
if(!require(dplyr))
  install.packages("dplyr", repos = "http://cran.us.r-project.org")
if(!require(tidyverse))
  install.packages("tidyverse", repos = "http://cran.us.r-project.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-project.org")
if(!require(lubridate))
  install.packages('lubridate', repos = "http://cran.us.r-project.org")
if(!require(recosystem))
  install.packages('recosystem', repos = "http://cran.us.r-project.org")
if(!require(rmarkdown))
  install.packages('rmarkdown', repos = "http://cran.us.r-project.org")
if(!require(tinytex))
  install.packages('tinytex', repos = "http://cran.us.r-project.org")
if(!require(kableExtra))
  install.packages('kableExtra', repos = "http://cran.us.r-project.org")
library('tidyverse')
library('caret')
library('data.table')
library('dplyr')
library('stringr')
library('lubridate')
library('rmarkdown')
library('recosystem')
library('rmarkdown')
```

```
library('tinytex')
library('kableExtra')
```

The following code was provided by the class to download the movielens data. The data is partitioned so that 10% of the dataset will be held for validation and 90% of the dataset will be used to develop models.

```
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.dat"))),
                 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(movieId),
                                           title = as.character(title),
                                           genres = as.character(genres))
movielens <- left_join(ratings, movies, by = "movieId")
set.seed(1, sample.kind="Rounding")
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)
edx <- rbind(edx, removed)
rm(dl, ratings, movies, test_index, temp, movielens, removed)
```

The edx (training) dataset will be further partitioned to facilitate parameter tuning and model evaluation during the model development process. The validation dataset will only be used to assess the performance of the final model.

```
set.seed(1, sample.kind="Rounding")
train_index <- createDataPartition(y = edx$rating, times = 1, p = 0.2, list = FALSE)
temp_train <- edx[-train_index,]
temp_test <- edx[train_index,]

edx_train <- temp_train %>%
  semi_join(edx, by = "movieId") %>%
  semi_join(edx, by = "userId")
edx_test <- temp_test %>%
  semi_join(edx, by = "movieId") %>%
  semi_join(edx, by = "userId")
```

Evaluation of the dataset: The full movielens dataset contains 10,000,054 rows and 6 columns. The fields included in the dataset are: userId, movieId, rating, timestamp, title, and genres. The dataset includes 10,677 movies and 69,878 users. The movie rating scale is between 0.5 and 5 in 0.5 increments. The edx (training) dataset contains 90% of the original dataset.

```
head(edx) %>% kable %>% kable_styling(latex_options="scale_down")
```

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

```
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. Methods/Analysis:

Data Cleaning Some data cleaning and transformation was done on the data to facilitate analysis and modeling. The timestamp and year_rated fields were converted to date fields. A new column called year_movie was created. The year_movie field represents the movie release year and was extracted from the movie title field using a regularized expression.

```
edx_train <- edx_train %>% mutate(timestamp = as_datetime(timestamp),
                                  year_rated = year(as_datetime(timestamp)),
                                  year_movie = as.numeric(
                                    str_extract(title,"(?<=\\()([0-9]{4})(?=\\)$)"))
                                 )
edx_test <- edx_test %>% mutate(timestamp = as_datetime(timestamp),
                                year_rated = year(as_datetime(timestamp)),
                                year_movie = as.numeric(
                                  str_extract(title,"(?<=\\()([0-9]{4})(?=\\)$)"))
                                )
validation <- validation %>% mutate(timestamp = as_datetime(timestamp),
                                    year_rated = year(as_datetime(timestamp)),
                                    year_movie = as.numeric(
                                      str_extract(title,"(?<=\\()([0-9]{4})(?=\\)$)"))
                                   )
```

A new field called year_diff was created. Year_diff accounts for the number of years from when a movie was released to when a movie was rated. Evaluating this field may help determine if older, more classic films are rated higher than newer releases or vice versa.

```
edx_train <- edx_train %>% mutate(year_diff = year_rated - year_movie)
edx_test <- edx_test %>% mutate(year_diff = year_rated - year_movie)
validation <- validation %>% mutate(year_diff = year_rated - year_movie)
```

There are 19 genre categories in the data. The genres field is a list. Each movie's genres' field lists every genre attributed to the movie separated by a "|" character. To analyze the impact of genre on rating, the column needs to be reformatted - firstly by lengthening the table so each movie has a separate row for each attributed genre, and then by pivoting the table so each genre has a binary indicator for each movie. Hyphens were removed from genre names for easier formatting.

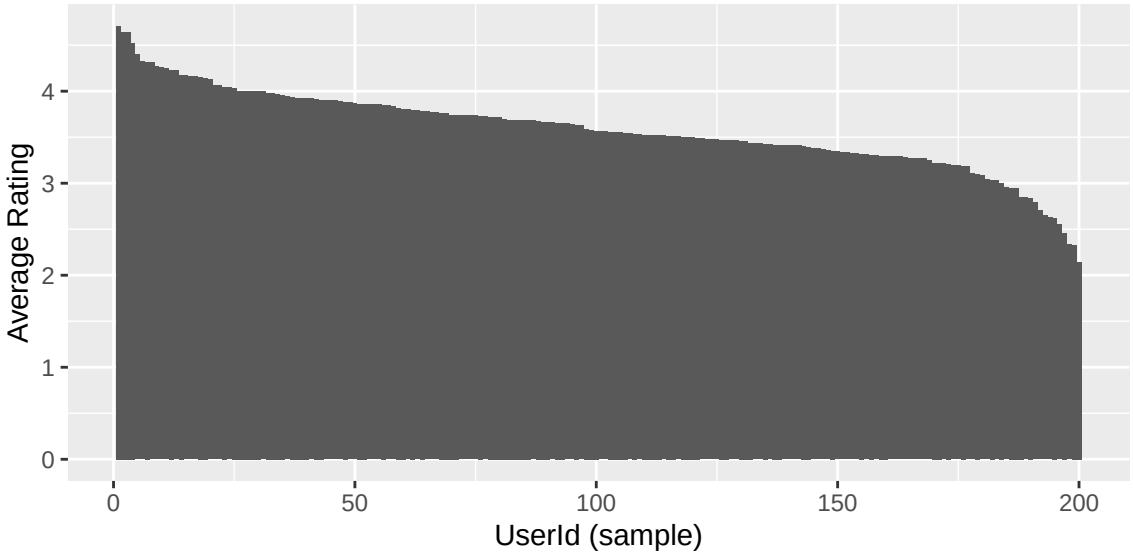
```
head(edx_train$genres)
```

```
## [1] "Action|Crime|Thriller"
## [2] "Action|Adventure|Sci-Fi"
## [3] "Action|Adventure|Drama|Sci-Fi"
## [4] "Children|Comedy|Fantasy"
## [5] "Adventure|Animation|Children|Drama|Musical"
## [6] "Action|Romance|Thriller"
```

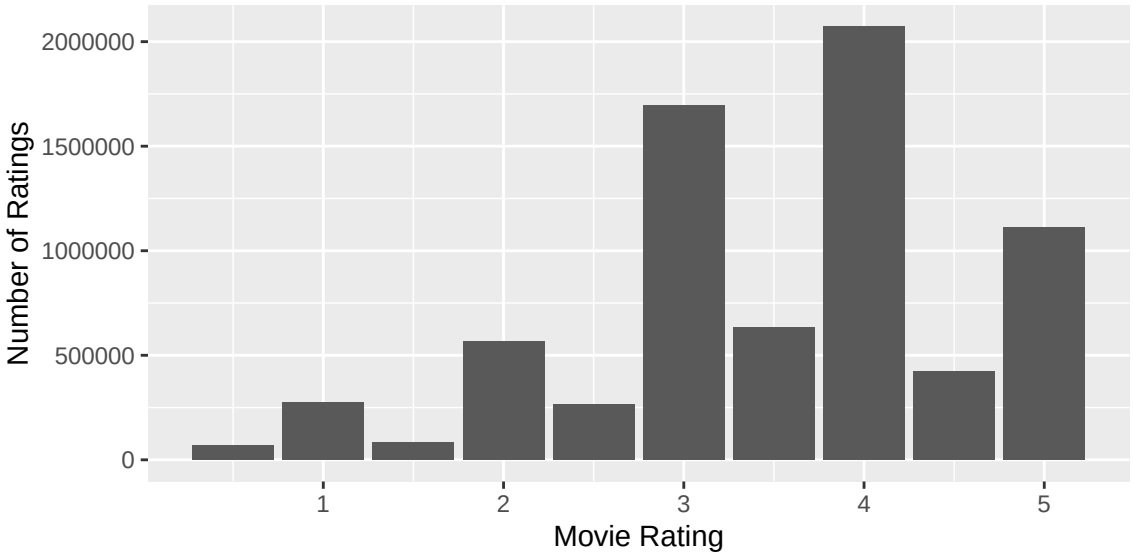
```
# Parse genre field
edx_genre <- edx_train %>% separate_rows(genres, sep ="\\|")
edx_test_genre <- edx_test %>% separate_rows(genres, sep ="\\|")
validation_genre <- validation %>% separate_rows(genres, sep ="\\|")
# Remove hyphens from genres column
edx_genre$genres <- gsub("-","",edx_genre$genres)
edx_test_genre$genres <- gsub("-","",edx_test_genre$genres)
validation_genre$genres <- gsub("-","",validation_genre$genres)
# Build table with binary genre columns
edx_use <- edx_genre %>% mutate(yesno = 1) %>% spread(genres,yesno,fill=0)
edx_test_use <- edx_test_genre %>% mutate(yesno = 1) %>% spread(genres,yesno,fill=0)
validation_use <- validation_genre %>% mutate(yesno = 1) %>% spread(genres,yesno,fill=0)
```

Data Exploration and Visualizations Evaluating variables and trends in the data

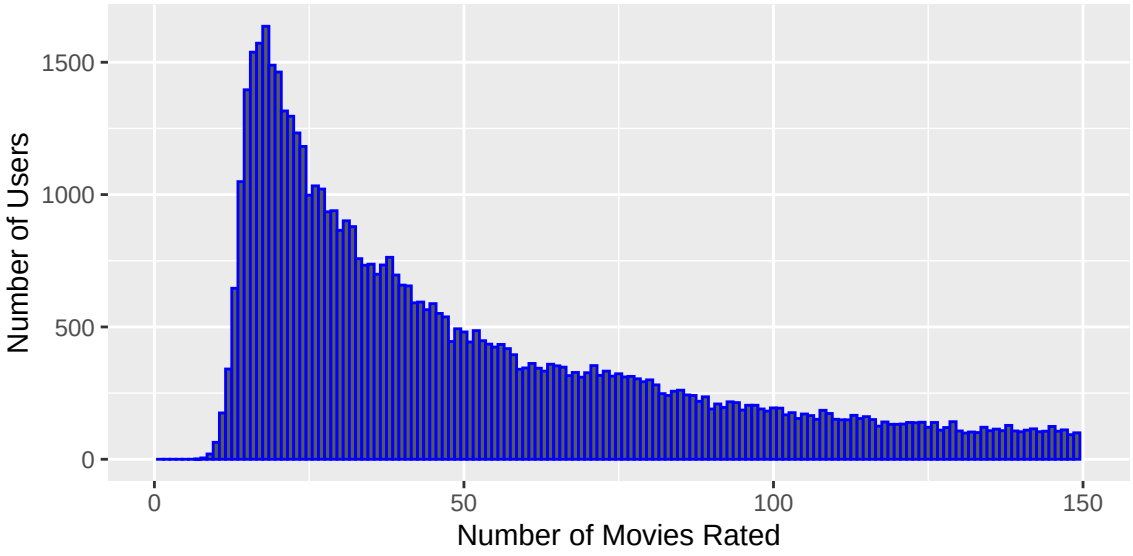
The graph below takes a sample of 200 users and evaluates the average movie ratings given by user. The graph illustrates variability in average rating as some users evaluate movies more harshly than others.



The number of ratings versus rating category graphed below illustrates a tendency for users to vote whole numbers as opposed to half ratings. 4 is the most frequent rating, followed by 3 and 5.



The average number of movies rated by a single user is 103. The median number of movies rated is 50. The graph below illustrates that most users tend to evaluate between 15 and 25 movies. The maximum number of movies rated by one user is 5,324. The histogram below shows the number of movies rated up until the 3rd quantile for view-ability.

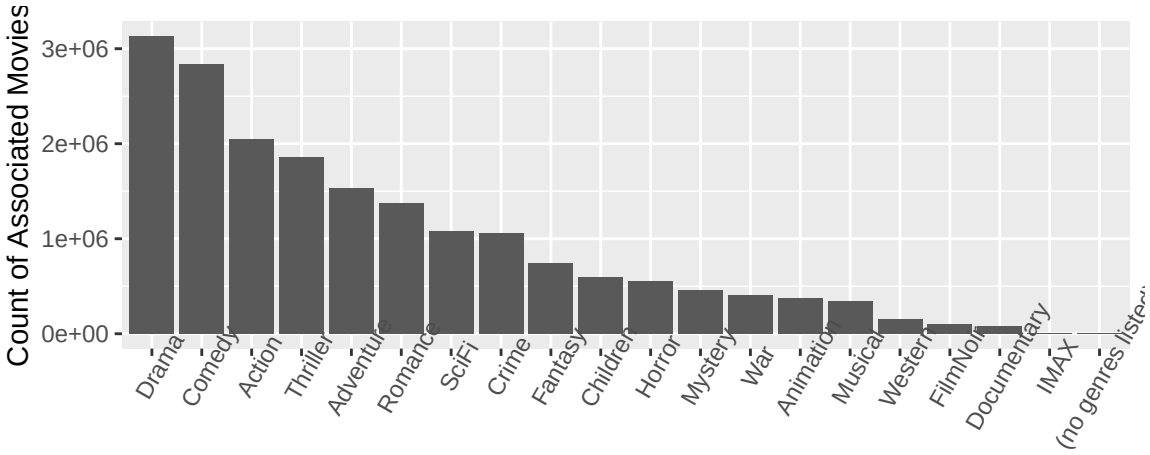



```
user_temp %>% group_by(movies_rated) %>% summarize(freq=n()) %>% arrange(desc(freq))
```

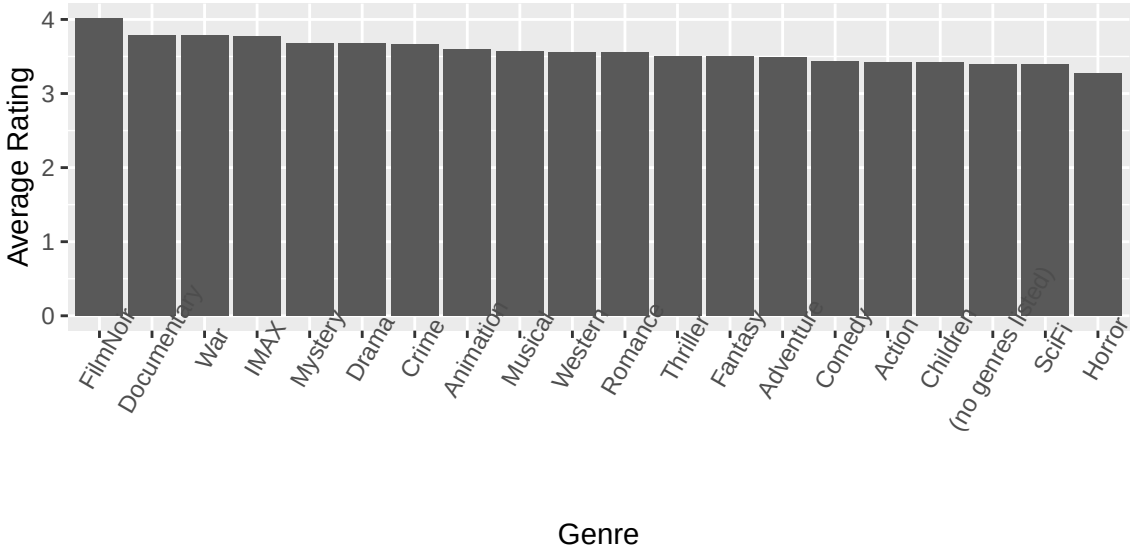
```
## `summarise()` ungrouping output (override with `.groups` argument)
```

```
## # A tibble: 1,196 x 2
##    movies_rated  freq
##           <int> <int>
##  1           18  1636
##  2           17  1572
##  3           16  1538
##  4           19  1489
##  5           20  1463
##  6           15  1396
##  7           21  1316
##  8           22  1296
##  9           23  1233
## 10           24  1182
## # ... with 1,186 more rows
```

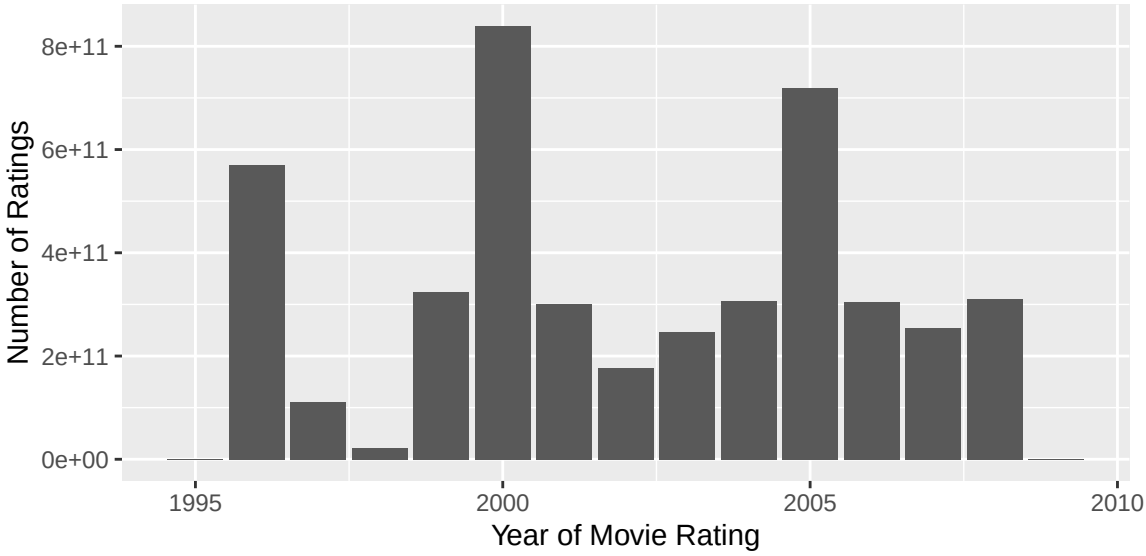
Drama, Comedy, and Action are the top 3 genre categories. Film-Noir, Documentary, and IMAX are the bottom 3 genre categories.

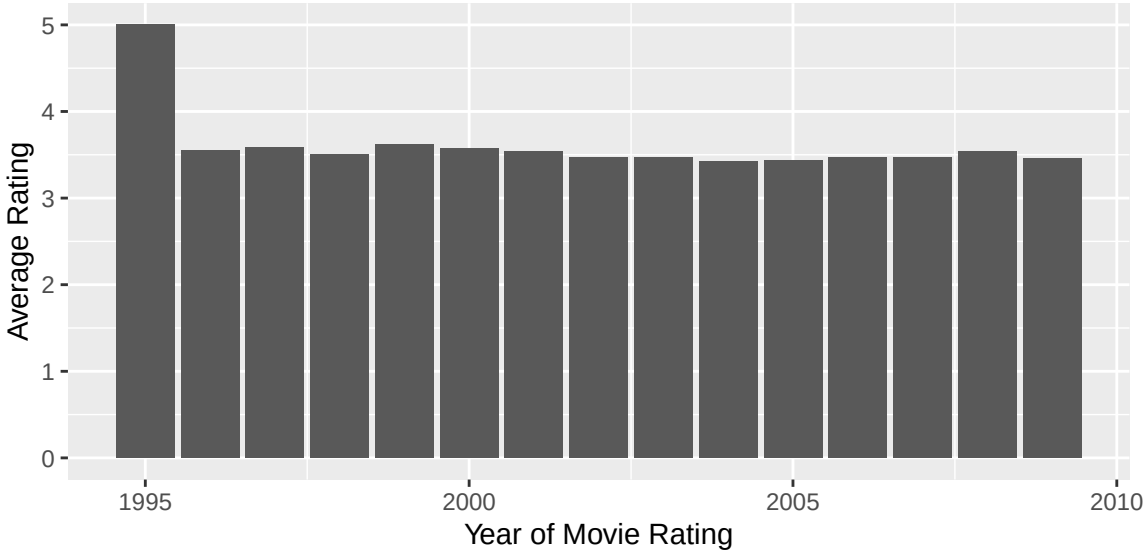


Film-Noir has the highest average rating, however this is may be a skewed result since there are not as many movies in this category (~1% of movies). The range of average rating across genres is ~0.75.

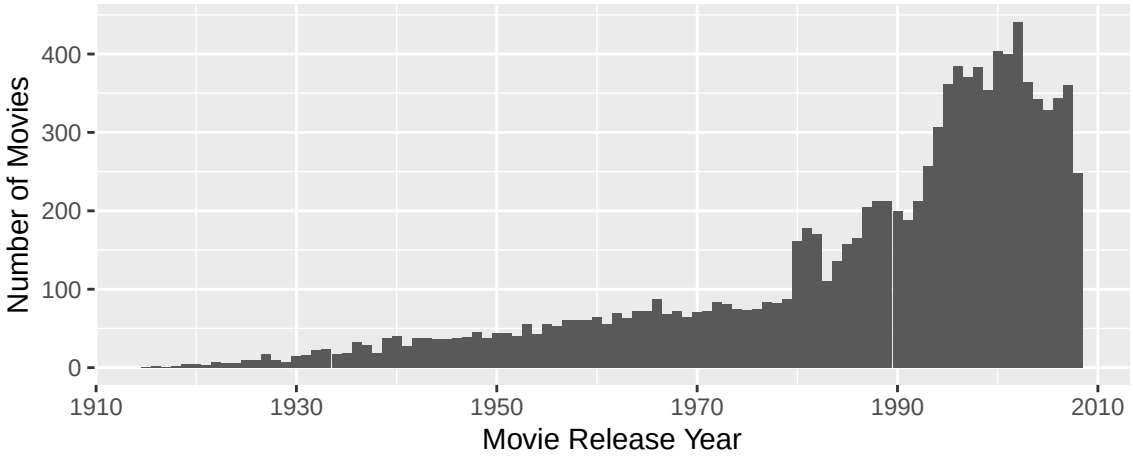


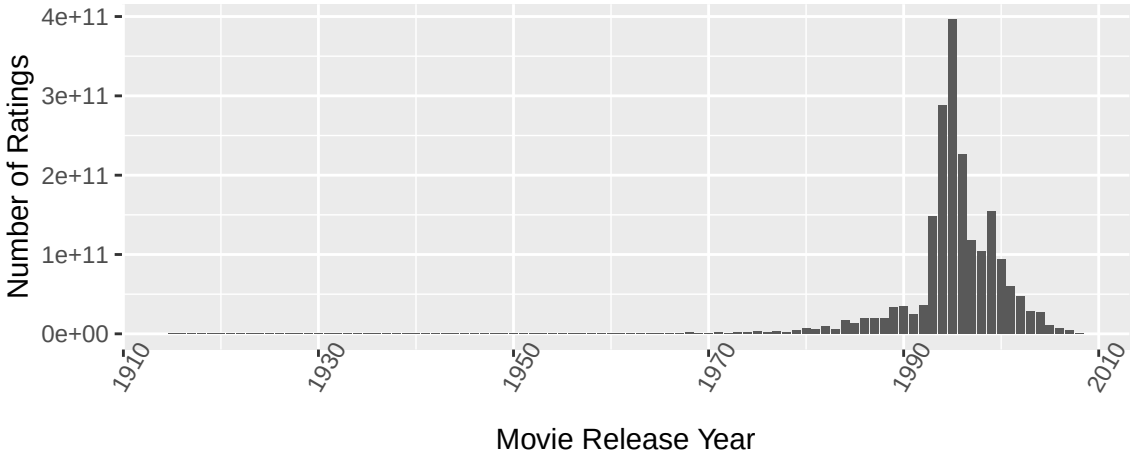
The dataset contains user ratings between 1995 and 2009. The averaged rating by rating year is consistent. Movies rated in 1995 have a slightly lower average rating compared with other years.

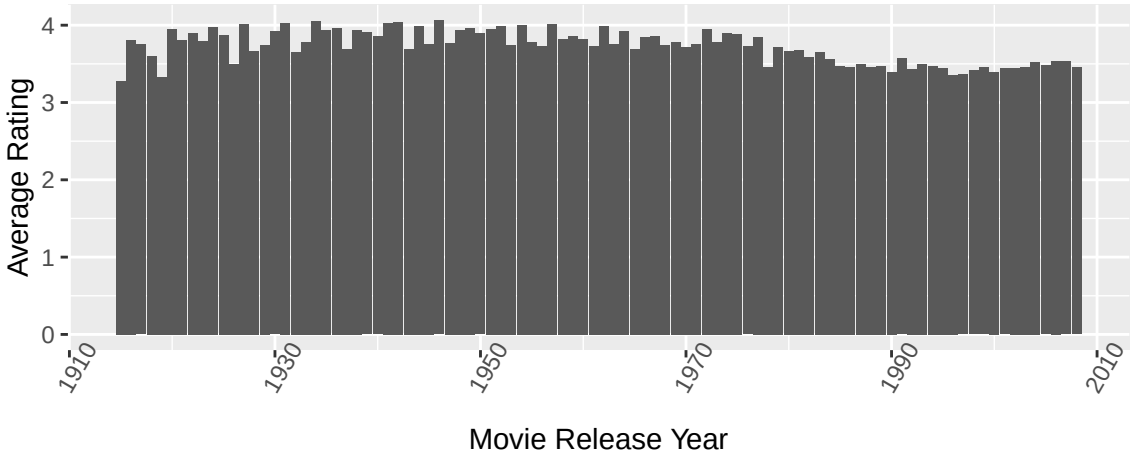




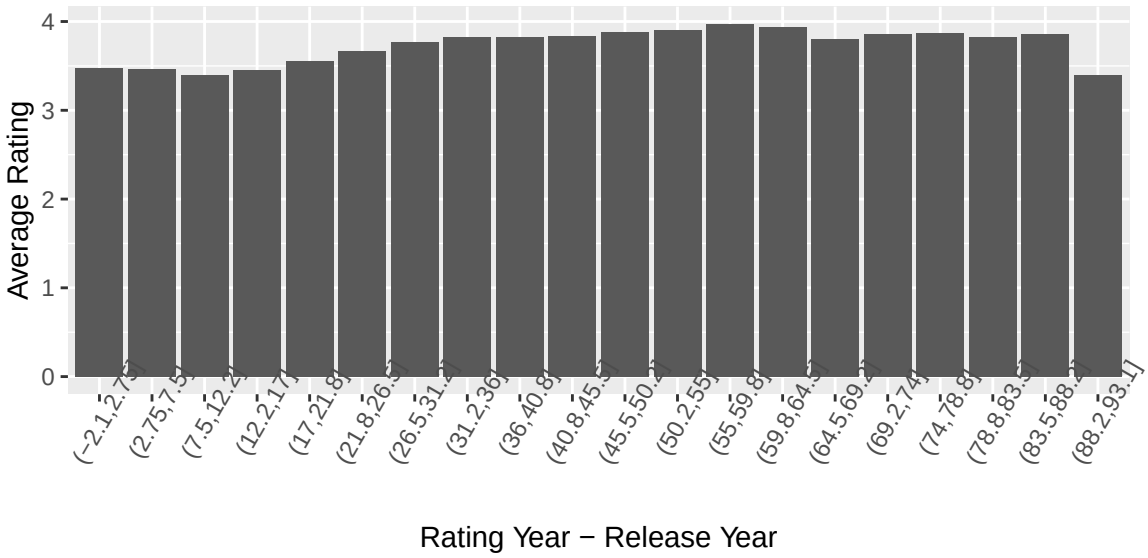
Movies in the dataset were released between 1915 and 2008. The dataset is skewed towards movies released between 1990 and 2010. The average rating for more recent movies is lower than older movies which may be due to larger rating count for those years.







There is limited variation of average rating based on when movie was rated versus when the movie premiered based on the plot below.



3. Results:

For model performance evaluation: Build function to calculate the root mean squared error (RMSE) between model predictions and actual values. For this application, the RMSE represents the typical error made when predicting a movie rating. The unit of the rmse is consistent with the rating variable, i.e. rmse greater than 1 indicates an error larger than 1 star.

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

The first model created is a model predicting rating based on the average movie rating over the entire dataset. The average rating model is a simplistic approach but will provide a baseline performance measure against which other models can be compared. Subsequent linear models will be created to evaluate the predictive nature of various variables within the dataset. The final model created is developed using matrix factorization, a method typically used in developing recommendation systems. Matrix factorization is useful in that it works well for large datasets and this method is able to determine latent effects that influence rating behavior.

Model 1: Average Rating Model – predicted rating is the average rating across the training dataset

```
mu <- mean(edx_use$rating)
naive_rmse <- RMSE(edx_test_use$rating,mu)
rmse_results <- tibble(method = "Average Rating", RMSE = naive_rmse)
rmse_results
```

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

Model 2: Movie Effects Model – linear model predicting rating based on average rating by movie

```
movie_avgs <- edx_use %>% group_by(movieId) %>% summarize(b_i = mean(rating - mu))
predicted_ratings <- mu + edx_test_use %>% left_join(movie_avgs,by='movieId') %>%
  pull(b_i)
movie_rmse <- RMSE(predicted_ratings, edx_test_use$rating)
rmse_results <- rmse_results %>% add_row(method = "Movie Effects", RMSE = movie_rmse)
```

Model 3: User Effects Model – linear model predicting rating based on average rating by user

```
user_avgs <- edx_use %>% group_by(userId) %>% summarize(b_u = mean(rating - mu))
predicted_ratings <- mu + edx_test_use %>% left_join(user_avgs,by='userId') %>%
  pull(b_u)
user_rmse <- RMSE(predicted_ratings, edx_test_use$rating)
rmse_results <- rmse_results %>% add_row(method = "User Effects", RMSE = user_rmse)
```

Model 4: Rating Year minus Premier Year Effects Model – linear model predicting rating based on average rating by number of years between model premier and rating date

```
yr_avgs <- edx_use %>% group_by(year_diff) %>% summarize(b_y = mean(rating - mu))
predicted_ratings <- mu + edx_test_use %>% left_join(yr_avgs,by='year_diff') %>%
  pull(b_y)
yr_rmse <- RMSE(predicted_ratings, edx_test_use$rating)
rmse_results <- rmse_results %>% add_row(method = "Year Diff Effects", RMSE = yr_rmse)
```

Model 5: Movie Premier Year Effects Model – linear model predicting rating based on average rating by movie premier year

```
movieyr_avgs <- edx_use %>% group_by(year_movie) %>% summarize(b_my = mean(rating - mu))
predicted_ratings <- mu + edx_test_use %>% left_join(movieyr_avgs,by='year_movie') %>%
  pull(b_my)
movieyr_rmse <- RMSE(predicted_ratings, edx_test_use$rating)
rmse_results <- rmse_results %>% add_row(method = "Movie Year Effects", RMSE = movieyr_rmse)
```

Model 6: Genre Effects Model – linear model predicting rating based on average rating by movie genre

```
genre_avgs <- edx_use %>% group_by(Comedy, Romance, Action, Crime, Thriller, Drama,
                                   SciFi, Adventure, Children, Fantasy, War, Animation,
                                   Musical, Western, Mystery, FilmNoir, Horror,
                                   Documentary, IMAX) %>% summarize(b_g = mean(rating - mu))
predicted_ratings <- mu + edx_test_use %>%
  left_join(genre_avgs,by=c('Comedy','Romance','Action','Crime','Thriller','Drama',
                                   'SciFi','Adventure','Children','Fantasy','War','Animation',
                                   'Musical','Western','Mystery','FilmNoir','Horror',
                                   'Documentary','IMAX')) %>% pull(b_g)
genre_rmse <- RMSE(predicted_ratings, edx_test_use$rating)
rmse_results <- rmse_results %>% add_row(method = "Genre Effects", RMSE = genre_rmse)
```

Model 7: Movie + User Effects model – linear model predicting rating based on average rating by both movie and user

```
predicted_ratings <- edx_test_use %>% left_join(movie_avgs, by='movieId') %>%
  left_join(user_avgs, by='userId') %>% mutate(pred = mu + b_i + b_u) %>% pull(pred)
movie_user_rmse <- RMSE(predicted_ratings, edx_test_use$rating)
rmse_results <- rmse_results %>% add_row(method = "Movie + User Effects",
                                         RMSE = movie_user_rmse)
```

Model 8: Movie + User + Genre Effects model – linear effects model predicting ratings based on movie, user and genre

```
predicted_ratings <- edx_test_use %>% left_join(movie_avgs, by='movieId') %>%
  left_join(user_avgs, by='userId') %>%
  left_join(genre_avgs,by=c('Comedy','Romance','Action','Crime','Thriller','Drama',
                            'SciFi','Adventure','Children','Fantasy','War','Animation',
                            'Musical','Western','Mystery','FilmNoir','Horror',
                            'Documentary','IMAX')) %>%
  mutate(pred = mu + b_i + b_u + b_g) %>% pull(pred)
movie_user_genre_rmse <- RMSE(predicted_ratings, edx_test_use$rating)
rmse_results <- rmse_results %>% add_row(method = "Movie + User + Genre Effects",
                                         RMSE = movie_user_genre_rmse)
rmse_results
```

```
## # A tibble: 8 x 2
##   method                        RMSE
##   <chr>                        <dbl>
## 1 Average Rating               1.06
## 2 Movie Effects                0.944
## 3 User Effects                 0.978
## 4 Year Diff Effects            1.05
## 5 Movie Year Effects           1.05
## 6 Genre Effects                1.02
## 7 Movie + User Effects         0.886
## 8 Movie + User + Genre Effects 0.946
```

Overall, movie and user appear to be the most predictive characteristics of the variables evaluated. Genre, rating year, and movie release year were less predictive of rating.

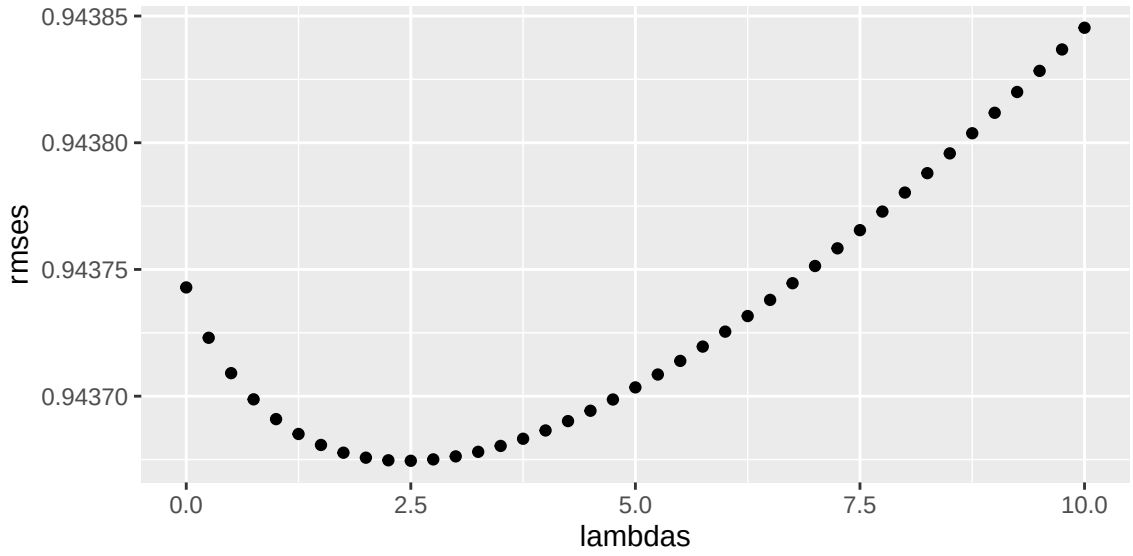
The movieid and userid single variable linear models had the lowest RMSE. The average rating, movie year, rating year, rating minus movie year variables all produce rmses greater than 1. The linear model including both movieid and userid produced the lowest RMSE of 0.886.

In the following model creation iterations, regularization is employed. Regularization is a tool that helps prevent over-fitting and decrease variability of the model. Regularization introduces a lambda term which penalizes coefficients with small sample sizes. As part of the model development, tuning is performed on a sequence of lambda values in order to pick a lambda that reduces the rmse. The use of regularization for this dataset is useful as certain movies, users have limited rating data.

Model 9: Regularized Movie Effects Model – regularized linear model predicting ratings based on average rating by movie

```
lambdas <- seq(0, 10, 0.25)
just_the_sum <- edx_use %>% group_by(movieId) %>% summarize(s = sum(rating - mu), n_i = n())
rmses <- sapply(lambdas, function(l){ predicted_ratings <- edx_test_use %>%
  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, edx_test_use$rating)) })
qplot(lambdas, rmses)
```

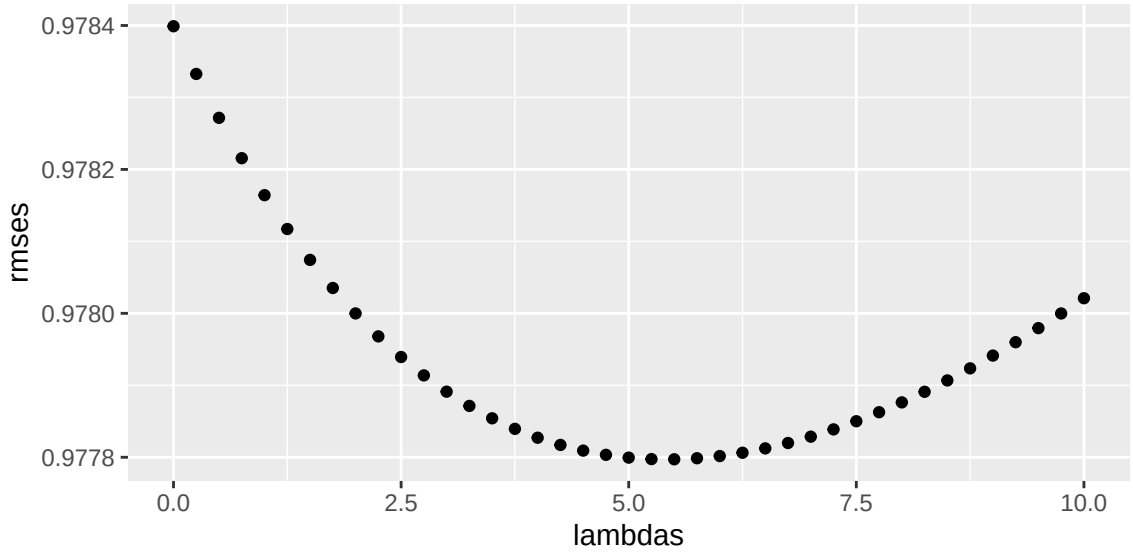



```
lambda <- lambdas[which.min(rmses)]
# 1.75
movie_reg_avgs <- edx_use %>% group_by(movieId) %>%
  summarize(b_i = sum(rating - mu)/(n()+lambda), n_i = n())

predicted_ratings <- edx_test_use %>% left_join(movie_reg_avgs, by='movieId') %>%
  mutate(pred = mu + b_i ) %>% pull(pred)
movie_reg_rmse <- RMSE(predicted_ratings, edx_test_use$rating)
rmse_results <- rmse_results %>% add_row(method = "Movie Regularized Effects",
                                         RMSE = movie_reg_rmse)
```

Model 10: Regularized User Effects Model – regularized linear model predicting ratings based on average rating by user

```
lambdas <- seq(0, 10, 0.25)
just_the_sum <- edx_use %>% group_by(userId) %>% summarize(s = sum(rating - mu), n_i = n())
rmses <- sapply(lambdas, function(l){
  predicted_ratings <- edx_test_use %>%
    left_join(just_the_sum, by='userId') %>% mutate(b_u = s/(n_i+l)) %>%
    mutate(pred = mu + b_u) %>%
    pull(pred)
  return(RMSE(predicted_ratings, edx_test_use$rating)) })
qplot(lambdas, rmses)
```

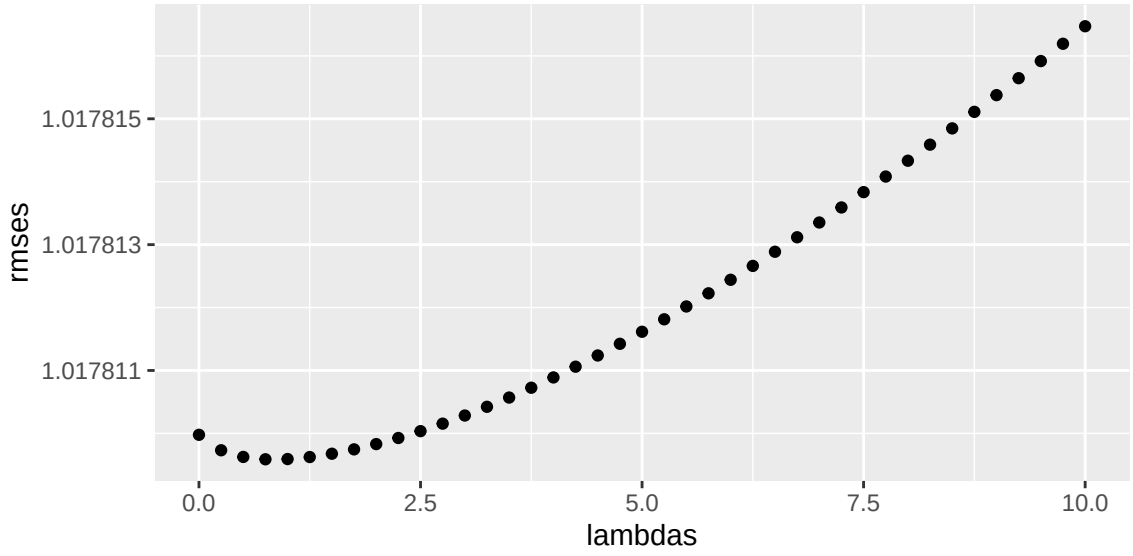



```
lambda <- lambdas[which.min(rmses)]
# 5.5
user_reg_avgs <- edx_use %>%
  group_by(userId) %>%
  summarize(b_u = sum(rating - mu)/(n()+lambda), n_i = n())

predicted_ratings <- edx_test_use %>% left_join(user_reg_avgs, by='userId') %>%
  mutate(pred = mu + b_u ) %>% pull(pred)
user_reg_rmse <- RMSE(predicted_ratings, edx_test_use$rating)
rmse_results <- rmse_results %>% add_row(method = "User Regularized Effects",
                                         RMSE = user_reg_rmse)
```

Model 11: Regularized Genre model – regularized linear effects model predicting ratings based on average rating by genre

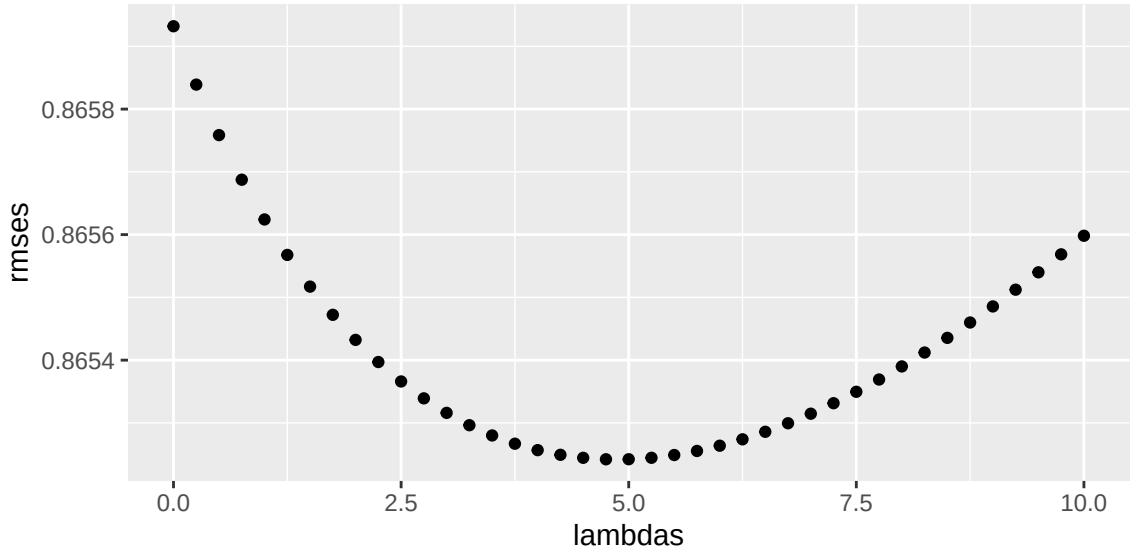
```
lambdas <- seq(0, 10, 0.25)
just_the_sum <- edx_use %>%
  group_by(Comedy, Romance, Action, Crime, Thriller, Drama,
           SciFi, Adventure, Children, Fantasy, War, Animation,
           Musical, Western, Mystery, FilmNoir, Horror,
           Documentary, IMAX) %>% summarize(s = sum(rating - mu), n_i = n())
rmses <- sapply(lambdas, function(l){
  predicted_ratings <- edx_test_use %>%
    left_join(just_the_sum, by=c('Comedy','Romance','Action','Crime','Thriller','Drama',
                                 'SciFi','Adventure','Children','Fantasy','War','Animation',
                                 'Musical','Western','Mystery','FilmNoir','Horror',
                                 'Documentary','IMAX')) %>%
    mutate(b_g = s/(n_i+l)) %>%
    mutate(pred = mu + b_g) %>%
    pull(pred)
  return(RMSE(predicted_ratings, edx_test_use$rating)) })
qplot(lambdas, rmses)
```




```
lambda <- lambdas[which.min(rmses)]
# 0.75
genre_reg_rmse <- min(rmses)
rmse_results <- rmse_results %>% add_row(method = "Genre Regularized Effects",
                                         RMSE = genre_reg_rmse)
```

Model 12: Regularized Movie + User Effects Model – regularized linear effects model predicting ratings based on average rating by movie and user

```
lambdas <- seq(0, 10, 0.25)
rmses <- sapply(lambdas, function(l){
  mu <- mean(edx_use$rating)
  b_i <- edx_use %>%
    group_by(movieId) %>%
    summarize(b_i = sum(rating - mu)/(n()+l))
  b_u <- edx_use %>%
    left_join(b_i, by="movieId") %>% group_by(userId) %>%
    summarize(b_u = sum(rating - b_i - mu)/(n()+l))
  predicted_ratings <-
    edx_test_use %>% left_join(b_i, by = "movieId") %>% left_join(b_u, by = "userId") %>%
    mutate(pred = mu + b_i + b_u) %>% pull(pred)
  return(RMSE(predicted_ratings, edx_test_use$rating)) })
qplot(lambdas, rmses)
```




```
lambda <- lambdas[which.min(rmses)]
# 4.75
reg_movie_user_rmse <- min(rmses)

rmse_results <- rmse_results %>% add_row(method = "Regularized Movie + User Effects",
                                         RMSE = reg_movie_user_rmse)
```

Model 13: Matrix Factorization Model – Matrix Factorization is often used when building recommendation systems because of its ability to process large datasets and uncover latent features within variables. Latent features for this data in particularly may be an actor that appears in multiple movies, the sentiment of the movies, similar filming styles, etc. In other words, features that are underlying observed features/variables in the data. The recosystem package builds and tunes a matrix factorization model based. For this model, movie_id and user_id will be incorporated.

```
mf_edx_use <- edx_use %>% select(movieId, userId, rating)
mf_edx_test_use <- edx_test_use %>% select(movieId, userId, rating)
mf_validation_use <- validation_use %>% select(movieId, userId, rating)

write.table(mf_edx_use,file="train.txt",sep=" ",row.names=FALSE,col.names=FALSE)
write.table(mf_edx_test_use,file="test.txt",sep=" ",row.names=FALSE,col.names=FALSE)
write.table(mf_validation_use,file="validation.txt",sep=" ",row.names=FALSE,col.names=FALSE)
mf_train <- data_file("train.txt")
mf_test <- data_file("test.txt")
mf_validation <- data_file("validation.txt")
r = Reco()

set.seed(1, sample.kind="Rounding")
```

```
## Warning in set.seed(1, sample.kind = "Rounding"): non-uniform 'Rounding' sampler
## used
```

```
opts <- r$tune(mf_train, opts = list(dim = c(10,20,30),lrate = c(0.1,0.2),
                                     costp_l1 = 0, costq_l1 = 0,
```

```
                                    nthread = 1, niter = 10))
opts
```

```
## $min
## $min$dim
## [1] 30
##
## $min$costp_l1
## [1] 0
##
## $min$costp_l2
## [1] 0.1
##
## $min$costq_l1
## [1] 0
##
## $min$costq_l2
## [1] 0.01
##
## $min$lrate
## [1] 0.1
##
## $min$loss_fun
## [1] 0.8063808
##
##
## $res
##    dim costp_l1 costp_l2 costq_l1 costq_l2 lrate  loss_fun
## 1   10        0     0.01        0     0.01   0.1 0.8313240
## 2   20        0     0.01        0     0.01   0.1 0.8196820
## 3   30        0     0.01        0     0.01   0.1 0.8296077
## 4   10        0     0.10        0     0.01   0.1 0.8328843
## 5   20        0     0.10        0     0.01   0.1 0.8107396
## 6   30        0     0.10        0     0.01   0.1 0.8063808
## 7   10        0     0.01        0     0.10   0.1 0.8325005
## 8   20        0     0.01        0     0.10   0.1 0.8129454
## 9   30        0     0.01        0     0.10   0.1 0.8136130
## 10  10        0     0.10        0     0.10   0.1 0.8411258
## 11  20        0     0.10        0     0.10   0.1 0.8339458
## 12  30        0     0.10        0     0.10   0.1 0.8301632
## 13  10        0     0.01        0     0.01   0.2 0.8265537
## 14  20        0     0.01        0     0.01   0.2 0.9251973
## 15  30        0     0.01        0     0.01   0.2 0.9365055
## 16  10        0     0.10        0     0.01   0.2 0.8284553
## 17  20        0     0.10        0     0.01   0.2 0.8098148
## 18  30        0     0.10        0     0.01   0.2 0.8110441
## 19  10        0     0.01        0     0.10   0.2 0.8275271
## 20  20        0     0.01        0     0.10   0.2 0.9182621
## 21  30        0     0.01        0     0.10   0.2 0.9677742
## 22  10        0     0.10        0     0.10   0.2 0.8434398
## 23  20        0     0.10        0     0.10   0.2 0.8283218
## 24  30        0     0.10        0     0.10   0.2 0.8280653
```

```
r$train(mf_train,opts=c(opts$min,nthread=1,niter=35))
```

```
## iter      tr_rmse          obj
##    0       0.9944   1.0052e+07
##    1       0.8798   8.1009e+06
##    2       0.8476   7.5092e+06
##    3       0.8255   7.1593e+06
##    4       0.8085   6.9082e+06
##    5       0.7955   6.7294e+06
##    6       0.7848   6.5926e+06
##    7       0.7754   6.4783e+06
##    8       0.7672   6.3845e+06
##    9       0.7601   6.3054e+06
##   10       0.7539   6.2364e+06
##   11       0.7483   6.1824e+06
##   12       0.7433   6.1315e+06
##   13       0.7388   6.0888e+06
##   14       0.7347   6.0520e+06
##   15       0.7309   6.0168e+06
##   16       0.7274   5.9852e+06
##   17       0.7242   5.9567e+06
##   18       0.7212   5.9327e+06
##   19       0.7185   5.9098e+06
##   20       0.7159   5.8871e+06
##   21       0.7136   5.8684e+06
##   22       0.7113   5.8504e+06
##   23       0.7092   5.8330e+06
##   24       0.7074   5.8184e+06
##   25       0.7055   5.8052e+06
##   26       0.7039   5.7914e+06
##   27       0.7023   5.7806e+06
##   28       0.7008   5.7682e+06
##   29       0.6994   5.7565e+06
##   30       0.6981   5.7481e+06
##   31       0.6968   5.7393e+06
##   32       0.6956   5.7295e+06
##   33       0.6945   5.7223e+06
##   34       0.6934   5.7157e+06
```

```
mf_prediction <- r$predict(mf_test,out_memory())
mf_test_ratings <- read.table("test.txt",header=FALSE,sep="")$V3
mf_rmse <- RMSE(mf_prediction, mf_test_ratings)
rmse_results <- rmse_results %>% add_row(method = "User + Movie Matrix Factorization",
                                         RMSE = mf_rmse)
rmse_results
```

```
## # A tibble: 13 x 2
##    method                              RMSE
##    <chr>                              <dbl>
##  1 Average Rating                     1.06
##  2 Movie Effects                      0.944
##  3 User Effects                       0.978
```

```
##  4 Year Diff Effects                 1.05
##  5 Movie Year Effects                1.05
##  6 Genre Effects                     1.02
##  7 Movie + User Effects              0.886
##  8 Movie + User + Genre Effects      0.946
##  9 Movie Regularized Effects         0.944
## 10 User Regularized Effects          0.978
## 11 Genre Regularized Effects         1.02
## 12 Regularized Movie + User Effects  0.865
## 13 User + Movie Matrix Factorization 0.790
```

4. Conclusion:

- RMSE $<$ 0.86490 to receive full marks

By iterating through various models, determining predictive variables, and evaluating the model RMSE, the matrix factorization model using movieid and userid was found to be the most predictive of movie rating. The model development and testing was performed using a training set consisting of 90% of the movielens dataset. The final model will be evaluated using the validation set in order to ensure that the model is not overfitted to the training data set.

```
mf_validation_prediction <- r$predict(mf_validation,out_memory())
mf_validaton_ratings <- read.table("validation.txt",header=FALSE,sep="")$V3
final_rmse <- RMSE(mf_validation_prediction,mf_validaton_ratings)
rmse_results <- rmse_results %>% add_row(method = "Final User + Movie Matrix Factorization",
                                         RMSE = final_rmse)
rmse_results
```

```
## # A tibble: 14 x 2
##    method                                  RMSE
##    <chr>                                  <dbl>
##  1 Average Rating                         1.06
##  2 Movie Effects                          0.944
##  3 User Effects                           0.978
##  4 Year Diff Effects                      1.05
##  5 Movie Year Effects                     1.05
##  6 Genre Effects                          1.02
##  7 Movie + User Effects                   0.886
##  8 Movie + User + Genre Effects           0.946
##  9 Movie Regularized Effects              0.944
## 10 User Regularized Effects               0.978
## 11 Genre Regularized Effects              1.02
## 12 Regularized Movie + User Effects       0.865
## 13 User + Movie Matrix Factorization      0.790
## 14 Final User + Movie Matrix Factorization 0.790
```

The final rmse using the matrix factorization movie and user model is 0.790. This is an improvement compared with the average rating model which had an rmse of 1.06. Given the movielens data size and the computing power of personal computers there were limitations to the complexity of models that could be developed/evaluated. If the project were to continue, I would wish to do more evaluation of interactions between variables, as well as, further investigate the impact of movie release year on rating. The movielens dataset is skewed toward ratings for movies in the 1990-2010 range. If a dataset included more ratings

for older movies then there could be further investigation into the impact of movie age on rating or its interaction with the movieid and userid variables. Additionally, investigation into more complex machine learning approaches and R libraries would facilitate continued learning and the ability to add additional complexity to future model development.