MovieLens Project Submission

HarvardX PH125.9x Data Science Capstone Project

Gerry Henstra

November 2019

INTRODUCTION

Objective

The purpose of this initiative is to construct a movie recommendation system using the MovieLens dataset. A machine learning algorithm will be trained to predict what movie ratings an individual would give to any particular movie title. We will start with a simple model and build on it as we learn from the data.

Our goal is to acheive an RMSE of 0.8649 or better.

Our approach will be:

1) Describe the datasets,
2) Explore the data to acheive an understanding of what we have to work with,
3) Build our model and refactor it to fine tune the results against the training set only,
4) Apply the final model to our validation/test set to see if the model provides a RMSE value of less than 0.8649, and
5) Discuss the results and summarize the overal outcome

Dataset Description

The dataset used in this project comes from the MovieLens 10M dataset. This dataset contains 10 million ratings and 100,000 tag applications applied to 10,000 movies by 72,000 users. It was released January 2009.

Users were selected at random for inclusion. All users selected had rated at least 20 movies. No demographic data is provided, so it will not be incorporated into the final prediction algorithm.

The dataset and additional information can be found at the following location : https://grouplens.org/datasets/movielens/10m/

Acknowledgement for the use of this dataset can be found at: F. Maxwell Harper and Joseph A. Konstan. 2015. The MovieLens Datasets: History and Context. ACM Transactions on Interactive Intelligent Systems (TiiS) 5, 4, Article 19 (December 2015), 19 pages. DOI=http://dx.doi.org/10.1145/2827872

Dataset Details

The data contained in this dataset consists of three files, movies.dat, ratings.dat and tags.dat.

movies.dat

The movies.dat file consists of movie information. Each row represents one movie and consists of three properties:

1) MovieID : MovieLens id
2) Title : movie title as found in IMDB and includes the year of release
3) Genres : pipe separated list of one or more genres assigned to the movie

The Genres included are :

- Action

- Adventure
- Animation
- Children's
- Comedy
- Crime
- Documentary
- Drama
- Fantasy
- Film-Noir
- Horror
- Musical
- Mystery
- Romance
- Sci-Fi
- Thriller
- War
- Western

ratings.dat

The ratings.dat file contains ratings of movies by users. Each row represents one rating for one movie by one user and consists of four properties :

1) UserID : Id of user
2) MovieID : MovieLens id
3) Rating (1 - 5)
4) Timestamp (seconds since 1/1/1970)

tags.dat

The tags.dat file contains one tag applied to one movie by one user. The tags are metadata about the movies and is a single word or short phrase. The meaning and value are determined by each user so the data is of little use in our algotithm, so we will not be using this file.

Helper Functions

A couple of helper functions have been created to assist.

1) setSeed : identifies the current version of R running and uses the appropriate set.seed function
2) RMSE : calculates the RMSE

```
#############################################################################################################
# Some custom functions
#############################################################################################################

# Get current major and minor version numbers of local R install
R_Major <- as.numeric(R.version$major)
R_Minor <- as.numeric(R.version$minor)

# Create a function to set the seed based on currently running R version
# setSeed(s) : s = seed to be set to
#############################################################################################################
setSeed <- function(s){
  if (R_Major >= 3 & R_Minor > 3.5) {
    set.seed(s, sample.kind="Rounding")
  } else {
```

```
    set.seed(s)
  }
}

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

Data Preparation

The ratings and movies datasets will be combined and the columns manipulated to extract the year from the name for use in our algorithm. A decade column will be added for further exporation.

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

##########################################################################################################
# Import Movies file
##########################################################################################################
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),
     titlenoyear = str_remove(title, "[/(]\\d{4}[/)]$"),
     genres = as.character(genres),
     year = as.numeric(str_extract(str_extract(title, "[/(]\\d{4}[/)]$"),regex("\\d{4}"))),
     decade = floor(year/10) * 10
  )

# The year released needs to be extracted from the title as shown above


##########################################################################################################
# Join Movies and Ratings into a working dataset
##########################################################################################################
movielens <- left_join(ratings, movies, by = "movieId")

#movielens <- movielens %>% mutate(datetime = as_datetime(timestamp), date = as.Date(datetime), yyyymmd
```

The structure of the working dataset if as follows:

```
str(movielens)
```

```
## 'data.frame':    10000054 obs. of  9 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
##  $ 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
##  $ titlenoyear: chr  "Boomerang " "Net, The " "Dumb & Dumber " "Outbreak " ...
```

```
##  $ year       : num  1992 1995 1994 1995 1994 ...
##  $ decade     : num  1990 1990 1990 1990 1990 1990 1990 1990 1990 1990 ...
```

A validation set will consist of 10% of the MovieLens data, the rest of the data will be used for training the algorithm.

```
################################################################################
# Create Validation Set representing 10% of the MovieLens Data
################################################################################
setSeed(1)
```

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

```
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", "titlenoyear", "year"
```

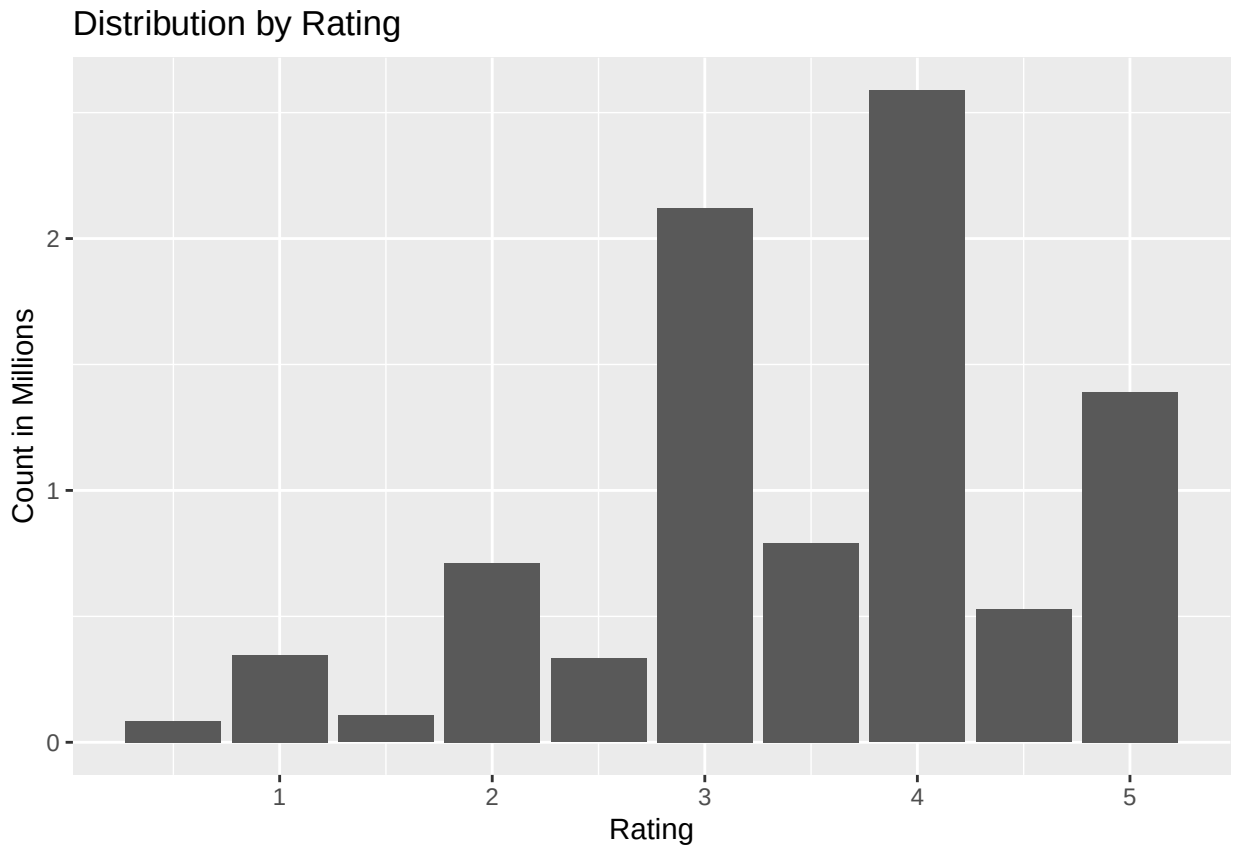
```
edx <- rbind(edx, removed)

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

DATA EXPLORATION

First let's look at the summary statistics for the ratings within our training set.

Minimum: 0.5
Median : 4
Mean : 3.5
Maximum: 5

With a median of 4 and a mean of 3.5 the data is right skewed meaning that the scores are generally on the higher side of a 5 star rating system as shown below.



Observations :

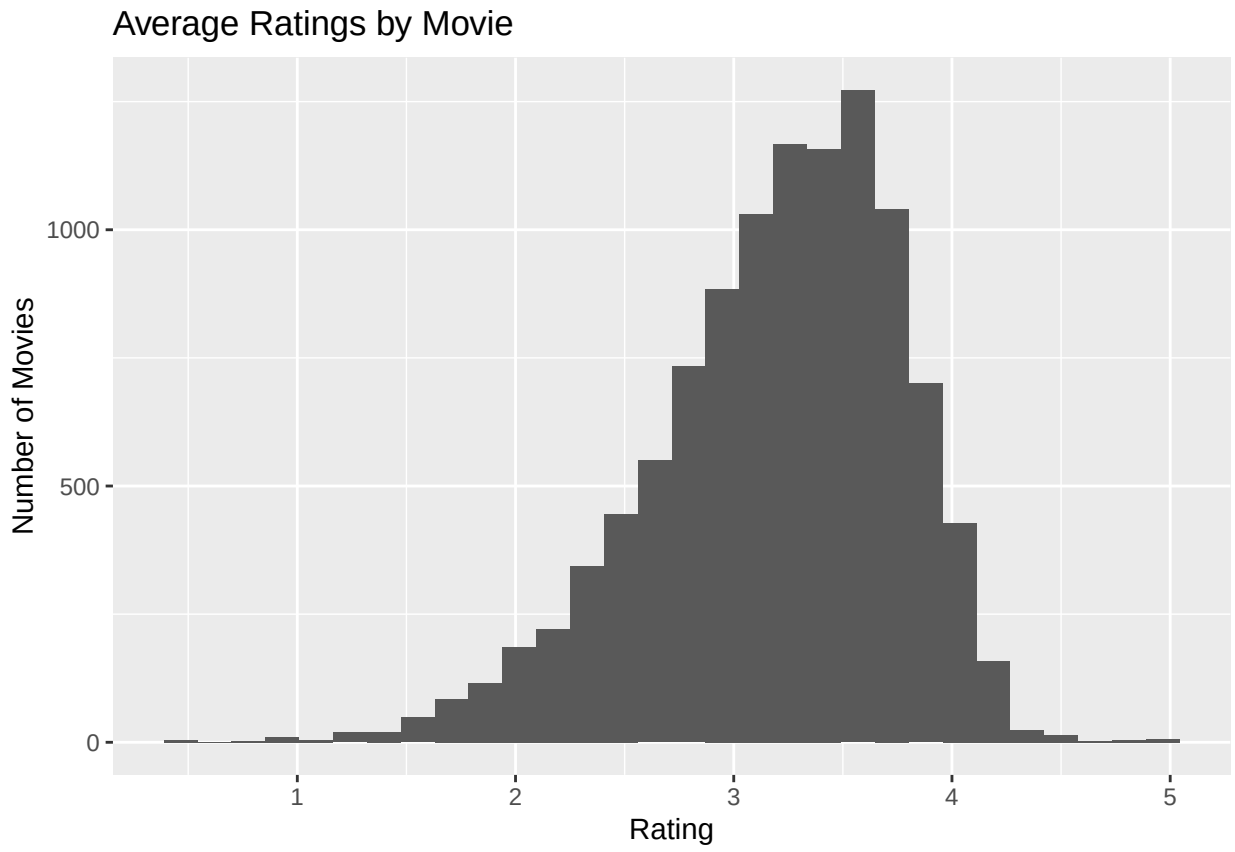
1) Ratings are in intervals of .5
2) There are no zero ratings
3) Most ratings are integers/whole numbers
4) A majority of ratings are 3 and above

The training dataset has the following contents:

Number of Users: `69878`
Number of Movies: `10677`
Number of Ratings: `9000055`
Lowest Rating: `0.5`
Highest Rating: `5`

Movie Exploration

Some movies will be generally rated higher than others. There are blockbusters, which will have an overall higher rating and there are flops which will generally have a low rating.



Observations

1) The majority of movies have a rating over 3
2) There are very few movies with ratings below 2 or above 4

Let's look at some of the movies with the most ratings.

```
## # A tibble: 10 x 4
## # Groups:   MovieID [10]
##    MovieID Title                                        Ratings   Avg
##      <dbl> <chr>                                          <int> <dbl>
##  1     296 Pulp Fiction (1994)                            31362  4.15
##  2     356 Forrest Gump (1994)                            31079  4.01
##  3     593 Silence of the Lambs, The (1991)               30382  4.20
##  4     480 Jurassic Park (1993)                           29360  3.66
##  5     318 Shawshank Redemption, The (1994)               28015  4.46
##  6     110 Braveheart (1995)                              26212  4.08
##  7     457 Fugitive, The (1993)                           25998  4.01
##  8     589 Terminator 2: Judgment Day (1991)              25984  3.93
##  9     260 Star Wars: Episode IV - A New Hope (a.k.a. Star W~   25672  4.22
## 10     150 Apollo 13 (1995)                               24284  3.89
```

Observations

1) The average number of ratings per movie is 843
2) The top 10 movies based on the number of ratings are clearly blockbusters with significantly more ratings
3) With the average rating being 3.5 we see that these blockbusters have higher than average ratings

Now let's look at the movies with the least ratings. We will also add average rating to the sort order.

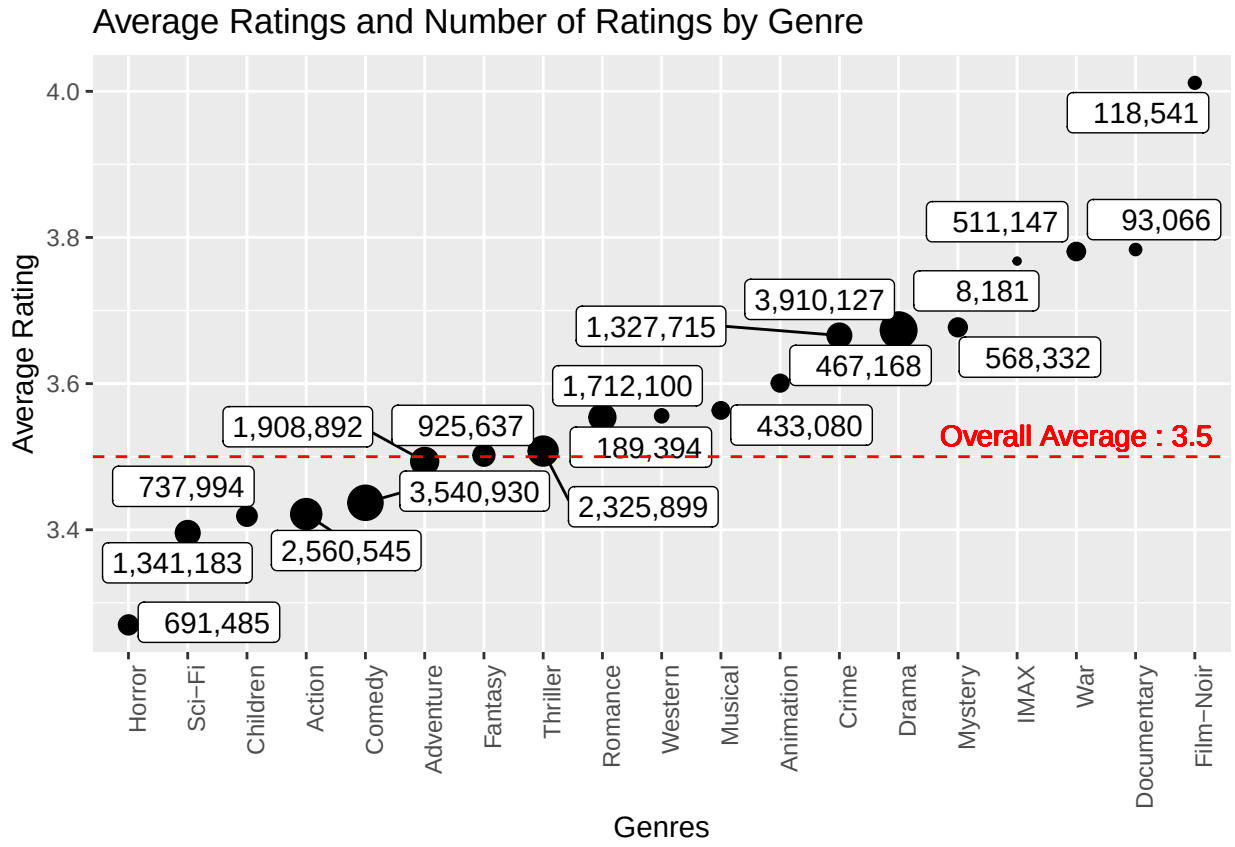
```
## # A tibble: 10 x 4
## # Groups:   MovieID [10]
##    MovieID Title                                                Ratings   Avg
##      <dbl> <chr>                                                  <int> <dbl>
##  1    8394 Hi-Line, The (1999)                                       1   0.5
##  2   61768 Accused (Anklaget) (2005)                                 1   0.5
##  3   63828 Confessions of a Superhero (2007)                         1   0.5
##  4    3561 Stacy's Knights (1982)                                    1   1
##  5    4071 Dog Run (1996)                                            1   1
##  6    4075 Monkey's Tale, A (Les ChÃ¢teau des singes) (1999)         1   1
##  7    5702 When Time Ran Out... (a.k.a. The Day the World En~        1   1
##  8    6189 Dischord (2001)                                           1   1
##  9   55324 Relative Strangers (2006)                                 1   1
## 10    6085 Neil Young: Human Highway (1982)                          1   1.5
```

Observations

1) These movies are obviously obscure movies that few people watch and rate
2) The ratings are generally very low

Movies are assigned one or more genres, so let's look at the individual genres first. In order to do this we have to separate the genres column into rows before plotting.

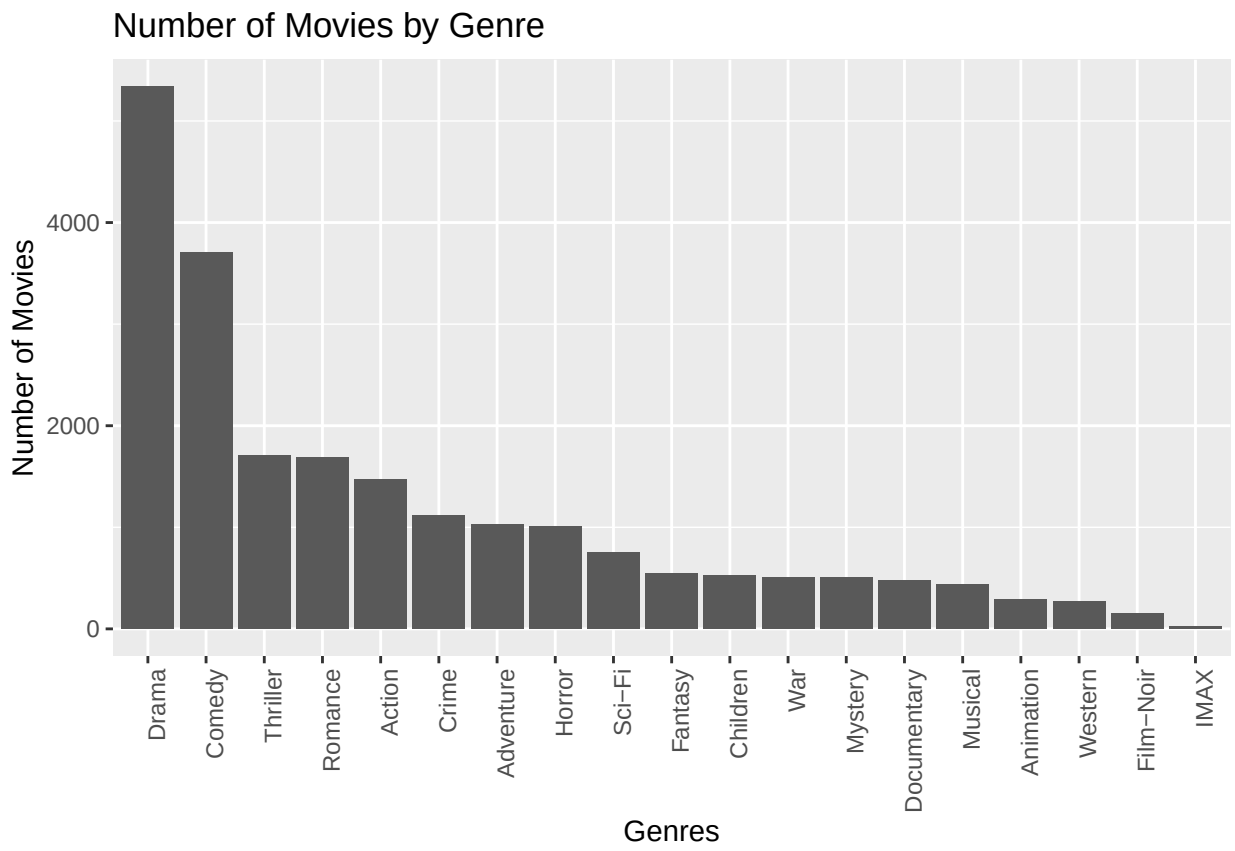
```
# generate a dataset of ratings by genre separating the genres column into rows based on the piped valu
ratings_by_genre <- edx %>%
  separate_rows(genres, sep="\\|") %>%
  select(movieId, genres, rating) %>%
  group_by(genres) %>%
  summarize(avg = mean(rating), movies = n_distinct(movieId), ratings = n())

# generate a plot showing the average rating by genre as well as the number of ratings by genre as a la
ratings_by_genre %>% filter(genres != "(no genres listed)") %>%
  ggplot(data=.) +
  geom_point(aes(reorder(genres, avg), avg, size=ratings), show.legend = FALSE) +
  theme(axis.text.x = element_text(angle=90, hjust=1)) +
  xlab("Genres") +
  ylab("Average Rating") +
  ggtitle("Average Ratings and Number of Ratings by Genre") +
  geom_label_repel(aes(genres, avg, label=format(ratings, big.mark=","))) +
  geom_hline(yintercept = round(as.numeric(edx_summary["Mean"]),1), linetype="dashed", color="red") +
  geom_text(aes(17,3.5, label="Overall Average : 3.5", vjust=-.5), color="red")
```



Observations

1) The Drama, Crime and Romance genres are major genres with average ratings higher than the average
2) Thriller, Fantasy and Adventure movies together represent a large number of average ratings
3) Sci-Fi, Horror, Action and Comedy are the remaining major genres and have lower than average ratings
4) There are a few smaller genres that have very high ratings. These include Film Noir, Documentary, War and IMAX
5) The Horror genre has a significantly lower average rating. This would be due to the number of "B" slasher movies available

Next let's look at the number of movies by genre.



Observations

1) Drama and Comedy have disproportionally larger number of movies than the other genres
2) The average number of movies per genre is `1077.8` with a median of `535.5`

Let's look at the movies by the genres assigned to them, which may include more than one.

```
#generate a datset of ratings by genres as they are assigned to the movies
#genres can have one or more assigned to a movie
ratings_by_genres <- edx %>%
  group_by(genres) %>%
  summarize(ratings = n(), avg=mean(rating), movies = n_distinct(movieId))

# calculate the number of genres combinations
number_of_genre_groups <- nrow(ratings_by_genres)

# calculate the top 10 genre groupings by number of movies within them
top_10_genres <- ratings_by_genres %>% arrange(desc(movies)) %>% head(10)

# calculate the bottom 10 genres groupings by the number of movies with them
bottom_10_genres <- ratings_by_genres %>% arrange(movies) %>% head(10)

#calculate the number of genre combinations that have only onr (1) movie assigned
genre_groups_with_one_movie <- ratings_by_genres %>% filter(movies == 1)
```

There are `797` unique combinations of genres.

Let's look at the top 10 groups based on number of movies.

```
## # A tibble: 10 x 4
##    genres                ratings   avg movies
##    <chr>                   <int> <dbl>  <int>
##  1 Drama                  733296  3.71   1815
##  2 Comedy                 700889  3.24   1047
##  3 Comedy|Drama           323637  3.60    551
##  4 Drama|Romance          259355  3.61    412
##  5 Comedy|Romance         365468  3.41    379
##  6 Documentary             70041  3.82    350
##  7 Horror                  68738  2.88    267
##  8 Comedy|Drama|Romance   261425  3.65    255
##  9 Drama|Thriller         145373  3.45    192
## 10 Drama|War              111029  3.98    173
```

No suprise here that Drama and Comedy, or combinations including them, are at the top.

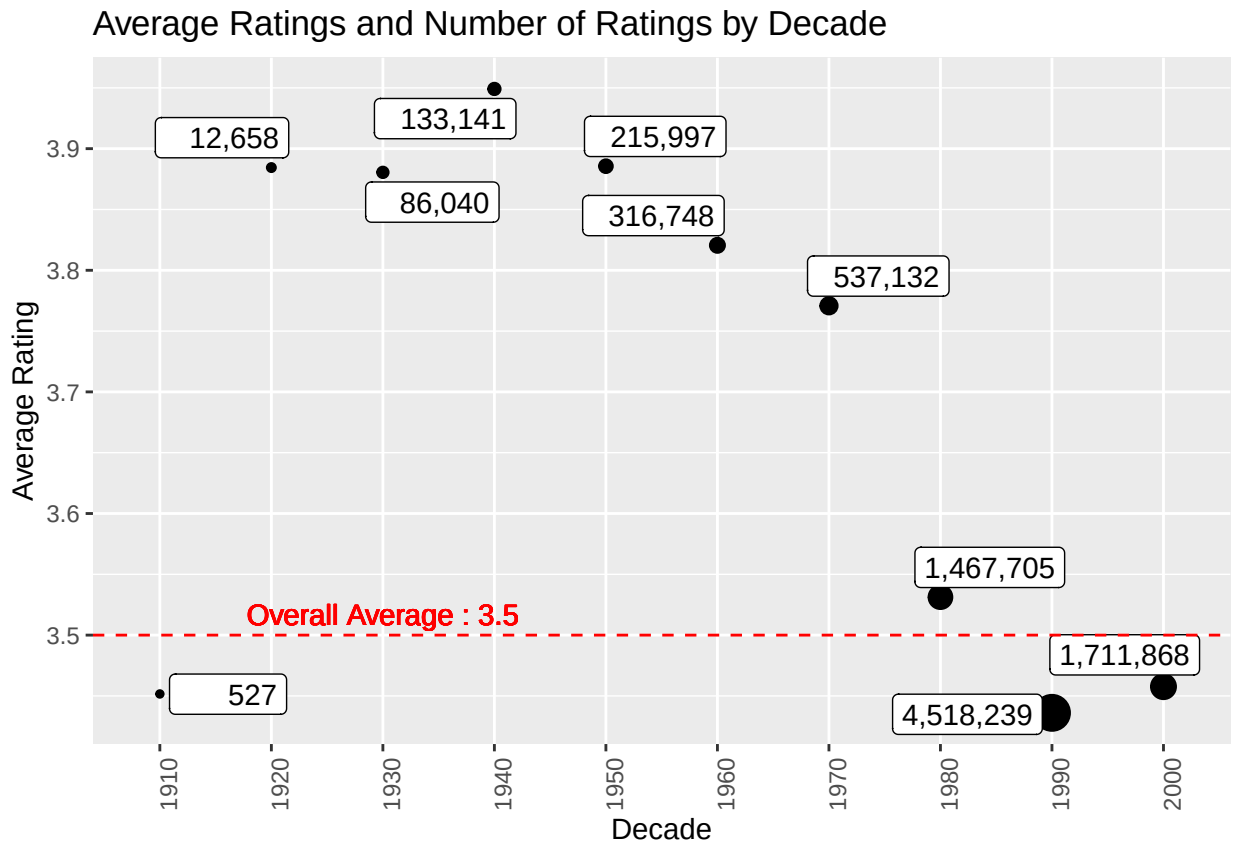
The bottom 10 are comprised of many genre combinations. So many that they have become obscure. There are 364 genre combinations with only one movie assigned to them. This many genres with a single movie provides little value when categorizing movies. When so many categories have one entry, then the categories are too finite.

```
bottom_10_genres
```

```
## # A tibble: 10 x 4
##    genres                                              ratings   avg movies
##    <chr>                                                 <int> <dbl>  <int>
##  1 (no genres listed)                                        7  3.64      1
##  2 Action|Adventure|Animation|Children|Comedy|Fantasy      187  2.99      1
##  3 Action|Adventure|Animation|Children|Comedy|IMAX          66  3.30      1
##  4 Action|Adventure|Animation|Children|Fantasy            737  2.70      1
##  5 Action|Adventure|Animation|Children|Sci-Fi              50  2.97      1
##  6 Action|Adventure|Animation|Comedy|Drama               1902  3.51      1
##  7 Action|Adventure|Animation|Comedy|Sci-Fi                 3  4         1
##  8 Action|Adventure|Animation|Drama|Fantasy|Sci-Fi        239  3.40      1
##  9 Action|Adventure|Animation|Fantasy|Sci-Fi              451  3.45      1
## 10 Action|Adventure|Animation|Horror|Sci-Fi              4087  3.38      1
```

Next, let's look at the years movies were released.

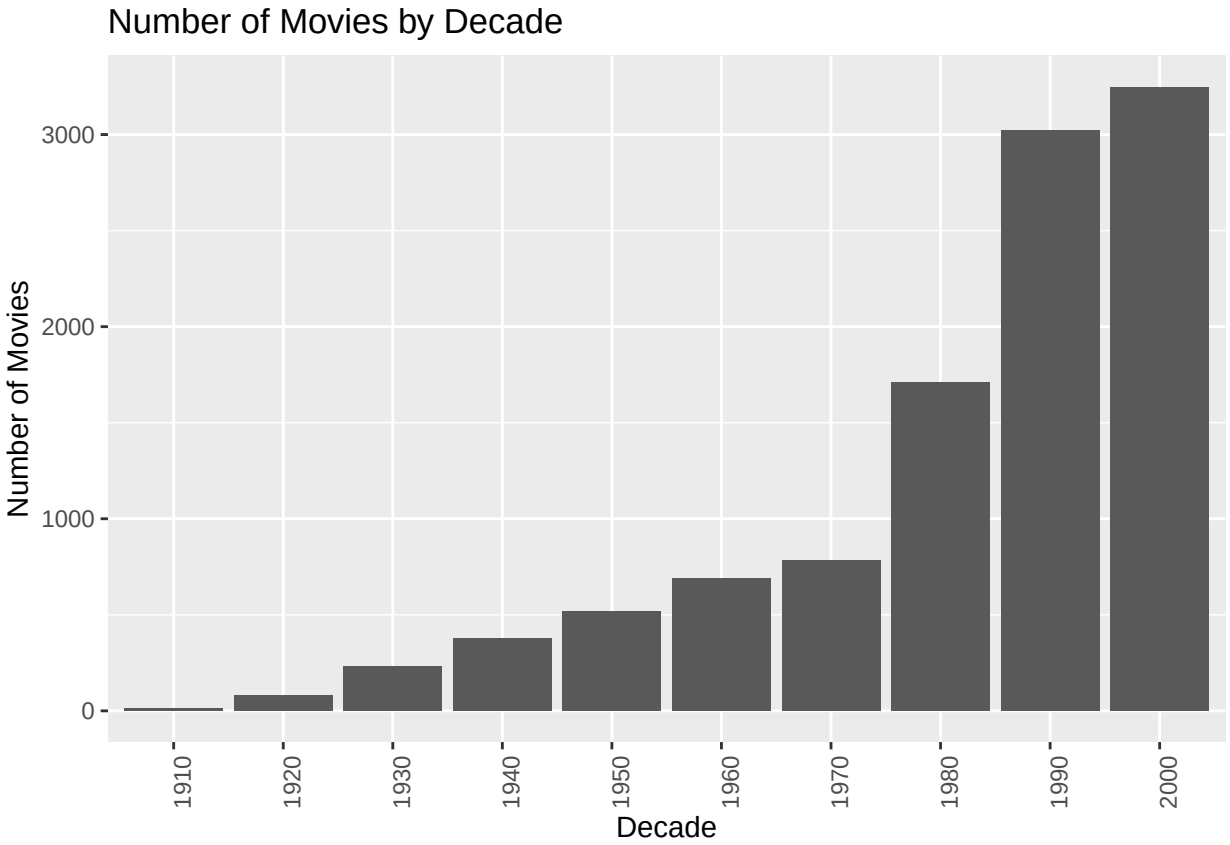
We will start looking at decades to help group them. We do this because we generally think of "movies from the 90's" for example. Like music, we tend to group releases into decades so we will start there.



Observations

1) Most ratings are for movies released in the past two decades
2) Movies released in the 90's make up the majority of the ratings
3) Movies from the past two decades have a lower than average rating
4) With the exception of the movies from 1910, movies prior to 1990 have higher than average ratings

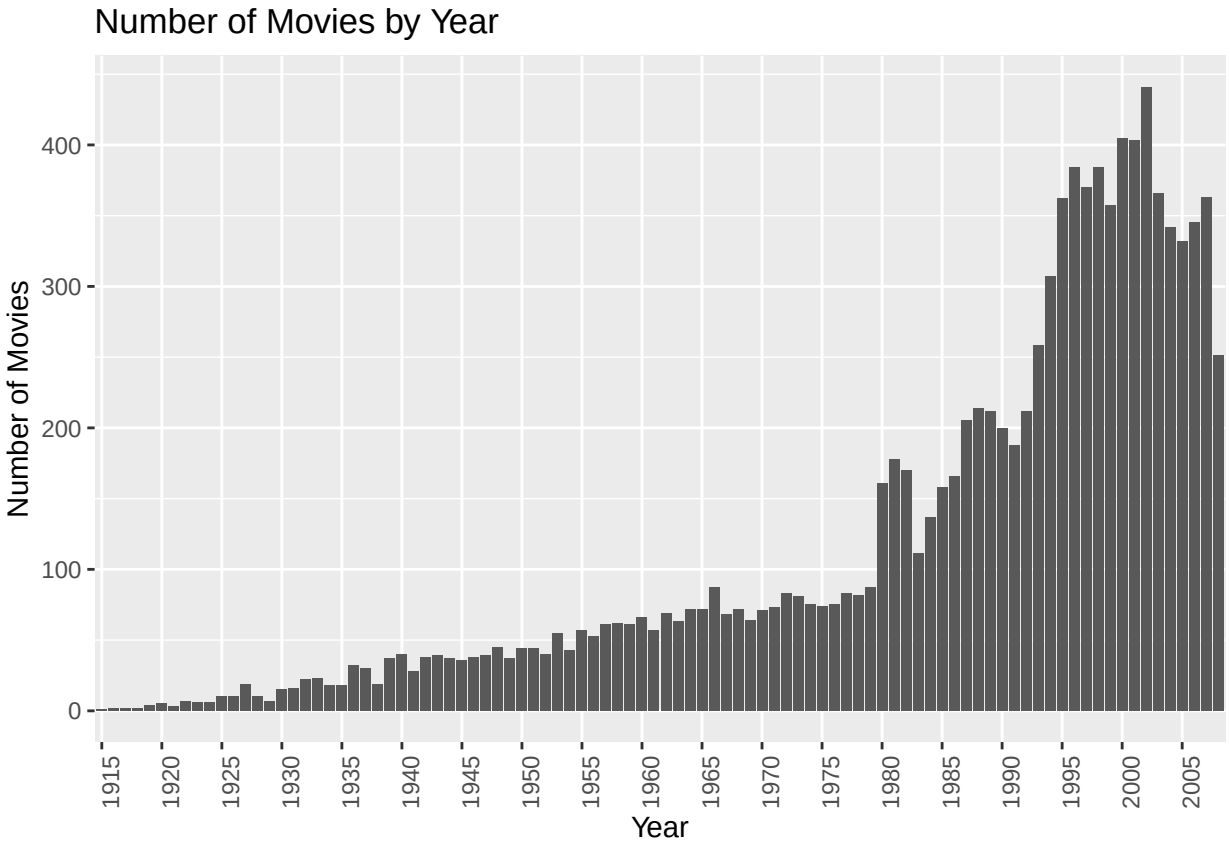
Let's look at the number of movies by decade.



Observations

1) Most of the movies were released in the past two decades
2) The number of movies by decade drops quickly as we step back in time

Let's drill into this to year.



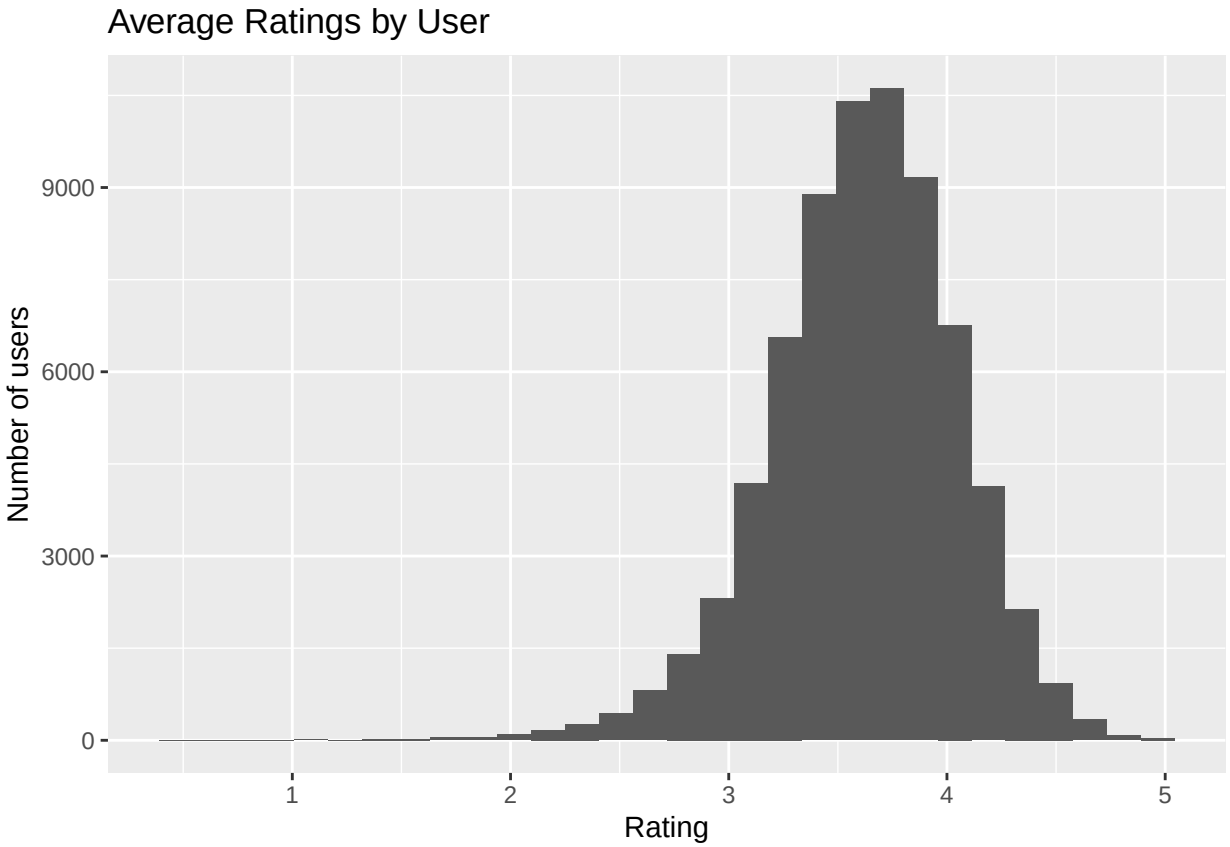

```
## # A tibble: 10 x 4
##    year  movies ratings   avg
##    <fct>  <int>   <int> <dbl>
##  1 2002     441  272180  3.45
##  2 2000     405  382763  3.40
##  3 2001     403  305705  3.44
##  4 1996     384  593518  3.36
##  5 1998     384  402187  3.42
##  6 1997     370  429751  3.36
##  7 2003     366  211397  3.46
##  8 2007     363   75788  3.53
##  9 1995     362  786762  3.44
## 10 1999     357  489537  3.45
```

Observations

1) We can see that there are some years that have disproportionately more movies than the other years within the same decade.
2) Not surprising, we see that the years with the most releases are over recent years.

User Exploration

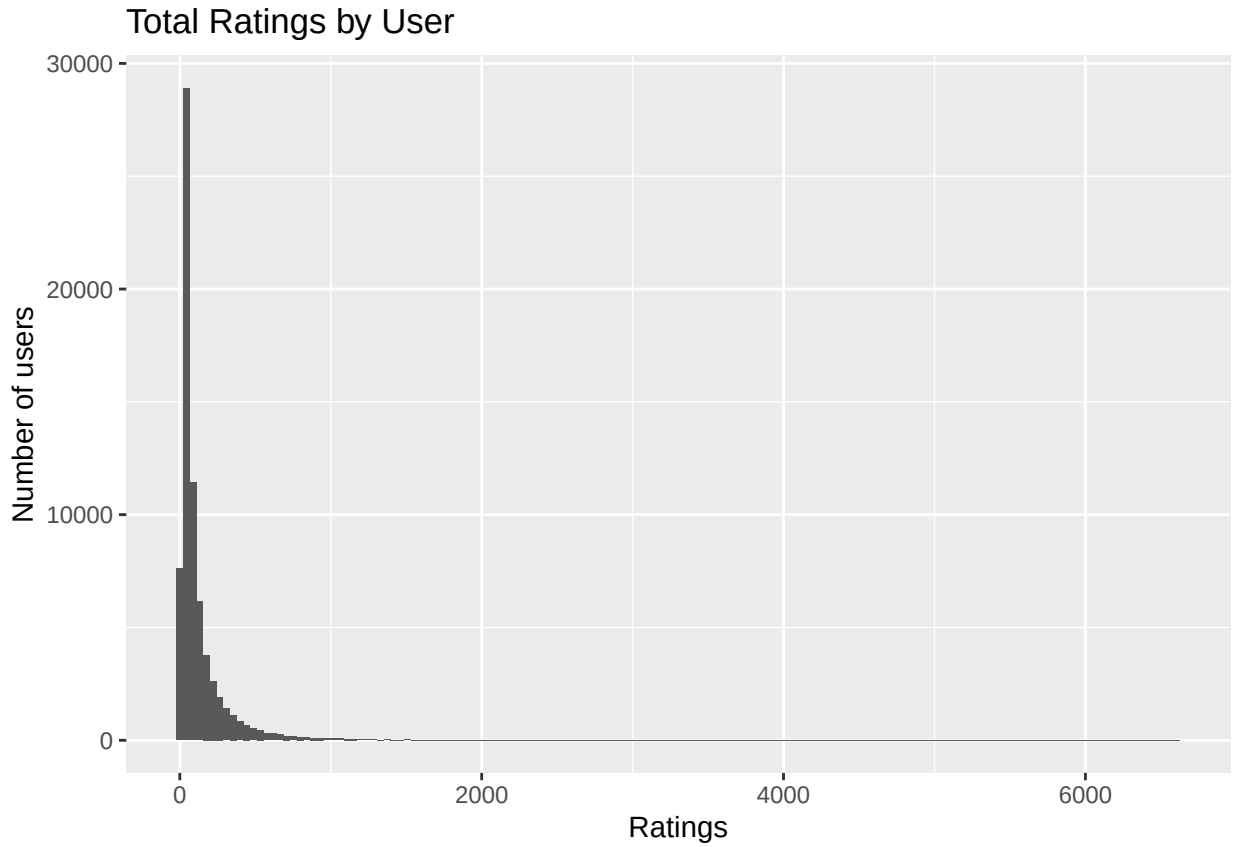
Let's look at how users rate movies on average. We expect that there are users who generally rate higher than others, or lower than others and we expect to see that some users rate more movies than others.



Observations

1) The user / rating relationship seems to follow a normal distribution
2) As we near the high and low end scores there are fewer users rating at those levels
3) With an overall average of 3.5 we see that most users actually rate higher so there is something else at play here

Let see how many movies users tend to rate.




```
## # A tibble: 10 x 3
##    userId ratings   avg
##     <int>   <int> <dbl>
##  1  59269    6616  3.26
##  2  67385    6360  3.20
##  3  14463    4648  2.40
##  4  68259    4036  3.58
##  5  27468    4023  3.83
##  6  19635    3771  3.50
##  7   3817    3733  3.11
##  8  63134    3371  3.27
##  9  58357    3361  3.00
## 10  27584    3142  3.00
```

Observations

1) The average number of ratings per user is `129`
2) There are `611` users who have 1,000 or more ratings
3) `1`% of the users (with 1,000 ratings or more) represent `10`% of the total ratings
4) There are two users with more than 6,000 ratings! Did they actually watch all these movies and remember them all so that they could rate them accurately? Not likely, however, we will leave this data in the set as we have no other evidence that disputes this.

Summary of Data Exploration

We looked at the data in many different ways to try to understand how the various dimensions of the data play out. We saw how blockbusters have many more ratings than the obscure movies. They also had higher

than average ratings. This would be expected, but we saw this first hand.

We looked at how the movies are assigned to genres and found that there are a few genres where the mojority of the movies and ratings can be found. We saw that Drama, Crime and Romance have higher than average ratings, while Horror saw the worst ratings overall.

Movies have more than one genre assigned to them in many cases and we saw that, while the top genre combinations made sense, there were many cominations that just had a single movie assigned to them, suggesting that these groupings have little value in their categorization of genres. A group of one, is not a group, and we saw many of these.

Most of the movies in our set were released in the past two decades and the majority of ratings were for those movies, however even though there were more movies released in the 2000's, there were significantly more ratings for movies in the 90's, suggesting that the movies of the 90's are more popular than the other decades.

Breaking down the releases into years showed that there are some years where the number of releases are higher than the other years in the same decade.

Average ratings for movies prior to 1990 have higher average ratings than movies released after.

Users tend to rate movies higher than average, however there are a disproportionate number of ratings from users who rated 1,000 or more movies. This group represented 1% of the users, but contributed 10% of the ratings. This group also has a lower than average rating for movies than those who rated less than 1,000 movies.

MODEL CREATION

Based on our observations, let's start building our RMSE model. Our goal is to acheive an RMSE of less than .8649. We will start with a baseline and then add from there. If an effect or preditor doesn't help us to attain our goal we will not add it to the final model.

We will work entirely on the training set, but will break this up into a training set and a test set. Once we get the model where we want it, we will do a final validation against the original validation/test set.

First, lets take our training set and break into a training set and a test set.

```
setSeed(1)
test_index <- createDataPartition(y=edx$rating, times=1, p=0.1, list=FALSE)
train_set <- edx[-test_index,]
temp <- edx[test_index,]

# Make sure userId and movieId in test set are also in train set

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

# Add rows removed from the test set back into training set

removed <- anti_join(temp, test_set)
train_set <- rbind(train_set, removed)


# calculate the overall average of the training set and assign it to mu
mu <- mean(train_set$rating)
```

Baseline RMSE

Let's start with the overall rating average of `3.5`

Method	RMSE
Baseline	1.060362

We are starting with an RMSE value of `1.0604`. This represents the effect of applying the overall rating average.

Movie Effect on RMSE

Based on our observations of our data, it is expected that individual movie ratings will be a major contributor to our model. We will add this next and see how that impacts our model

```
setSeed(1)

# create movie effect model
model_movie <- train_set %>%
  group_by(movieId) %>%
  summarize(b_movie = mean(rating - mu))

# apply it to the test set we created from the full training set
prediction <- test_set %>%
  left_join(model_movie, by="movieId") %>%
  mutate(pred = mu + b_movie) %>%
  .$pred

# calculate the RMSE
rmse_movie <- RMSE(prediction, test_set$rating)
rmse_movie
```

```
## [1] 0.9429615
```

The movie effect was significant so we will add that to our model.

Method	RMSE
Baseline	1.0603622
+ Movie Effect	0.9429615

User Effect on RMSE

Users have basic preferences and they tend to be generally easy on their ratings or hard, so it is expected that the user effect will also have a significant positive effect on in our model.

```
setSeed(1)

# add user effect to model
model_user <- train_set %>%
  left_join(model_movie, by="movieId") %>%
  group_by(userId) %>%
  summarize(b_user = mean(rating - mu - b_movie))

# apply model to test set
```

```
prediction <- test_set %>%
  left_join(model_movie, by="movieId") %>%
  left_join(model_user, by="userId") %>%
  mutate(pred = mu + b_movie + b_user) %>%
  .$pred

# calculate RMSE
rmse_user <- RMSE(prediction, test_set$rating)
rmse_user
```

```
## [1] 0.8646843
```

The user effect has provided a significant improvement in our model as well so we will add it as well.

Method	RMSE
Baseline	1.0603622
+ Movie Effect	0.9429615
+ User Effect	0.8646843

We have passed our goal of 0.8649, but we are not done.

We will now add decade.

Decade Effect on RMSE

```
setSeed(1)

# add decade to our model
model_decade <- train_set %>%
  left_join(model_movie, by="movieId") %>%
  left_join(model_user, by="userId") %>%
  group_by(decade) %>%
  summarize(b_decade = mean(rating - mu - b_movie - b_user))

# apply it to the test set
prediction <- test_set %>%
  left_join(model_movie, by="movieId") %>%
  left_join(model_user, by="userId") %>%
  left_join(model_decade, by="decade") %>%
  mutate(pred = mu + b_movie + b_user + b_decade) %>%
  .$pred

# calculate the RMSE
rmse_decade <- RMSE(prediction, test_set$rating)
rmse_decade
```

```
## [1] 0.8644729
```

Decade provides a very small increase, but we will keep it in our model.

Method	RMSE
Baseline	1.0603622
+ Movie Effect	0.9429615

Method	RMSE
+ User Effect	0.8646843
+ Decade Effect	0.8644729

Can year assist our model? Let's check it out.

Year Effect on RMSE

```
setSeed(1)

# add year to model
model_year <- train_set %>%
  left_join(model_movie, by="movieId") %>%
  left_join(model_user, by="userId") %>%
  left_join(model_decade, by="decade") %>%
  group_by(year) %>%
  summarize(b_year = mean(rating - mu - b_movie - b_user - b_decade))

# apply to test set
prediction <- test_set %>%
  left_join(model_movie, by="movieId") %>%
  left_join(model_user, by="userId") %>%
  left_join(model_decade, by="decade") %>%
  left_join(model_year, by="year") %>%
  mutate(pred = mu + b_movie + b_user + b_decade + b_year) %>%
  .$pred

#calculate RMSE
rmse_year <- RMSE(prediction, test_set$rating)
rmse_year
```

```
## [1] 0.8643301
```

Another small improvement. We will add it to our model.

Method	RMSE
Baseline	1.0603622
+ Movie Effect	0.9429615
+ User Effect	0.8646843
+ Decade Effect	0.8644729
+ Year Effect	0.8643301

The final effect is movie genres.

Genres Effect on RMSE

```
setSeed(1)

# add genres to model
model_genres <- train_set %>%
  left_join(model_movie, by="movieId") %>%
```

```
  left_join(model_user, by="userId") %>%
  left_join(model_decade, by="decade") %>%
  left_join(model_year, by="year") %>%
  group_by(genres) %>%
  summarize(b_genres = mean(rating - mu - b_movie - b_user - b_decade - b_year))

# apply to test set
prediction <- test_set %>%
  left_join(model_movie, by="movieId") %>%
  left_join(model_user, by="userId") %>%
  left_join(model_decade, by="decade") %>%
  left_join(model_year, by="year") %>%
  left_join(model_genres, by="genres") %>%
  mutate(pred = mu + b_movie + b_user + b_decade + b_year + b_genres) %>%
  .$pred

# calculate RMSE
rmse_genres <- RMSE(prediction, test_set$rating)
rmse_genres
```

```
## [1] 0.8640801
```

Another improvement.

Method	RMSE
Baseline	1.0603622
+ Movie Effect	0.9429615
+ User Effect	0.8646843
+ Decade Effect	0.8644729
+ Year Effect	0.8643301
+ Genres Effect	0.8640801

We can still apply one more component to our model: Regularization.

Regularization

Regularization allows us to accomodate oddities and inconsitencies in our model. For example, some movies have few ratings and may have unusualy high or low ratings as seen below.

The impact of this on our model can be illustrated by looking at the top 10 worst movies based on it's predictor.

```
## # A tibble: 10 x 2
##    Title                                    Error
##    <chr>                                    <dbl>
##  1 Besotted (2001)                           3.01
##  2 Hi-Line, The (1999)                       3.01
##  3 Accused (Anklaget) (2005)                 3.01
##  4 Confessions of a Superhero (2007)         3.01
##  5 War of the Worlds 2: The Next Wave (2008) 3.01
##  6 SuperBabies: Baby Geniuses 2 (2004)       2.77
##  7 Disaster Movie (2008)                     2.75
##  8 From Justin to Kelly (2003)               2.64
##  9 Hip Hop Witch, Da (2000)                  2.60
```

```
## 10 Criminals (1996)                    2.51
```

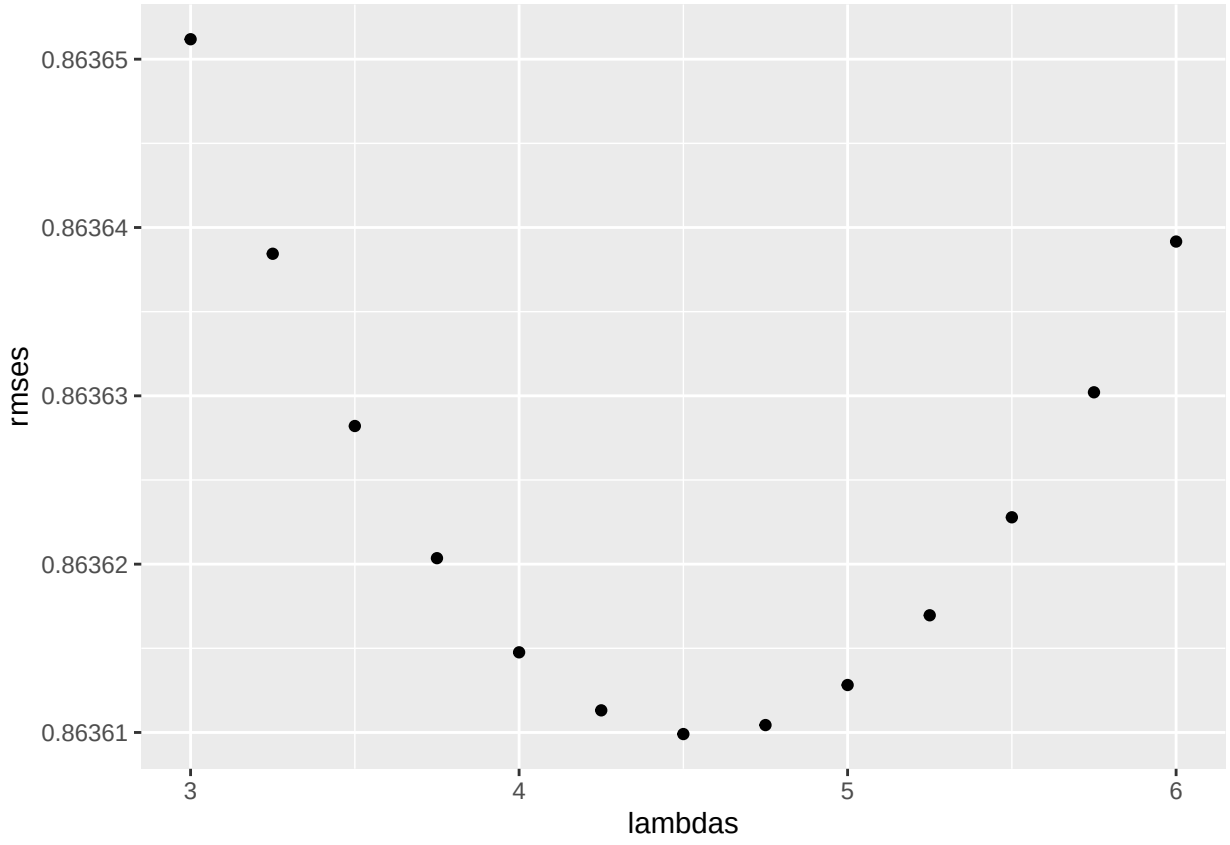
We can see that these are obscure movies and have a ver poor score.

This creates noise in our model and can increase our RMSE. We want to introduce penalized estimates in order to fine tune our model.

First we have to find the right tuning parameter using cross-validation.

```
setSeed(1)

# check for lambdas between 3 and 6 using .25 increments
lambdas <- seq(3,6, .25)

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

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

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

  b_decade <- train_set %>%
    left_join(b_movie, by="movieId") %>%
    left_join(b_user, by="userId") %>%
    group_by(decade) %>%
    summarize(b_decade =sum(rating - mu - b_movie - b_user)/(n() + l) )

  b_year <- train_set %>%
    left_join(b_movie, by="movieId") %>%
    left_join(b_user, by="userId") %>%
    left_join(b_decade, by="decade") %>%
    group_by(year) %>%
    summarize(b_year =sum(rating - mu - b_movie - b_user - b_decade)/(n() + l) )

  b_genres <- train_set %>%
    left_join(b_movie, by="movieId") %>%
    left_join(b_user, by="userId") %>%
    left_join(b_decade, by="decade") %>%
    left_join(b_year, by="year") %>%
    group_by(genres) %>%
    summarize(b_genres =sum(rating - mu - b_movie - b_user - b_decade - b_year)/(n() + l) )


  prediction <- test_set %>%
    left_join(b_movie, by="movieId") %>%
    left_join(b_user, by="userId") %>%
    left_join(b_decade, by="decade") %>%
    left_join(b_year, by="year") %>%
    left_join(b_genres, by="genres") %>%
    mutate(pred = mu + b_movie + b_user + b_decade + b_year + b_genres) %>%
    .$pred
```

```
  return(RMSE(prediction, test_set$rating))
})

# plot lambas versu rmse results
qplot(lambdas, rmses)
```




```
# identify the lowest RMSE value
lowest_rmse <- min(rmses)

# identity which lambda produced the lowest RMSE
lowest_lambda <- lambdas[which.min(rmses)]
```

The lowest lambda is `4.5`. We will use this in our final model.

Here are the results of our model against the training set.

Method	RMSE
Baseline	1.0603622
+ Movie Effect	0.9429615
+ User Effect	0.8646843
+ Decade Effect	0.8644729
+ Year Effect	0.8643301
+ Genres Effect	0.8640801
Final Regularized Model	0.8636099

By applying more predictors and adjusting for errors, we were able to get from a baseline RMSE of 1.06 to

0.8636099 and beat our goal.

FINAL RESULTS

Now that we have defined a model that meets our objective, it's time to apply it to our validation/test set created at the beginning to see how it will perform against an unknown data source.

Here is our final model.

```
lambda <- lowest_lambda

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

b_user <- edx %>%
  left_join(b_movie, by="movieId") %>%
  group_by(userId) %>%
  summarize(b_user = sum(rating - mu - b_movie)/(n() + lambda))

b_decade <- edx %>%
  left_join(b_movie, by="movieId") %>%
  left_join(b_user, by="userId") %>%
  group_by(decade) %>%
  summarize(b_decade =sum(rating - mu - b_movie - b_user)/(n() + lambda) )

b_year <- edx %>%
  left_join(b_movie, by="movieId") %>%
  left_join(b_user, by="userId") %>%
  left_join(b_decade, by="decade") %>%
  group_by(year) %>%
  summarize(b_year =sum(rating - mu - b_movie - b_user - b_decade)/(n() + lambda) )

b_genres <- edx %>%
  left_join(b_movie, by="movieId") %>%
  left_join(b_user, by="userId") %>%
  left_join(b_decade, by="decade") %>%
  left_join(b_year, by="year") %>%
  group_by(genres) %>%
  summarize(b_genres =sum(rating - mu - b_movie - b_user - b_decade - b_year)/(n() + lambda) )


prediction <- validation %>%
  left_join(b_movie, by="movieId") %>%
  left_join(b_user, by="userId") %>%
  left_join(b_decade, by="decade") %>%
  left_join(b_year, by="year") %>%
  left_join(b_genres, by="genres") %>%
  mutate(pred = mu + b_movie + b_user + b_decade + b_year + b_genres) %>%
  .$pred

rmse_final <- RMSE(prediction, validation$rating)
rmse_final
```

```
## [1] 0.8642542
```

The results of our model against the validation/test set was successful. Not quite as low as we estimated, but still better than our goal of .8649.

CONCLUSIONS

As we analyzed the data different ways we were able to select a number of effects that we felt would provide a lower RMSE value. Unfortunately, we had limited predictors to work with.

There are many more predictors that would make the model much better, such as demographic data for the user, directors and actors in the movie and so forth, however we did not have this data available in our dataset, so we have attained the best RMSE we could based on the data we have.

We found that two of the effects contributed the most to attaining our goal. Applying movies and users. This is not a surprise at all. We expected these two would have the largest effect.

Surprisingly, adding decade, year and genres brought us down further. It was not expected that all these would have a positive effect, but even though small, adding them all gave us an even lower RMSE score, allowing us to go beyond the goal.