HarvardX Data Science Capstone - MovieLens Recommendation System

Terry McLaughlin

6/7/2020

Contents

1 Overview

1.1 Introduction

This document discusses my analysis of the MovieLens dataset. This is the first of two projects required for the Capstone class, the ninth and final course HarvardX Data Science Program.

In this project I apply skills learned throughout the HarvardX Data Science course series to create a movie recommendation system using the MovieLens dataset. Code was provided in the instructions to generate the datasets used.

I compared various models to determine which Model produces the lowest root mean square error (RMSE). The RMSE tells me which model's predicted rating come the closest to "true" ratings. RMSE is the typical calculation used made when predicting a movie rating. This is similar to how the popular Netflix challenge decided on a winner.

RMSE can be calculated as follows:

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

Project RMSE scoring

0 points: No RMSE reported AND/OR code used to generate the RMSE appears to violate the edX Honor Code.
5 points: RMSE >= 0.90000 AND/OR the reported RMSE is the result of overtraining (validation set ratings used for anything except reporting the final RMSE value)
10 points: 0.86550 <= RMSE <= 0.89999
15 points: 0.86500 <= RMSE <= 0.86549
20 points: 0.86490 <= RMSE <= 0.86499
25 points: RMSE < 0.86490

For this project, I replicated the Machine Learning class exercise using Regularization. Then I developed additional models to see if I could produce a lower RMSE than shown in the course material. The goal is to obtain an RMSE lower than .86490 to obtain the maximum points.

Note: I also tried to produce models using other popular algorithms including knn, lm, and glm, however, I ran out of memory in all cases.

1.2 Dataset Summary

The MovieLens dataset provided has the following dimensions:

Description	Value
edx Rows	9,000,055
Validation set Rows	999,999
Data Columns	6 initial, 9 after further preparing dataset

2 Method and Analysis

2.1 Project Setup

Steps to prepare project

1. Data Loading: Load initial dataset using code provided in course materials
2. Load Libraries: Load libraries needed for project
3. Transform Dataset: Add additional columns to both edx and validation datasets for use in analysis

2.1.1 Data Loading

The r script to load the data was provided in the project instructions. The code created the edx dataset and a validation set of the MovieLens data. The MovieLens dataset is divided as follows:

1. edx data.frame: ~90% of MovieLens dataset (used to model)
2. validation data.frame: ~10% of MovieLens dataset (used to validate results of model)

2.1.2 Load Libraries

The libraries listed below are needed for my project.

```
##########################################
# Load libraries needed for project
##########################################
library(dslabs)
library(tidyverse)
library(lubridate)
library(caret)
library(dplyr)
library(kableExtra)
library(formattable)
library(ggrepel)
```

2.2 Approach / Overview

For this project, regularization was used. As demonstrated in the course material, regularization is a form of regression, that constrains / regularizes or shrinks the coefficient estimates towards zero. In other words, this technique discourages learning a more complex or flexible model, so as to avoid the risk of overfitting.

I wanted a better understanding of "regularization" so I did some research. I found an article that explained regularization at https://towardsdatascience.com/regularization-in-machine-learning-76441ddcf99a.

> *Regularization, significantly reduces the variance of the model, without substantial increase in its bias. So the tuning parameter λ, used in the regularization techniques described above, controls the impact on bias and variance. As the value of λ rises, it reduces the value of coefficients and thus reducing the variance. Till a point, this increase in λ is beneficial as it is only reducing the variance(hence avoiding overfitting), without loosing any important properties in the data. But after certain value, the model starts loosing important properties, giving rise to bias in the model and thus underfitting. Therefore, the value of λ should be carefully selected.*

For the models using Regularization, I tried several values of λ and the selected the value that would yield the lowest RMSE. The equation below is minimized to add a penalty.

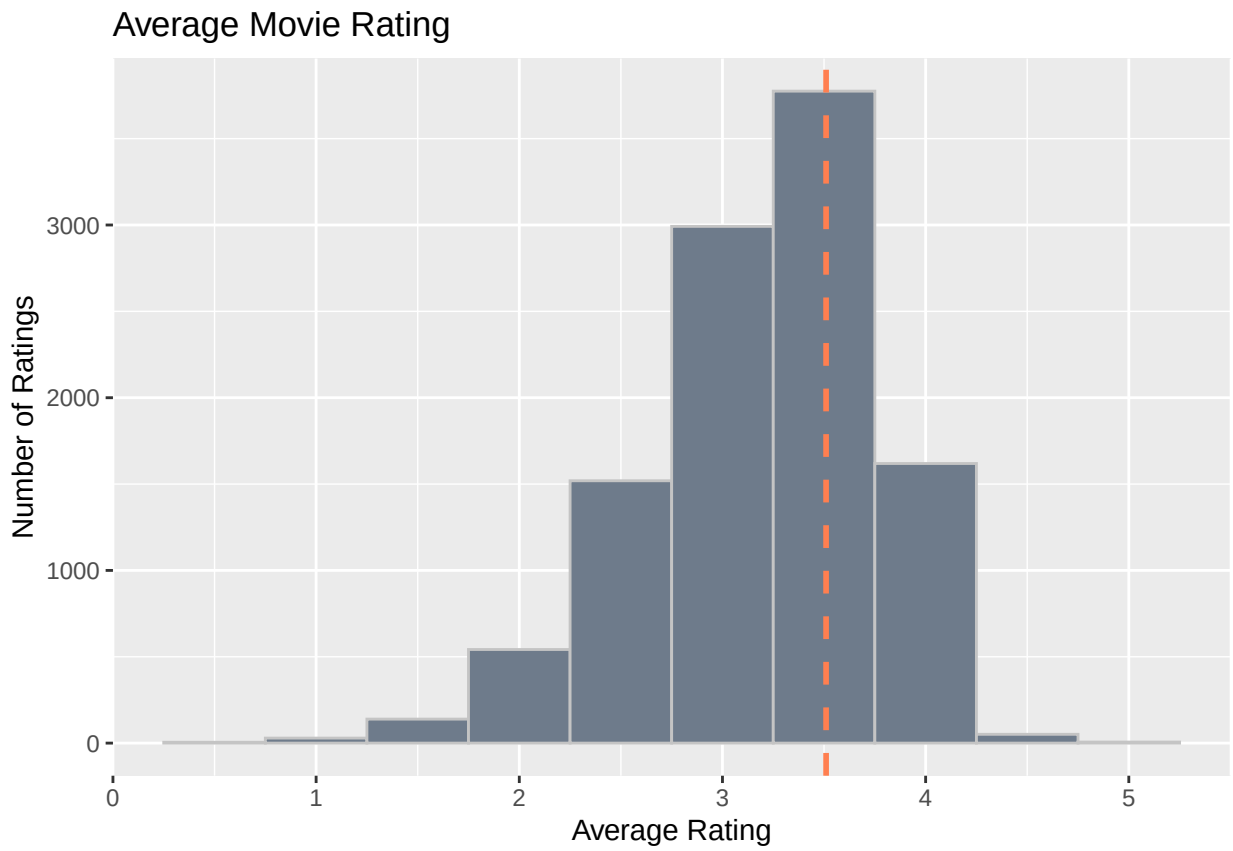
Note equation below is specifically for Regularization using both Movie and User variables. This equation can be simplified or expanded, depending on the number of variables used.

$$\frac{1}{N}\sum_{u,i}(y_{u,i}-\mu-b_i-b_u)^2+\lambda(\sum_i b_i^2+\sum_u b_u^2)$$

where:

n_i: Number of ratings for movie i
λ: Tuning parameter

A simple histogram of the average movie rating shows the mean for all movie ratings. The red vertical line indicates the mean is at 3.51.



The approach I took was to create models just using the data. Then I added in Regularization. This is a similar approach used in the course. I reran the approach used in class, and added in a time effect between the release year and the year the movie was rated.

2.3 Data Cleanup and Transformation

The initial dataset was transformed to add 3 additional columns: year_rated, year_release, and years_between. The following is the code I used.

```
################################################################################
# Prepare the edx and validation datasets to add columns needed for project
################################################################################
#FIRST add a column for year rated, release year, and number of years between
#    for both EDX and Validations sets
edxdata <-mutate(edx,  year_rated = year(as_datetime(timestamp)),
                 year_released = as.numeric(str_sub(title,-5,-2)),
                 year_between = year_rated - year_released)
validationdata <- mutate(validation, year_rated = year(as_datetime(timestamp)),
                         year_released = as.numeric(str_sub(title,-5,-2)),
                         year_between = year_rated - year_released)
```

3 Dataset General Statistics and Overview

Prior to building models to predict movie ratings, we need to examine and familiarize ourselves with the data.

3.1 Dataset Overview

First let's look at the columns in the dataset and the first few rows of data to understand what we have to work with. Both edx and validation datasets have identical format, just different number of rows.

Columns in Dataset

Columns	Provided or Added
userId	Provided
movieId	Provided
rating	Provided
timestamp	Provided
title	Provided
genres	Provided
year_rated	Added
year_released	Added
year_between	Added

EDX Data Header

	userId	movieId	rating	timestamp	title	genres
1	1	122	5	838985046	Boomerang (1992)	Comedy|Romance
2	1	185	5	838983525	Net, The (1995)	Action|Crime|Thriller
4	1	292	5	838983421	Outbreak (1995)	Action|Drama|Sci-Fi|Thriller
5	1	316	5	838983392	Stargate (1994)	Action|Adventure|Sci-Fi
6	1	329	5	838983392	Star Trek: Generations (1994)	Action|Adventure|Drama|Sci-Fi
7	1	355	5	838984474	Flintstones, The (1994)	Children|Comedy|Fantasy

Transformed EDX Data Header - scaled to fit page

userId	movieId	rating	timestamp	title	genres	year_rated	year_released	year_between
1	122	5	838985046	Boomerang (1992)	Comedy|Romance	1996	1992	4
1	185	5	838983525	Net, The (1995)	Action|Crime|Thriller	1996	1995	1
1	292	5	838983421	Outbreak (1995)	Action|Drama|Sci-Fi|Thriller	1996	1995	1
1	316	5	838983392	Stargate (1994)	Action|Adventure|Sci-Fi	1996	1994	2
1	329	5	838983392	Star Trek: Generations (1994)	Action|Adventure|Drama|Sci-Fi	1996	1994	2
1	355	5	838984474	Flintstones, The (1994)	Children|Comedy|Fantasy	1996	1994	2

Now lets look at summary statistics of the data, comparing the edx dataset to the Validation data set.

Summary Statistics

	EDX Data	Validation Data
No_Users	69878	68534
No_Movies	10677	9809
No_Genres	797	773
Earliest_Movie_Release_Year	1915	1915
Earliest_Movie_Rated_Year	1995	1995
Latest_Movie_Release_Year	2008	2008
Latest_Movie_Rated_Year	2009	2009
Year_Between	96	95
Median_Rating	4	4
Average_Rating	3.512465	3.512033

Note: Genres in the original dataset were concatenated. I split out the 19 genres for analysis.

We can also take a look at the top 10 and bottom 10 movies with over 1,000 ratings per movie.

Top 10 Movies with over 1,000 ratings each

Title	Movie ID	Count	Average Rating
Shawshank Redemption, The (1994)	318	28015	4.455131
Godfather, The (1972)	858	17747	4.415366
Usual Suspects, The (1995)	50	21648	4.365854
Schindler's List (1993)	527	23193	4.363493
Casablanca (1942)	912	11232	4.320424
Rear Window (1954)	904	7935	4.318651
Sunset Blvd. (a.k.a. Sunset Boulevard) (1950)	922	2922	4.315880
Third Man, The (1949)	1212	2967	4.311426
Double Indemnity (1944)	3435	2154	4.310817
Paths of Glory (1957)	1178	1571	4.308721

Bottom 10 Movies with over 1,000 ratings each

Title	Movie ID	Count	Average Rating
Battlefield Earth (2000)	3593	1869	1.568218
Police Academy 6: City Under Siege (1989)	2383	1092	1.723443
Spice World (1997)	1760	1288	1.739907
Police Academy 5: Assignment: Miami Beach (1988)	2382	1111	1.783078
Jaws 3-D (1983)	1389	1052	1.794202
Home Alone 3 (1997)	1707	1221	1.818591
Speed 2: Cruise Control (1997)	1556	2566	1.873733
Mighty Morphin Power Rangers: The Movie (1995)	181	1316	1.883739
Look Who's Talking Now (1993)	5452	1059	1.883853
Grease 2 (1982)	1381	1866	1.943462

There are 19 different genres in the dataset. Below is the average rating for each genre.

Ratings by Genre sorted descending by average rating

Genre	# Ratings	Average Rating
Film-Noir	118541	4.011625
Documentary	93066	3.783487
War	511147	3.780813
IMAX	8181	3.767693
Mystery	568332	3.677001
Drama	3910127	3.673131
Crime	1327715	3.665925
Animation	467168	3.600644
Musical	433080	3.563305
Western	189394	3.555918
Romance	1712100	3.553813
Thriller	2325899	3.507676
Fantasy	925637	3.501946
Adventure	1908892	3.493544
Comedy	3540930	3.436908
Action	2560545	3.421405
Children	737994	3.418715
Sci-Fi	1341183	3.395743
Horror	691485	3.269815

Interestingly, I had to google “Film-Noir”. According to the Oxford dictionary, it is a style or genre of cinematographic film marked by a mood of pessimism, fatalism, and menace. The term was originally applied (by a group of French critics) to American thriller or detective films made in the period 1944–54 and to the work of directors such as Orson Welles, Fritz Lang, and Billy Wilder. These are lowkey black and white movies that are associated with dark meanings. Examples are *Double Indemnity, The Big Sleep, Sunset Boulevard, Maltese Falcon, and The Big Heat* to name a few.

Next lets look at the number of ratings for each genre.

Ratings by Genre sorted descending by number of reviews

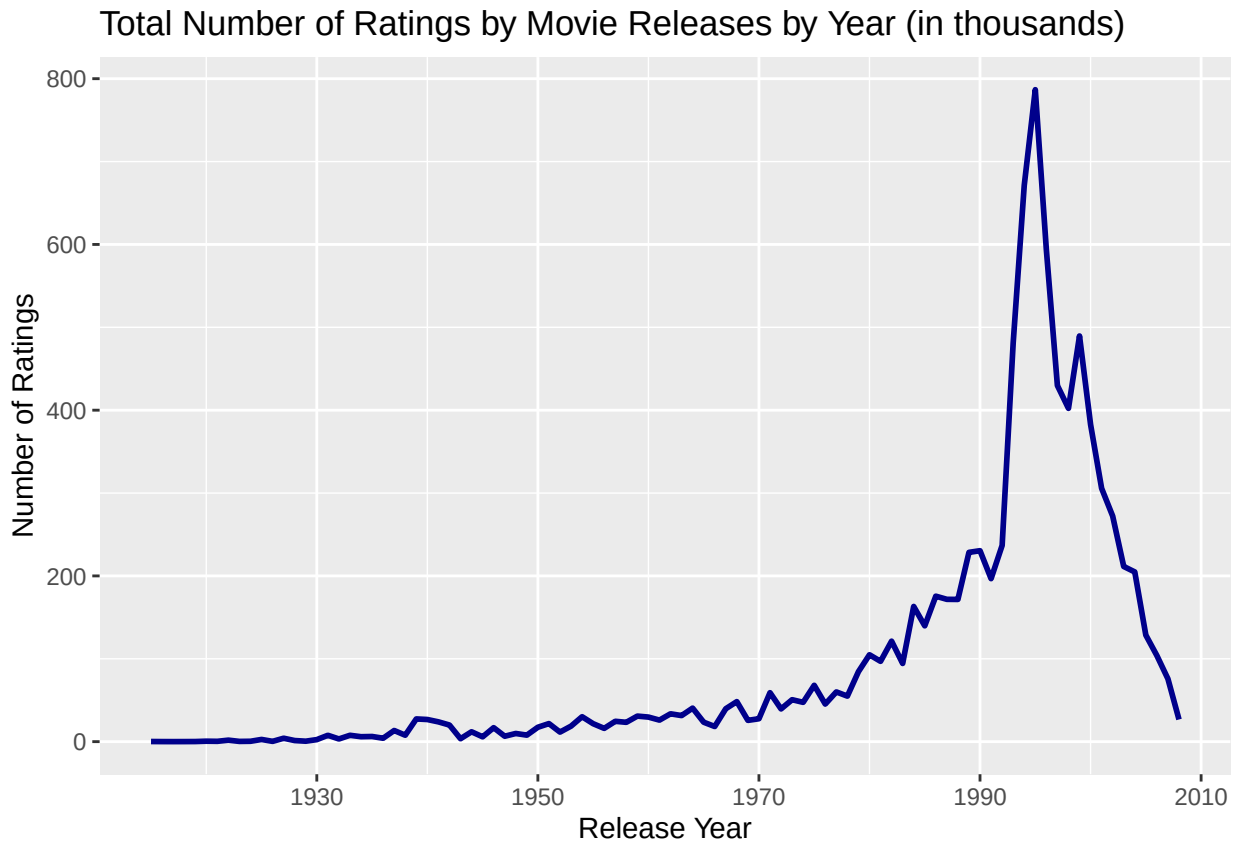
Genre	# Ratings	Average Rating
Drama	3910127	3.673131
Comedy	3540930	3.436908
Action	2560545	3.421405
Thriller	2325899	3.507676
Adventure	1908892	3.493544
Romance	1712100	3.553813
Sci-Fi	1341183	3.395743
Crime	1327715	3.665925
Fantasy	925637	3.501946
Children	737994	3.418715
Horror	691485	3.269815
Mystery	568332	3.677001
War	511147	3.780813
Animation	467168	3.600644
Musical	433080	3.563305
Western	189394	3.555918
Film-Noir	118541	4.011625
Documentary	93066	3.783487
IMAX	8181	3.767693

3.2 Dataset Visualization

It really helps to visualize the data to understand the dataset. I am going to start with some charts about the data in general, along with the r Code. Next, I will display charts that assist with the models I developed, specifically the year between release and review. Lastly, I included genre charts.

As shown in the chart below the number of ratings in the data spiked in the mid-1990s. The last movie release date in the dataset is 2008.

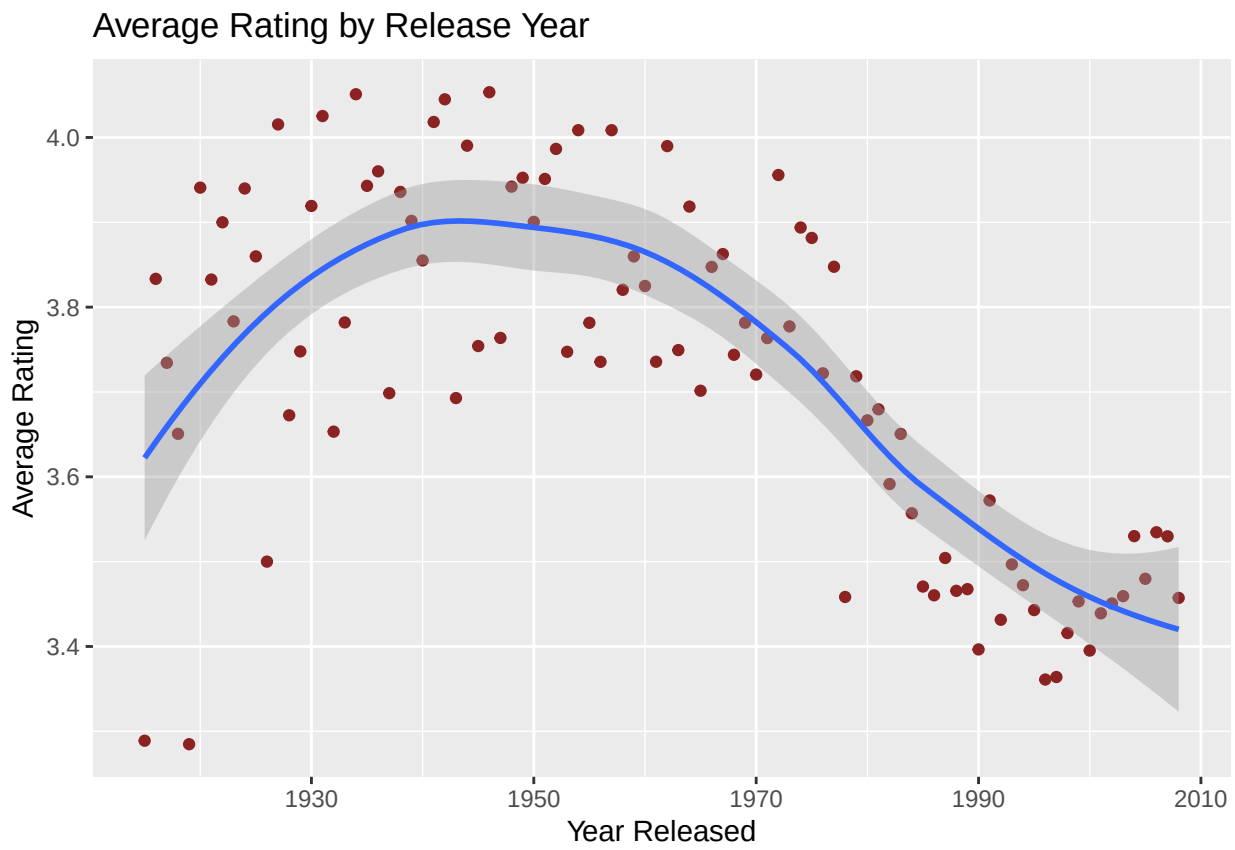
```
#Number of user ratings by release per year
  edxdata %>%
    group_by(year_released)  %>%
    summarize(Total=n())  %>%
    ggplot(aes(x=year_released, y=Total/1000)) +
    geom_line(color = "blue4", size=1) +
    ggtitle("Total Number of Ratings by Movie Releases by Year (in thousands)") +
    xlab("Release Year")+
    ylab("Number of Ratings")
```



The next 3 charts look at the “time” variable average ratings.

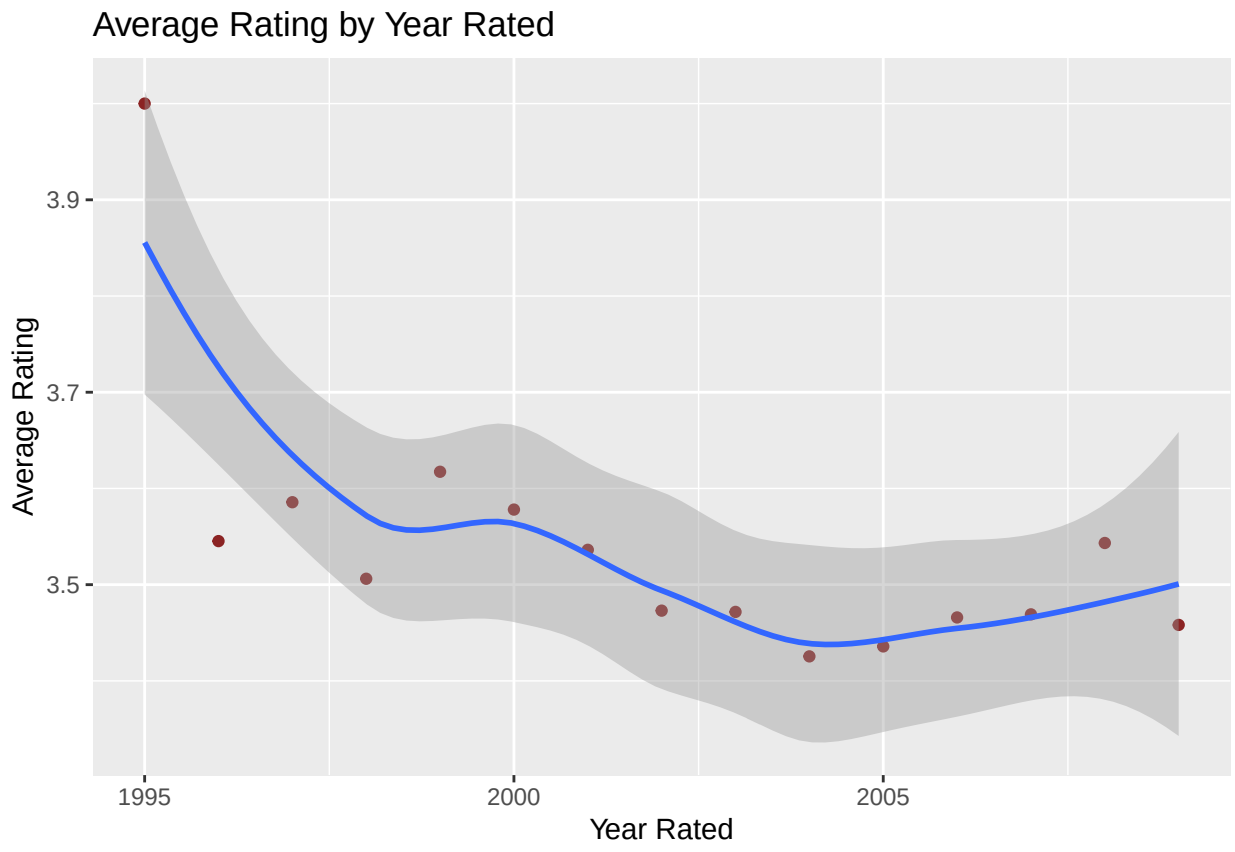
The first smooth density chart displays the average rating by release year. We discovered in our data analysis above that “Film-Noir” genre, movies in the 40s and 50s, has the highest ratings of all the genres. This chart validates that finding, with the curve peaking in the 40s and 50s.

```
#Average rating by year released
edxdata %>%
  group_by(year_released) %>%
  summarize(AverageRating = mean(rating)) %>%
  ggplot(aes(year_released, AverageRating)) +
  geom_point(color = "brown4") +
  geom_smooth() +
  ggtitle("Average Rating by Release Year") +
  xlab("Year Released") +
  ylab("Average Rating")
```



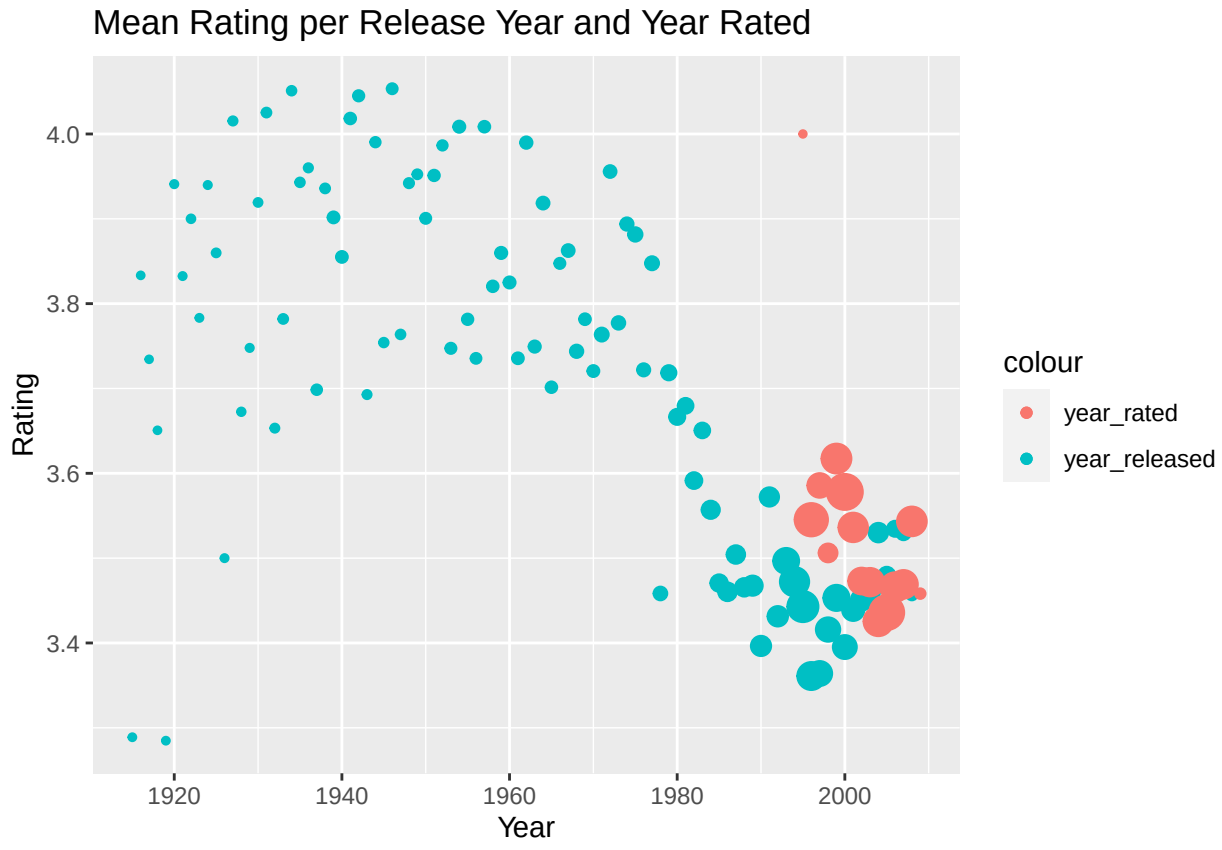
Next is a smooth density chart of average rating by year rated. The ratings appear to decline by year rated at first and then the ratings level off.

```
#Average rating by year rated
edxdata %>%
  group_by(year_rated) %>%
  summarize(AverageRating = mean(rating)) %>%
  ggplot(aes(year_rated, AverageRating)) +
  geom_point(color = "brown4") +
  geom_smooth() +
  ggtitle("Average Rating by Year Rated") +
  xlab("Year Rated") +
  ylab("Average Rating")
```



The last chart in this series combines both movie release year and the year the movies were rated. It visually tells us information about the “time” variable in the data. It displays the movies date back to the early 1900s, whereas the ratings only date after the 1990s. Each point represents a year. The blue points indicate the number of movies, and the red points indicate the number of movie reviews.

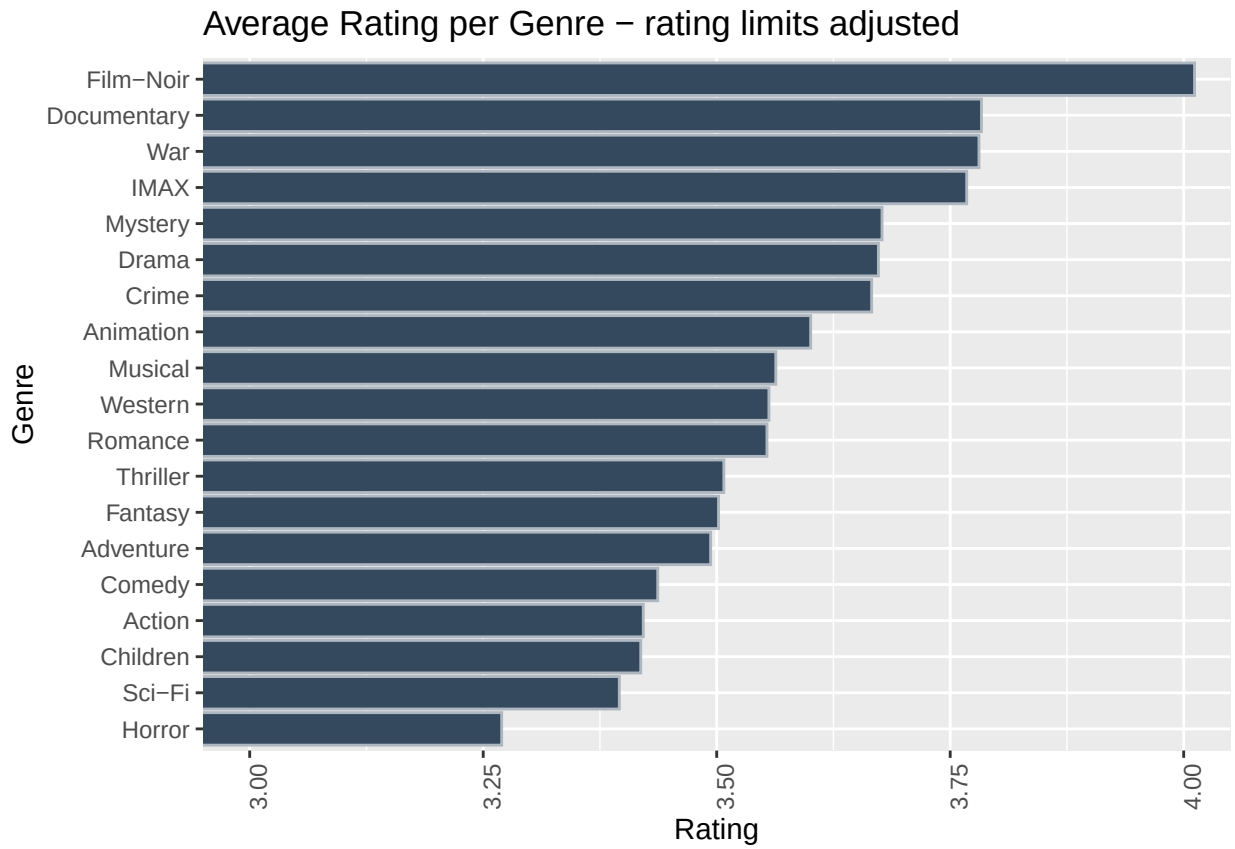
```
#Both Together - year vs rating and year released vs rating
  ggplot() +
    geom_point(data = edxdata %>%
    group_by(year_released) %>%
    summarize(Average_Rating = mean(rating), number=n()),
    aes(year_released, Average_Rating, col="year_released", size=number)) +
    geom_point(data = edxdata %>% group_by(year_rated) %>%
    summarize(Average_Rating = mean(rating), number=n()),
    aes(year_rated, Average_Rating, col="year_rated", size=number)) +
    ggtitle("Mean Rating per Release Year and Year Rated")  +
    xlab("Year") +
    ylab("Rating") +
    guides(size = FALSE)
```



We can also look at data provided for genre.

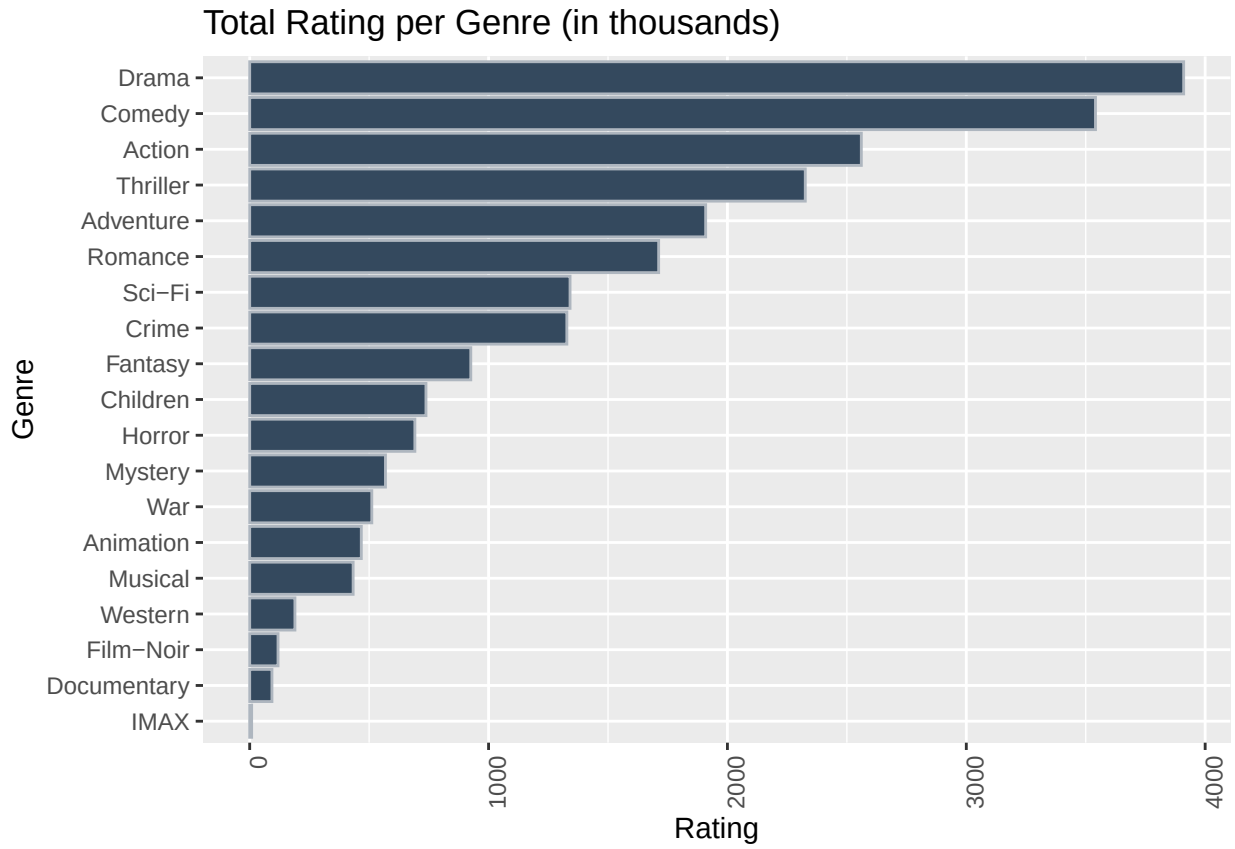
- The first chart shows Average Rating. Note due to all averages falling in the 3-4 range, the scale was adjusted between 3 and 4.
- The second chart shows the Total Number of Ratings variables. I was not surprised to see the number of total ratings chart. I suspected the order would be similar to that displayed, with Drama, Comedy, Action, and Thriller at the top of the chart.
- The third chart in this series shows the average rating per genre, with the size of the points indicated the number of ratings.

```
##############################################################################
# Display Genre Charts in Report
##############################################################################
#Average Rating Genre with rating axis adjusted
number_of_genres %>%
  ggplot(aes(x= reorder(genres, AverageRating), y = AverageRating)) +
  geom_bar(stat = "identity", color = "#aeb6bf", fill = "#34495e") +
  ggtitle("Average Rating per Genre - rating limits adjusted") +
  xlab("Genre")+
  ylab("Rating") +
  theme(axis.text.x = element_text(angle = 90, hjust = 1)) +
  coord_flip(ylim = c(3,4))
```

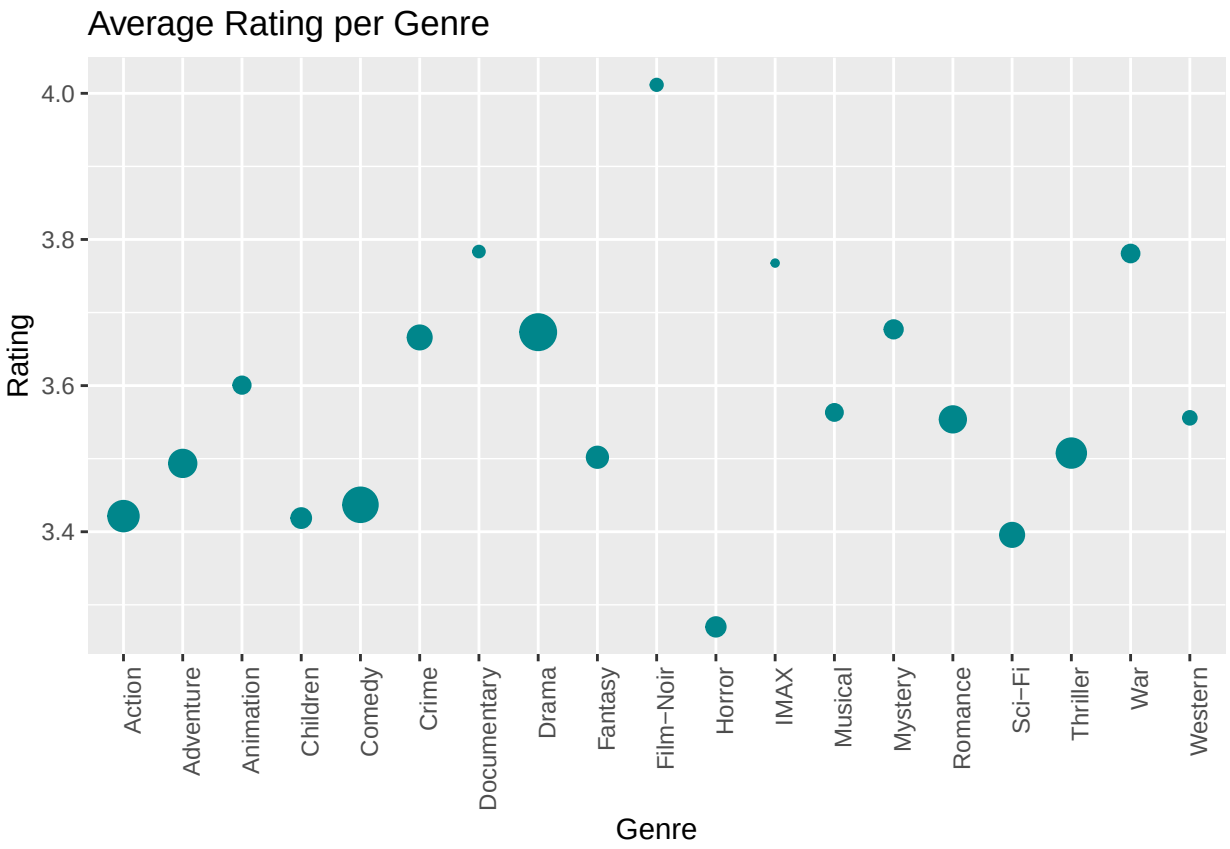



```
#Total Ratings per Genre Bar Chart
number_of_genres %>%
```

```
  ggplot(aes(x= reorder(genres, TotalRatings), y = TotalRatings/1000)) +
  geom_bar(stat = "identity", color = "#aeb6bf", fill = "#34495e") +
  ggtitle("Total Rating per Genre (in thousands)") +
  xlab("Genre")+
  ylab("Rating") +
  theme(axis.text.x = element_text(angle = 90, hjust = 1)) +
  coord_flip()
```




```
#Mean Ratings by Genre
number_of_genres %>%
  ggplot(aes(x= genres, y = AverageRating )) +
  geom_point(aes(size=TotalRatings/1000), color = "turquoise4") +
  ggtitle("Average Rating per Genre") +
  xlab("Genre")+
  ylab("Rating") +
  theme(axis.text.x = element_text(angle = 90, hjust = 1)) +
  guides(size=FALSE)
```


Average Rating per Genre
Rating
4.0
3.8
3.6
3.4
Action
Adventure
Animation
Children
Comedy
Crime
Documentary
Drama
Fantasy
Film−Noir
Horror
IMAX
Musical
Mystery
Romance
Sci−Fi
Thriller
War
Western
Genre

4 Results

In this section, I will present the models one by one with a summary section of the RMSE for each model.

4.1 Model 1 - Average Only

This is most basic of all of the models. This model computes the mean of the edx dataset compares it to the average of the validation dataset. The equation for this model is as follows:

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

where:

$Y_{u,i}$: Prediction
μ: "True Rating"
$\epsilon_{u,i}$: Independent errors sampled from the same distribution centered at p

The histogram for this model was presented earlier in this document in the Approach / Overview section. As a reminder the mean is 3.51.

```
##########################################################################
# MODEL 1 - AVERAGE ONLY
##########################################################################

#-------------------------AVERAGE ONLY ----------------------------------------------
# First lets just get the average of the ratings, not taking anything else into account
#     Most basic of all the analysis
rmse_results <- ''
options(digits=7)
mu_hat <- mean(edxdata$rating)
RMSE <- RMSE(validationdata$rating, mu_hat)

#-------------------------AVERAGE ONLY HISTOGRAM-----------------------------------------



#-------------------------AVERAGE ONLY RMSE-----------------------------------------
rmse_results <- tibble(method = "Model 1 - Average only", RMSE)
```

The RMSE obtained was 1.0612. This RMSE is not achieving the results we need, but will serve as a baseline for all other models.

Model 1 Results: Average Only

Method	RMSE
Model 1 - Average only	1.061202

4.2 Model 2 - Movie Effect

Model 2 accounts for the fact that different movies yield different ratings, which is known as “The Movie Effect”. For example, movie popularity due to star power and advertisement is one effect on movie ratings.

The equation for this model is as follows:

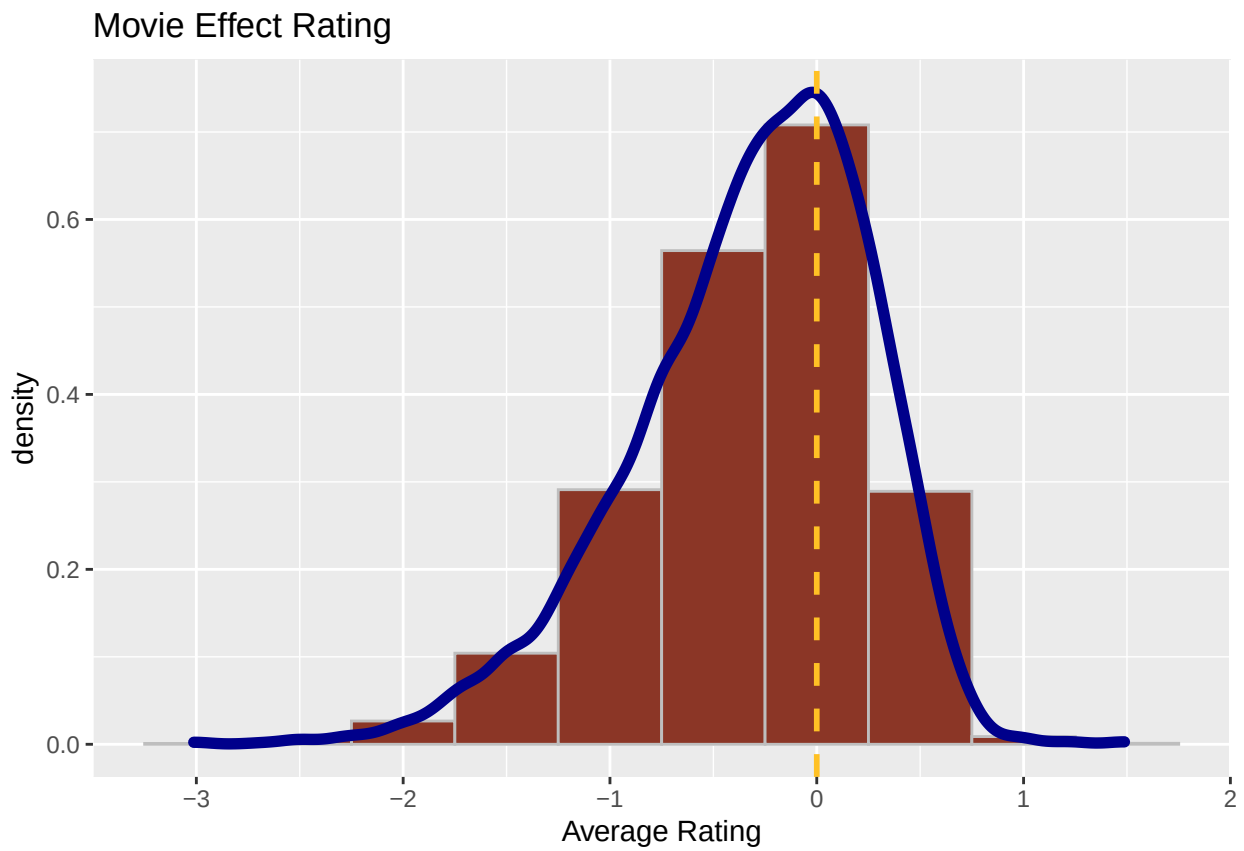
$$Y_{u,i} = \mu + b_i + \epsilon_{u,i}$$

where:

$Y_{u,i}$: Prediction
μ: “True Rating”
b_i: Indicates Movie Bias
$\epsilon_{u,i}$: Independent errors sampled from the same distribution centered at p

The Movie Effect summarizes the mean for each move and then calculates the bias. The bias is the individual movie rating minus the average move rating. We can see the shift in the histogram to center around zero. This model has a negative effect on ratings.

```
##############################################################################
# MODEL 2 - MOVIE EFFECT
##############################################################################

mu <- mean(edxdata$rating)
movie_avgs <- edxdata %>%
  group_by(movieId) %>%
  summarize(b_i = mean(rating - mu))
predicted_ratings <- mu + validationdata   %>%
  left_join(movie_avgs, by='movieId') %>%
  .$b_i

#-------------------------MOVIE EFFECT HISTOGRAM---------------------------------------
# Movie Effect Rating
movie_avgs%>%
  ggplot(aes(x=b_i, y = ..density..)) +
  geom_histogram(bins = 10, color = "gray", fill = "tomato4") +
  geom_density(method = "loess", size = 2, color = "darkblue") +
  geom_vline(xintercept=0, linetype="dashed", color = "goldenrod1",size=1) +
  ggtitle("Movie Effect Rating") +
  xlab("Average Rating")
```




```
#-------------------------MOVIE EFFECT RMSE ---------------------------------------------
model_m_rmse <- RMSE(validationdata$rating, predicted_ratings)

#add rows to a running table to summarize all model results
rmse_results <- bind_rows(rmse_results,
                 tibble(method="Model 2 - Movie Effect Model",RMSE = model_m_rmse))
```

The RMSE for the Movie Effect gets us a little closer to our goal at .9439. However, is not quite where we need to be. Here is where we stand.

Model 2 Results: Movie Effect

Method	RMSE
Model 1 - Average only	1.0612018
Model 2 - Movie Effect Model	0.9439087

4.3 Model 3 - Movie + Time Effect

For this model are adding an a variable to the movie effect that will have the number of years between when a movie was release to when the movie was rated. We will call this the time effect.

Again we will display our histogram, centering around zero. This is a little closer to normal.

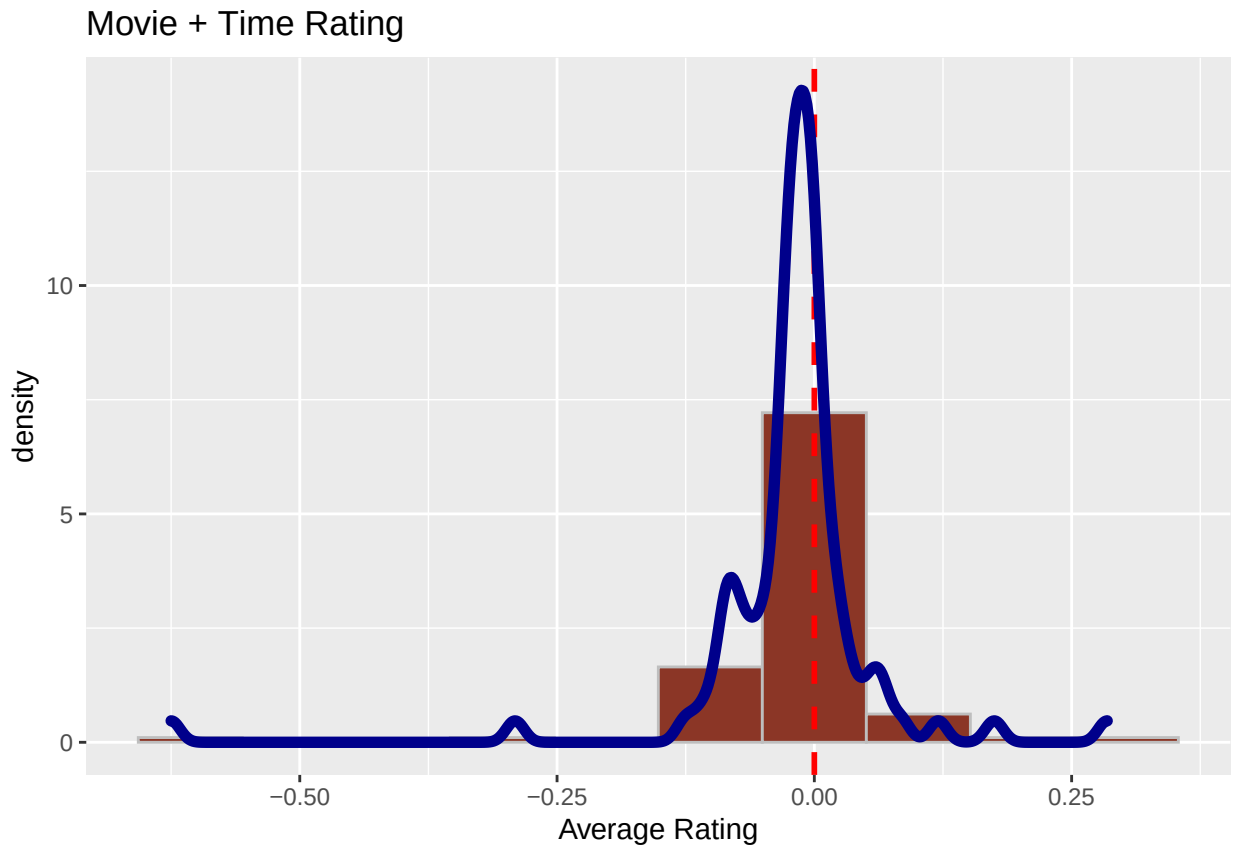
The equation for this model is as follows:

$$Y_{t,i} = \mu + b_i + b_t + \epsilon_{t,i}$$

where:

$Y_{u,i}$: Prediction
μ: "True Rating"
b_i: Indicates Movie Bias
b_t: Indicates Time (year between release and rating) Bias
$\epsilon_{t,i}$: Independent errors sampled from the same distribution centered at p

```
##########################################################################
# MODEL 3 - MOVIE + TIME EFFECT
##########################################################################
#-------------------------MOVIE + TIME EFFECT ------------------------------------

year_between_avgs <- edxdata %>%
  left_join(movie_avgs, by='movieId') %>%
  group_by(year_between) %>%
  summarize(b_t = mean(rating - mu - b_i))

predicted_ratings <- validationdata %>%
  left_join(movie_avgs, by='movieId') %>%
  left_join(year_between_avgs, by='year_between') %>%
  mutate(pred = mu + b_i + b_t) %>%
  pull(pred)

#-----------------------MOVIE + TIME EFFECT HISTOGRAM--------------------------------
# Movie Effect Rating
year_between_avgs %>%
  ggplot(aes(x=b_t, y = ..density..)) +
  geom_histogram(bins = 10, color = "gray", fill = "tomato4") +
  geom_vline(xintercept=0, linetype="dashed", color = "red",size=1) +
  geom_density(method = "loess", size = 2, color = "darkblue") +
  ggtitle("Movie + Time Rating") +
  xlab("Average Rating")
```




```
#-----------------------MOVIE + TIME EFFECT RMSE--------------------------------
model_my_rmse <- RMSE(predicted_ratings, validationdata$rating)

#add rows to a running table to summarize all model results
rmse_results <- bind_rows(rmse_results,
                          tibble(method="Model 3 - Movie + Time Effect Model",
                                 RMSE = model_my_rmse ))
```

The RMSE for the Movie + Time Effect puts us at .9425. We are tad closer, yet still not achieving our goal.

Model 3 Results: Movie + Time Effect

Method	RMSE
Model 1 - Average only	1.0612018
Model 2 - Movie Effect Model	0.9439087
Model 3 - Movie + Time Effect Model	0.9424734

4.4 Model 4 - Movie + User Effect

This is similar to the Model 2 the Movie Effect. In this model, we are adding an additional user variable. Users tend to rate things differently. Some rate high and some rate low. The scale is not consistent between individuals. I tend to be an optimist and rating things higher when I review things.

Again we will display our histogram, centering around zero, but closer to a normal distribution than the Movie + Time Effect model. You can see by the histogram that we are getting closer to the zero mark, and the deviations are not as wide as just the Movie Effect.

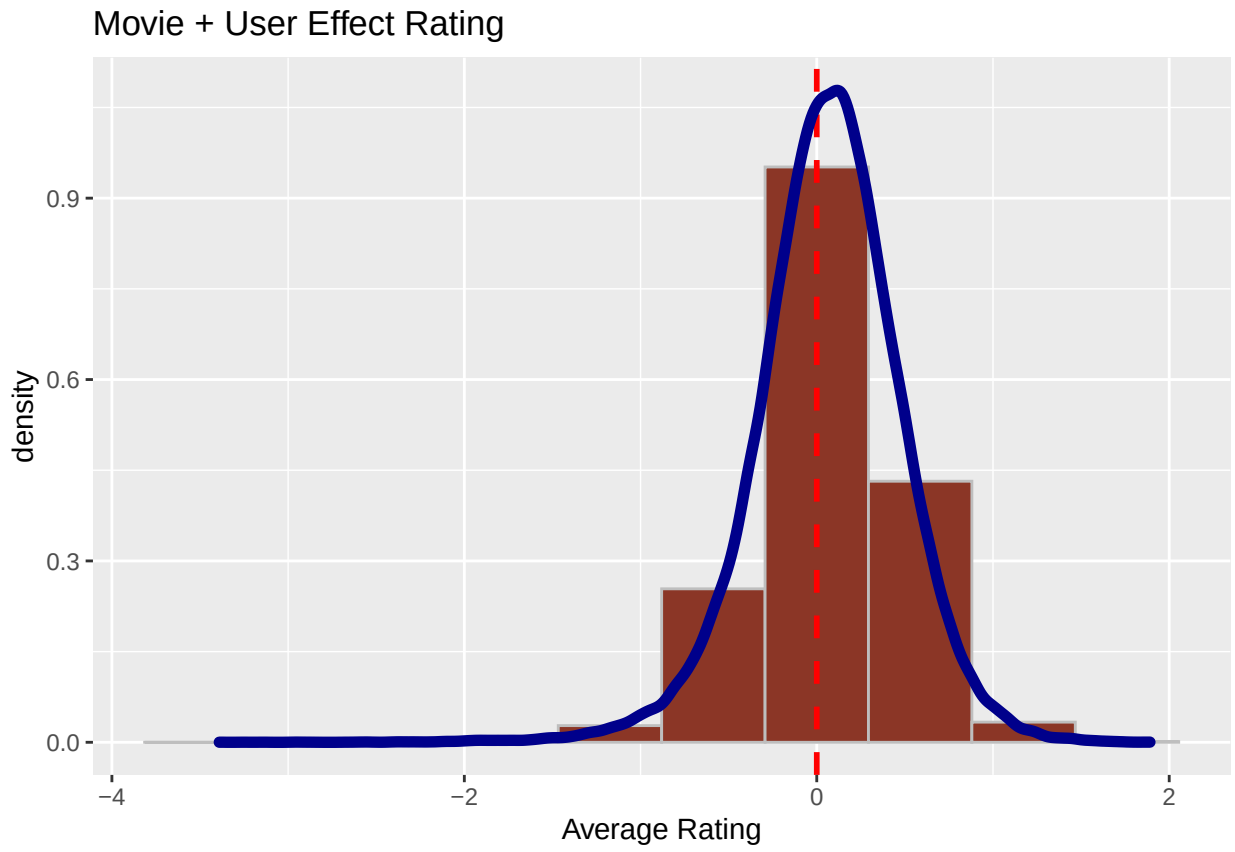
The equation for this model is as follows:

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

where:

$Y_{u,i}$: Prediction
μ: "True Rating"
b_i: Indicates Movie Bias
b_u: Indicates User Bias
$\epsilon_{u,i}$: Independent errors sampled from the same distribution centered at p

```
##############################################################################
# MODEL 4 - MOVIE + USER EFFECT
##############################################################################

#-------------------------MOVIE + USER EFFECT ---------------------------------------
#Now the Movie + user effect
user_avgs <- edxdata %>%
  left_join(movie_avgs, by='movieId') %>%
  group_by(userId) %>%
  summarize(b_u = mean(rating - mu - b_i))
predicted_ratings <- validationdata %>%
  left_join(movie_avgs, by='movieId') %>%
  left_join(user_avgs, by='userId') %>%
  mutate(pred = mu + b_i + b_u) %>%
  pull(pred)

#-------------------------MOVIE + USER EFFECT HISTOGRAM-----------------------------------------
# Movie Effect Rating
user_avgs%>%
  ggplot(aes(x=b_u, y = ..density..)) +
  geom_histogram(bins = 10, color = "gray", fill = "tomato4") +
  geom_vline(xintercept=0, linetype="dashed", color = "red",size=1) +
  geom_density(method = "loess", size = 2, color = "darkblue") +
  ggtitle("Movie + User Effect Rating") +
  xlab("Average Rating")
```




```
#-------------------------MOVIE + USER EFFECT RMSE ------------------------------------------

model_mu_rmse <- RMSE(validationdata$rating, predicted_ratings)

#add rows to a running table to summarize all model results
rmse_results <- bind_rows(rmse_results,
                          tibble(method="Model 4 - Movie + User Effect Model",
                                 RMSE = model_mu_rmse ))
```

The RMSE for the Movie + User Effect again gets us a little closer to our goal at .8653. We can still do better.

Model 4 Results: Movie + User Effect

Method	RMSE
Model 1 - Average only	1.0612018
Model 2 - Movie Effect Model	0.9439087
Model 3 - Movie + Time Effect Model	0.9424734
Model 4 - Movie + User Effect Model	0.8653488

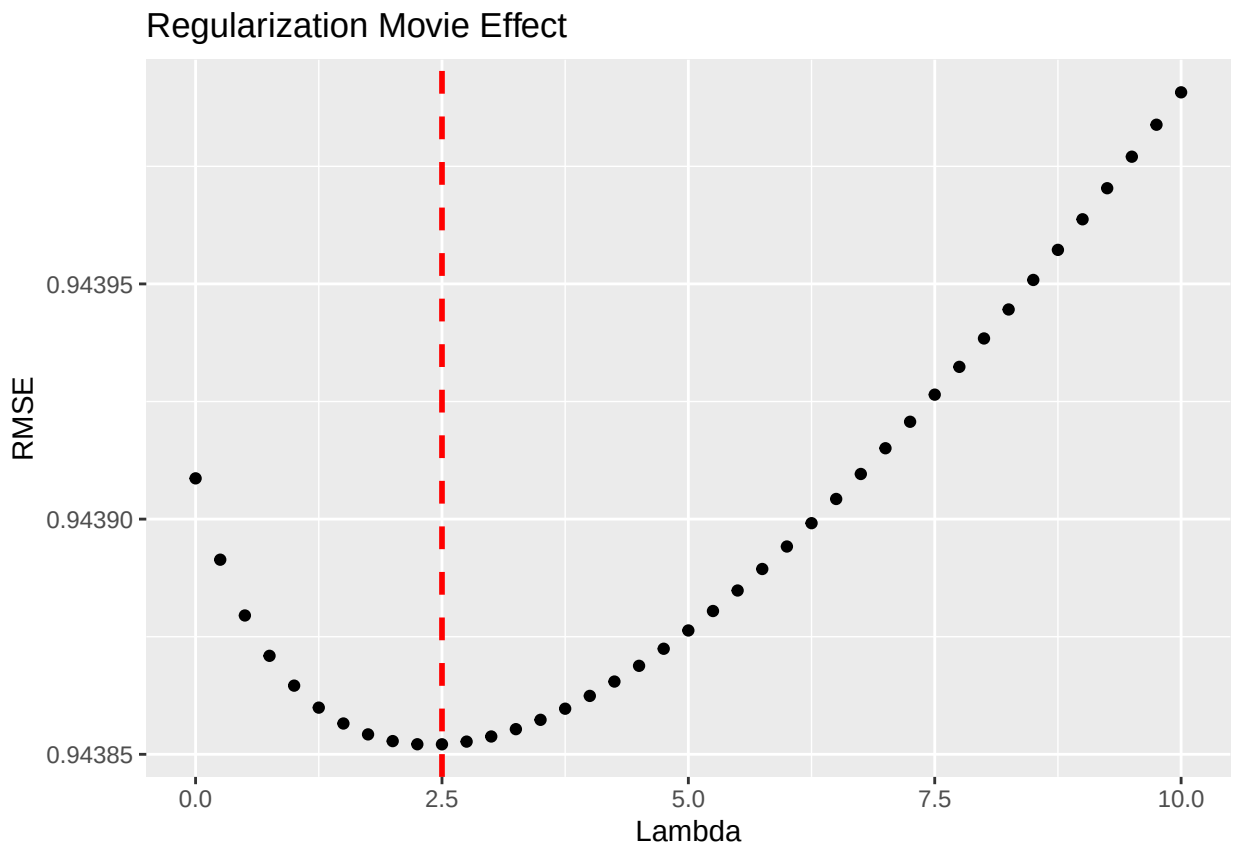
4.5 Model 5 - Regularization Movie Effect

Regularization takes care of noise and outliers in data, because they are penalized. It addresses the fact there are obscure movies not seen by many people, but yet rated by a few. What we do is run through the equation additional values of λ, our tuning parameter, to minimize the RMSE.

For reference here is our equation again:

$$\frac{1}{N}\sum_{i}(y_i - \mu - b_i)^2 + \lambda(\sum_{i} b_i^2)$$

```
##############################################################################
# MODEL 5 - REGULARIZATION MOVIE EFFECT
##############################################################################
#------------------------REGULARIZATION MOVIE EFFECT ------------------------------------
#Choose lambda with cross-validation
lambdas <- seq(0, 10, 0.25)

mu <- mean(edxdata$rating)

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

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

#----------------Plot the lambdas-----------------------------------------------------
best_Lambda <- lambdas[which.min(rmses)]
chartData <- as.data.frame(cbind(rmses, lambdas))

chartData %>% ggplot(aes(lambdas, rmses)) +
  geom_point() +
  geom_vline(xintercept=best_Lambda, linetype="dashed", color = "red",size=1)+
  ggtitle("Regularization Movie Effect") +
  xlab("Lambda") +
  ylab("RMSE")
```




```
#-------------------------REGULARIZATION MOVIE EFFECT RMSE ----------------------------------------
RMSE_RM = min(rmses)

#add rows to a running table to summarize all model results
rmse_results <- bind_rows(rmse_results,
                  tibble(method="Model 5 - Regularized Movie Effect Model",
                  RMSE= RMSE_RM, Lambda = best_Lambda))
```

The RMSE for the Regularization Movie Effect again gets us a little closer to our goal at .9439, with Lambda = 2.5. We will have to drill down further with more variables to get better results.

Model 5 Results: Regularization Movie Effect

Method	RMSE	Lambda
Model 1 - Average only	1.0612018	NA
Model 2 - Movie Effect Model	0.9439087	NA
Model 3 - Movie + Time Effect Model	0.9424734	NA
Model 4 - Movie + User Effect Model	0.8653488	NA
Model 5 - Regularized Movie Effect Model	0.9438521	2.5

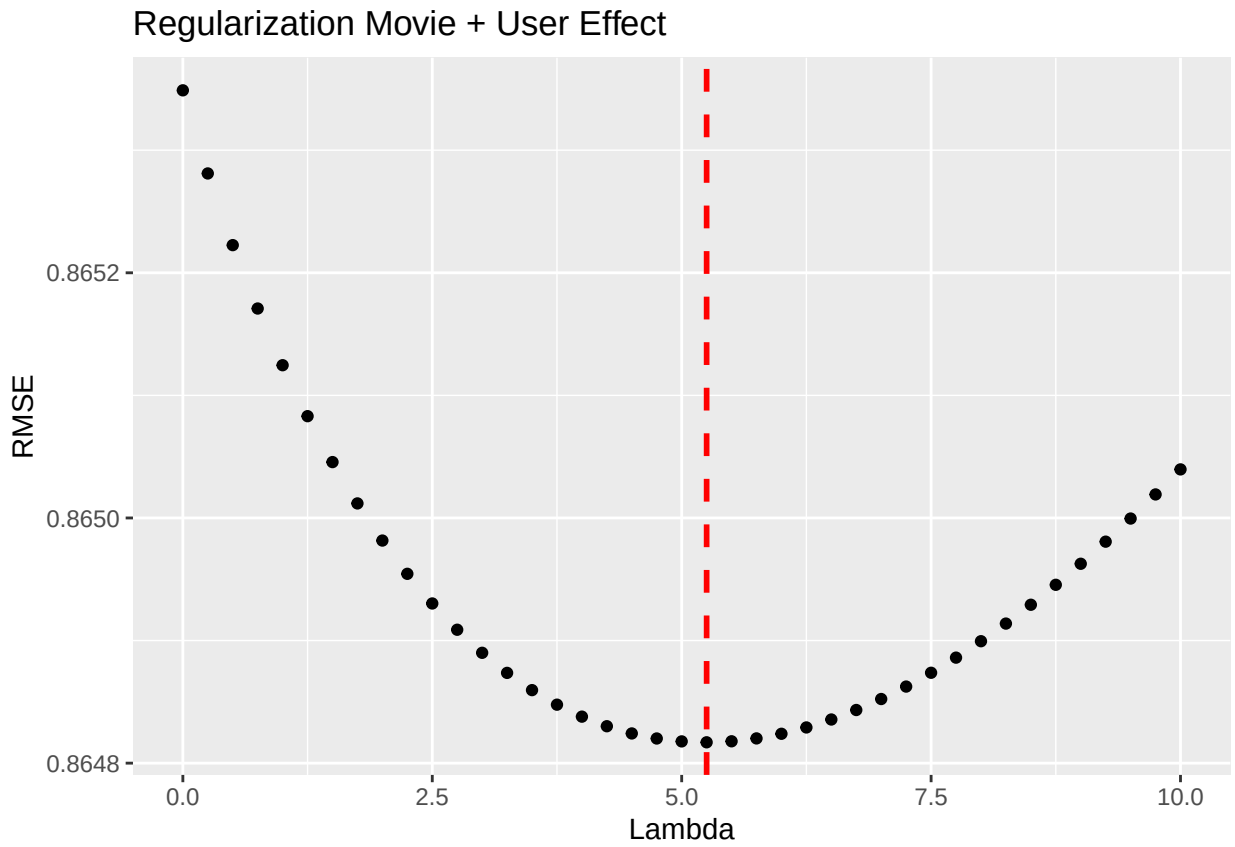
4.6 Model 6 - Regularization Movie + User Effect

This model we will add user to our regularization model.

For reference here is our equation again:

$$\frac{1}{N}\sum_{u,i}(y_{u,i}-\mu-b_i-b_u)^2+\lambda(\sum_i b_i^2+\sum_u b_u^2)$$

```
##############################################################################
# MODEL 6- REGULARIZATION MOVIE + USER EFFFECT
##############################################################################
#-------------------------REGULARIZATION MOVIE + USER EFFFECT ------------------------------------
#Choose lambda with cross-validation
lambdas <- seq(0, 10, 0.25)
rmses <- sapply(lambdas, function(l){
  mu <- mean(edx$rating)

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

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

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

  return(RMSE( predicted_ratings, validationdata$rating))
})

#----------------Plot the lambdas-------------------------------------------------------
best_Lambda <- lambdas[which.min(rmses)]
chartData <- as.data.frame(cbind(rmses, lambdas))

chartData %>% ggplot(aes(lambdas, rmses)) +
  geom_point() +
  geom_vline(xintercept=best_Lambda, linetype="dashed", color = "red",size=1)+
  ggtitle("Regularization Movie + User Effect") +
  xlab("Lambda") +
  ylab("RMSE")
```




```
#-------------------------REGULARIZATION MOVIE + USER EFFECT RMSE -------------------------------------
RMSE_RMU = min(rmses)

#add rows to a running table to summarize all model results
rmse_results <- bind_rows(rmse_results,
                tibble(method="Model 6 - Regularized Movie + User Effect Model",
                RMSE= RMSE_RMU, Lambda = best_Lambda))
```

The RMSE for the Regularization Movie + User Effect again gets us a little closer to our goal at .8648, with Lambda = 5.25. We have achieved our RMSE goal to be lower than .86490. However, let's add one more variable to see if we can get even better results.

Model 6 Results: Regularization Movie + User Effect

Method	RMSE	Lambda
Model 1 - Average only	1.0612018	NA
Model 2 - Movie Effect Model	0.9439087	NA
Model 3 - Movie + Time Effect Model	0.9424734	NA
Model 4 - Movie + User Effect Model	0.8653488	NA
Model 5 - Regularized Movie Effect Model	0.9438521	2.50
Model 6 - Regularized Movie + User Effect Model	0.8648170	5.25

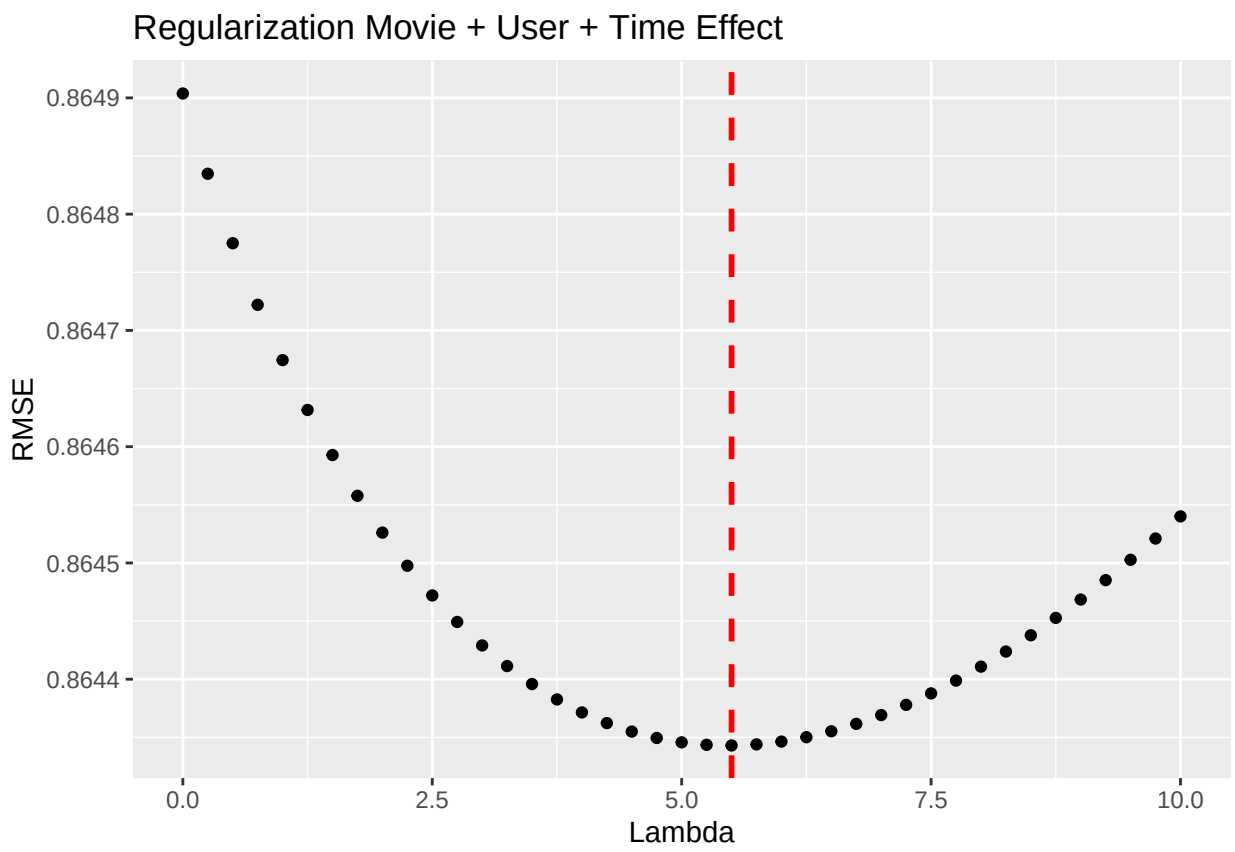
4.7 Model 7 - Regularization Movie + User + Time Effect

This model we will add time to our regularization model. Time is the number of years between when the movie was release to when the move was rated in years.

For reference here is our equation again with the additional variable:

$$\frac{1}{N}\sum_{t,u,i}(y_{t,u,i}-\mu-b_i-b_u-b_t)^2+\lambda(\sum_i b_i^2+\sum_u b_u^2+\sum_t b_t^2)$$

```
################################################################################
# MODEL 7 - REGULARIZATION MOVIE + USER + TIME EFFFECT
################################################################################
#-------------------------REGULARIZATION MOVIE + USER + TIME EFFFECT ---------------------------------
#Choose lambda with cross-validation
lambdas <- seq(0, 10, 0.25)
rmses <- sapply(lambdas, function(l){
  mu <- mean(edxdata$rating)

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

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

  b_yb <- edxdata %>%
    left_join(b_i, by="movieId") %>%
    left_join(b_u, by='userId') %>%
    group_by(year_between) %>%
    summarize(b_yb = sum(rating - b_i - b_u - mu)/(n()+l), n_yb = n())

  predicted_ratings <-
    validationdata %>%
    left_join(b_i, by = "movieId") %>%
    left_join(b_u, by = "userId") %>%
    left_join(b_yb, by = 'year_between') %>%
    mutate(pred = mu + b_i + b_u + b_yb) %>%
    .$pred

  return(RMSE(predicted_ratings, validationdata$rating))
})

#----------------Plot the lambdas-------------------------------------------------------
best_Lambda <- lambdas[which.min(rmses)]
chartData <- as.data.frame(cbind(rmses, lambdas))

chartData %>% ggplot(aes(lambdas, rmses)) +
  geom_point() +
  geom_vline(xintercept=best_Lambda, linetype="dashed", color = "red",size=1)+
  ggtitle("Regularization Movie + User + Time Effect") +
  xlab("Lambda") +
  ylab("RMSE")
```




```
#-------------------------REGULARIZATION MOVIE + + USER + TIME EFFFECT RMSE ----------------------------
RMSE_RMUT = min(rmses)

#add rows to a running table to summarize all model results
rmse_results <- bind_rows(rmse_results,
                tibble(method="Model 7 - Regularized Movie + User + Time Effect Model",
                RMSE= RMSE_RMUT, Lambda = best_Lambda))
```

The RMSE for the Regularization Movie + User + Time Effect achieves and even better result, with Lambda = 5.5 and RMSE of .8643.

Model 7 Results: Regularization Movie + User + Time Effect

Method	RMSE	Lambda
Model 1 - Average only	1.0612018	NA
Model 2 - Movie Effect Model	0.9439087	NA
Model 3 - Movie + Time Effect Model	0.9424734	NA
Model 4 - Movie + User Effect Model	0.8653488	NA
Model 5 - Regularized Movie Effect Model	0.9438521	2.50
Model 6 - Regularized Movie + User Effect Model	0.8648170	5.25
Model 7 - Regularized Movie + User + Time Effect Model	0.8643430	5.50

5 Conclusion

The goal of this project was to construct a machine learning algorithm from the MovieLens dataset to obtain an RMSE less than .86490. Two of my models achieved this goal: Regularized Movie + User Effect and Regularized Movie + User + Time Effect. Both of these models used the Regularization method demonstrated in the course materials. By using regularization, I was able to fulfill the project requirements.

The data provided was split with 90% used to “train” the model and 10% was used to validate the model. Additionally, I added three data columns to calculate the number of years between the movie release and the movie rating. I presented results using seven different models. Of these, the Regularization Movie + User + Time (years between release and review) Effect model produced the lowest RMSE at .8653, which meets the project requirement.

My key takeaways while doing this project:

- I liked using a real-world example, the Netflix challenge. This shows the practical use of the HarvardX Data Science program.
- I was unable to use other modeling tools learned in the course materials on this project. My computer wouldn’t handle using knn, lm, or others. However, I have one more capstone project to work through and turn in.
- It took a long time to format this document using R Markdown. Simple things like centering and indentation took investigation along with trial and error. Even this list took research, as it was different from the RMarkdown cheatsheet syntax.
- Formattable is a cool function, but I couldn’t get it to work with latex in RMarkdown. I wound up using Kable to format tables.
- It seemed to take a long time to ”knit” the RMarkdown pdf file. However, my neighbor is a data scientist. She told me some of her models take up to 2 weeks to run. I think I was just impatient, because mine ran in a matter of minutes.
- I learned a lot in the HarvardX Data Science program. I am extremely glad I took this class. Now, when I watch the briefings on the COVID-19 pandemic or even election results, I have a basic understanding of ”models” when they discuss them.