HarvardX PH125.9x Capstone Project: Movielens 10m

David Mc Manus

6/14/2022

Contents

Introduction

Background

Launched in 2006,and running through 2009, the Netflix Prize open competition sought submissions of recommendation systems that would represent a 10% improvement on their own system, Cinematch. This competition became quite popular with thousands of teams competing for the grand prize of $1,000,000 in 2009. The wining team, BellKor's Pragmatic Chaos, achieved a final validation RSME of 0.8567(1).

Data Set

The Movielens 10M dataset, compiled by GroupLens Research, contains 10 million ratings for 10,000 movies, that were submitted to movielens.com. Each row represents an individual rating, and contains the user id, movie id, rating, timestamp, rating, and genre.

Aim

Although the competition is no longer running, this project aims to create a movie recommendation system that will improve upon the winning test RSME of 0.8567.

Required Packages

Memory Allocation

In order to help improve speed and resolve memory related errors, it is advised to change the memory allocation. This can be done with the following code

```
## [1] 56000
```

Please note that "size=56000" sets the memory allocation to 7gb.

Importing Data Set

10% of the dataset has be partitioned, and used only for the final validation test.

Analysing the Dataset

Before we start to build any models, we should learn more about the data we are working with.

General Analysis

Structure of the Dataset

Table 1: edX Data Set

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

As we can see, while each row represents a single rating, it also includes the user ID, movie ID, a timestamp, the movie's title including year of release, and the genre(s).

Number of Movies and Users

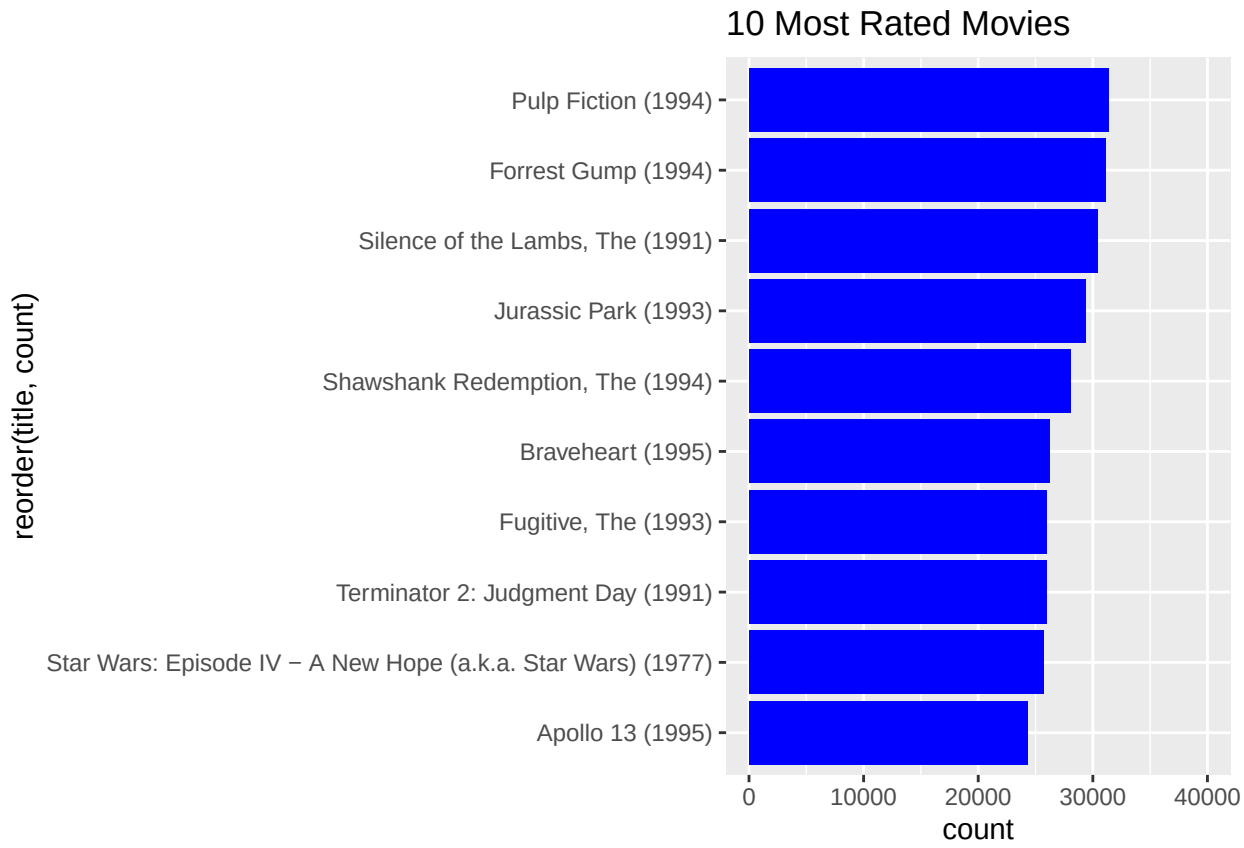
Here we can see how many users and movies make up the set

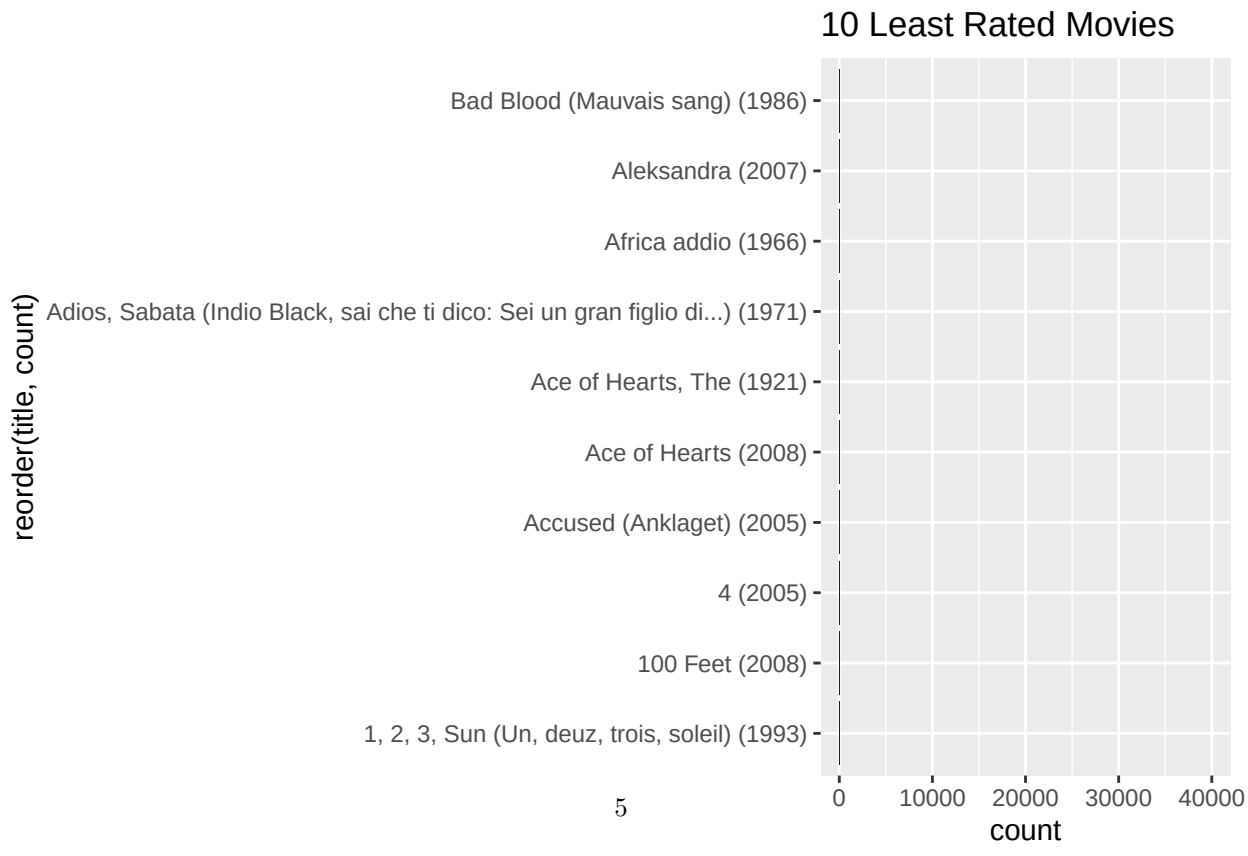
Table 2: Number of User and Movies

n_users	n_movies
69878	10677

Analysing Rating Patterns

The Least and Most Rated Movies





As we can clearly see, some movie are rated more often then others.

Average Number of Ratings for Top 10 Most and Least Rated Movies

Table 3: Average Number of Ratings for the 10 Most Rated Movies

Average
27834.8

Table 4: Average Number of Ratings for the 10 Least Rated Movies

Average
1

In fact, as we can see the difference between the averages of two groups is very stark.

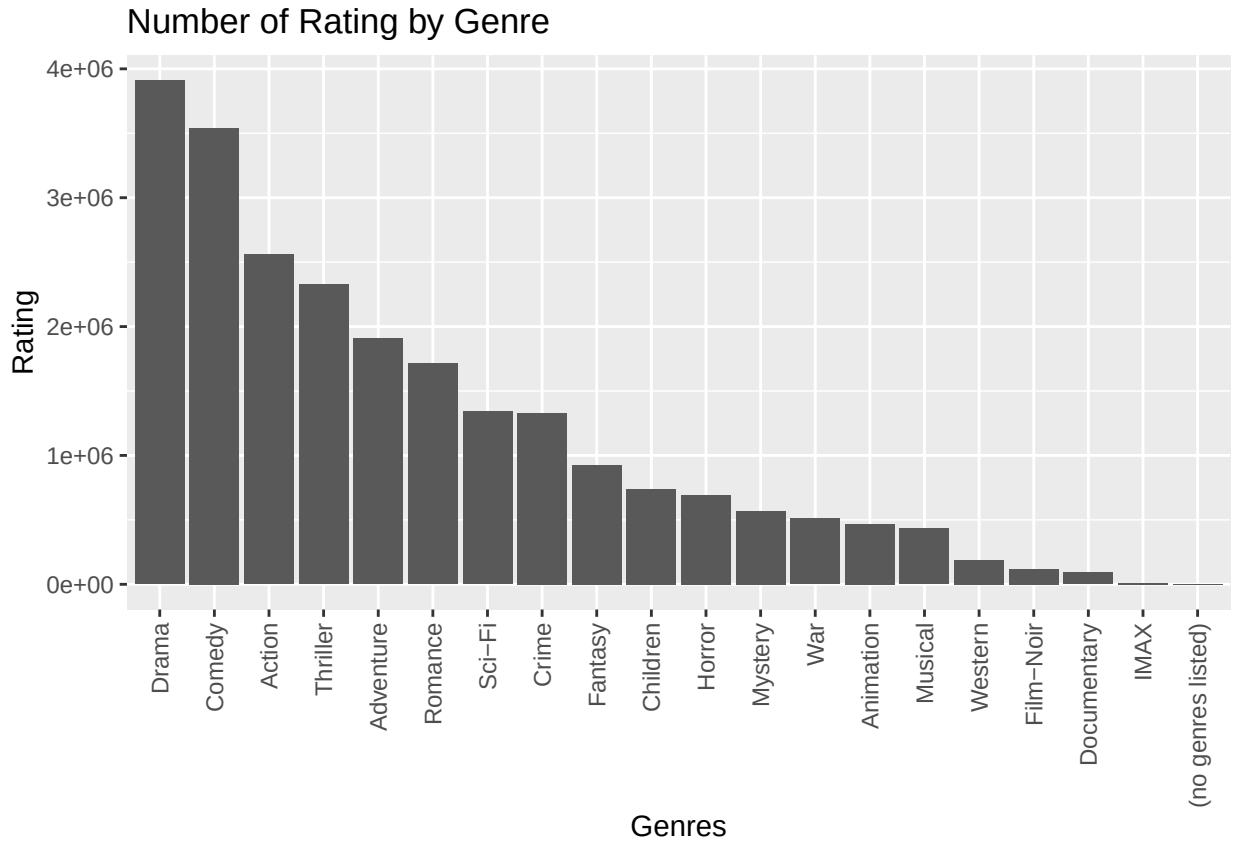
Number of Ratings for 10 Highest Rated Movies

Table 5: Top 10 Highest Rated Movies

title	mean	count
Blue Light, The (Das Blaue Licht) (1932)	5.00	1
Fighting Elegy (Kenka erejii) (1966)	5.00	1
Hellhounds on My Trail (1999)	5.00	1
Satan's Tango (SÃ¡tÃ¡ntangÃ³) (1994)	5.00	2
Shadows of Forgotten Ancestors (1964)	5.00	1
Sun Alley (Sonnenallee) (1999)	5.00	1
Constantine's Sword (2007)	4.75	2
Human Condition II, The (Ningen no joken II) (1959)	4.75	4
Human Condition III, The (Ningen no joken III) (1961)	4.75	4
Who's Singin' Over There? (a.k.a. Who Sings Over There) (Ko to tamo peva) (1980)	4.75	4

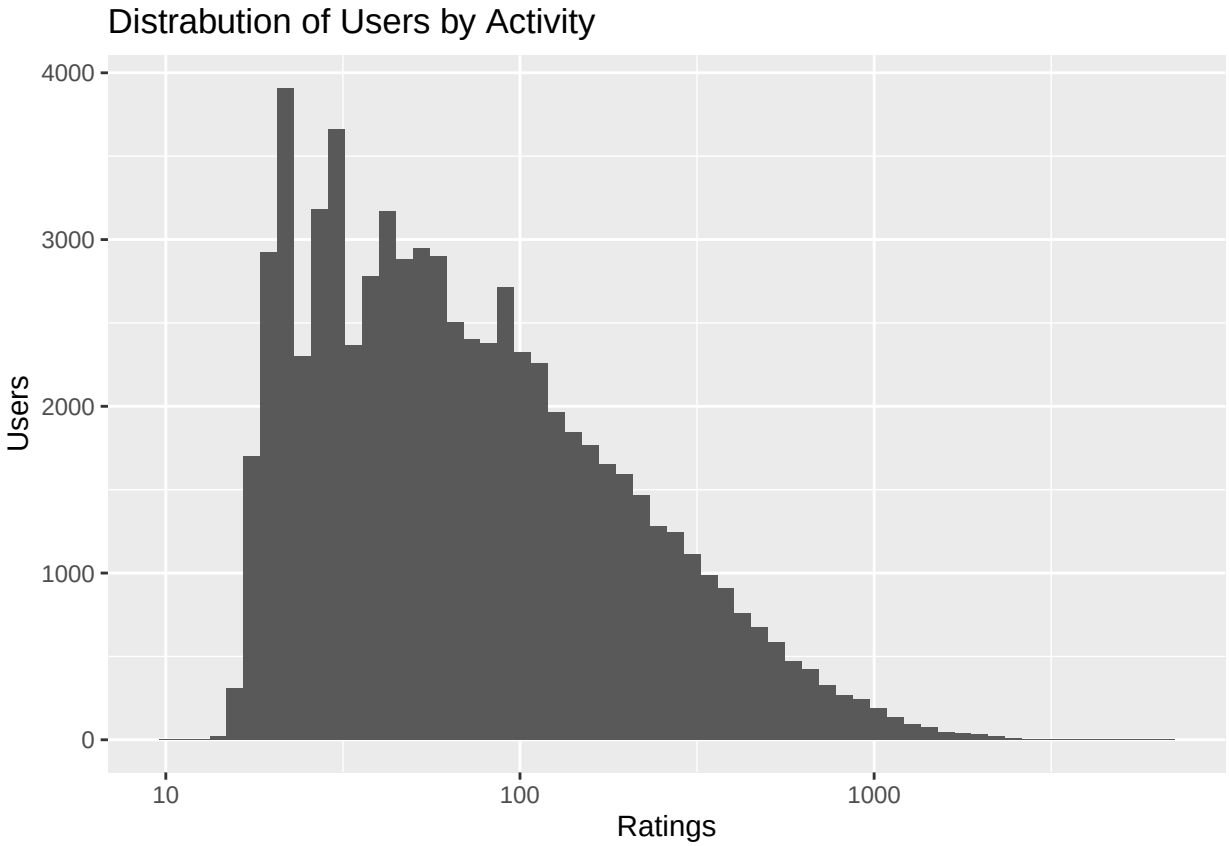
The low number of ratings are having a considerable impact on the mean scores, on what appears to be rather obscure movies.

Ratings by Genre



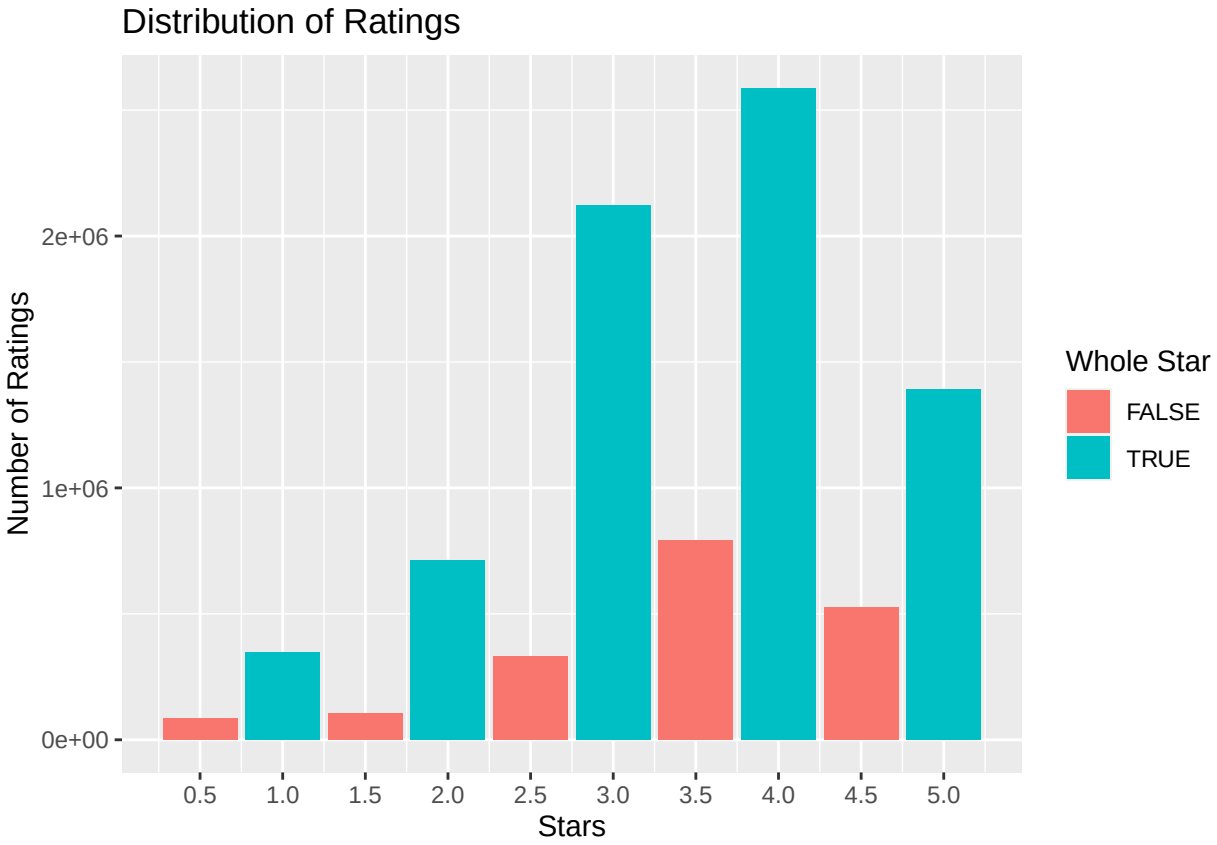
Some genres are also rated more often than others.

Users by Number of Ratings



We can see that some users are more active than others.

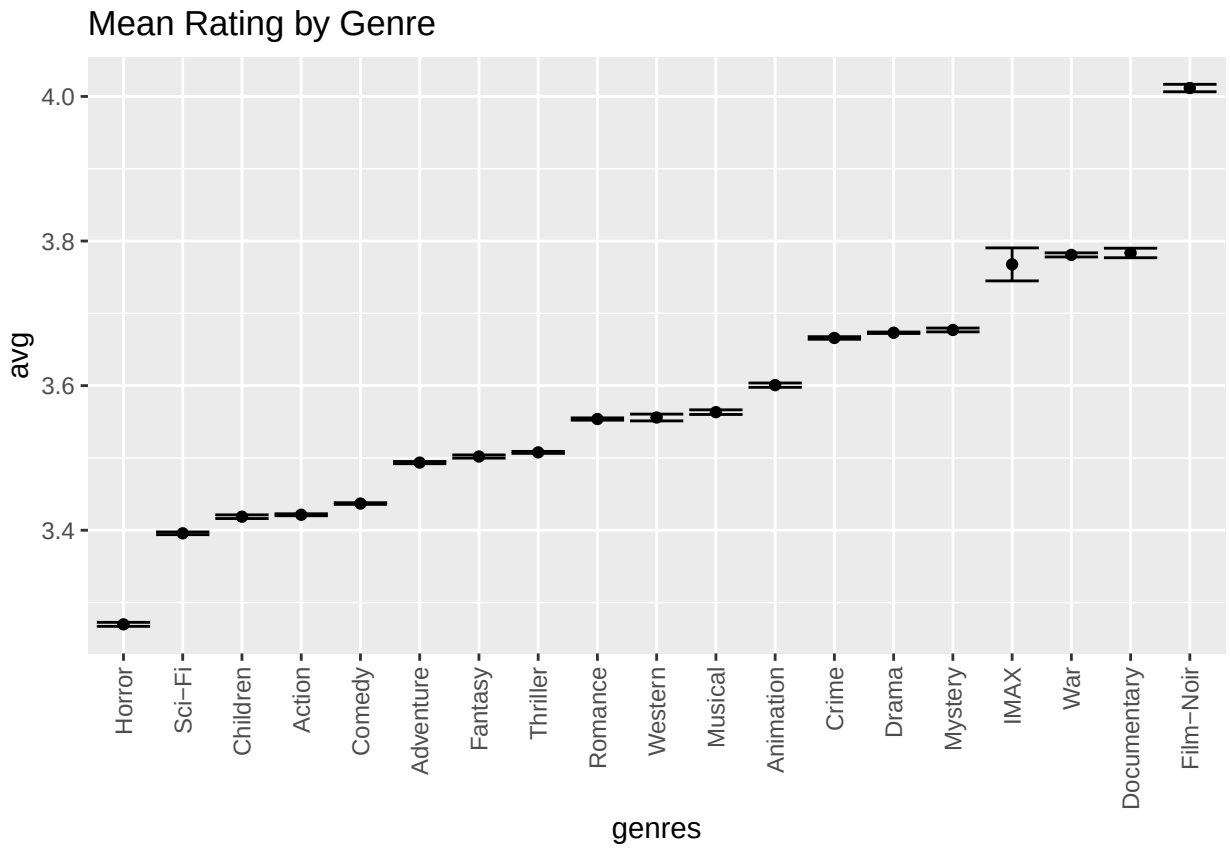
Frequency of Ratings



Users are also more likely to give full star ratings, with 4, 3, and 5 being the most popular.

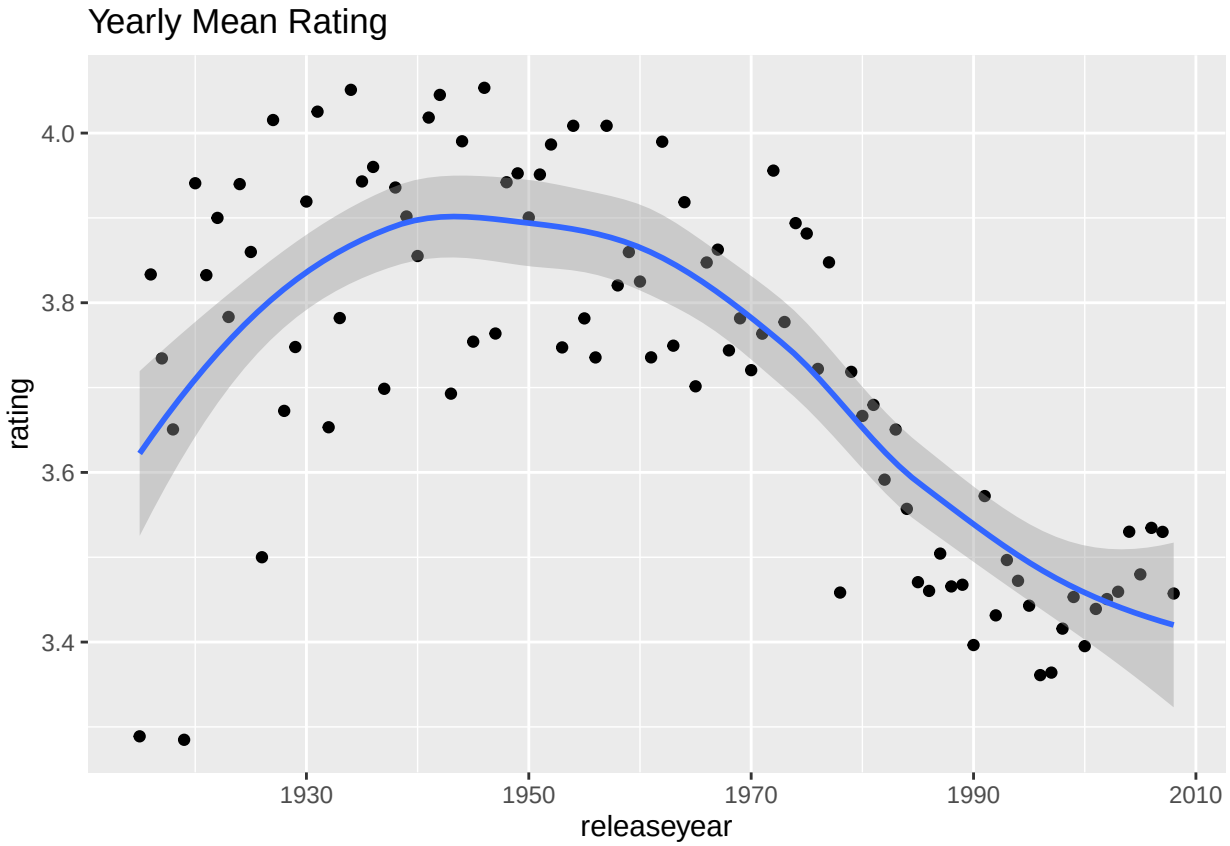
Analysing the Mean Ratings

Mean Rating by Genre



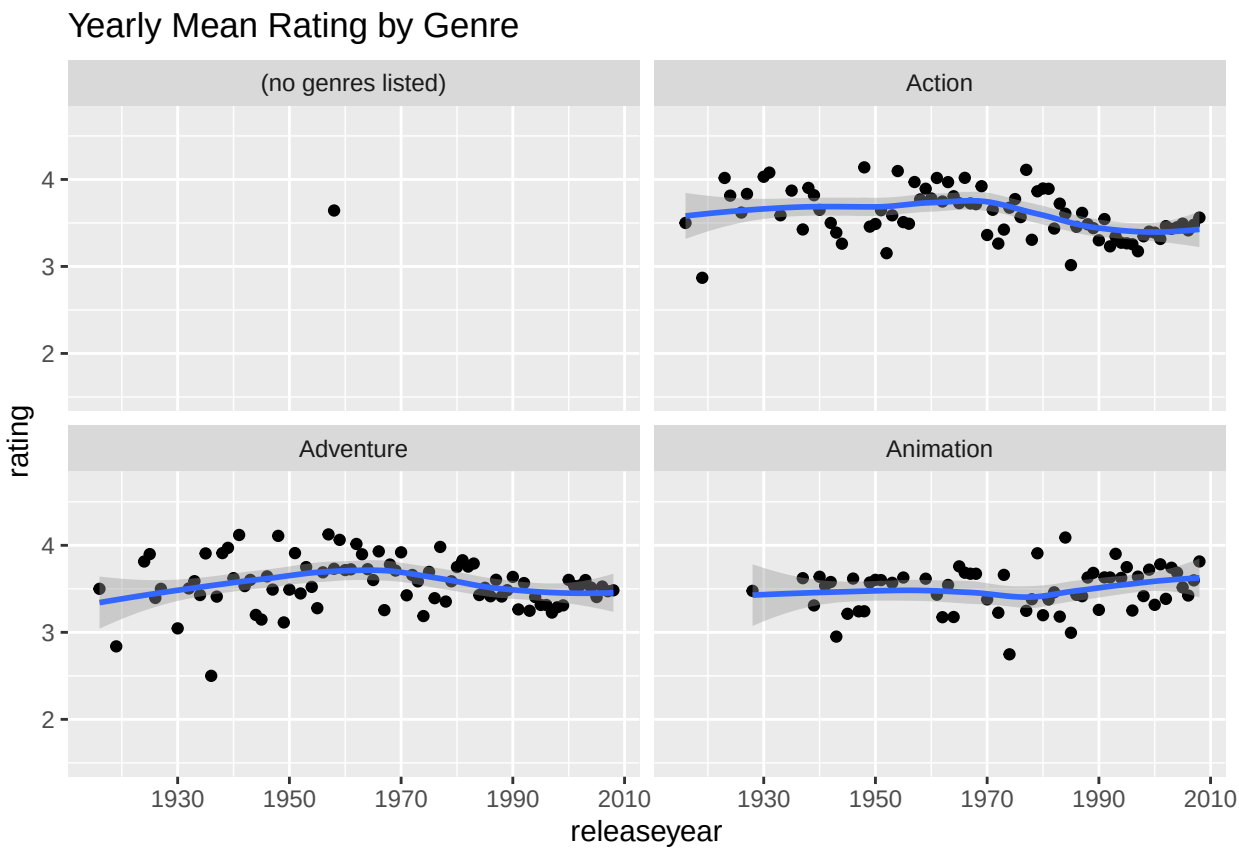
Some genres are rated higher than others, with Film-Noir being rated the highest, followed by Documentary and War movies being second and third, respectively.

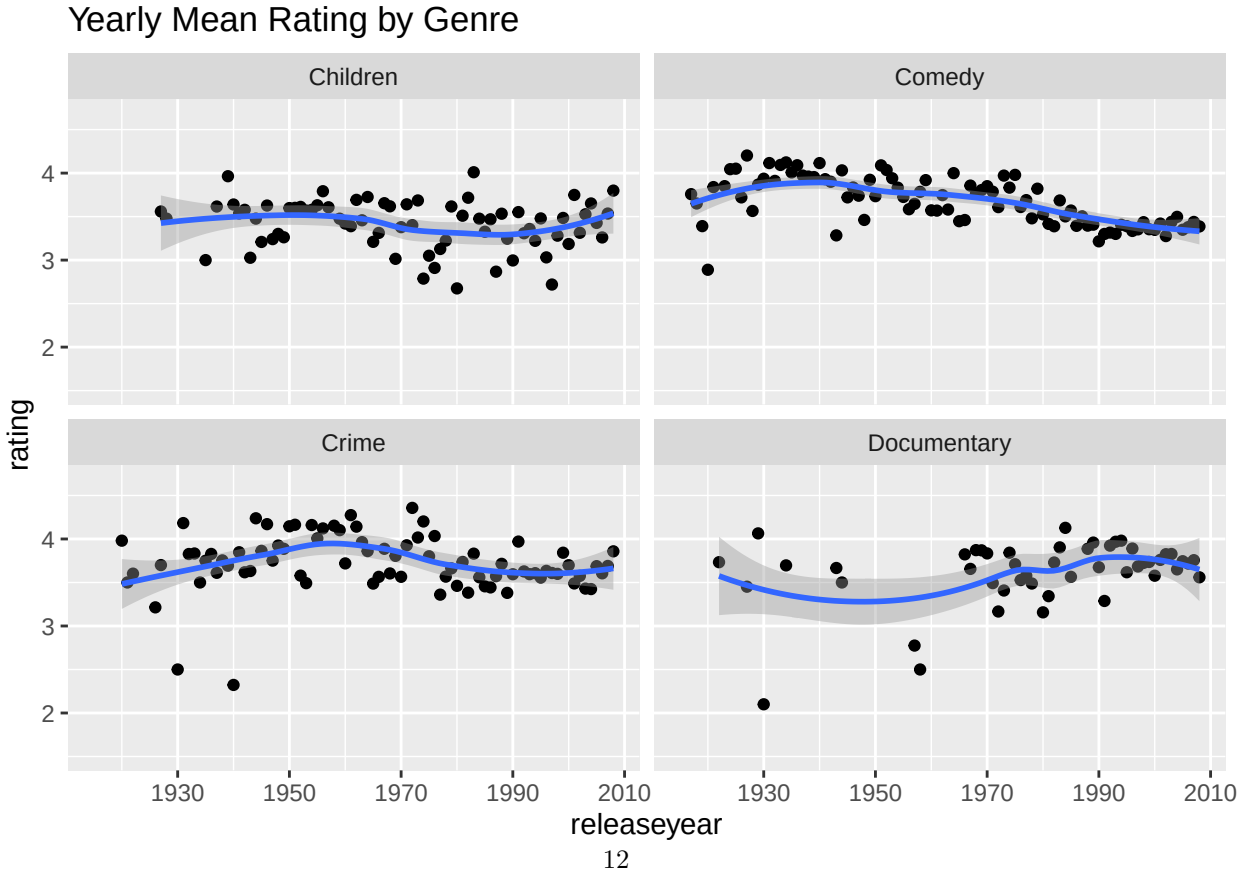
Mean Rating by Release Year

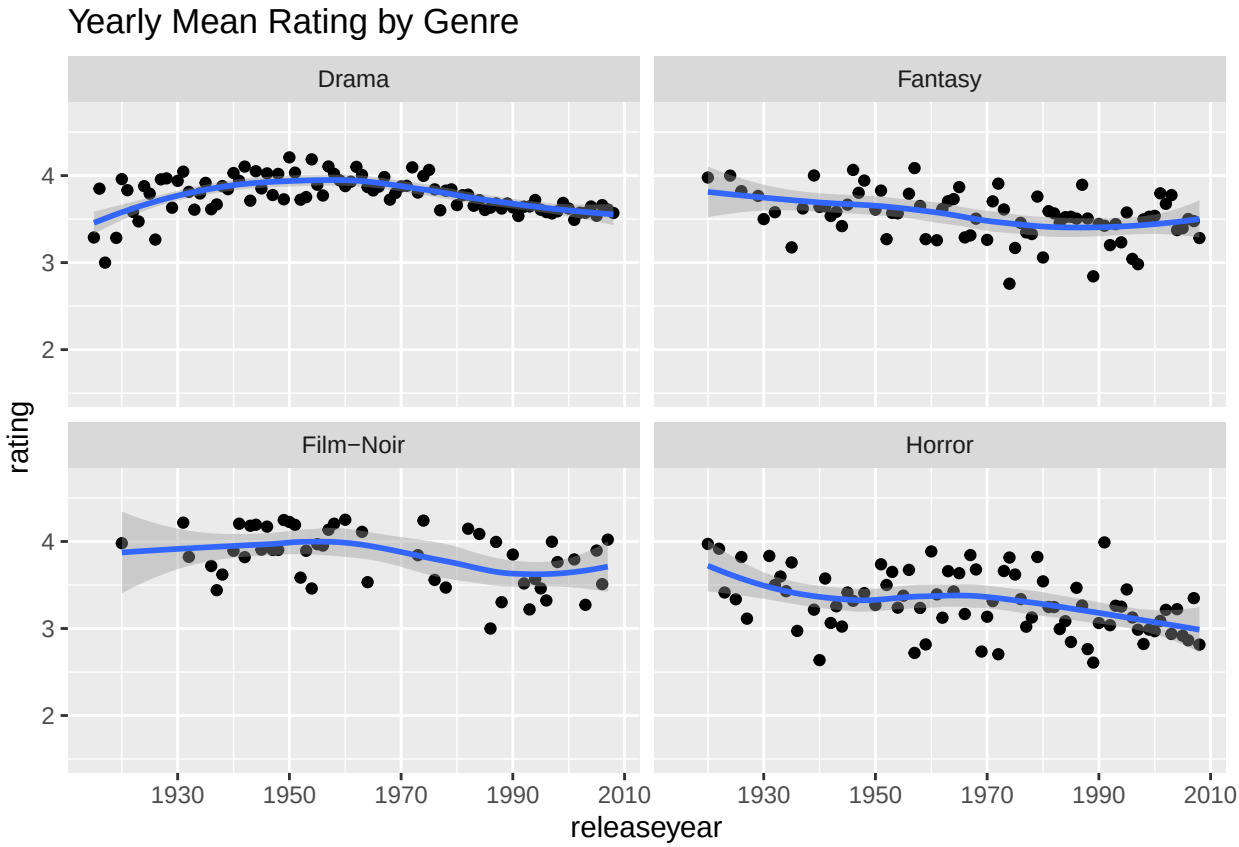


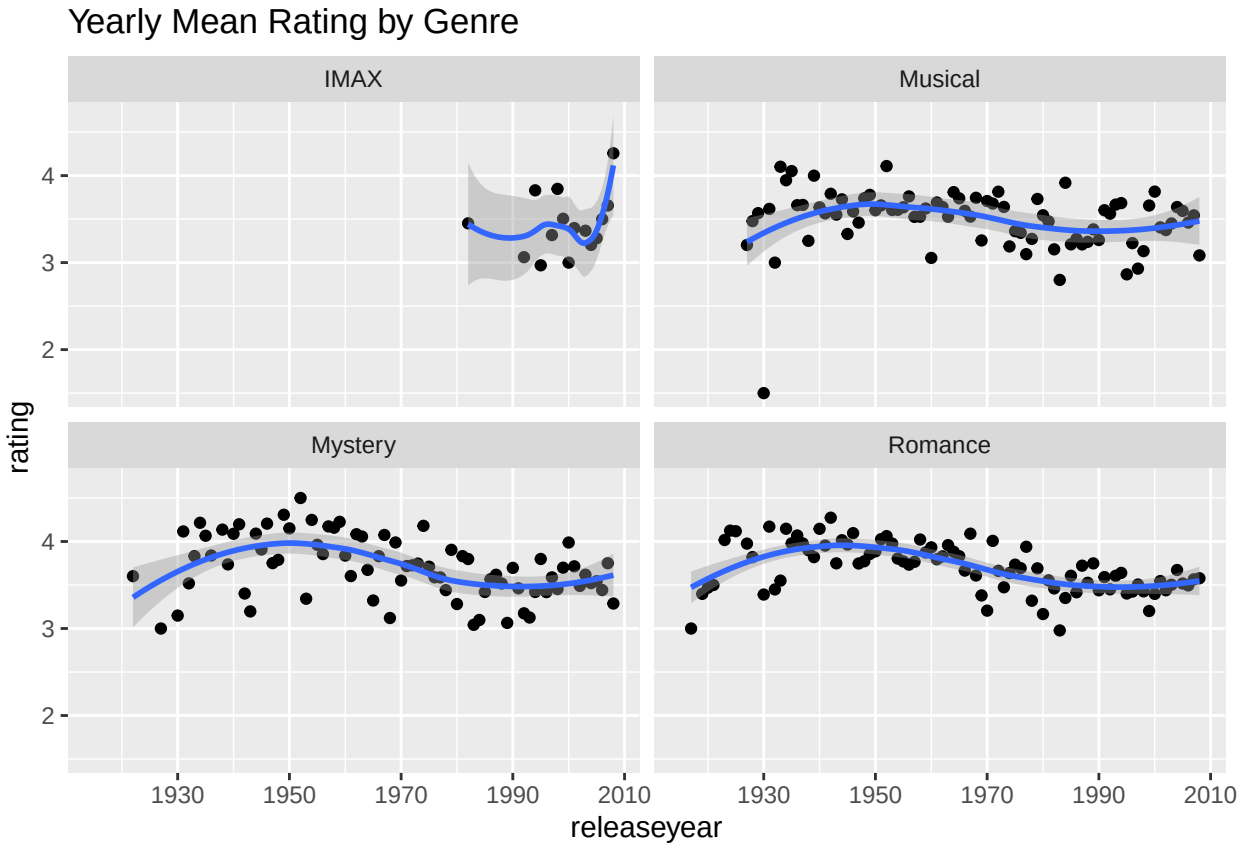
There appears to be a pattern with the mean increasing until the 40's, starts to fall in the 50's with the rate of fall increasing around the 60's.

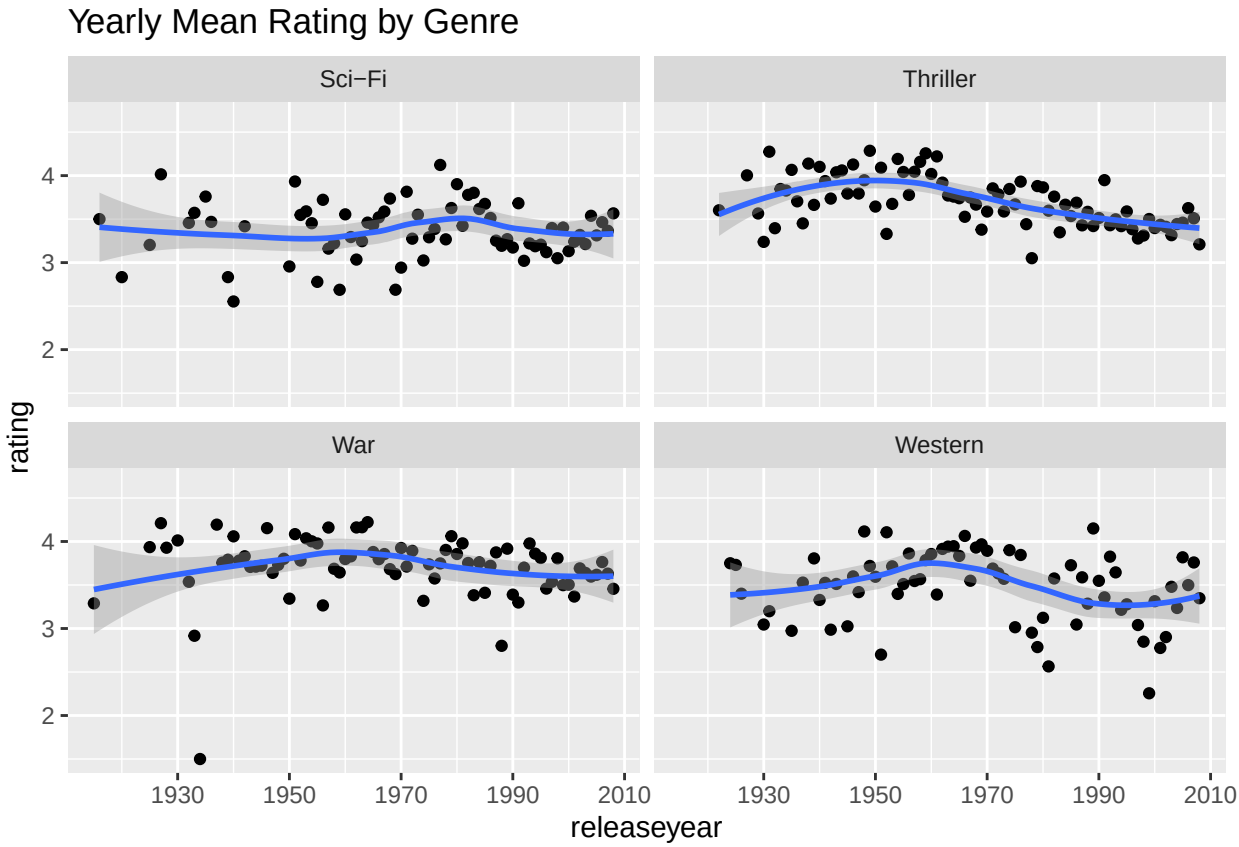
Release Year's Mean Rating by Genre





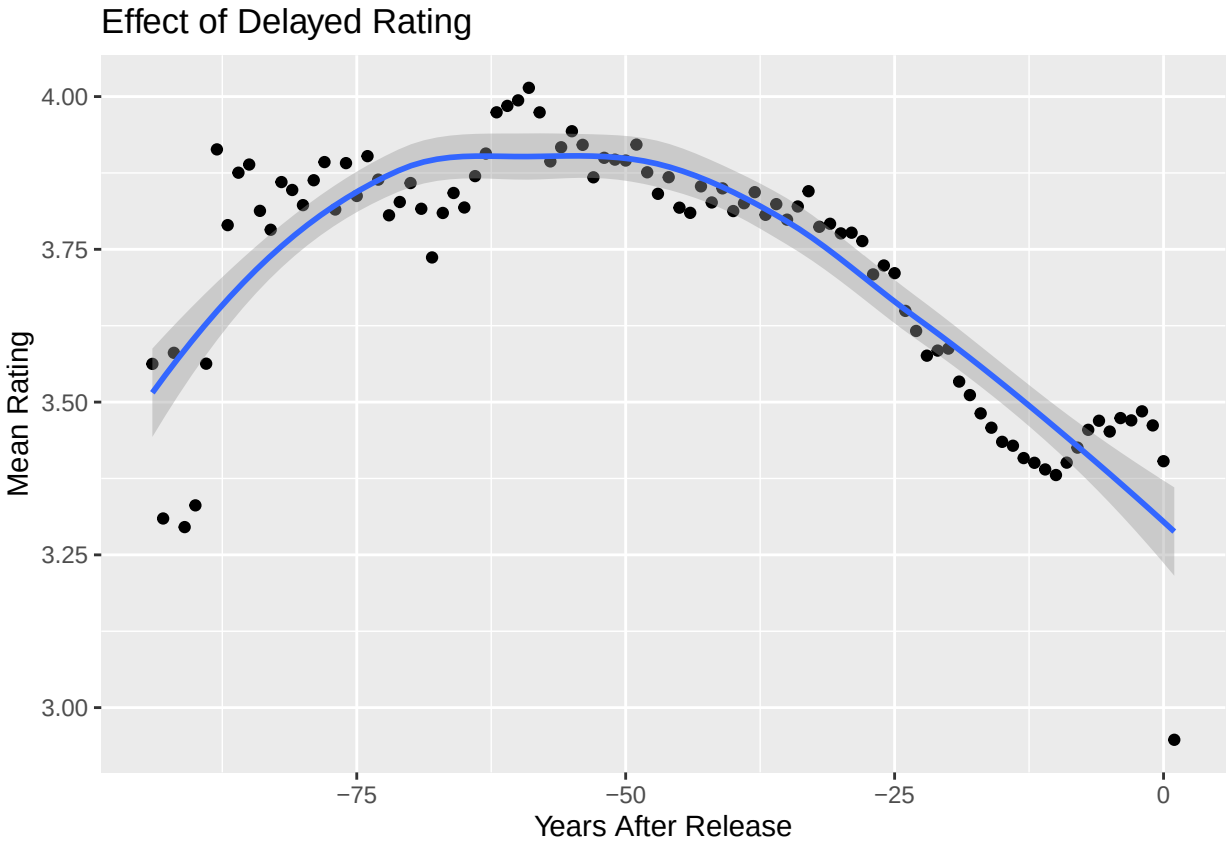

Yearly Mean Rating by Genre
Drama
Fantasy
Film−Noir
Horror
rating
releaseyear
4
3
2
1930
1950
1970
1990
2010


Yearly Mean Rating by Genre
IMAX
Musical
Mystery
Romance
rating
releaseyear
4
3
2
1930
1950
1970
1990
2010



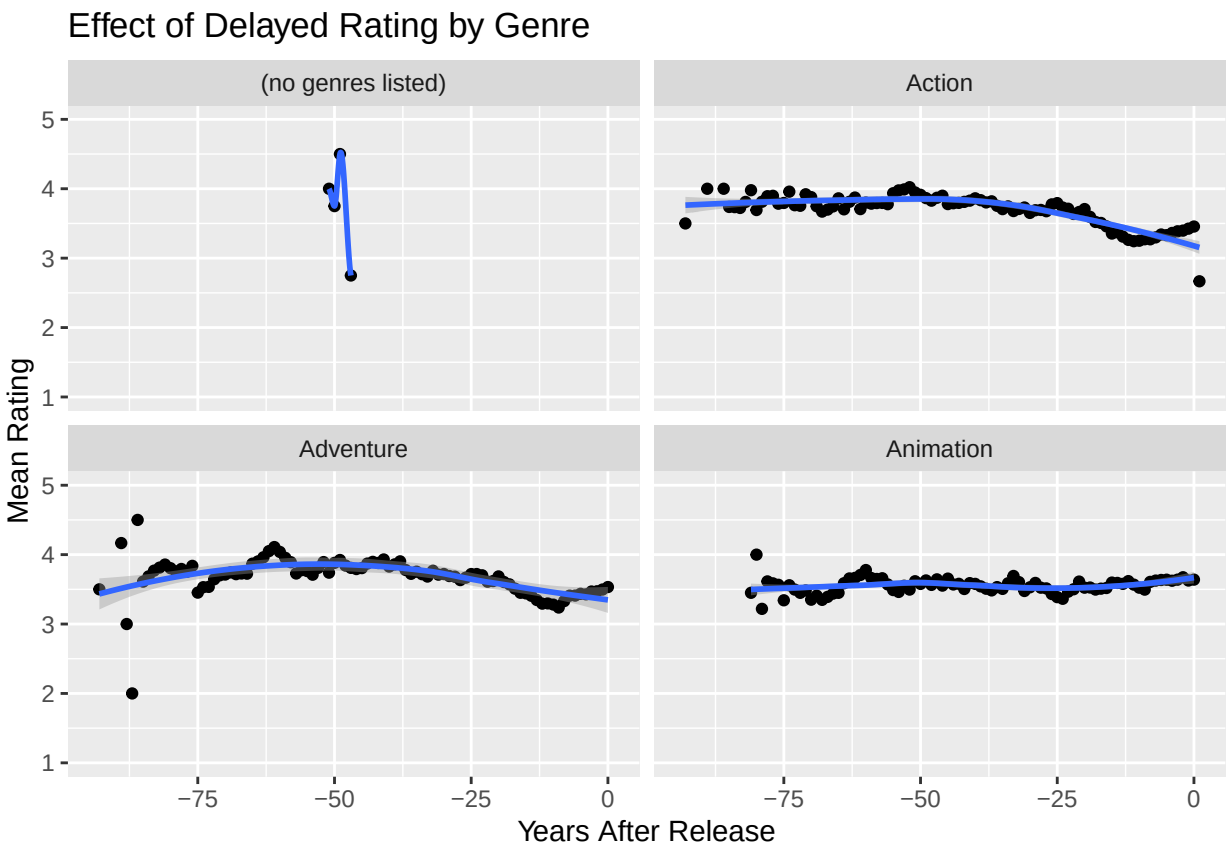
When seperated by genre, it appears that many genres follow the general pattern, but others, such as Documentary and Horror, displaying a different pattern.

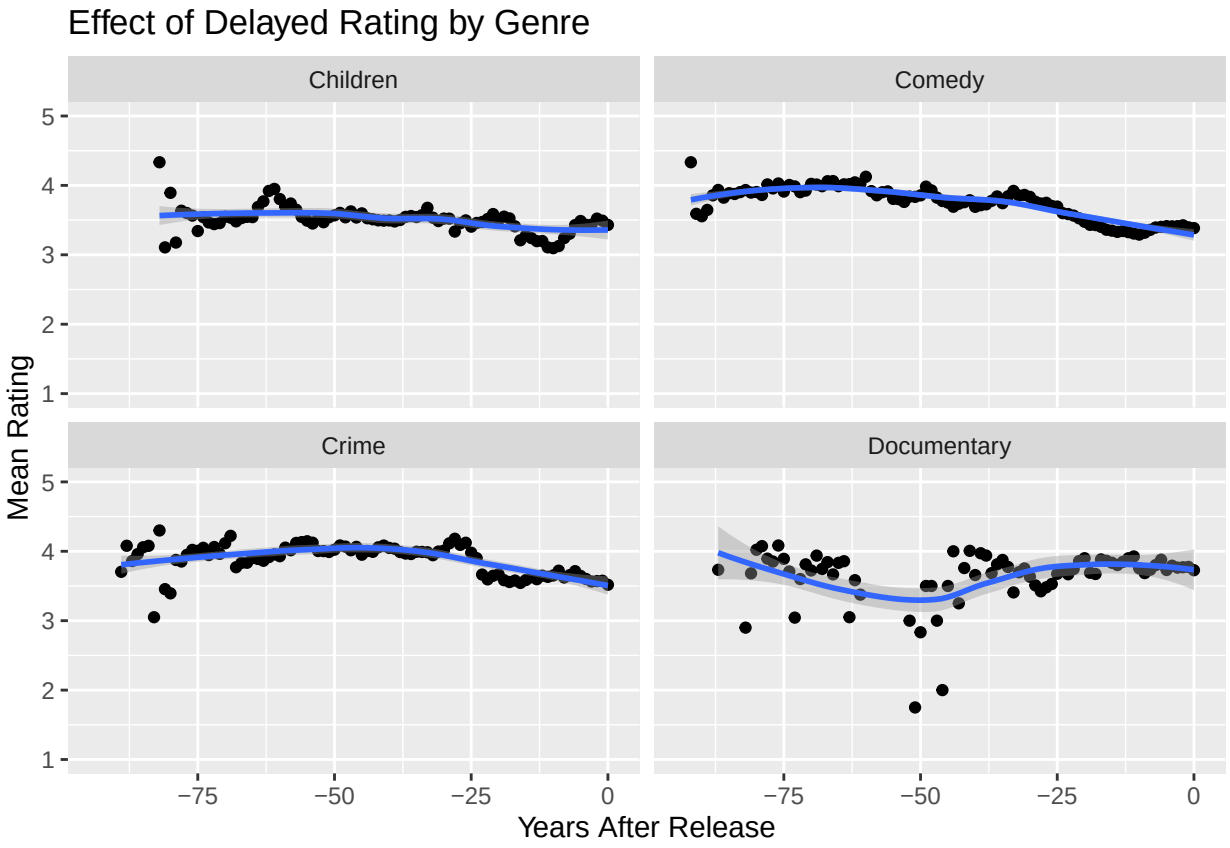
Effect of Delayed Rating

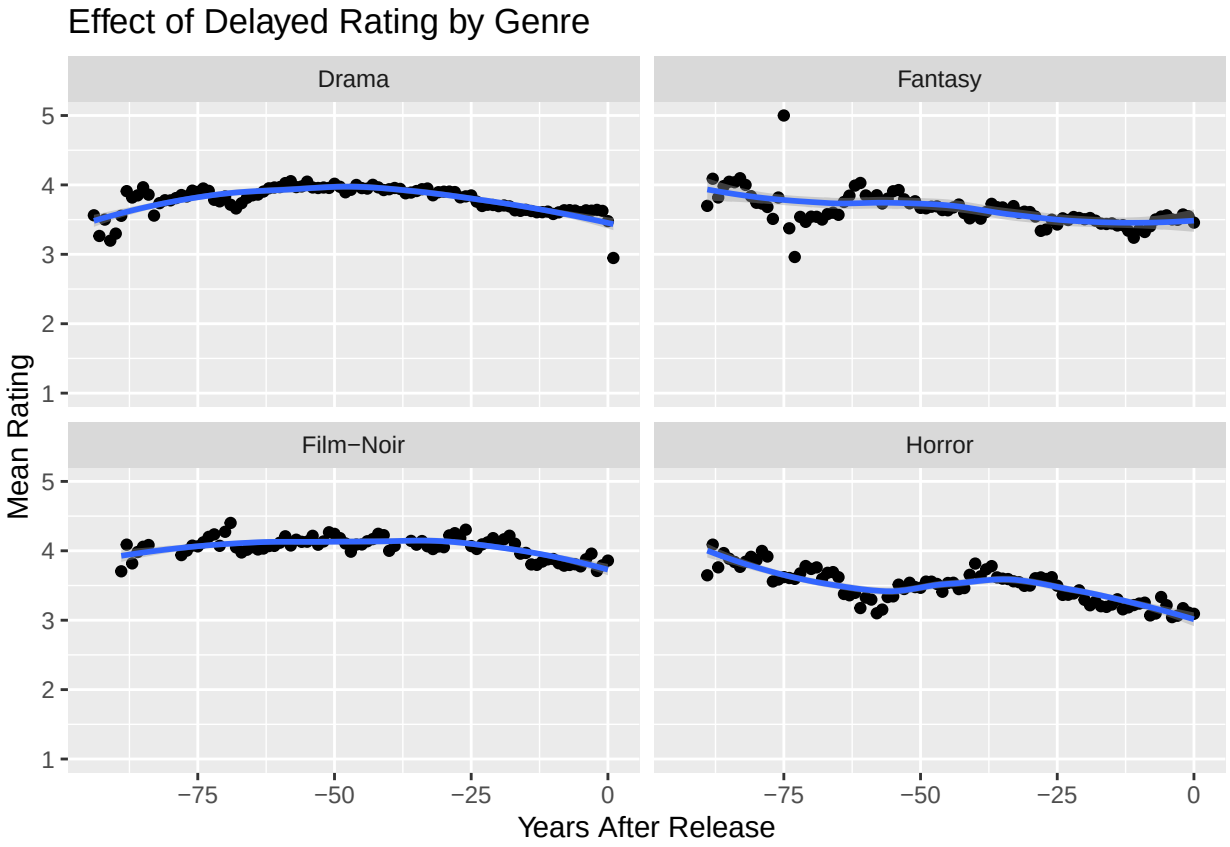


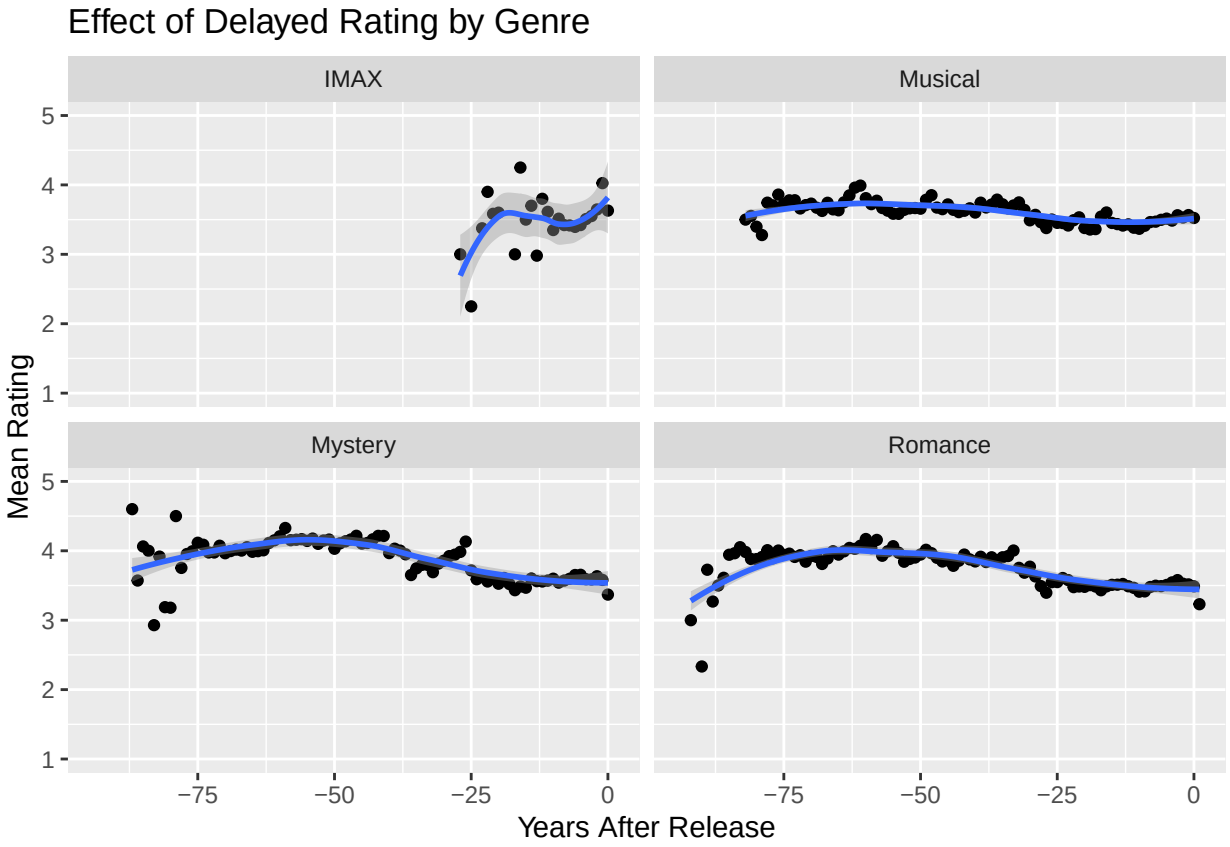
The delay between the movies release and its rating also appears to have an effect.

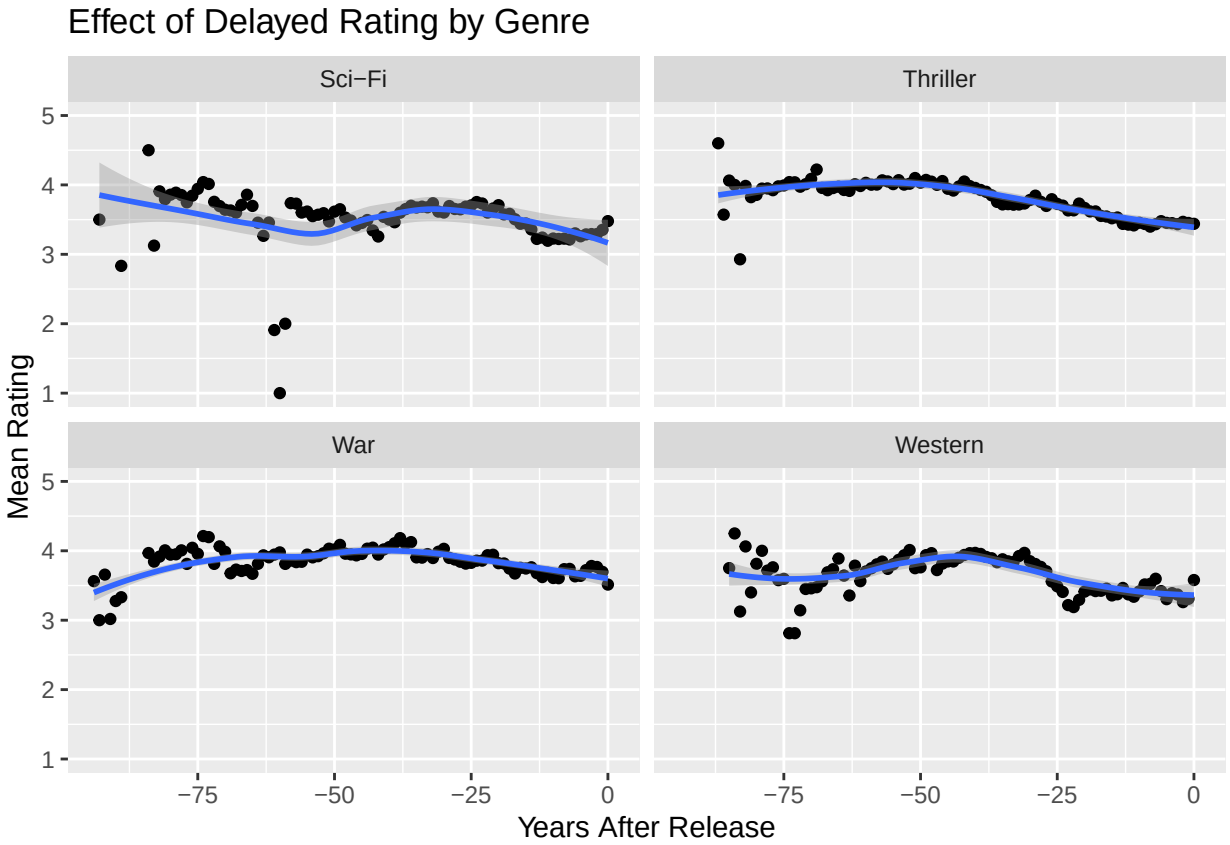
Delayed Rating's Effect by Genre






Effect of Delayed Rating by Genre
Drama
Fantasy
Film−Noir
Horror
Mean Rating
5
4
3
2
1
−75
−50
−25
0
Years After Release


Effect of Delayed Rating by Genre
IMAX
Musical
Mystery
Romance
Mean Rating
Years After Release
5
4
3
2
1
−75
−50
−25
0



Similarly to above, we can see some genres follow the general trend and some do not.

Conclusions

From our analysis, we can conclude that:

1. While there are some users at the extremes, in regard to activity, the majority have rated between 50 and 100 movies, approximately.
2. The number of ratings per movie varies greatly. The most frequently rated have tens of thousands, while the least have a single rating. Movies with an extremely low number of ratings may have an artificially high mean.
3. Some genres are rated higher than others. Film-Noir and Documentary movies have the highest and second highest mean rating, respectively, while also being the third and second least rated genre, excluding no genre listed. This could be linked to the above point, and may be caused by an extremely small number of high rating from users who are fans of the particular genre.
4. The year the movie was released, as well as the length of time from release to being reviewed appears to influence ratings. This influence appears to effect different genres to different degrees.

Recommendation Systems

Defining RMSE

Before we start to create and test our models, we first need to define how we will judge them. For this, as in the competition, we will use the Root Mean Squared Error.

Creating Train and Test Sets

In order to gauge the accuracy of our recommendation systems, we will partition 10% of the EdX set to use as a test set. This will allow us to keep the Validation set we already created for our final test, and help insure the reliability of our systems.

Mean and Biases Based Models

Simple Mean Rating

This is the simplest recommendation system. It simply uses the mean rating, and is based on the assumption that all remaining differences can be explained by random variation.

Table 6: RMSE Results

method	RMSE
Just the average	1.060054

This RMSE equates to our typical error being just over 1 star. This is not particularly good, and falls short of our target RMSE.

Adding Movie Bias

As common experience tells us, some movies are generally rated higher than other. In this model, we will incorporate that, and base the model on the assumption that the differences in ratings can be explained by the mean plus the individual movie bias, which we will call bi, and assume that all other variance is random.

Table 7: RMSE Results

method	RMSE
Just the average	1.0600537
With Movie Bias	0.9429615

This approach has reduced our RMSE by ~ .1 of a star. Although we are moving in the right direction, more explanation of the remaining variance is needed.

Adding User Bias

We also know from experience that some people generally rate movies higher than other. We can incorporate that by adding the variable bi.

Table 8: RMSE Results

method	RMSE
Just the average	1.0600537
With Movie Bias	0.9429615
With Movie Bias and User Bias	0.8646844

We have gotten a reduction of ~.08 on our previous RMSE score. However, it is still above our target RMSE.

Adding Release Year Bias

As we saw earlier, the year the movie was released appears to have an effect on the rating. This variable can be added to our model, which we will refer to as br.

Table 9: RMSE Results

method	RMSE
Just the average	1.0600537
With Movie Bias	0.9429615
With Movie Bias and User Bias	0.8646844
With Movie Bias and User Bias and Release Year Bias	0.8643302

Although we still see a reduction in our RMSE, it only represents a tiny fraction of a star.

Adding Genre Bias

Some genres are rated higher than other, we have seen in our analysis that Film-Noir, Documentary, and War movies are the highest rated. This can be added to our model, using a variable we will call bg.

Table 10: RMSE Results

method	RMSE
Just the average	1.0600537
With Movie Bias	0.9429615
With Movie Bias and User Bias	0.8646844
With Movie Bias and User Bias and Release Year Bias	0.8643302
With Movie Bias, User Bias, Release Year Bias, and Genre Bias	0.8623595

This results in another small decrease in our RMSE.

Replacing Release Year and Genre Bias for Release Year by Genre Bias

We have already seen how the Release Year Bias has a different effect depending on the genre. Lets try to model that effect. We will call this variable brg.

Table 11: RMSE Results

method	RMSE
Just the average	1.0600537
With Movie Bias	0.9429615
With Movie Bias and User Bias	0.8646844
With Movie Bias and User Bias and Release Year Bias	0.8643302
With Movie Bias, User Bias, Release Year Bias, and Genre Bias	0.8623595
With Movie Bias, User Bias, and Release Year by Genre Bias	0.8622556

This approach results in almost no difference.

Table 12: RMSE Results

method	RMSE
Just the average	1.0600537
With Movie Bias	0.9429615
With Movie Bias and User Bias	0.8646844
With Movie Bias and User Bias and Release Year Bias	0.8643302
With Movie Bias, User Bias, Release Year Bias, and Genre Bias	0.8623595
With Movie Bias, User Bias, and Release Year by Genre Bias	0.8622556
With Movie Bias, User Bias, and Release/Review Bias	0.8622619

Replacing Release Year by Genre Bias for Delayed Rating Bias

We already know that the amount of time between a movies release and its rating has an effect on its rating. We can name this brr.

We reduction is extremely small.

Adding Genre Bias

Lets add the genre bias back into the model. We should recalculate this first. We will refer to this as brrg.

Table 13: RMSE Results

method	RMSE
Just the average	1.0600537
With Movie Bias	0.9429615
With Movie Bias and User Bias	0.8646844
With Movie Bias and User Bias and Release Year Bias	0.8643302
With Movie Bias, User Bias, Release Year Bias, and Genre Bias	0.8623595
With Movie Bias, User Bias, and Release Year by Genre Bias	0.8622556
With Movie Bias, User Bias, and Release/Review Bias	0.8622619
With Movie Bias, User Bias, Release/Review Bias, and genre	0.8621862

The reduction is negligible.

Replacing Delayed Review Bias for Delayed Review by Genre Bias.

We've shown that the effect of a Delayed Review can be very different depending on the genre. For this we will use brrg2.

This approach results in NAs being created. The following code can be used to identify the cause.

We can see that the issue is not having any training data for a documentary with a delayed rating of -48, -49, and -52 years. There are possible solutions for this issue, However that is beyond the scope of this project.

We have seen a diminishing return for our efforts with this strategy. Instead of continuing to try and explain the remaining variance by adding more variables, let's try to a new approach.

Regularization

Earlier in analysis, we noted that there is a substantial variance in how often users rated individual movies. We also say that can skew the mean ratings, with some movies having a perfect 5 star rating from a single

Table 14: RMSE Results

method	RMSE
Just the average	1.0600537
With Movie Bias	0.9429615
With Movie Bias and User Bias	0.8646844
With Movie Bias and User Bias and Release Year Bias	0.8643302
With Movie Bias, User Bias, Release Year Bias, and Genre Bias	0.8623595
With Movie Bias, User Bias, and Release Year by Genre Bias	0.8622556
With Movie Bias, User Bias, and Release/Review Bias	0.8622619
With Movie Bias, User Bias, Release/Review Bias, and genre	0.8621862
With Movie Bias, User Bias, and Release/Review Bias by Genre	NA

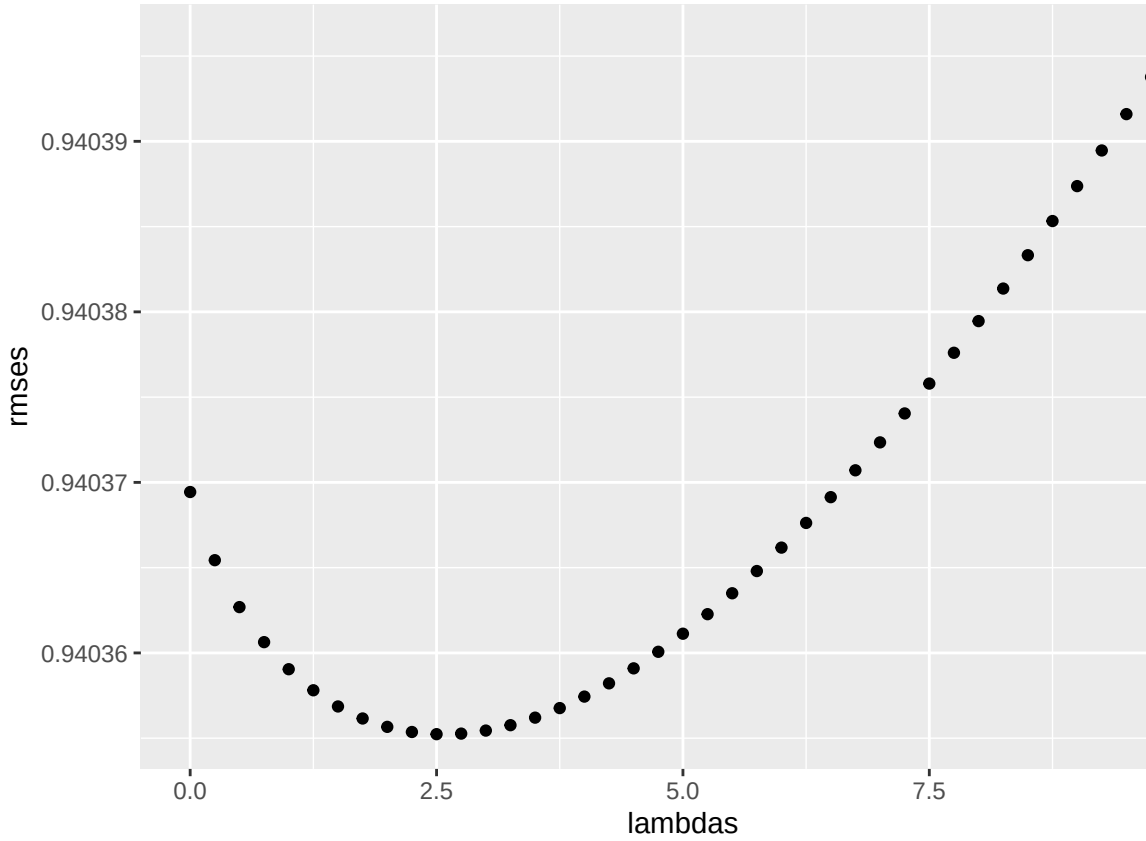
Table 15: Entries with NA's

userId	movieId	rating	timestamp	title	genres	releaseyear	date	release_r
6477	3136	3.5	1136352253	James Dean Story, The (1957)	Documentary	1957	2006	
16929	3136	3.0	1221163759	James Dean Story, The (1957)	Documentary	1957	2009	
30013	3136	3.5	1099285247	James Dean Story, The (1957)	Documentary	1957	2005	

rating. In order to address this, we will use a Penalised Least Squares approach.

Penalised Least Squares

The first step is to select a value for lambda, in this case the one which produces the lowest RMSE. To help us select the best choice, we can evaluate a sequence of them.



Regularized Movies

Table 16: Lambda with Lowest RMSE

x
2.5

We can now use this value for our model.

Table 17: RMSE Results

method	RMSE
Just the average	1.0600537
With Movie Bias	0.9429615
With Movie Bias and User Bias	0.8646844
With Movie Bias and User Bias and Release Year Bias	0.8643302
With Movie Bias, User Bias, Release Year Bias, and Genre Bias	0.8623595
With Movie Bias, User Bias, and Release Year by Genre Bias	0.8622556
With Movie Bias, User Bias, and Release/Review Bias	0.8622619
With Movie Bias, User Bias, Release/Review Bias, and genre	0.8621862
With Movie Bias, User Bias, and Release/Review Bias by Genre	NA
With Regularised Movie	0.9403552

Regularizing movies has resulted a modest reduction when compared to the non-regularized movie bias alone. However, we still have not reached our target RMSE.

Lets try to add some biases.

Regularized Movie and User Bias We can start with adding User Bias, in much the same way we did before.

Table 18: RMSE Results

method	RMSE
Just the average	1.0600537
With Movie Bias	0.9429615
With Movie Bias and User Bias	0.8646844
With Movie Bias and User Bias and Release Year Bias	0.8643302
With Movie Bias, User Bias, Release Year Bias, and Genre Bias	0.8623595
With Movie Bias, User Bias, and Release Year by Genre Bias	0.8622556
With Movie Bias, User Bias, and Release/Review Bias	0.8622619
With Movie Bias, User Bias, Release/Review Bias, and genre	0.8621862
With Movie Bias, User Bias, and Release/Review Bias by Genre	NA
With Regularised Movie	0.9403552
With Regularised Movie and User Bias	0.8629317

Adding the User bias has brought the RMSE down to 0.863

Let's add another bias.

Regularized Movie and User Bias, with Release Year by Genre Bias Let's try to add the Release Year by Genre Bias.

Table 19: RMSE Results

method	RMSE
Just the average	1.0600537
With Movie Bias	0.9429615
With Movie Bias and User Bias	0.8646844
With Movie Bias and User Bias and Release Year Bias	0.8643302
With Movie Bias, User Bias, Release Year Bias, and Genre Bias	0.8623595
With Movie Bias, User Bias, and Release Year by Genre Bias	0.8622556
With Movie Bias, User Bias, and Release/Review Bias	0.8622619
With Movie Bias, User Bias, Release/Review Bias, and genre	0.8621862
With Movie Bias, User Bias, and Release/Review Bias by Genre	NA
With Regularised Movie	0.9403552
With Regularised Movie and User Bias	0.8629317
With Regularised Movie. User Bias, and Release Year by Genre Bias	0.8623673

We have now gotten the RMSE down to 0.862.

Although we have gotten the RMSE down, we still haven't been able to reach our target.

Matrix Factorization

What is recosystem

The recosystem package is a r wrapper for the open source, LIBMF library for recommendation systems. Developed by Yu-Chin Juan, Wei-Sheng Chin, et al. (3), this package holds many advantages, not only over our previous approaches, but also when compared other systems that utilize a matrix factorization method.

Recosystem, as well as the LIBMF library it is based on, allows users to take advantage of multi-core processing, improving performance times, as well better memory management stratigies.

Setings

The recosystem allows for a wide range of customization through it's peramiters and options. This includes the nthread parameter, which defines the number of threads for parallel computing. For this project, it has not been changed from the default value of 1. This is due to when nthread is set to a value greater than 1, the training result can not be guaranteed to be reproducable, even if random seed is set.(4)

Recosystem Model

The recosystem requires us to first specify our train train and test set, before we can use it's inbuilt tuning feature to optimize it's options.

```
## iter      tr_rmse          obj
##    0       0.8927   2.3616e+07
##    1       0.8057   2.0209e+07
##    2       0.7633   1.8697e+07
##    3       0.7390   1.7949e+07
##    4       0.7231   1.7479e+07
##    5       0.7118   1.7179e+07
##    6       0.7036   1.6966e+07
##    7       0.6973   1.6800e+07
##    8       0.6924   1.6683e+07
##    9       0.6883   1.6598e+07
##   10       0.6851   1.6515e+07
##   11       0.6823   1.6457e+07
##   12       0.6799   1.6411e+07
##   13       0.6779   1.6368e+07
##   14       0.6760   1.6331e+07
##   15       0.6744   1.6302e+07
##   16       0.6729   1.6265e+07
##   17       0.6717   1.6247e+07
##   18       0.6705   1.6232e+07
##   19       0.6694   1.6212e+07
```

Final Validation

By using the Recosystem, we have gotten the lowest RMSE, one well below our target. However, this has only been tested on data from our Edx set. We must now test it on our Validation set, in order to get a more reliable RMSE.

```
## iter      tr_rmse          obj
##    0       0.9568   1.0889e+07
##    1       0.8606   9.2349e+06
##    2       0.8150   8.5443e+06
##    3       0.7881   8.1842e+06
##    4       0.7698   7.9687e+06
##    5       0.7560   7.8076e+06
```

Table 20: RMSE Results

method	RMSE
Just the average	1.0600537
With Movie Bias	0.9429615
With Movie Bias and User Bias	0.8646844
With Movie Bias and User Bias and Release Year Bias	0.8643302
With Movie Bias, User Bias, Release Year Bias, and Genre Bias	0.8623595
With Movie Bias, User Bias, and Release Year by Genre Bias	0.8622556
With Movie Bias, User Bias, and Release/Review Bias	0.8622619
With Movie Bias, User Bias, Release/Review Bias, and genre	0.8621862
With Movie Bias, User Bias, and Release/Review Bias by Genre	NA
With Regularised Movie	0.9403552
With Regularised Movie and User Bias	0.8629317
With Regularised Movie. User Bias, and Release Year by Genre Bias	0.8623673
Recosystem	0.7896290

```
##    6       0.7454   7.6957e+06
##    7       0.7368   7.6063e+06
##    8       0.7298   7.5403e+06
##    9       0.7241   7.4804e+06
##   10       0.7193   7.4371e+06
##   11       0.7152   7.3990e+06
##   12       0.7117   7.3648e+06
##   13       0.7089   7.3405e+06
##   14       0.7062   7.3169e+06
##   15       0.7040   7.2993e+06
##   16       0.7019   7.2794e+06
##   17       0.7001   7.2644e+06
##   18       0.6985   7.2516e+06
##   19       0.6970   7.2381e+06
```

Table 21: RMSE Results

method	RMSE
Just the average	1.0600537
With Movie Bias	0.9429615
With Movie Bias and User Bias	0.8646844
With Movie Bias and User Bias and Release Year Bias	0.8643302
With Movie Bias, User Bias, Release Year Bias, and Genre Bias	0.8623595
With Movie Bias, User Bias, and Release Year by Genre Bias	0.8622556
With Movie Bias, User Bias, and Release/Review Bias	0.8622619
With Movie Bias, User Bias, Release/Review Bias, and genre	0.8621862
With Movie Bias, User Bias, and Release/Review Bias by Genre	NA
With Regularised Movie	0.9403552
With Regularised Movie and User Bias	0.8629317
With Regularised Movie. User Bias, and Release Year by Genre Bias	0.8623673
Recosystem	0.7896290
Final Validation: Recosystem	0.7845712

Our final RMSE score is 0.7845, far below our stated target of 0.8567.

Conclusions

This project aimed to create a movie recommendation system, from the MovieLens 10m data set, with a lower RMSE than 0.8567. From our initial model, using just the average rating, we added different biases; movie, user, genre, etc., trying to account for the remaining variables in the ratings. This approach reduced our RMSE score from 1.06 down to 0.8621. At that point we change approach, and implemented a Penalised Least Squares method to account for issues arising from extremely low number of ratings for some movies. Although this did result in a small RMSE reduction, compared to similar non-regularised methods, this approach started to see a diminished return at roughly the same point as our previous models, ~ 0.864. In our final model, we took advantage of the recosystem, a r wapper for the LIBMF matrix factorization library for recommender system. Using this matrix factorization approach gave us a RMSE of 0.7896, which reduced further in a final validation test to 0.7846. This is despite the recosystem's customization not being fully utilized. Through further refinement, the RMSE may be reduced further still, or more preferable balance between speed and accuracy may be found by utilizing its multicore processing capabilities.

References and Acknowledgements

References

1. Wikipedia, Netflix Prize: https://en.wikipedia.org/wiki/Netflix_Prize
2. Grouplens.org, MovieLens 10M Dataset : https://grouplens.org/datasets/movielens/10m/
3. Yixuan Qiu, recosystem: Recommender System Using Parallel Matrix Factorization, https://cran.r-project.org/web/packages/recosystem/vignettes/introduction.html
4. Yixuan's Blog - R, https://www.r-bloggers.com/2016/07/recosystem-recommender-system-using-parallel-matrix-factorization/

Acknowledgements

Numerous parts of the above code, including, but not limited to, the code for importing and creating the data sets, were either supplied as a part of this capstone project, or were directly taken from related HarvardX course materials: Rafael A. Irizarry, Introduction to Data Science:Data Analysis and Prediction Algorithms with R, https://rafalab.github.io/dsbook/

Appendix

Script

Introduction

```
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(tidyr)) install.packages("tidyr", repos = "http://cran.us.r-project.org" )
if(!require(ds4psy)) install.packages("ds4psy", repos = "http://cran.us.r-project.org" )
if(!require(dplyr)) install.packages("dplyr", repos = "http://cran.us.r-project.org" )
if(!require(recosystem)) install.packages("recosystem", repos = "http://cran.us.r-project.org" )
if(!require(lubridate)) install.packages("lubridate", repos = "http://cran.us.r-project.org" )
if(!require(ggforce))install.packages("ggforce", repos = "http://cran.us.r-project.org" )
```

```
if(!require(rmarkdown))install.packages("rmarkdown", repos = "http://cran.us.r-project.org"  )
if(!require(knitr))install.packages("knitr", 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(tidyr)
library(ds4psy)
library(dplyr)
library(lubridate)
library(recosystem)
library(ggforce)
library(rmarkdown)
library(knitr)
library(kableExtra)
```

```
memory.limit(size=56000)
```

```
############################################################
# Create edx set, validation set (final hold-out test set)
############################################################

# Note: this process could take a couple of minutes
# MovieLens 10M dataset:
# https://grouplens.org/datasets/movielens/10m/
# http://files.grouplens.org/datasets/movielens/ml-10m.zip

dl <- tempfile()
download.file("https://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")

# if using R 3.6 or earlier:
#movies <- as.data.frame(movies) %>% mutate(movieId = as.numeric(levels(movieId))[movieId],
#                                           title = as.character(title),
#                                           genres = as.character(genres))
# if using R 4.0 or later:
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")

# Validation set will be 10% of MovieLens data
set.seed(1, sample.kind="Rounding") # if using R 3.5 or earlier, use `set.seed(1)`
```

```
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)
```

Analysing the Dataset

```
edx %>%
  summarize(n_users = n_distinct(userId),
            n_movies = n_distinct(movieId)) %>%
  kbl(caption = "Number of User and Movies") %>%
  kable_styling(latex_options = c("striped", "hold_position"))
```

General Analysis

```
#most rated movies
 edx %>% group_by(title) %>%
  summarise(count = n()) %>%
 slice_max(count,n=10, with_ties = FALSE)%>%
   ggplot(aes(x= reorder(title, count),y = count)) +
   geom_bar(stat='identity', fill="blue") + coord_flip(y=c(0, 40000))+
   ggtitle("10 Most Rated Movies")
```

```
#least rated movies
 edx %>% group_by(title) %>%
   summarise(count = n()) %>%
   slice_min(count, n=10, with_ties = FALSE)%>%
   ggplot(aes(x= reorder(title, count),y = count)) +
   geom_bar(stat='identity', fill="blue") + coord_flip(y=c(0, 40000))+
   scale_y_continuous()+
   ggtitle("10 Least Rated Movies")
```

```
edx %>% group_by(movieId) %>%
   summarise(count = n()) %>%
```

```
   slice_max(count, n=10, with_ties = FALSE) %>%
  summarise("Average" = mean(count)) %>%
  kbl(caption = "Average Number of Ratings for the 10 Most Rated Movies") %>%
  kable_styling(latex_options = c("striped", "hold_position"))

edx %>% group_by(movieId) %>%
   summarise(count = n()) %>%
   slice_min(count, n=10, with_ties = FALSE) %>%
  summarise("Average" = mean(count)) %>%
  kbl(caption = "Average Number of Ratings for the 10 Least Rated Movies") %>%
  kable_styling(latex_options = c("striped", "hold_position"))
```

```
# Top 10 Highest Rated Movies
 edx %>% group_by(title) %>%
   summarise(mean = mean(rating),  count = n()) %>%
   slice_max(mean, n=10, with_ties = FALSE) %>%
  kbl(caption = "Top 10 Highest Rated Movies")%>%
  kable_styling(latex_options = c("striped", "hold_position"))
```

```
#Most Rated Genres
edx %>% separate_rows(genres, sep = "\\|") %>%
  group_by(genres) %>%
  summarise(count = n()) %>%
  arrange(desc(count)) %>%
  ggplot(aes(x = reorder(genres, -count), y = count)) +
  geom_bar(stat = "identity")+
  theme(axis.text.x=element_text(angle=90,hjust=1,vjust=0.5))+
  xlab("Genres")+
  ylab("Rating")+
  ggtitle("Number of Rating by Genre")
```

Analysing Rating Patterns

```
# mean rating by genre, with error
edx %>% separate_rows(genres, sep = "\\|") %>%
  group_by(genres) %>%
  summarise(count = n(), avg = mean(rating), se = sd(rating)/sqrt(count)) %>%
  mutate(genres = reorder(genres, avg)) %>%
  filter(genres != "(no genres listed)")%>%
  ggplot(aes(x = genres, y = avg, ymin = avg - 2*se, ymax = avg + 2*se)) +
  geom_point() +
  geom_errorbar()+
  theme(axis.text.x=element_text(angle=90,hjust=1,vjust=0.5))+
  ggtitle("Mean Rating by Genre")
```

```
#year of release
edx %>% mutate(releaseyear = as.numeric(str_extract(str_extract(title, "[/(]\\d{4}[/)]$"), regex("\\d{4}
  group_by(releaseyear) %>%
  summarize(rating = mean(rating)) %>%
  ggplot(aes(releaseyear, rating)) +
  geom_point() +
  geom_smooth()+
  ggtitle("Yearly Mean Rating")
```

```
#year of release by genre
#use ggforce to split into graphs into multiple pages
#first process the data
release_year_genre <- edx %>% mutate(releaseyear = as.numeric(str_extract(str_extract(title, "[/(]\\d{4}
  separate_rows(genres, sep = "\\|") %>%
  group_by(releaseyear,genres)%>%
  summarize(rating = mean(rating))

#find how many pages are needed
release_year_genre%>%
  ggplot(aes(releaseyear, rating)) +
  geom_point() +
  geom_smooth()+
  facet_wrap_paginate(~genres, nrow = 2, ncol = 2)+
  ggtitle("Yearly Mean Rating by Genre") ->p

required_n_pages <- n_pages(p)

#loop to print each page
for(i in 1:required_n_pages){
  release_year_genre%>%
    ggplot(aes(releaseyear, rating)) +
    geom_point() +
    geom_smooth()+
    facet_wrap_paginate(~genres, nrow = 2, ncol = 2, page = i)+
    ggtitle("Yearly Mean Rating by Genre") ->p

print(p)
}
```

```
#date of rating v release
edx %>% mutate(releaseyear = as.numeric(str_extract(str_extract(title, "[/(]\\d{4}[/)]$"), regex("\\d{4}
  mutate(date = as.numeric(what_year(round_date(as_datetime(timestamp),"year")))) %>%
  mutate(Release_Review = releaseyear - date) %>%
  group_by(Release_Review) %>%
  summarise(rating = mean(rating))%>%
  ggplot(aes(Release_Review, rating)) +
  geom_point() +
  geom_smooth()+
```

```
  xlab("Years After Release")+
  ylab("Mean Rating")+
  ggtitle("Effect of Delayed Rating")
```

```
#date of rating v release by genre
#use ggforce to break up into multiple pages
rating_release_genre <- edx %>% mutate(releaseyear = as.numeric(str_extract(str_extract(title, "[/(]\\d
  mutate(date = as.numeric(what_year(round_date(as_datetime(timestamp),"year")))) %>%
  mutate(Release_Review = releaseyear - date) %>%
  separate_rows(genres, sep = "\\|") %>%
  group_by(Release_Review, genres) %>%
  summarise(rating = mean(rating))

#find how many pages needed
rating_release_genre %>%
  ggplot(aes(Release_Review, rating)) +
  geom_point() +
  geom_smooth()+
  facet_wrap_paginate(~genres, nrow = 2, ncol = 2)+
  xlab("Years After Release")+
  ylab("Mean Rating")+
  ggtitle("Effect of Delayed Rating by Genre") -> p

required_n_pages <- n_pages(p)

#run loop to print each page
for (i in 1:required_n_pages){
  rating_release_genre %>%
    ggplot(aes(Release_Review, rating)) +
    geom_point() +
    geom_smooth()+
    facet_wrap_paginate(~genres, nrow = 2, ncol = 2, page = i)+
    xlab("Years After Release")+
    ylab("Mean Rating")+
    ggtitle("Effect of Delayed Rating by Genre") -> p

  print(p)
}
```

Analysing the Mean Ratings

Recommendation Systems

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

```r
#creating training and testing sets from edx
#test set will be 10% of edx
set.seed(1, sample.kind="Rounding") # if using R 3.5 or earlier, use `set.seed(1)`
test_index <- createDataPartition(y = edx$rating, times = 1, p = 0.1, list = FALSE)
train_set <- edx[-test_index,]
test_set <- edx[test_index,]
#insuring test movie and users are in training set
test_set <- test_set %>%
  semi_join(train_set, by = "movieId") %>%
  semi_join(train_set, by = "userId")
```

```r
#Model 1 - Simple Mean Rating

mu_hat <- mean(train_set$rating)

rmse_native <- RMSE(test_set$rating, mu_hat)

#save rmse results
rmse_results <- tibble(method = "Just the average", RMSE = rmse_native)

rmse_results %>%
  kbl(caption = "RMSE Results") %>%
  kable_styling(latex_options = c("striped", "hold_position"))
```

```r
#Model 2 - adding movie bias

mu <- mean(train_set$rating)

movie_avgs <- train_set %>%
  group_by(movieId) %>%
  summarize(b_i = mean(rating - mu))

predicted_ratings <- mu + test_set %>%
  left_join(movie_avgs, by='movieId') %>%
  pull(b_i)

rmse_movie_bias <-RMSE(predicted_ratings, test_set$rating)

rmse_results <- bind_rows(rmse_results, tibble(method ="With Movie Bias", RMSE = rmse_movie_bias ))

rmse_results %>%
  kbl(caption = "RMSE Results") %>%
  kable_styling(latex_options = c("striped", "hold_position"))
```

```r
#Model 3 - adding user bias
user_avgs <- train_set %>%
  left_join(movie_avgs, by='movieId') %>%
```

```
  group_by(userId) %>%
  summarize(b_u = mean(rating - mu - b_i))

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

rmse_user_bias <- RMSE(predicted_ratings, test_set$rating)

rmse_results <- bind_rows(rmse_results, tibble(method = "With Movie Bias and User Bias", RMSE = rmse_us

rmse_results %>%
  kbl(caption = "RMSE Results") %>%
  kable_styling(latex_options = c("striped", "hold_position"))
```

```
#Model 4 - adding Release Year
#adding Release Date to data sets
test_set <- test_set %>% mutate(releaseyear = as.numeric(str_extract(str_extract(title, "[/(]\\d{4}[/)]$
train_set <- train_set %>% mutate(releaseyear = as.numeric(str_extract(str_extract(title, "[/(]\\d{4}[/

release_avgs <- train_set %>%
  left_join(movie_avgs, by='movieId') %>%
  left_join(user_avgs, by='userId') %>%
  group_by(releaseyear) %>%
  summarize(b_r = mean(rating - mu - b_i - b_u))

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

rmse_release_bias <- RMSE(predicted_ratings, test_set$rating)

rmse_results <- bind_rows(rmse_results, tibble(method = "With Movie Bias and User Bias and Release Year

rmse_results %>%
  kbl(caption = "RMSE Results") %>%
  kable_styling(latex_options = c("striped", "hold_position"))
```

```
#Model 5 - Adding Genre Bias
#seperating genres in data sets
test_set <- test_set  %>% separate_rows(genres, sep = "\\|")
train_set <- train_set %>% separate_rows(genres, sep = "\\|")

genres_avgs <- train_set %>%
  left_join(movie_avgs, by='movieId') %>%
```

```
  left_join(user_avgs, by='userId') %>%
  left_join(release_avgs, by='releaseyear') %>%
  group_by(genres) %>%
  summarize(b_g = mean(rating - mu - b_i - b_u - b_r))

predicted_ratings <- test_set %>%
  left_join(movie_avgs, by= 'movieId') %>%
  left_join(user_avgs, by='userId') %>%
  left_join(release_avgs, by='releaseyear') %>%
  left_join(genres_avgs, by= "genres") %>%
  mutate(pred = mu + b_i + b_u + b_r + b_g) %>%
  pull(pred)

  rmse_genre_bias <- RMSE(predicted_ratings, test_set$rating)
  rmse_results <- bind_rows(rmse_results, tibble(method = "With Movie Bias, User Bias, Release Year Bias

  rmse_results %>%
  kbl(caption = "RMSE Results") %>%
  kable_styling(latex_options = c("striped", "hold_position"))
```

```
#Model 7 - subsituting release year by genre for release year - review year
#getting the year of review
  test_set <- test_set %>% mutate(date = as.numeric(what_year(round_date(as_datetime(timestamp),"year"))
    mutate(release_review = releaseyear - date)
  train_set <- train_set %>% mutate(date = as.numeric(what_year(round_date(as_datetime(timestamp),"year
    mutate(release_review = releaseyear - date)

  release_review_avgs <- train_set %>%
    left_join(movie_avgs, by='movieId') %>%
    left_join(user_avgs, by='userId') %>%
    group_by(release_review) %>%
    summarize(b_r_r = mean(rating - mu - b_i - b_u))

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

  rmse_release_review_bias <- RMSE(predicted_ratings, test_set$rating)
  rmse_results <- bind_rows(rmse_results, tibble(method = "With Movie Bias, User Bias, and Release/Revi

  rmse_results %>%
  kbl(caption = "RMSE Results") %>%
  kable_styling(latex_options = c("striped", "hold_position"))
```

```
#Model 9 - substituting release year - review year for release year - review year by genre
  #adding release-review
  # NA's being caused in movieId 3136, Documentary, r_r c(-48, -49, -52)
```

```
  test_set <- test_set %>% mutate(date = as.numeric(what_year(round_date(as_datetime(timestamp),"year"))
    mutate(release_review = releaseyear - date)
  train_set <- train_set %>% mutate(date = as.numeric(what_year(round_date(as_datetime(timestamp),"year
    mutate(release_review = releaseyear - date)


  release_review_genre_avgs <- train_set %>%
    left_join(movie_avgs, by='movieId') %>%
    left_join(user_avgs, by='userId') %>%
    group_by(release_review, genres) %>%
    summarize(b_r_r_g2 = mean(rating - mu - b_i - b_u))

  predicted_ratings <- test_set %>%
    left_join(movie_avgs, by= 'movieId') %>%
    left_join(user_avgs, by='userId') %>%
    left_join(release_review_genre_avgs, by= c("release_review", "genres")) %>%
    mutate(pred = mu + b_i + b_u + b_r_r_g2) %>%
    pull(pred)


  rmse_release_review_genre_bias <- RMSE(predicted_ratings, test_set$rating)
  rmse_results <- bind_rows(rmse_results, tibble(method = "With Movie Bias, User Bias, and Release/Revi

rmse_results %>%
  kbl(caption = "RMSE Results") %>%
  kable_styling(latex_options = c("striped", "hold_position"))
```

Mean and Biases Based Models

```
#Model 10 - Regularized Movies - PLS
lambdas <- seq(0, 10, 0.25)

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

  mu <- mean(train_set$rating)

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

  predicted_ratings <-
    test_set %>%
    left_join(b_i, by = "movieId") %>%
    mutate(pred = mu + b_i ) %>%
    pull(pred)

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

qplot(lambdas, rmses)
```

```
lambdas[which.min(rmses)] %>%
  kbl(caption = "Lambda with Lowest RMSE") %>%
  kable_styling(latex_options = c("striped", "hold_position"))
```

```
#Model 11 - reg movie and user bias
lambda <-  2.5
mu <- mean(train_set$rating)

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

user_avgs_new <- train_set %>%
  left_join(movie_reg_avgs, by= "movieId") %>%
  group_by(userId)%>%
  summarise(b_u = mean(rating - mu - b_i))

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

rmse_reg_movie_and_user <- RMSE(predicted_ratings, test_set$rating)
rmse_results <- bind_rows(rmse_results, tibble(method = "With Regularised Movie and User Bias ", RMSE =

rmse_results %>%
  kbl(caption = "RMSE Results") %>%
  kable_styling(latex_options = c("striped", "hold_position"))
```

Regularization

```
#Model 13 - recosystem
set.seed(1, sample.kind="Rounding") # if using R 3.5 or earlier, use `set.seed(1)
train_reco <- with(train_set, data_memory(user_index = userId, item_index = movieId, rating = rating))
test_reco <- with(test_set, data_memory(user_index = userId, item_index = movieId, rating = rating))

r <- Reco()
tuned_reco <- r$tune(train_reco,opts = list(dim = c(10, 20, 30), lrate = c(0.1, 0.2),
                                            costp_l1 = 0, costq_l1 = 0,
                                            nthread = 1, niter = 10))

r$train(train_reco, opts = c(tuned_reco$min, nthread = 1, niter = 20))
reco_results <- r$predict(test_reco, out_memory())

rmse_reco <- RMSE(reco_results, test_set$rating)
rmse_results <- bind_rows(rmse_results, tibble(method = "Recosystem ", RMSE = rmse_reco ))
```

```
rmse_results %>%
  kbl(caption = "RMSE Results") %>%
  kable_styling(latex_options = c("striped", "hold_position"))
```

Matrix Factorization

```
train_reco <- with(edx, data_memory(user_index = userId, item_index = movieId, rating = rating))
test_reco <- with(validation , data_memory(user_index = userId, item_index = movieId, rating = rating))

r <- Reco()

tuned_reco <- r$tune(train_reco,opts = list(dim = c(10, 20, 30), lrate = c(0.1, 0.2),
                                            costp_l1 = 0, costq_l1 = 0,
                                            nthread = 1, niter = 10))

r$train(train_reco, opts = c(tuned_reco$min, nthread = 1, niter = 20))
reco_results <- r$predict(test_reco, out_memory())

final_rmse_reco <- RMSE(reco_results, validation$rating)
rmse_results <- bind_rows(rmse_results, tibble(method = "Final Validation: Recosystem ", RMSE = final_rm

rmse_results %>%
  kbl(caption = "RMSE Results") %>%
  kable_styling(latex_options = c("striped", "hold_position"))
```

Final Validation