Regularization for MovieLens Recommendation System

Andres E. Osuna

6/17/2020

Contents

1. Introduction

This project consists on the analysis of the MovieLens dataset with the objective of creating a recommendation system that predicts the rating that a certain user will give to a specific movie. This dataset consists of over 10 millions observations, recorded alongside 6 features. It includes userId, movieId, timestamp, movie title, movie genre and rate given.

To start the analysis process, a data partition was done to divide the dataset into training and test set.The training set will be used to calculate different effects that determine the calculation of a predicted rating. Then, an important dataset exploration analysis was performed to understand the structure of the data and gain insights on what the best approach is.

Moreover, a regularization approach was determined to be the best one because of the different ammount of observations given to different categorizations. Some have many more observations than others. In addition, the objective of these models is to minimize the Root Mean Squared Error (RMSE):

$$\frac{1}{N}\sqrt{\sum_{t=1}^{n} e_t^2}$$

The expectation is that the RMSE can be decreased by using movie, user, year, and genre effects. The goal of achieving an RMSE lower than 0.8649 was achieved.

2. Dataset Exploratory Analysis

First step is to obviously download the dataset and do a random data set parition that will create two sets. The dataset was divided into sets: edx (training set) and validation (test set). The edx dataset contaings 90% of the total MovieLens dataset, meaning the validation set has only 10%, which is a good amount to test our regularized effects. A preview of the data can be seen in the table below:

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

Let's examine the structure of both our sets. Both the edx and the validation set contains 6 variables, which were mentioned recently in the introduction. You can see the classes of all features in the output below:

Var	class
userId	integer
timestamp	integer
movieId	numeric
title	character
genres	character
rating	numeric

As shown below, the observation size of the validation set is apprixmately 1 million observations while edx contains over 9 million observations.

set	N_rows	N_columns
edx	9000055	6
validation	999999	6

Next, it was important to check how many rating entries were 0 to understand a little better how this dataset is structured and how rough the users grade the movies. As you can observe in the code output below, no entries have a rating of 0, meaning user will either not grade a movie or rate it with a 0.5 or higher.

```r
N_zeros<- edx %>% filter(rating==0) %>% summarize(n=n())
N_zeros%>% kable() %>% kable_styling()
```

n
0

Furthemore, let's try to understand how many users actually participate in the rating procedure of our training set. The code below shows that a total of 69,878 unique users are registered in our edx set. In addition, the total number of unique movies in the edx set is 10,677.

```r
N_different_movies<- n_distinct(edx$movieId)
N_different_users<- n_distinct(edx$userId)
paste("Our data set contains", N_different_movies, "movies","&",N_different_users, " users")
```

```
## [1] "Our data set contains 10677 movies & 69878  users"
```

Lets keep drilling down into our dataset. Another thing to notice is some of the main genres the dataset contains. Drama seems to be one of the most common genres in our movie dataset

```r
N_drama<- edx %>% filter(str_detect(genres, "Drama")) %>% nrow()
N_comedy<- edx %>% filter(str_detect(genres, "Comedy")) %>% nrow()
N_thriller<- edx %>% filter(str_detect(genres, "Thriller")) %>% nrow()
N_romance<- edx %>% filter(str_detect(genres, "Romance")) %>% nrow()
data.frame(Genre= c("Drama", "Comedy", "Thriller", "Romance"),
                 N=c(N_drama, N_comedy, N_thriller, N_romance))%>%
  kable() %>% kable_styling()
```

Genre	N
Drama	3910127
Comedy	3540930
Thriller	2325899
Romance	1712100

This shows the 10 movies with more ratings. This could also be considered the most popular movies with respect to the amount of observations

```r
#movies with the greatest number of ratings:
top_10<-edx %>% group_by(title) %>% summarize(n=n(), .groups= "drop") %>%
  top_n(10, n) %>% arrange(desc(n))
top_10%>% kable() %>% kable_styling()
```

title	n
Pulp Fiction (1994)	31362
Forrest Gump (1994)	31079
Silence of the Lambs, The (1991)	30382
Jurassic Park (1993)	29360
Shawshank Redemption, The (1994)	28015
Braveheart (1995)	26212
Fugitive, The (1993)	25998
Terminator 2: Judgment Day (1991)	25984
Star Wars: Episode IV - A New Hope (a.k.a. Star Wars) (1977)	25672
Apollo 13 (1995)	24284

Now we can check the top 5 ratings given by users:

```r
#the five most given ratings:
top_5<-edx %>% group_by(rating)%>% summarize(n=n(), .groups= "drop") %>%
  top_n(5, n) %>% arrange(desc(n))
top_5%>% kable() %>% kable_styling()
```

rating	n
4.0	2588430
3.0	2121240
5.0	1390114
3.5	791624
2.0	711422

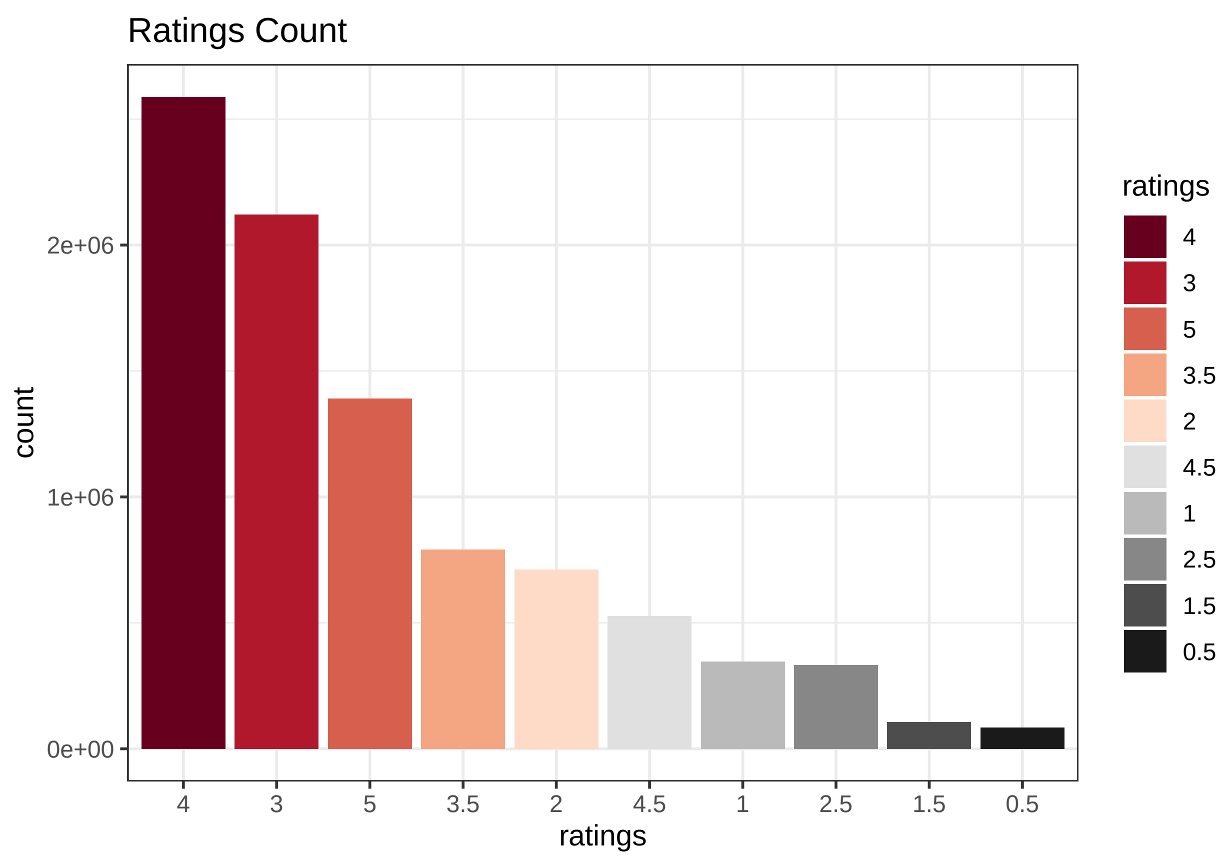
The next summary represents the count of how many timese each rating appear on the dataset. As you can start to observe, it is more common to give whole number as ratings. Like 4,3,5, being the top 3 ratings given.

```r
#half star rating are less common than the whole star ratings
summary_ratings<-edx %>% group_by(rating)%>% summarize(n=n(), .groups= "drop") %>%
  arrange(desc(n))
summary_ratings %>% kable() %>% kable_styling()
```

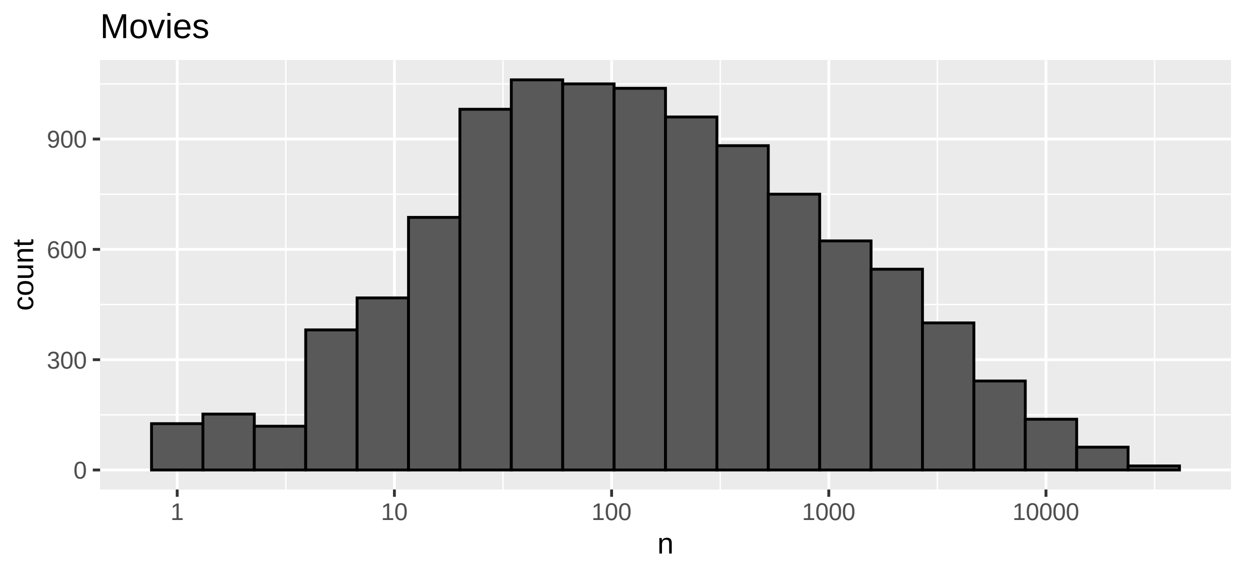
rating	n
4.0	2588430
3.0	2121240
5.0	1390114
3.5	791624
2.0	711422
4.5	526736
1.0	345679
2.5	333010
1.5	106426
0.5	85374

3. Data Visualization Insights

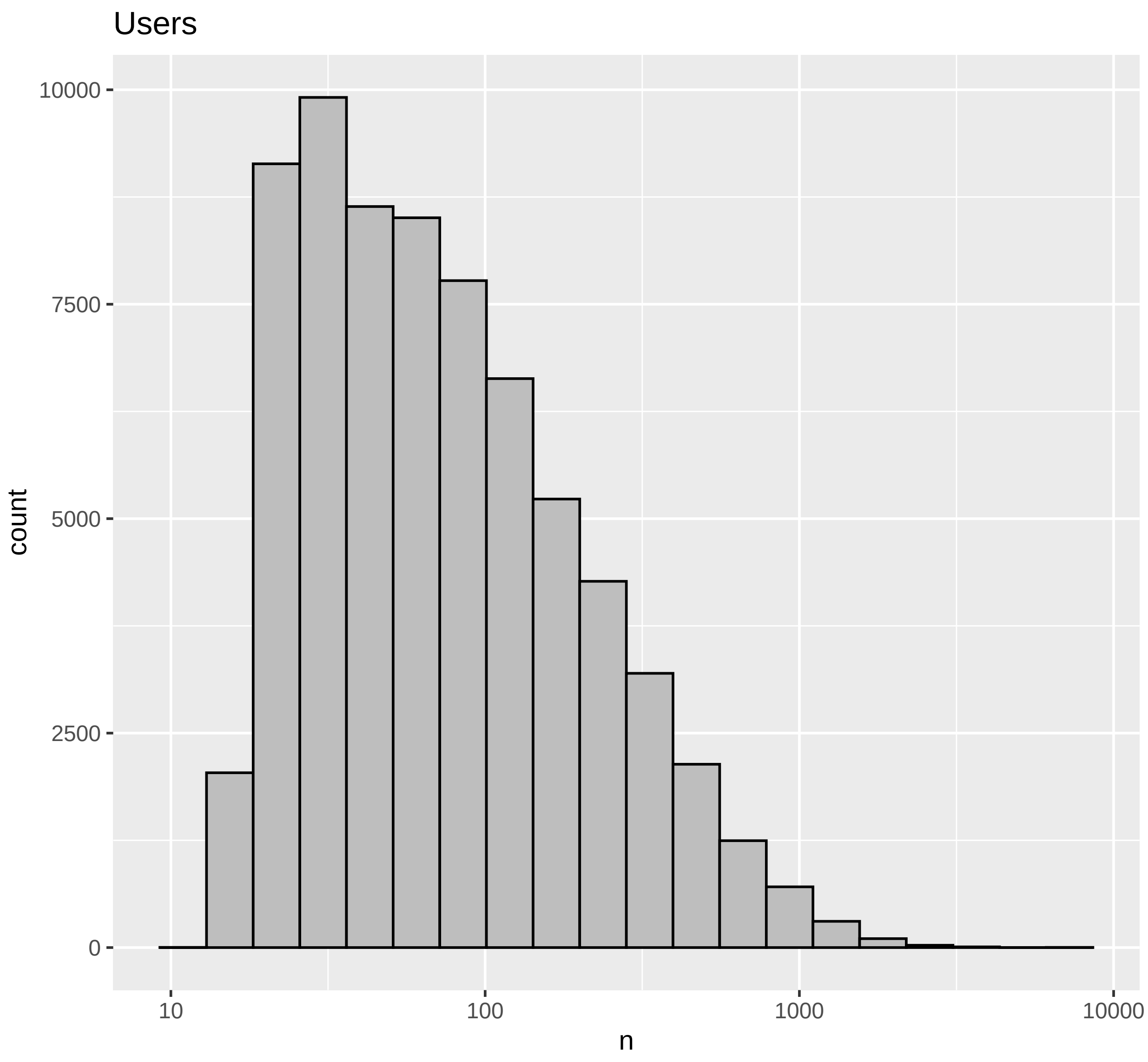
Let's start by checking the ratings frecuency count:



Note the distribution of ratings and how common ratings of a 3, 4 or 5 are. From this we can expect that our movie avg rating is within 3 and 4.

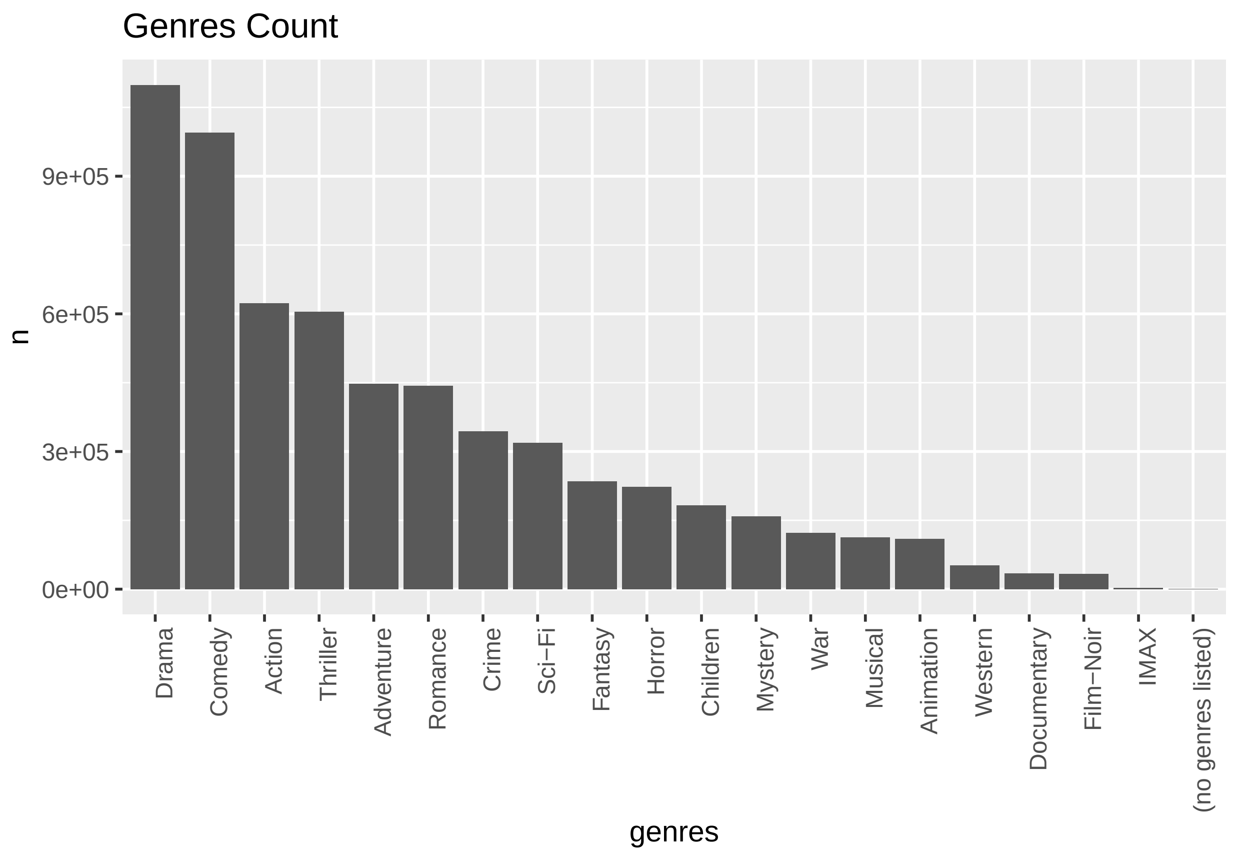


As seen in the figure above, there is high variability within how often a certain movieId is rated, meaning that some movies have more tendency to be watched and rated by the other users. This make senses if you think about it. As shown before, a table with the top rated movies was presented. Most of these movies could be considered “classics”, “great”, or even as some of the “most popular films” ever. Therefore, it makes sense to think that users will always tend to focus more on well recognized movies.



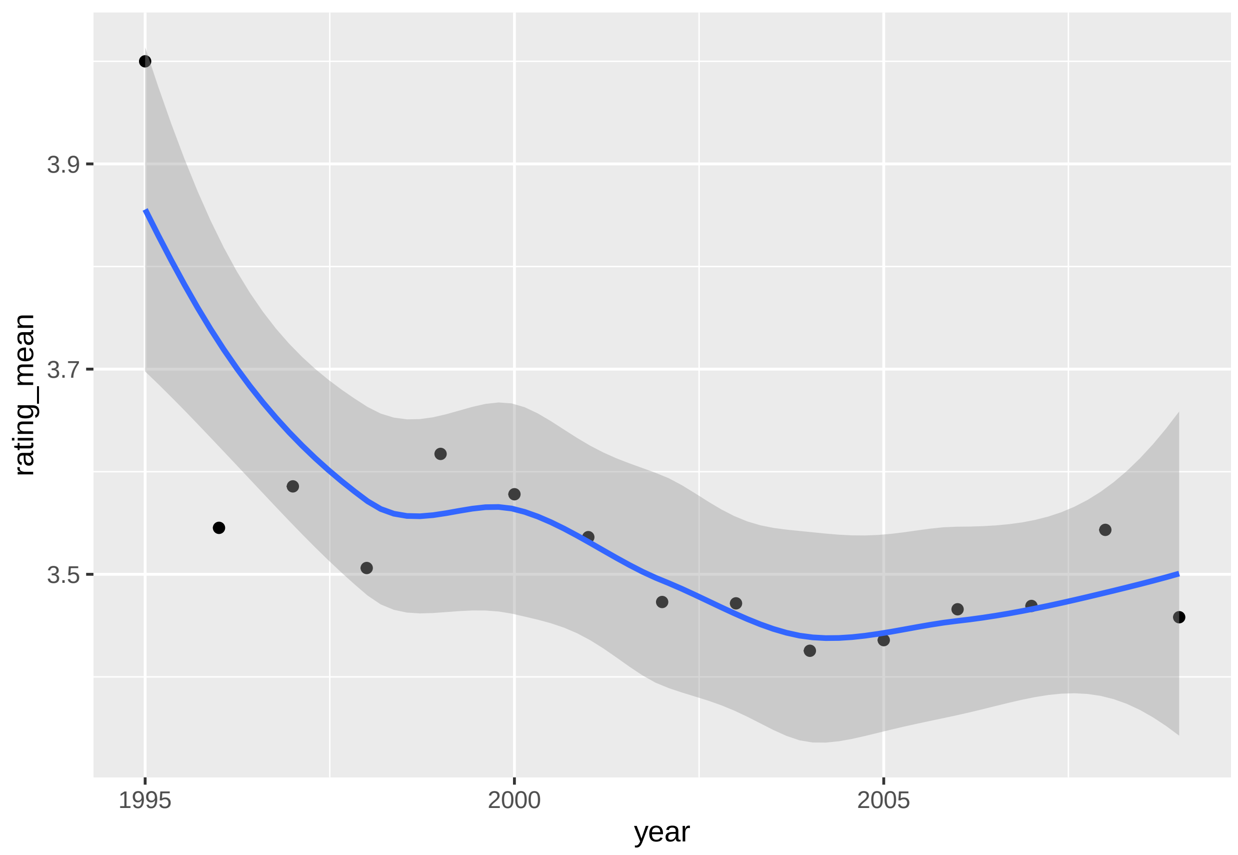
Furthermore, there is high variability within how often a certain userId rates movies, meaning that some users are more actively rating than others. This also make senses. Not everyone is the same, and not everyone takes the time to rate a movie.The figure below represents the amount of times different users rate movies.

Also remember that not every user behaves the same way, they have different tastes in movies. So, taking into account the genre variability might also be important. As seen below, some genres are more rated than others. Maybe more people like drama, comedy, and action movies, so they get rated more often.



Another interesting effect could be how users behave through time. As you can observe in the figure below, movies would get rated higher back in the 1990s, meaning that movie's year variability might also have an impact on how often some movies get rated and how. The figure shows a clear downward trend. It is also important to mention that this could be happening because more movies are being released after the 2000s, bringing the movie rating average down with time.

```
## ‘geom_smooth()‘ using method = ’loess’ and formula ’y ~ x’
```



All of these figures give use insight on what features to use for our modeling approach. Ratings seem to have a movie, user, genre, and released year effect. All of these will be tested in a later section of this report.

4. Dataset Pre-processing

Before getting into the details of the modeling approach. There is some work to be done. It is important to wrangle with a couple of things within the data. First, we need to create a new feature called "year_released". This will represent the year the movie was released. This will be part of the calculation of the year effect later on. The timestamp variable was also modified so that it can show actual useful information about the date in which the observation was recorded. The new usable dataset is called "new_edx".

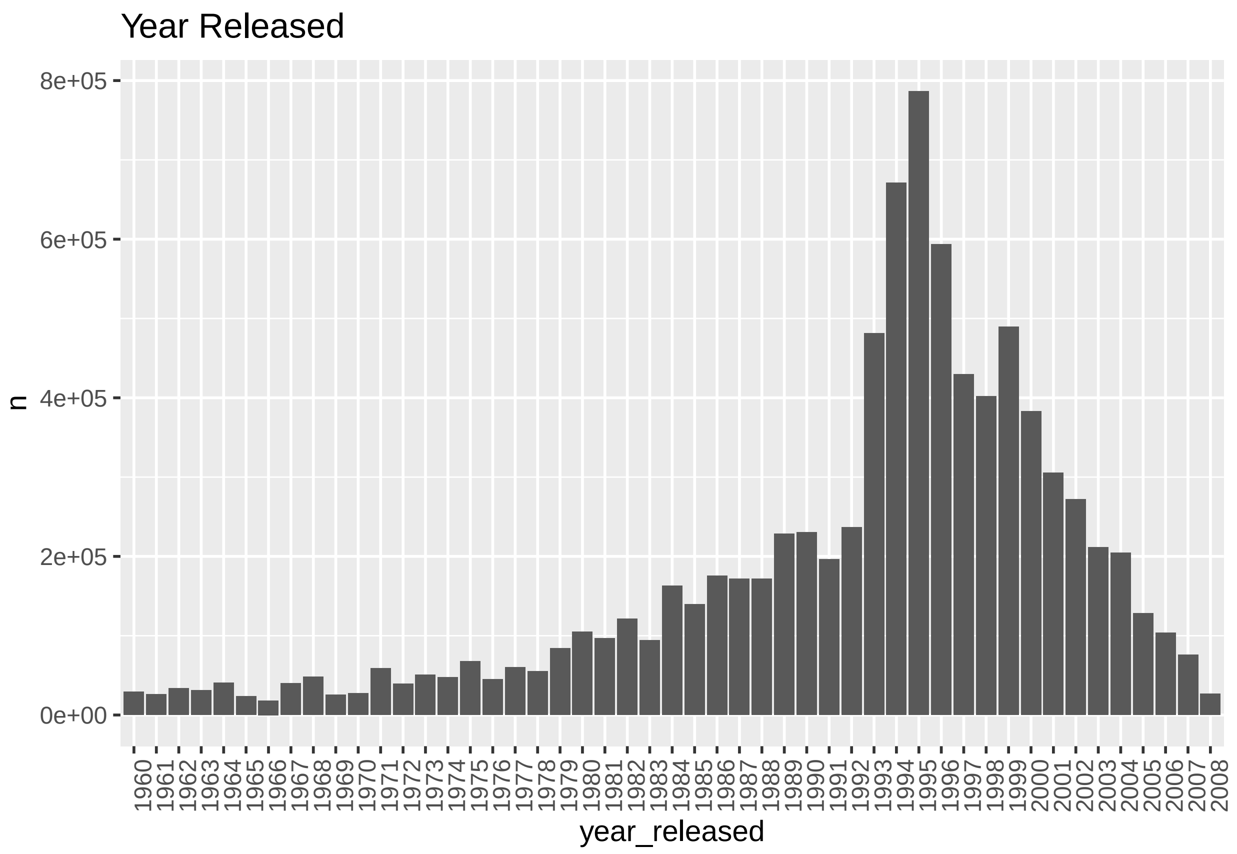
The code below provides specifc on how this was done:

```
#extracting released year from the title to create a new feature
new_edx<-edx %>%
  mutate(year_released= str_extract(title,("\\(\\d{4}\\)"))) %>% #extracting year inside()
  mutate(year_released= str_extract(year_released, "\\d+")) %>%
  rename(date= timestamp) %>%
  mutate(date= as_datetime(date)) %>% #convert the timestap to a date everyone can read
  mutate(date= round_date(date, unit= "month"))
```

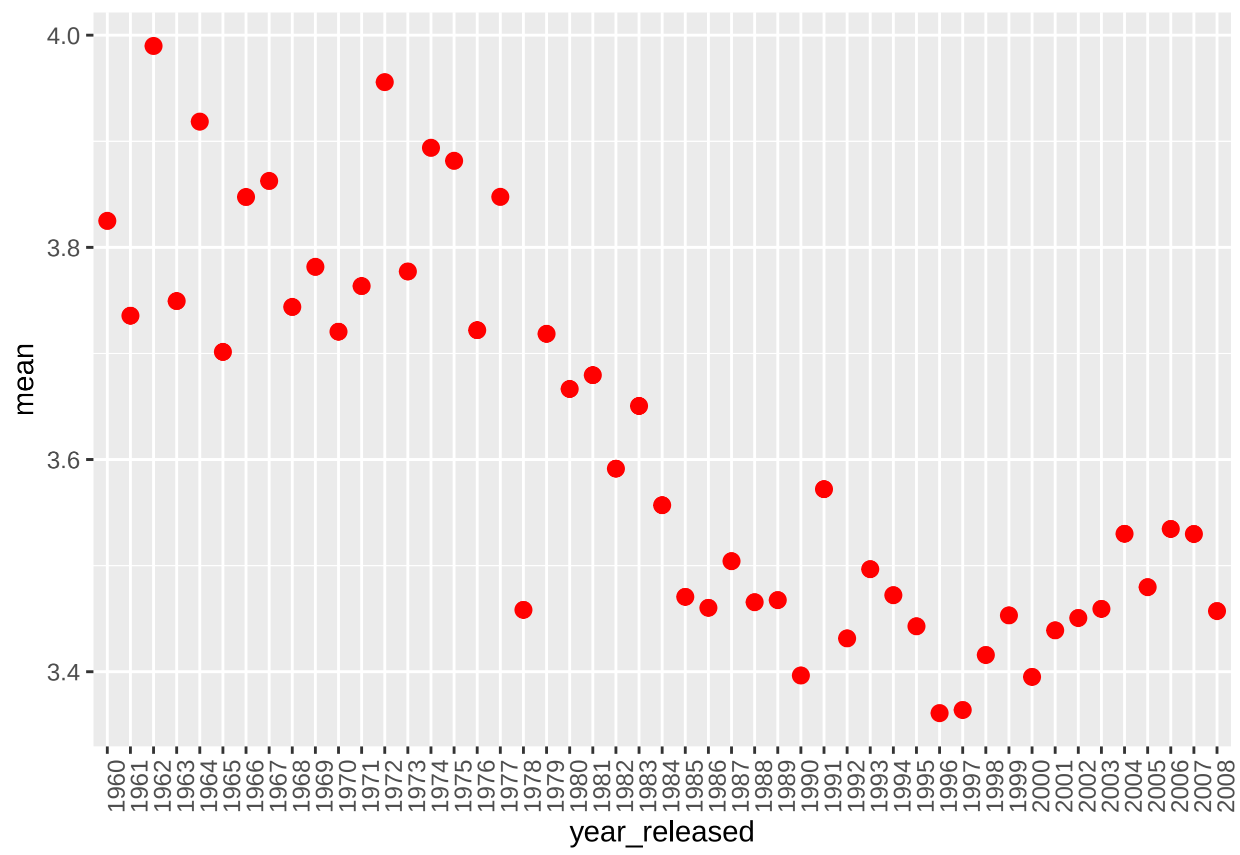
We also need to do the same procedure for the validation set since we will use this to test out our effects estimates.

```
#It is also necessary to do it to the validation set
validation<-validation %>%
  mutate(year_released= str_extract(title,("\\(\\d{4}\\)"))) %>% #same for validation set
  mutate(year_released= str_extract(year_released, "\\d+")) %>%
  rename(date= timestamp) %>%
  mutate(date= as_datetime(date)) %>% #do the same for the validation set
  mutate(date= round_date(date, unit= "month"))
```

After wrangling with the data, it seems that the hypothesis that there is an upward trend within our feature is confirmed. The figure below shows that there are many more ratings for recent movies within the 1990s than for some other years.



As seen in the figure below, there is a strong drop in the rating avg after the 1980s. There seems to be an interesting trend and variability that should help us predict the ratings of the valdiation set. The figure above determined that the edx and validation data set should have an inbalanced amount of obseravtions with respect to the year released.



5. Methodlogy - Modeling Approach

Naive Mean Baseline Approach

First, a Naive mean approach was used to create a baseline benchmark. The predictions will be made using the following formula:

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

In this equation, $\hat{\mu}$ will represent the mean of all observations (ratings), and $\epsilon_{u,i}$ represents the independent error or noise associated with the sample, which comes from the same distribution.

```
avg_rating<- mean(new_edx$rating) #we calculate the mean of the training set

#this function calculates the RMSE between the validation ratings and the predicted values
RMSE<- function(true_ratings, predicted_rating){
  sqrt(mean((true_ratings-predicted_rating)^2))
}

baseline_rmse<-RMSE(validation$rating, avg_rating)
#the simples approach, in which we predict all ratings will be equal to the mean
```

This baseline provides a poor RSME of approximately of 1.06.

If we start from the baseline approach, it is clear that many effects and variables that should be taking into consideration are missing. Currently, none of these variables are present in the Naive mean approach model. These new variables are introduced with following notation:

m_i: movie effect for movie i. u_u: user effect for user u. r_j: year effect for each year j. g_g: genre effect for each genre g.

The following formula for the prediction $Y_{u,i}$ now captures all of the new effects:

$$Y_{u,i,g,j} = \hat{\mu} + m_i + u_u + r_j + g_g + \epsilon_{u,i,g,j}$$

Instead of using these variables-based model approach, it will be more appropiate to directly use the regularization approach since it was shown that within our dataset, some smaller samples are present.These smaller samples directly affect model estimates.

Regularization Approach

Regularization is an approach used to penalize large estimates that come from smaller sample sizes. The general idea is to add a penalty for large values of the effect's estimate to the sum of squares equations that is being minimized.

In the data insights section, it was shown that many of these variables have a high variance with respect to the amount of times present in the data. Some movies are rated more than others. Some users rate more often than others. Some genres are also more often rated than others, and finally, the movie released year seems to also have an effect on the rating decision. For this problem, a regularization approach is commonly used for better performance.

To check the impact of each new variable introduced, 4 new models were developed, following a regularization approach.

Regularized Movie Effect Model

After explaining the regularization approach, let's incorporate one effect at a time for better understanding. In this way, the improvement can be seen little by little, by adding new terms to the following equations. By using a regularization approach, the new objective is to minimize the following formula:

$$\frac{1}{N}\sqrt{\sum_{i,u}^{n}(y_{u,i} - m_i - \hat{\mu})^2} + \lambda\sum_{i}^{n} m_i^2$$

The first term is the RMSE and the second term represents the penalty (λ) to all estimates that have a smaller sample size compared to the rest of the sample space.

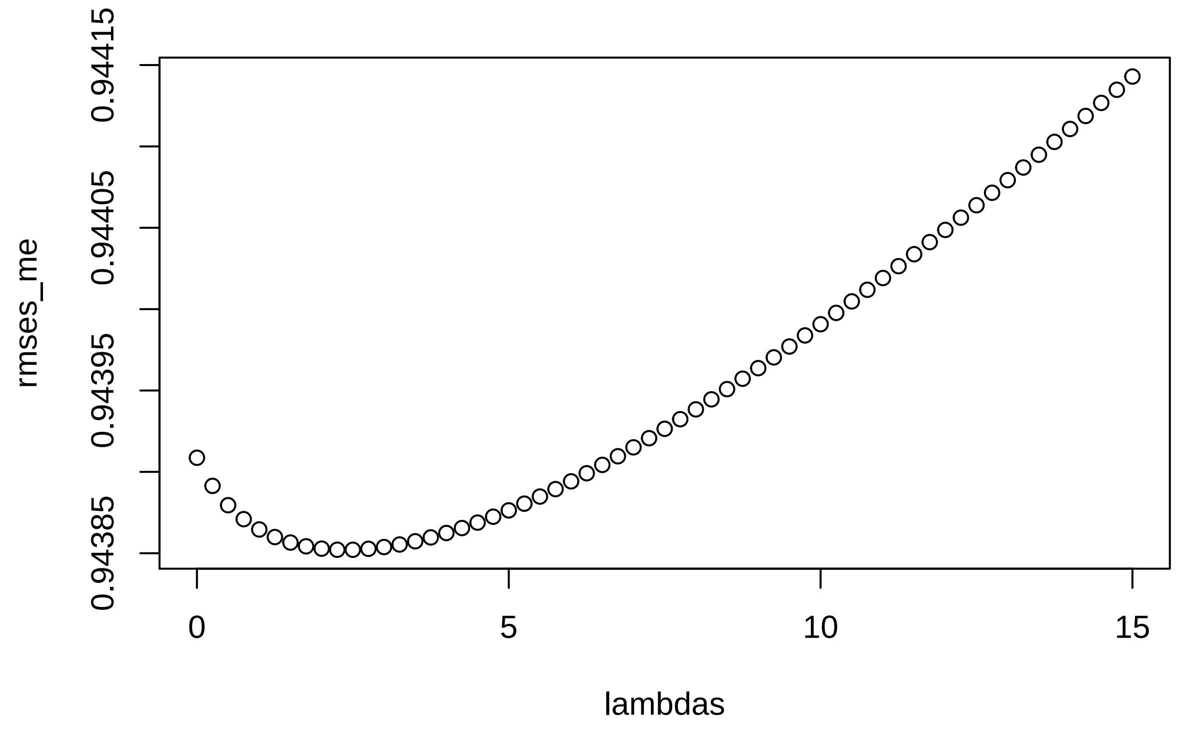
By using simple math, this objective can be minimized by the following formula:

$$\hat{m_i}(\lambda) = \frac{1}{\lambda + n_i}\sum_{i,u}^{n}(Y_{u,i} - \hat{\mu})$$

We can calculate the new RMSE using the following code:

```r
#Regularized Movie effect model

#set a vector of different lambda to calculate best penalty for movie effect
lambdas<- seq(0,15,0.25)
```

```r
#apply the vector of lambdas to a function that will calculate the rmses
rmses_me<- sapply(lambdas, function(lambda){
  m_i<- new_edx %>%
    group_by(movieId) %>%
    summarize(m_i= sum(rating-avg_rating)/(n()+ lambda), .groups= "drop")
  #calculate the movie effect and penalize the groups with small number of ocurrences

  predicted_ratings<-validation %>% #use the test set to predict movie ratings by users
    left_join(m_i, by= "movieId") %>%
    mutate(prediction= avg_rating+ m_i) %>%
    .$prediction #extract predictions
  return(RMSE(validation$rating, predicted_ratings))
  #call the RMSE function developed before
})


plot(lambdas, rmses_me) #plot penalties vs rmses
```



```r
optlm<- lambdas[which.min(rmses_me)] #extracts optimal penalty for movie effect
paste("The best achieved RMSE is ", round(rmses_me[which.min(rmses_me)],5),
      "with a penalty of ", optlm)
```

```
## [1] "The best achieved RMSE is  0.94385 with a penalty of  2.5"
```

In the code above, the training set(new_edx) was used to calculate the estimates for m_i and then predict the ratings in the validation set, using different values for (λ). In this case, we get an RMSE OF 0.94385. This is a good, but not enough, improvement from the Naive mean approach. This RMSE was found using a penalty of 2.5.

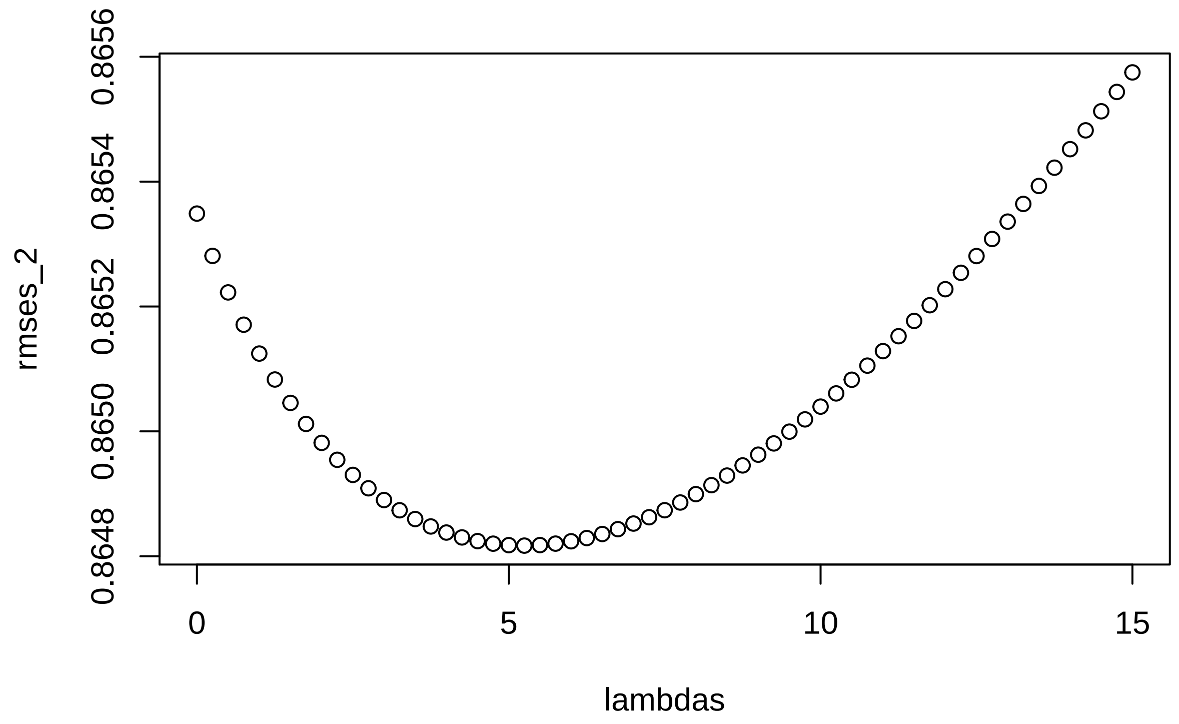
Regularized Movie + User Effect Model

Now, for better prediction, let's include the user effect with the same approach. The formula was modified to incorporate the new effect:

$$\frac{1}{N}\sqrt{\sum_{i,u}^{n}(y_{u,i} - m_i - u_u - \hat{\mu})^2} + \lambda\sum_{i}^{n} m_i^2 + \lambda\sum_{u}^{n} u_u^2$$

As we increase the number of variables, the formula basically extends. This can be observed in the formula above and the code bellow. The solution to this model can be solved by:

```
#apply the vector of lambdas to a function that will calculate the rmses
rmses_2<- sapply(lambdas, function(lambda){

m_i<- new_edx %>%
    group_by(movieId) %>%
    summarize(m_i= sum(rating-avg_rating)/(n()+ lambda), .groups= "drop")
#calculate the movie effect and penalize the groups with small number of ocurrences

u_u <- new_edx %>%
    left_join(m_i, by= "movieId") %>%
    group_by(userId) %>%
    summarize(u_u= sum(rating- m_i- avg_rating)/(n()+lambda), .groups= "drop")
#calculate the user effect and penalize the groups with small number of ocurrences

predicted_ratings<-validation %>%
    left_join(m_i, by= "movieId") %>%
    left_join(u_u, by="userId") %>%
    mutate(prediction= avg_rating+ m_i+ u_u) %>%
    .$prediction

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

plot(lambdas, rmses_2) #it plots penalties vs rmses
```



```
optl2<- lambdas[which.min(rmses_2)] #extracts optimal penalty for movie + user effect
paste("The best achieved RMSE is ", round(rmses_2[which.min(rmses_2)],5),
      "with a penalty of ", optl2)
```

```
## [1] "The best achieved RMSE is  0.86482 with a penalty of  5.25"
```

In this case, we get a major improvement, and beat expectations with an RMSE of 0.86482. This was found by using a penalty of 5.25

Regularized Movie + User + Year Effect Model

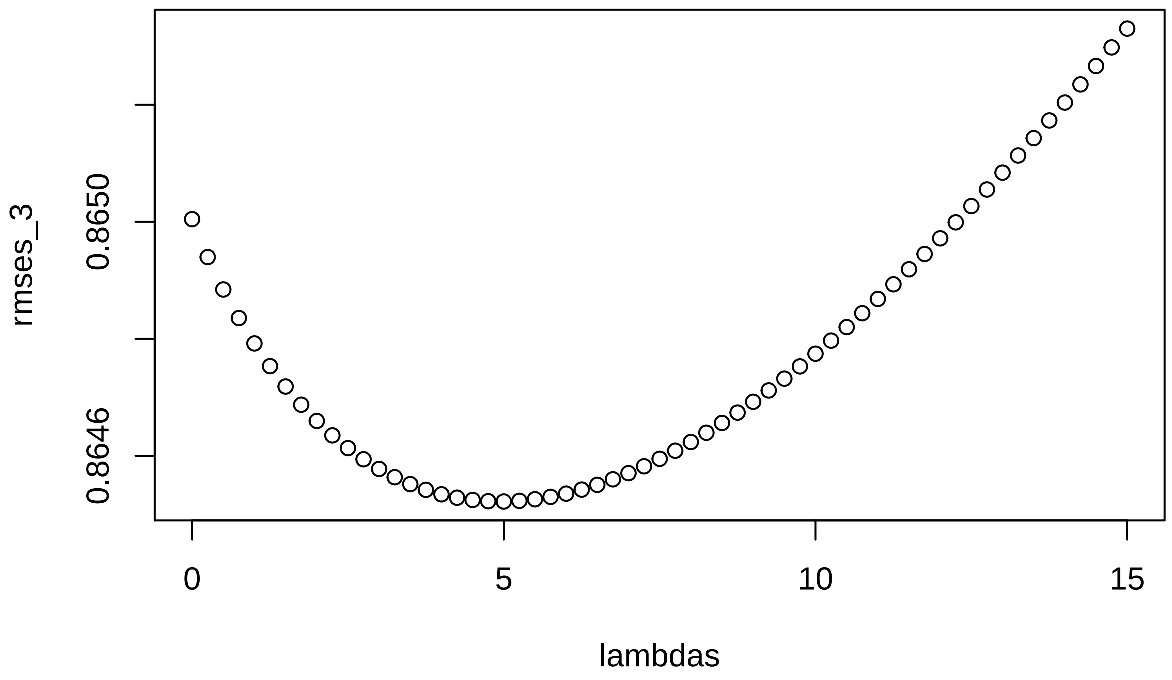
The third model includes the movie's release year as an effect on the quality and rating of the movies. The new minimization ojective is as follows:

$$\frac{1}{N}\sqrt{\sum_{i,u,j}^{n}(y_{u,i,j}-m_i-u_u-r_j-\hat{\mu})^2}+\lambda\sum_{i}^{n}m_i^2+\lambda\sum_{u}^{n}u_u^2+\lambda\sum_{j}^{n}r_j^2$$

By running the code below the new RMSE is found:

```
#apply the vector of lambdas to a function that will calculate the rmses
rmses_3<- sapply(lambdas, function(lambda){

m_i<- new_edx %>%
```

```r
    group_by(movieId) %>%
    summarize(m_i= sum(rating-avg_rating)/(n()+ lambda), .groups="drop")
#calculate the movie effect and penalize the groups with small number of ocurrences

u_u <- new_edx %>%
    left_join(m_i, by= "movieId") %>%
    group_by(userId) %>%
    summarize(u_u= sum(rating- m_i- avg_rating)/(n()+lambda), .groups= "drop")
#calculate the user effect and penalize the groups with small number of ocurrences

r_j<- new_edx %>%
    left_join(m_i, by= "movieId") %>%
    left_join(u_u, by= "userId") %>%
    group_by(year_released) %>%
    summarize(r_j= sum(rating-m_i-u_u-avg_rating)/ (n()+ lambda), .groups= "drop")
#calculate the year effect and penalize the groups with small number of ocurrences

predicted_ratings<-validation %>%
    left_join(m_i, by= "movieId") %>%
    left_join(u_u, by="userId") %>%
    left_join(r_j, by= "year_released") %>%
    mutate(prediction= avg_rating+ m_i+ u_u+ r_j) %>%
    .$prediction

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


plot(lambdas, rmses_3) #it plots penalties vs rmses
```



```
optl3<- lambdas[which.min(rmses_3)]
#extracts optimal penalty for movie + user effect + year_released effect
paste("The best achieved RMSE is ", round(rmses_3[which.min(rmses_3)],5),
      "with a penalty of ", optl3) #we print the rmse results
```

```
## [1] "The best achieved RMSE is  0.86452 with a penalty of  5"
```

This model still provides a slight improvement for an RMSE of 0.86452 with a penalty of 5.

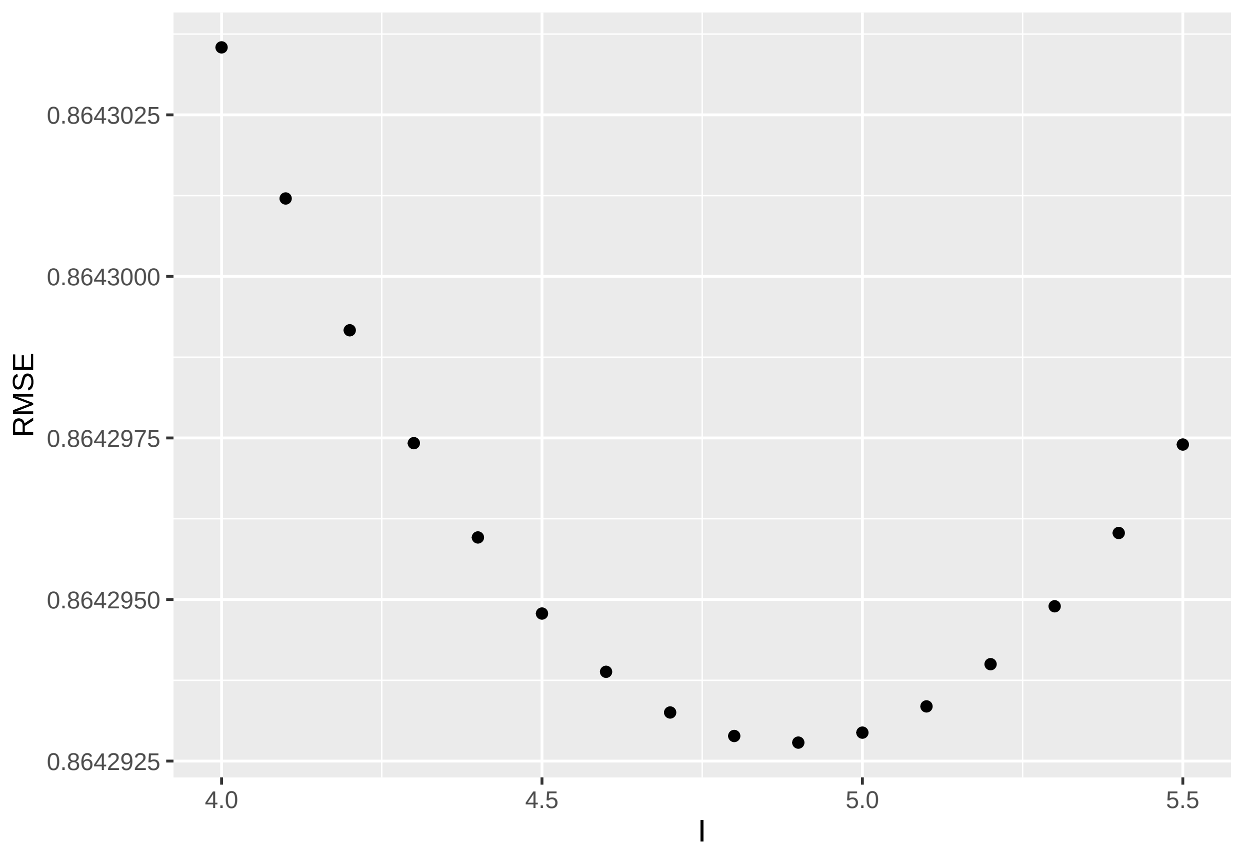
Regularized Movie + User + Year+ Genre Effect Model

The last model includes the movie's genre as an effect on the quality and rating of the movies. The new minimization ojective is as follows:

$$\frac{1}{N}\sqrt{\sum_{i,u,g,j}^{n}(y_{u,i}-m_i-u_u-r_j-g_g-\hat{\mu})^2}+\lambda\sum_{i}^{n}m_i^2+\lambda\sum_{u}^{n}u_u^2+\lambda\sum_{j}^{n}r_j^2+\lambda\sum_{g}^{n}g_g^2$$

```
#in this case we will a more defined subset of lambdas for greater precision
l<- seq(4,5.5,0.10) #sets a vector of penalties for last model

#apply the vector of lambdas to a function that will calculate the rmses
calculate_rmses<- sapply(l, function(lambda){
```

```
m_i<- new_edx %>%
    group_by(movieId) %>%
    summarize(m_i= sum(rating-avg_rating)/(n()+ lambda),.groups="drop")
#calculate the movie effect and penalize the groups with small number of ocurrences

u_u <- new_edx %>%
    left_join(m_i, by= "movieId") %>%
    group_by(userId) %>%
    summarize(u_u= sum(rating- m_i- avg_rating)/(n()+lambda),.groups="drop")
#calculate the user effect and penalize the groups with small number of ocurrences


g_g<- new_edx %>%
  left_join(m_i, by= "movieId") %>%
  left_join(u_u, by= "userId") %>%
  group_by(genres) %>%
  summarize(g_g= sum(rating-m_i-u_u-avg_rating)/(n()+lambda),.groups="drop")
#calculate the genre effect and penalize the groups with small number of ocurrence

r_j<- new_edx %>%
  left_join(m_i, by= "movieId") %>%
  left_join(u_u, by= "userId") %>%
  left_join(g_g, by= "genres") %>%
  group_by(year_released) %>%
  summarize(r_j= sum(rating-m_i-g_g-u_u-avg_rating)/ (n()+ lambda),.groups="drop")
#calculate the year effect and penalize the groups with small number of ocurrences

predicted_ratings<-validation %>%
  left_join(m_i, by= "movieId") %>%
  left_join(u_u, by="userId") %>%
  left_join(g_g, by= "genres") %>%
  left_join(r_j, by= "year_released") %>%
  mutate(prediction= avg_rating+ m_i+g_g+ u_u+ r_j) %>%
  .$prediction

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

data.frame(l=l, RMSE=calculate_rmses) %>%
  ggplot(aes(l, RMSE)) +
  geom_point() #curve between penalty and rmse
```



```
optimal_lambda<-l[which.min(calculate_rmses)] #extracts optimal penalty
paste("The best achieved RMSE is ", round(calculate_rmses[which.min(calculate_rmses)],5),
      "with a penalty of ", optimal_lambda) #prints the RMSE solution
```

```
## [1] "The best achieved RMSE is  0.86429 with a penalty of  4.9"
```

This improved the RMSE even more with a result of 0.86429 with a penalty of 4.9.

6. Results

In this section, all results are summarized and provided for a better look into the approach. As you can see, the best approach is the last model, which includes all 4 effects. The year effect generated and effect, but not as much as expected.The effect that impacted the most seems to be the movie and user effect. Clearly, users behave differently and defining a user behaviour is the key to predict the rating they would give to a certain movie.

The table below shows all the results:

Model	penalty	RMSE
Baseline Naive-mean Model	0.00	1.06120
Regularized Movie effect	2.50	0.94385
Regularized Movie + User effect	5.25	0.86482
Regularized Movie + User + Year effect	5.00	0.86452
Regularized Movie+ User+ Year+ Genre	4.90	0.86429

These modeling approaches worked quite well. The final model was able to beat expectations with an RMSE of 0.86429.

7. Conclusions

The report presented a regularization approach focused on movie and user effect.The final model also included the combination of genres, and movie's released year as an effect that could further improve the predictive approach. Many models were presented and compared against a Naive-mean baseline model.Regularization is clearly a powerful approach taken for recommendation systems and other machine learning solutions. The objective of obtaining an RMSE lower than expectations was achieved. It is important to mention that a more precise genre effect could be estimated by separating the feature into different variables and really capturing the movies genres reality.Future work would include engineering and researching more features that could further capture users' behavior.

This work was heavily limited by computer power and time. Because of the size of this dataset and a repetitive approach, with several penalty calculations, the computer used for this calculations was restrained by its hardware and sofware capabilities.