Hussam Hanafi MovieLens Submission

Hussam Hanafi

10/24/2019

Introduction

This report will detail the work done to build a recommendation system using the 10M version of the MovieLens package.

In order to achieve this we will estimate a linear model using the various variables available to us in the data set.

The Data Set

The data set is comprised of ratings from 69,878 users across 10,677 movies defined by a movieId. It also includes the movie title, when the movie was rated as well as the genres it can be categorized by.

Below is a sample of the data set to illustrate:

Data Sample

userId	movieId	rating	timestamp	title	genres
605	172	2	857996769	Johnny Mnemonic (1995)	Action|Sci-Fi|Thriller
23285	260	4	938540022	Star Wars: Episode IV - A New Hope (a.k.a. Star Wars) (1977)	Action|Adventure|Sci-Fi
302	2334	4	981600729	Siege, The (1998)	Action|Thriller

We divide this data set into a Training and Validation set; and then further divide the training set to generate a training subset and test set that we will use to optimize any parameters.

Thus our validation set will only be used in the final prediction but not throughout the training process to avoid over-training.

Goal

The end goal is to generate a linear model where each variable represents an effect that can be extracted from the data. We will use the Root Mean Square Error (RMSE) as an indicator of the models success thus our target will be to minimize it.

Methodology

Throughout the report we will take each possible effect and visualize it to determine if it is worth including in our model.

We will use the mean rating of each effect as a base line and include it in our model.

Just the Mean

To start with we will calculate the mean of all the ratings as a base line for our model.

Thus our Model will be $Y = \hat{\mu} + \varepsilon$, with $\mu = 3.51257$ and RMSE:

Method	RMSE	Lamda
Just the average	1.0602	NA

The Movie Effect

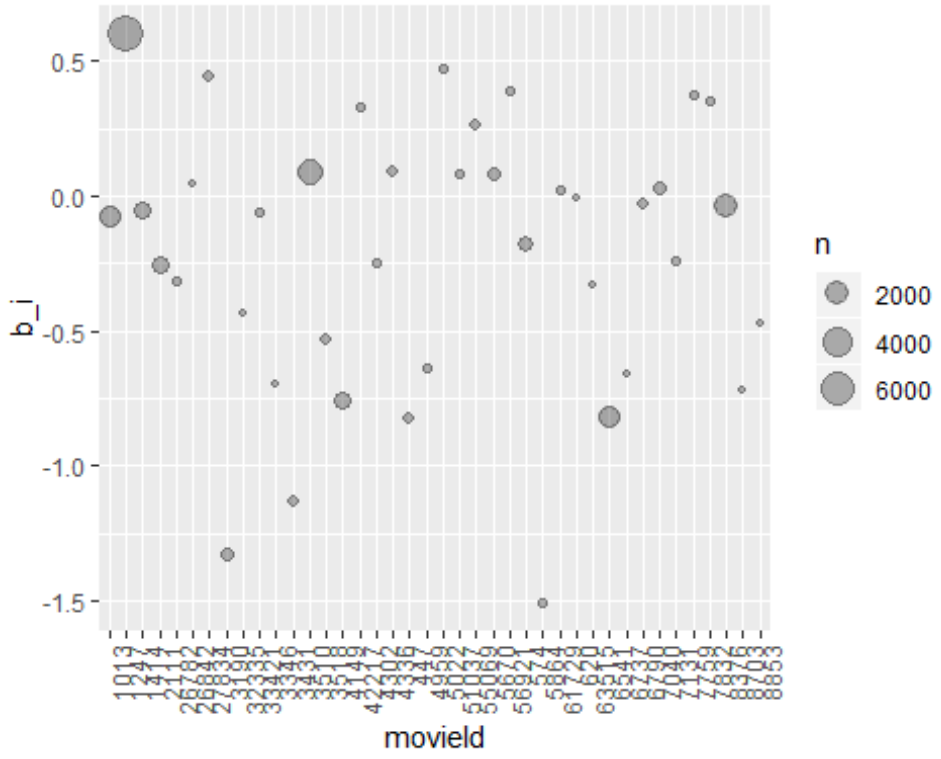
The first effect we will take into consideration is the movie effect b_i, or the average rating for each movie. This makes intuitive sense as some movies will be more popular or critically acclaimed than the rest.

To model the movie effect we will calculate the average rating for each movie regularized with a penalty term λ to minimize the effect of rarely rated movies on our average. therefore $\widehat{b_i} = \frac{1}{n+\lambda}\sum_{i=1}^{n_i}(Y_i - \mu)$

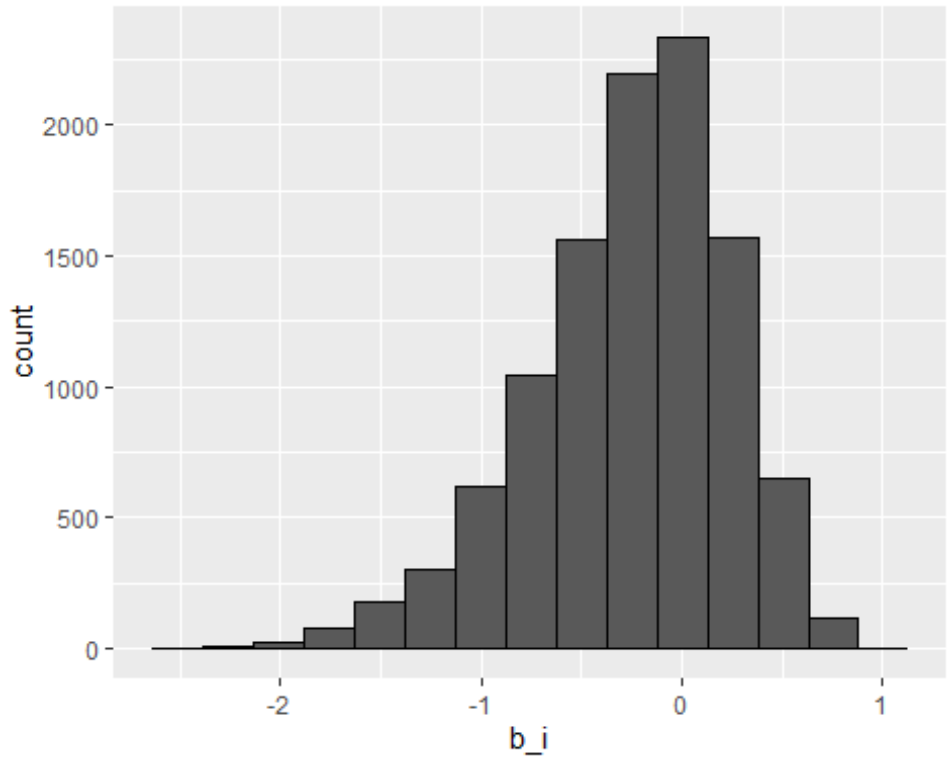
Thus our model now becomes $Y_i = \mu + b_i + \varepsilon$

Visualizations

A quick visualization of a sample of the movies shows that there is indeed variation across the different movies.



Below we can see how the averages are distributed:

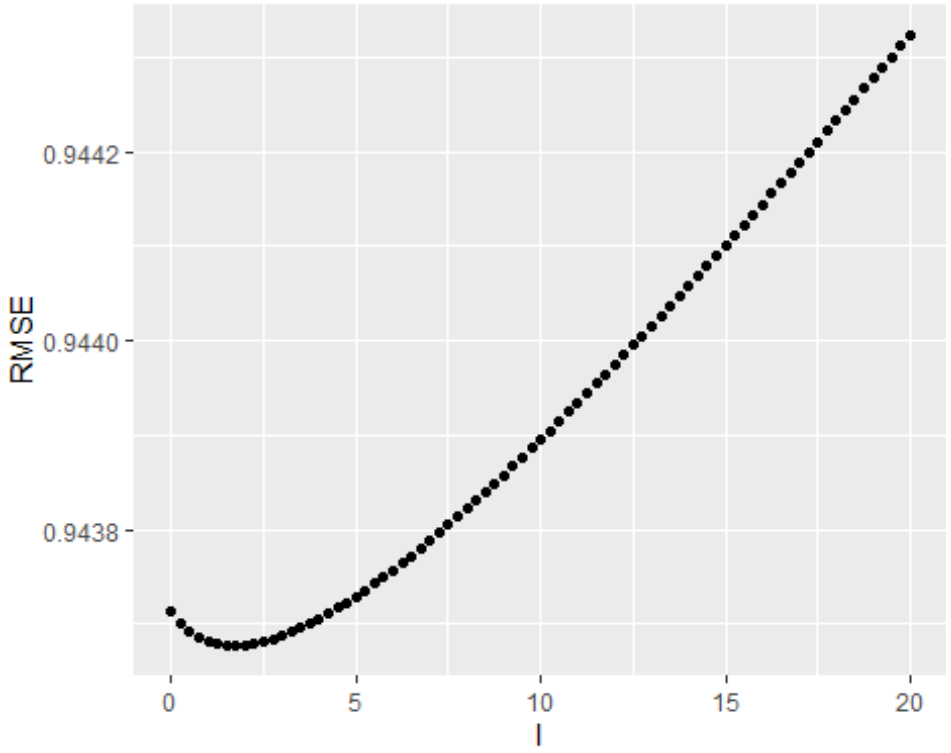


The Model

Now that we have confirmed the existence of the movie effect we can work on identifying the λ that optimizes the RMSE on our test set using the below code:

```
#Finding optimal Lamda Parameter
l <- seq(0,20,0.25)

RMSE_lamda <- map_df(l,function(l){

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

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

  data.frame(l = l, RMSE = RMSE(predicted_ratings,test$rating))
})
```

We can confirm that the optimal λ is within the range we trained for by visualizing it's effect on the RMSE:



Thus our optimal λ and respective RMSE is:

Method	RMSE	Lamda
Just the average	1.06024	NA
Mean + b_i	0.94368	1.75

The User Effect

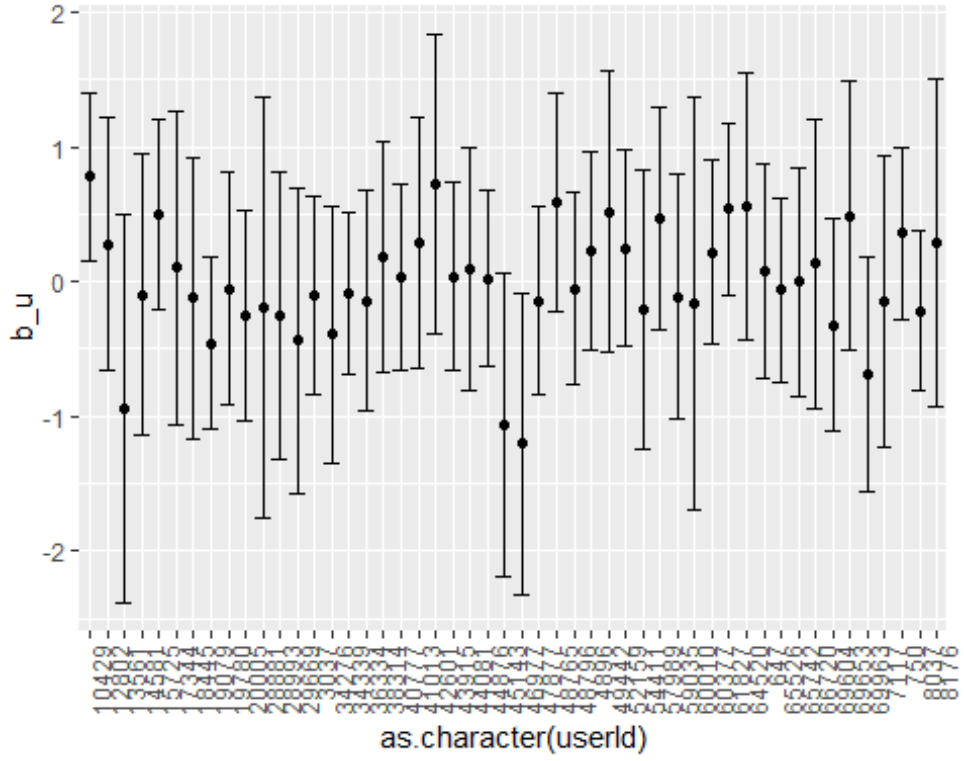
Next We will consider the User effect. Once again it would make intuitive sense that some users would be harsher critics than others.

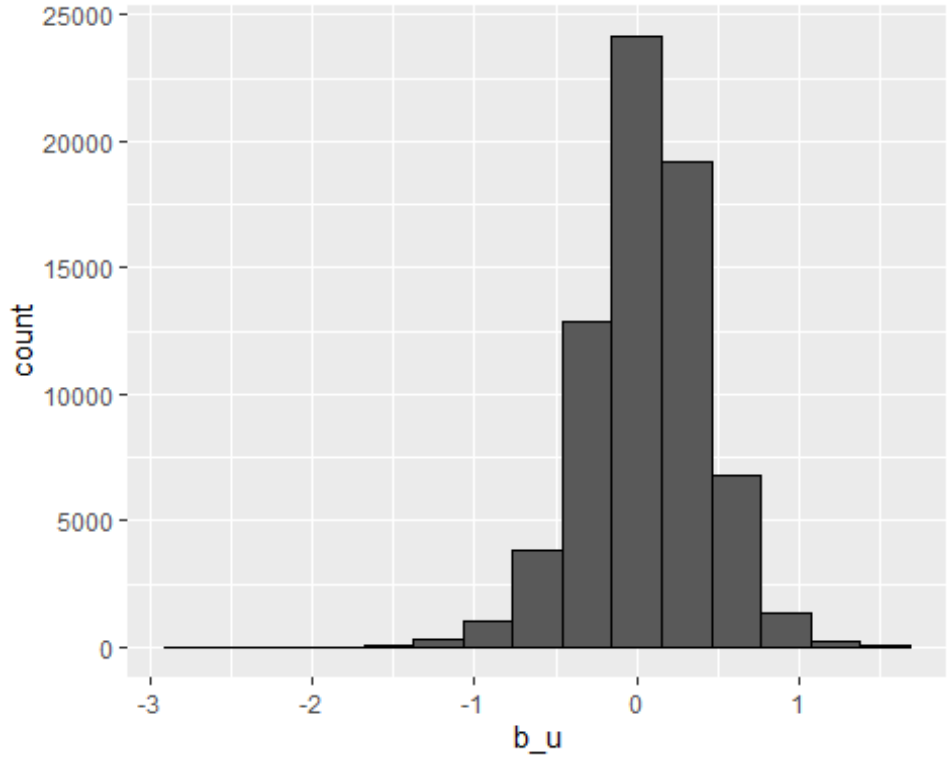
Once again we will calculate the average rating each user gives regularized with our parameter λ similar to what was done with the movie effect.

Our new model will be $Y_{iu} = \mu + b_i + b_u + \varepsilon$

Visualizations

A quick Scatter plot and histogram will confirm our intuition



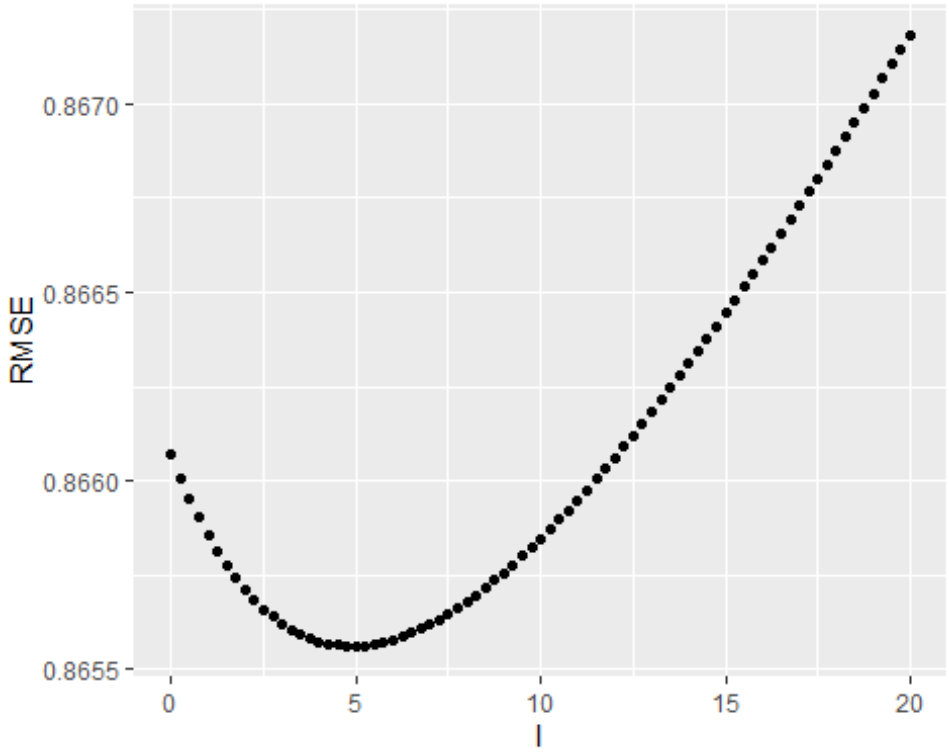


The Model

Our next step is to optimize our λ parameter for the User effect:

```
l <- seq(0,20,0.25)
RMSE_lamda <- map_df(l,function(l){

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

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

  data.frame(l = l, RMSE = RMSE(predicted_ratings,test$rating))
})
```

We confirm that our optimal λ is within the range we tested:



Thus our optimal λ and respective RMSE:

Method	RMSE	Lamda
Just the average	1.06024	NA
Mean + b_i	0.94368	1.75
Mean + b_i + b_u	0.86556	5.00

Genre Effect

Next We will consider the Genre effect. Intuitively some genres would be more popular than others.

The Genre variable is coded as a string containing all relevant genres the movie falls under:

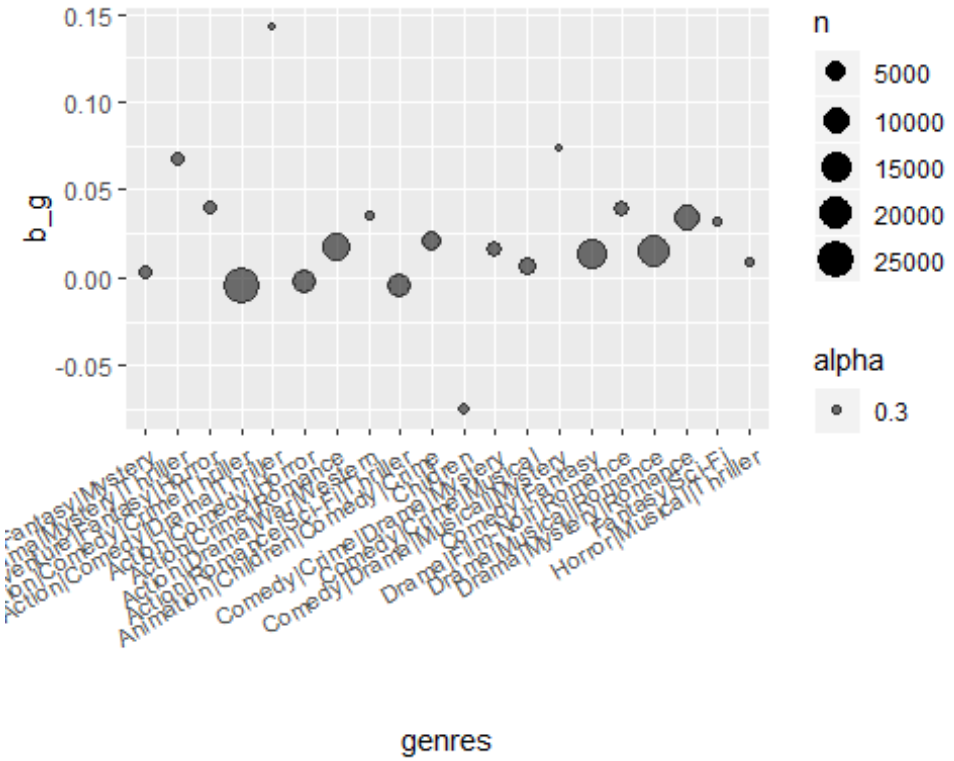
title	genres
Mission: Impossible 2 (2000)	Action|Thriller
Kingpin (1996)	Comedy
Naked Gun: From the Files of Police Squad!, The (1988)	Action|Comedy|Crime|Romance
Eternal Sunshine of the Spotless Mind (2004)	Comedy|Drama|Romance|Sci-Fi
Fugitive, The (1993)	Thriller

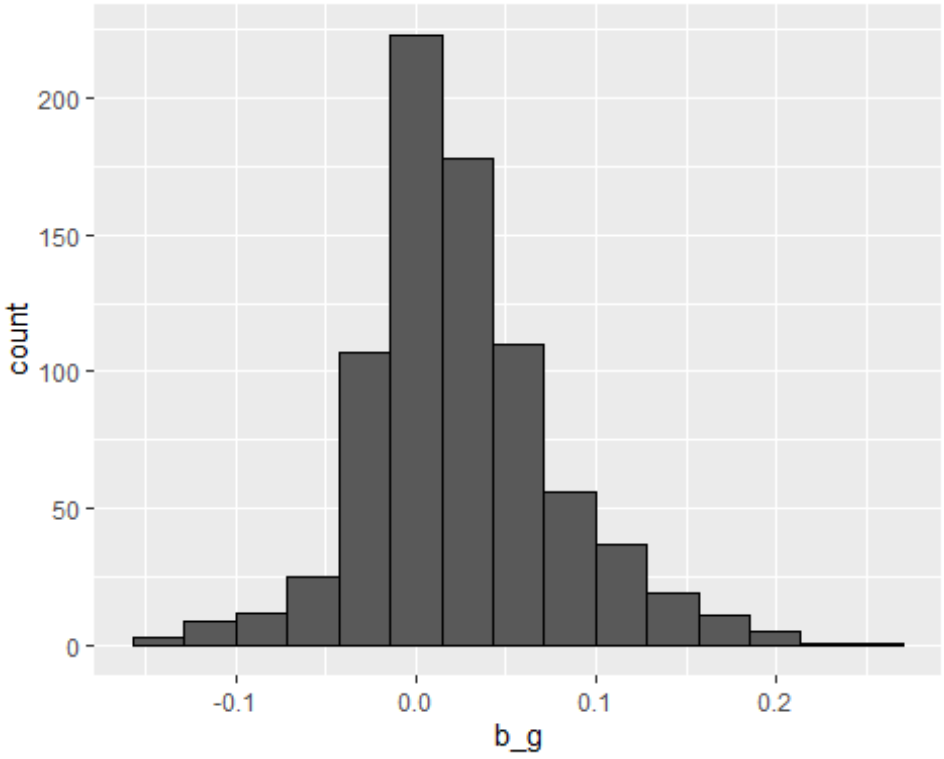
We will calculate the regularized average rating for each genre combination similar to what was done with the movie effect.

Our new model will be $Y_{iu} = \mu + b_i + b_u + b_g + \varepsilon$

Visualizations

Once again we confirm our intuition with some visualizations



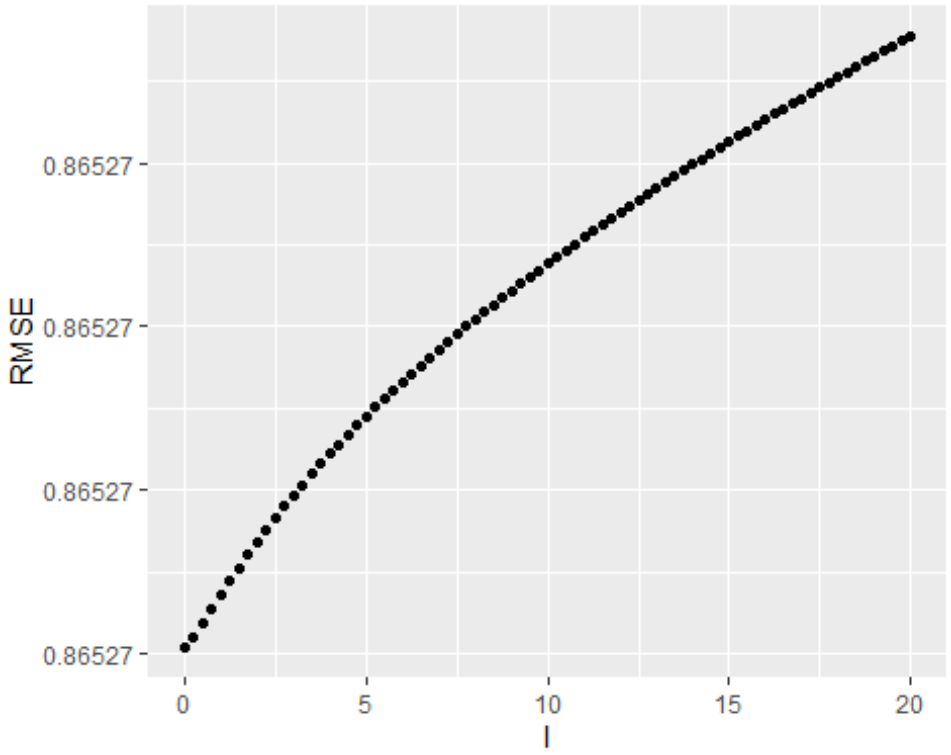


The Model

Now that we have confirmed a genre effect we can find our optimal λ and integrate the genre effect into our model.

```
l <- seq(0,20,0.25)
RMSE_lamda <- map_df(l,function(l){

  b_g <- train %>% left_join(b_i, by= "movieId") %>% left_join(b_u, by = "use
rId") %>% group_by(genres) %>%
    summarize(b_g = sum(rating - mu - b_i - b_u)/(n()+l))

  predicted_ratings <-  test %>% left_join(b_i, by = "movieId") %>% left_join
(b_u, by = "userId") %>%   left_join(b_g, by = "genres") %>%
    mutate(pred = mu + b_i + b_u + b_g) %>% pull(pred)

  data.frame(l = l, RMSE = RMSE(predicted_ratings,test$rating))
})
```

We confirm our optimal Lamda is in the range we tested for



Our Optimal Lamda and the respective RMSE

Method	RMSE	Lamda
Just the average	1.06024	NA
Mean + b_i	0.94368	1.75
Mean + b_i + b_u	0.86556	5.00
Mean + b_i + b_u + b_g	0.86527	0.00

Detailed Genre Effect

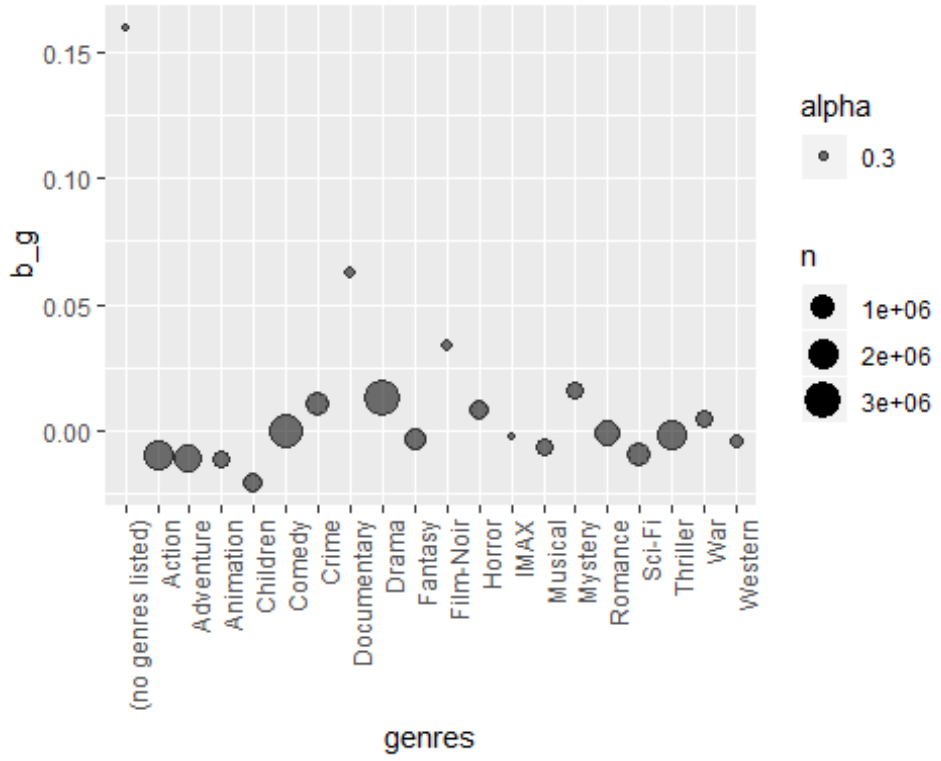
A closer look at the Genre Variable shows potential for improvement. As it is every observation of the genre variable is a combination of multiple possible genres. By splitting it into its components we can capture the effect of each individual genre; ending up with fewer possible observations each with a larger number of observations leading to a more robust estimation.

The below code achieves the split:

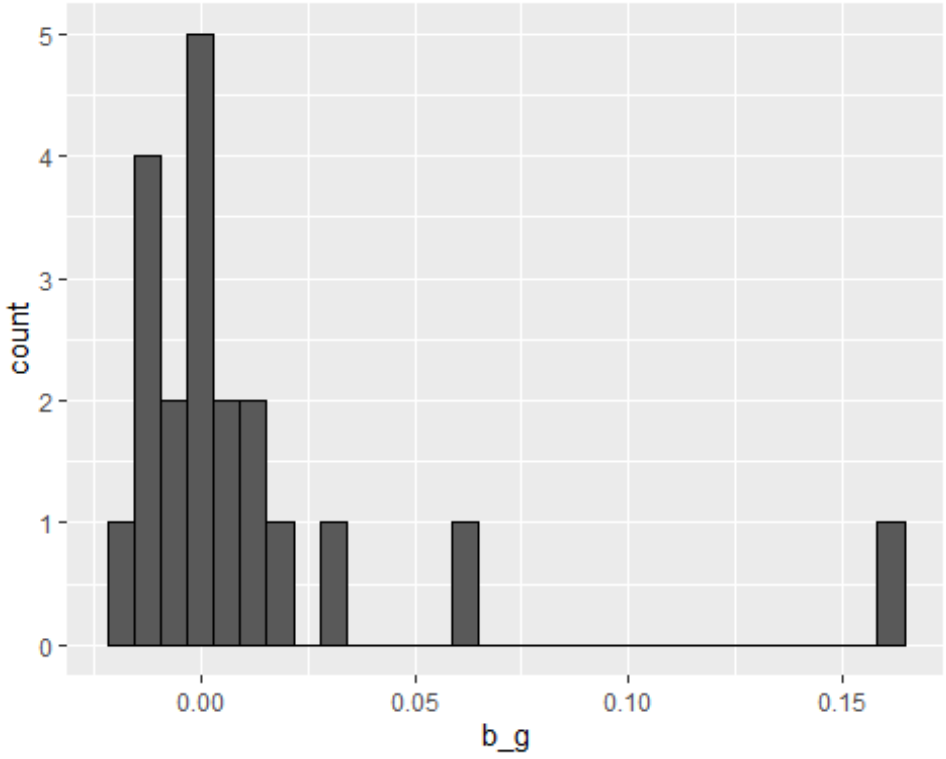
```
#Seperating the genres variable into its components in both sets
train <- train %>% separate_rows(genres, sep = "\\|")
test <- test %>% separate_rows(genres, sep = "\\|")
```

Visualizations

We can expect that this new version of genre variable will show us a confirmation of our assumptions



Our distribution also shows us that there are some clearer variations as well as some



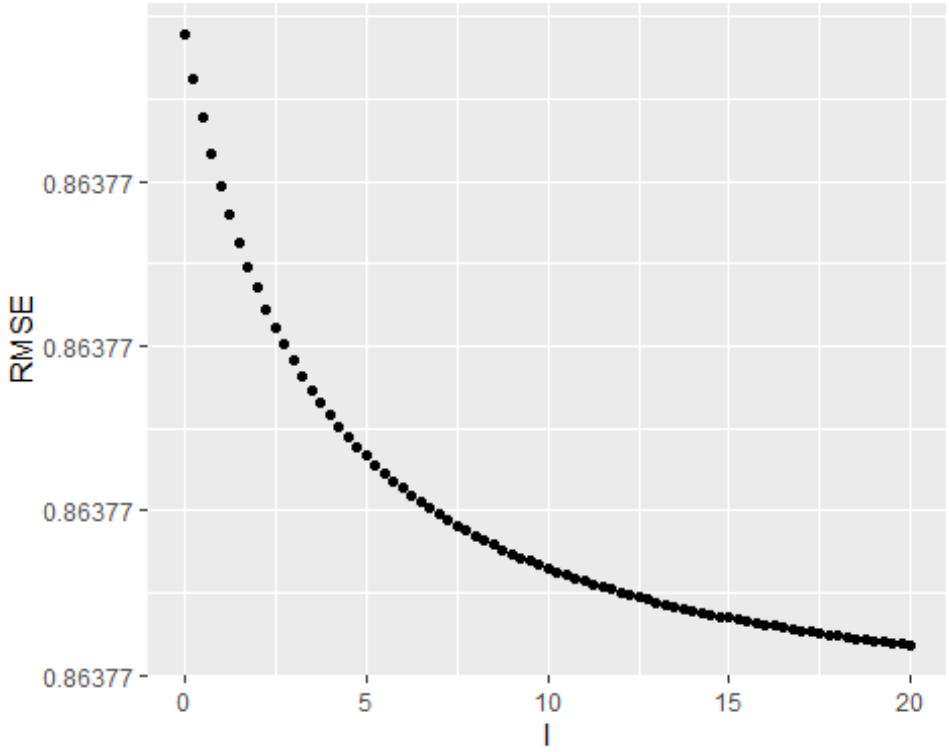
outliers

Model

We rebuild our model in the same way now that we have Wrangled our genre variable:

```
l <- seq(0,20,0.25)
RMSE_lamda <- map_df(l,function(l){

  b_g <- train %>% left_join(b_i, by= "movieId") %>% left_join(b_u, by = "use
rId") %>% group_by(genres) %>%
    summarize(b_g = sum(rating - mu - b_i - b_u)/(n()+l))

  predicted_ratings <-  test %>% left_join(b_i, by = "movieId") %>% left_join
(b_u, by = "userId") %>% left_join(b_g, by = "genres") %>%
    mutate(pred = mu + b_i + b_u + b_g) %>% pull(pred)

  data.frame(l = l, RMSE = RMSE(predicted_ratings,test$rating))
})
```

Confirm the optimization of our λ parameter:



And finally take a look at our results:

Method	RMSE	Lamda
Just the average	1.06024	NA
Mean + b_i	0.94368	1.75
Mean + b_i + b_u	0.86556	5.00
Mean + b_i + b_u + b_g	0.86527	0.00
mu + b_i + b_u + detailed_b_g	0.86377	20.00

Effect of the release year

Looking at the title variable we find that the release year of the movie is encoded as part of the character string as we can see below:

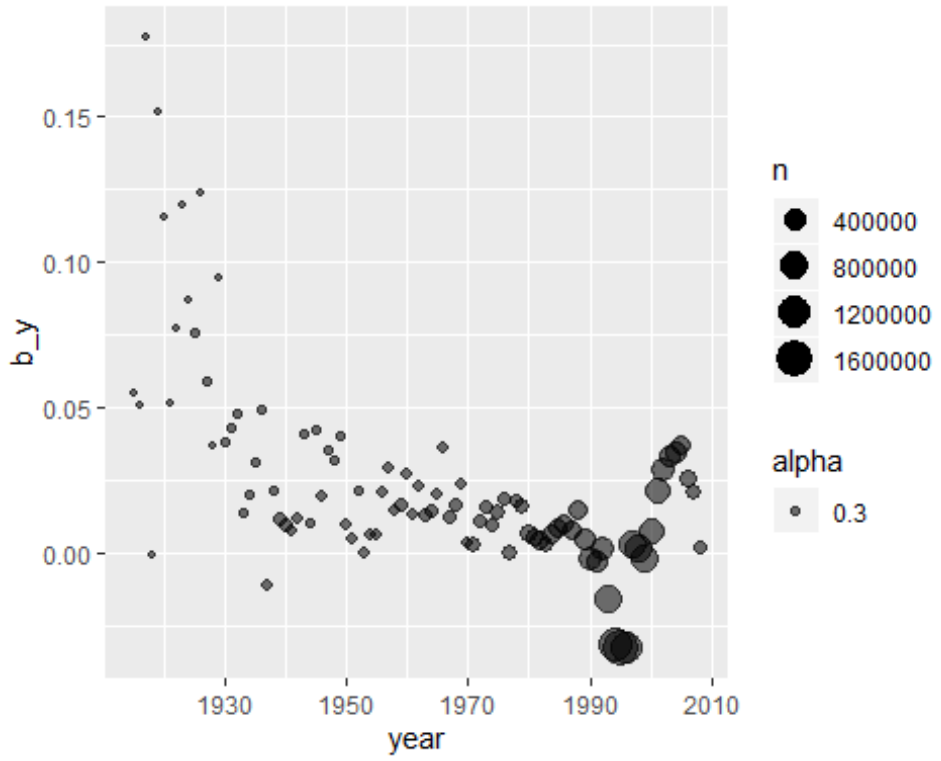
title
Intermission (2003)
Legends of the Fall (1994)
Lord of the Rings: The Fellowship of the Ring, The (2001)
Amadeus (1984)
Knight's Tale, A (2001)

We will analyze the year effect to see if it can add to our model, the first step is to wrangle the release year information from the title variable.

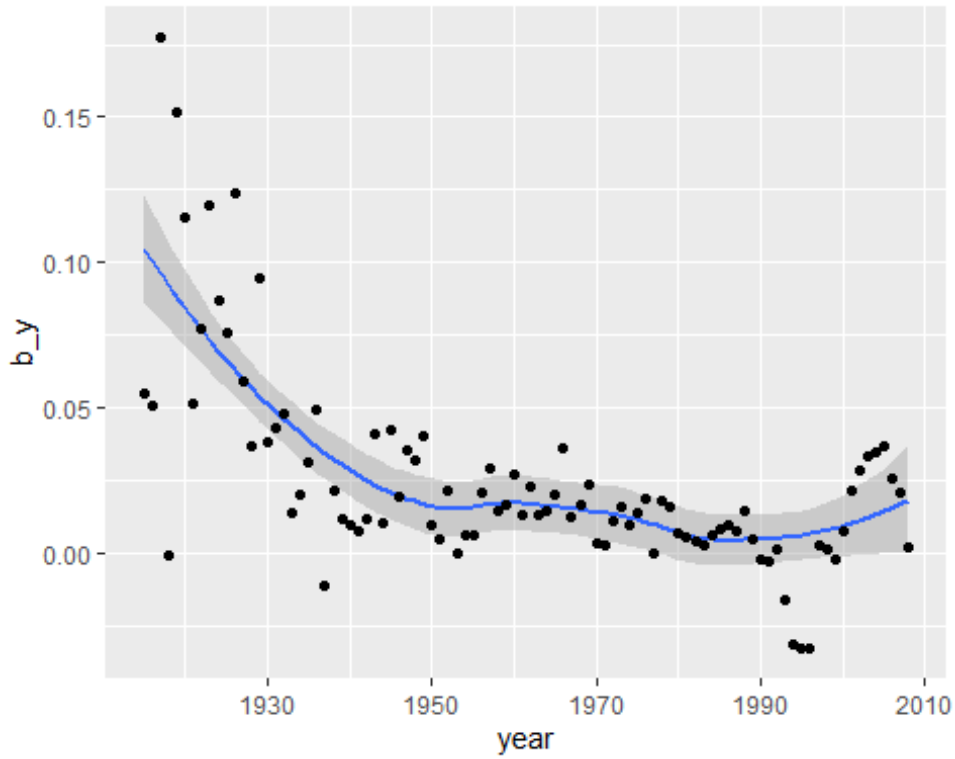
```
#Extracting the release of the Movie from the title
train <- train %>%
  mutate(year = str_extract(title,"\\(\\d{4}\\)")) %>%
  mutate(year = str_replace_all(year, "[//(//)]", ""))

test <- test %>%
  mutate(year = str_extract(title,"\\(\\d{4}\\)")) %>%
  mutate(year = str_replace_all(year, "[//(//)]", ""))
```

Visualizations

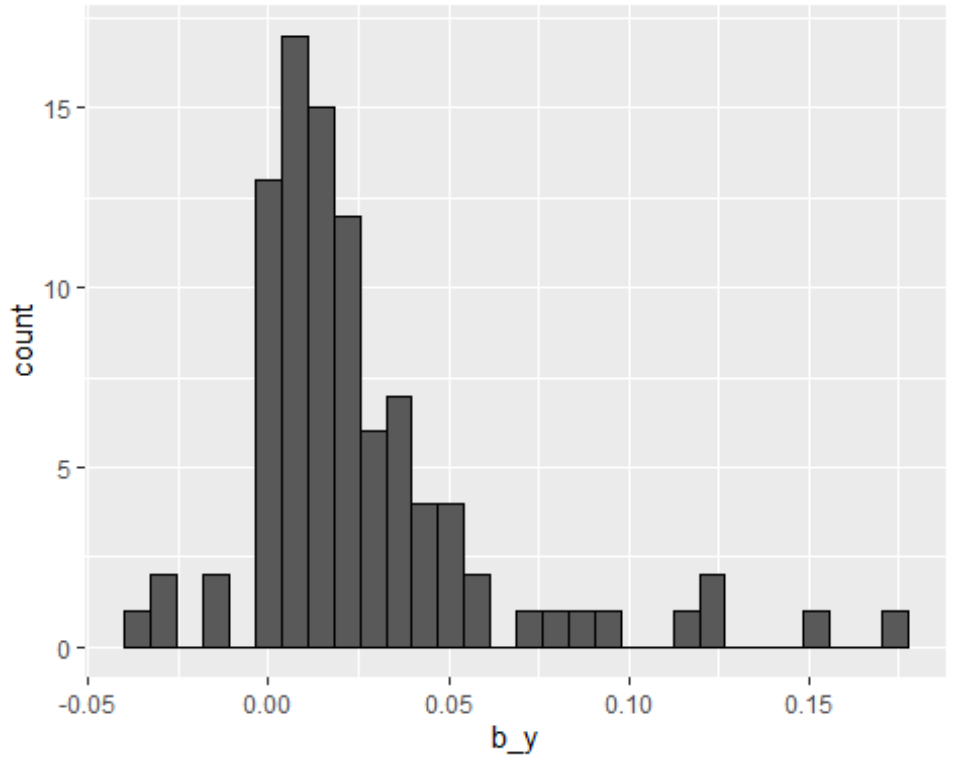
Visualizing the Scatter Plot shows that there appears to be a trend in how movies from different years are related:



This pattern is more visible in the below plot:



And in the below histogram:



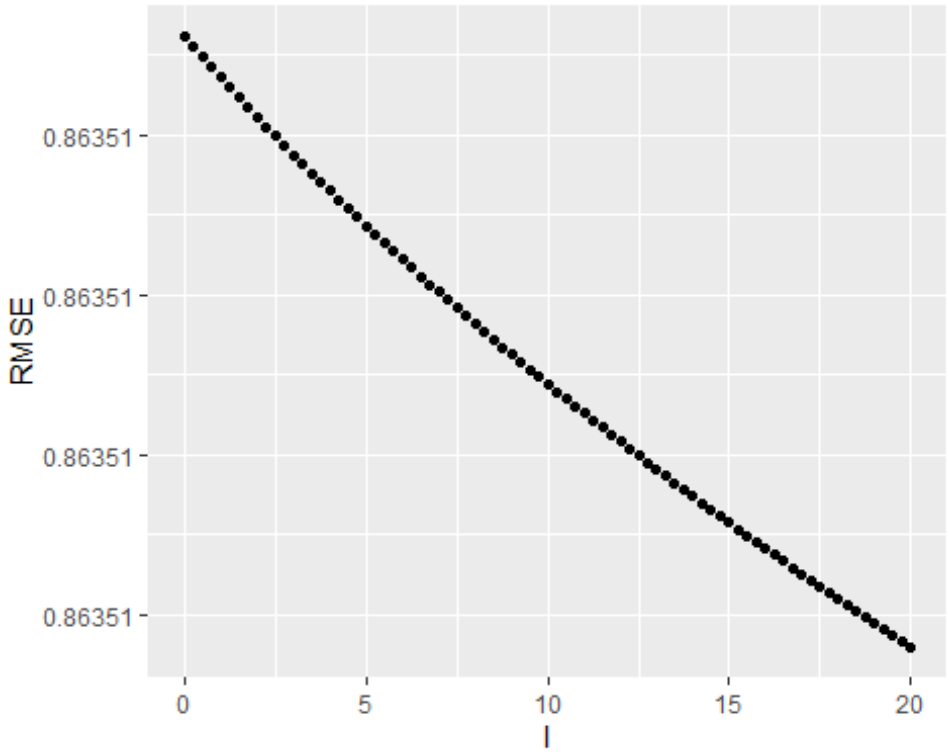
As such our new model will be $Y_{iu} = \mu + b_i + b_u + b_g + b_y + \varepsilon$

Model

We build our model in the same way as before:

```
l <- seq(0,20,0.25)
RMSE_lamda <- map_df(l,function(l){

  b_y <- train %>% left_join(b_i, by = "movieId") %>% left_join(b_u, by = "us
erId") %>% left_join(b_g, by = "genres") %>%
    group_by(year) %>%
    summarize(b_y = sum(rating - mu - b_i - b_u - b_g)/(n()+l))

  predicted_ratings <-  test %>%
    left_join(b_i, by = "movieId") %>% left_join(b_u, by = "userId") %>% left
_join(b_g, by = "genres") %>% left_join(b_y, by = "year") %>%
    mutate(pred = mu + b_i + b_u + b_g + b_y) %>% pull(pred)

  data.frame(l = l, RMSE = RMSE(predicted_ratings,test$rating))
})
```

Visualizing the optimal Lamda parameter:



Taking a look at our results:

Method	RMSE	Lamda
Just the average	1.06024	NA
Mean + b_i	0.94368	1.75
Mean + b_i + b_u	0.86556	5.00
Mean + b_i + b_u + b_g	0.86527	0.00
mu + b_i + b_u + detailed_b_g	0.86377	20.00
mu + b_i + b_u + detailed_b_g + b_y	0.86351	20.00

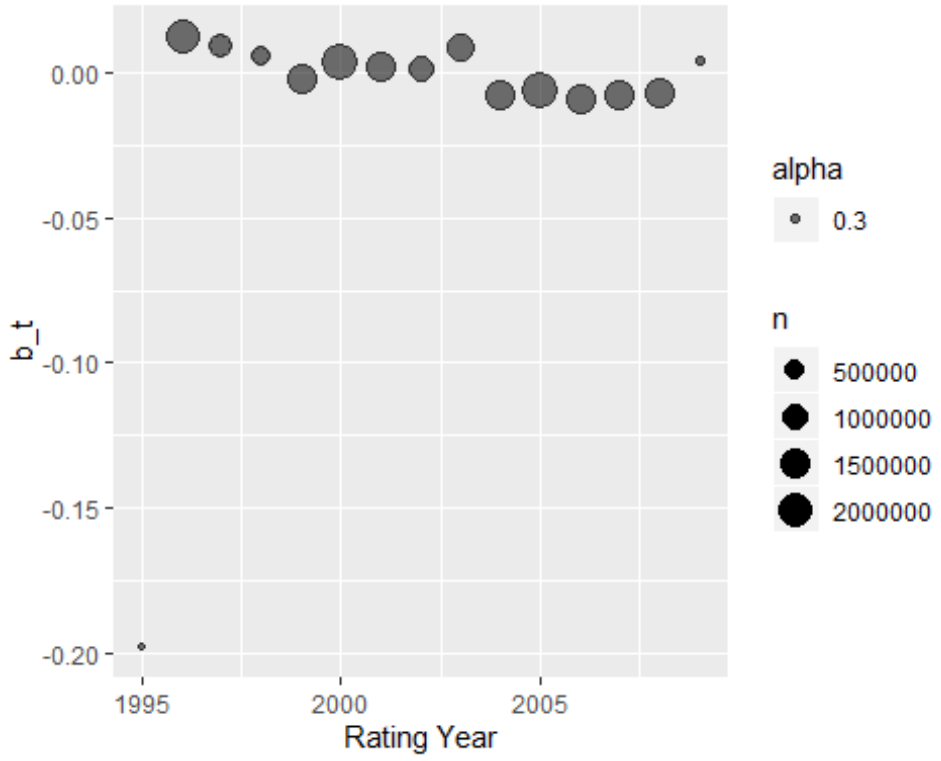
Effect of the review time

The Final piece of information that can be gleaned from the data is in the timestamp variable which encodes the time the rating was placed

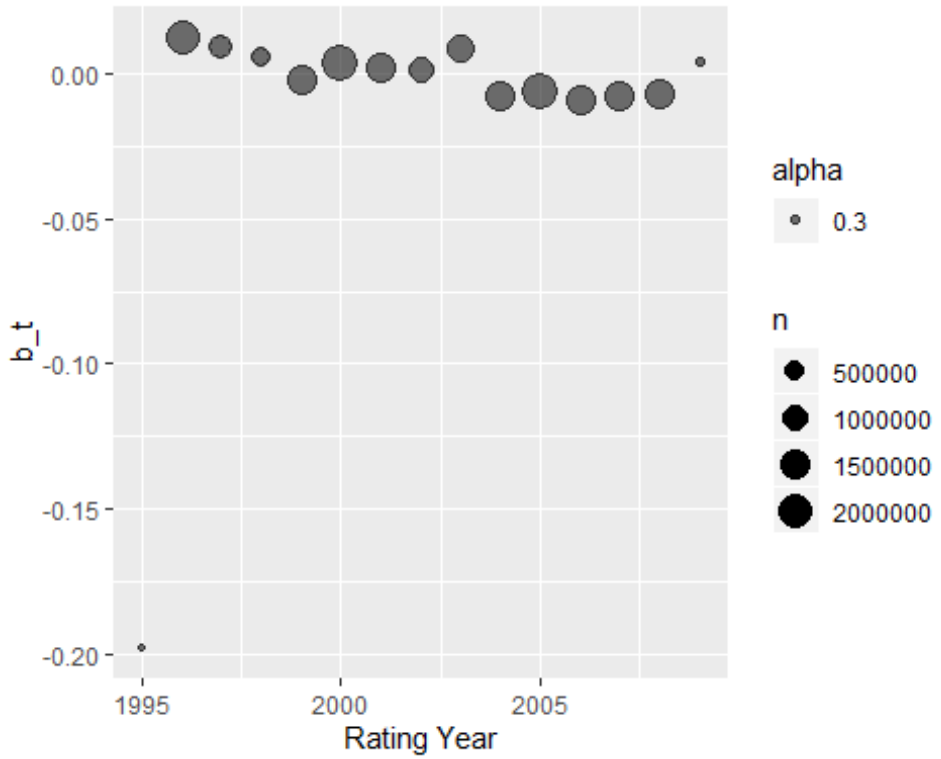
userId	timestamp
10171	2001-04-15 23:38:59
37175	2000-02-14 15:55:02
27447	2001-08-13 03:30:04

Visualizations

Visualizing the effect of the year the review was placed:



Visualizing the effect of the month:



If we continue to analyze based on different time-frames we will find that the effect continues to average around zero so we can ignore the timestamp variable in our model.

Results

Now That we have settled on our model $Y_{iu} = b_i + b_u + b_g + b_y + \varepsilon$ and saved the effects that optimize our RMSE we can build our final model and calculate the RMSE using the below code:

```
Model <- function(validation){
  validation <- validation %>% separate_rows(genres, sep = "\\|")
  validation <- validation %>%
    mutate(year = str_extract(title,"\\(\\d{4}\\)")) %>%
    mutate(year = str_replace_all(year, "[//(//)]", ""))

  validation %>% left_join(b_i, by = "movieId") %>%
    left_join(b_u, by = "userId") %>%
    left_join(b_g, by = "genres") %>%
    left_join(b_y, by = "year") %>%
    mutate(pred = mu + b_i + b_u + b_g + b_y) %>% pull(pred)
}

RMSE_final <- function(validation,pred){
  validation <- validation %>% separate_rows(genres, sep = "\\|")
  validation <- validation %>%
    mutate(year = str_extract(title,"\\(\\d{4}\\)")) %>%
    mutate(year = str_replace_all(year, "[//(//)]", ""))

  RMSE(validation$rating,pred)
}
```

Testing our model on our validation data:

```
pred <- Model(validation)
Model_RMSE <- RMSE_final(validation,pred)
```

Our Model returns a RMSE of 0.86316

Conclusion

We built our model by calculating regularized averages for the various possible effects that could be extracted from the data. With these effects; b_i, b_u, b_g, b_y, we were able to build a model with a RMSE of 0.86316.

Technical Limitations

The size of the data set makes it difficult if not impossible to run any Machine Learning algorithms on home based devices. More powerful Workstation environments or cloud based computing would allow for more complex machine learning that could improve our results

Limitations of the Data

There is also the possibility that the data is not capturing all the relevant effects. Additional Information on User demographics, more granular information on the movie could also further improve our model even without access to more powerful machine learning.