Capstone Project Choose Your Own - Anime Ratings

Arturo Inda

June 30, 2021

An introduction/overview/executive summary section that describes the dataset and variables, and summarizes the goal of the project and key steps that were performed. A methods/analysis section that explains the process and techniques used, including data cleaning, data exploration and visualization, any insights gained, and your modeling approach. At least two different models or algorithms must be used, with at least one being more advanced than linear or logistic regression for prediction problems. A results section that presents the modeling results and discusses the model performance. A conclusion section that gives a brief summary of the report, its potential impact, its limitations, and future work.

1. Introduction

For this project, I decided to use the data set **Anime-Planet Recommendation Database 2020** from Kaggle, uploaded on June 28, 2021 as I was looking if I could put a machine learning model on Anime movies, which is something I enjoy. We need to create a model that we choose, with certain characteristics. This dataset (attached on .csv) contains 1,048575 of ratings (column named **Rating**) numbered from 0 to 5, for Anime movies that hare identified by an id on the column **ProductId**. Finally, the individual rater username id is stored in column **username**. Finally, there is a column named **null**, that I use in order to test some formulas. Won't be used in the coding below. The goal of this project is to create a model that has a better rating via the Root Mean Square Error. I wont go into statistical details, but concentrate on the coding and results.

Before I begin, I need to install some packages we will use in the project: gdata, caret, dplyr and recosystem.

```
if(!require(gdata)) install.packages("gdata", repos = "http://cran.us.r-
project.org")

## Loading required package: gdata

## gdata: Unable to locate valid perl interpreter
## gdata:
## gdata: read.xls() will be unable to read Excel XLS and XLSX files
## gdata: unless the 'perl=' argument is used to specify the location of a
## gdata: valid perl intrpreter.
## gdata:
## gdata: (To avoid display of this message in the future, please ensure
## gdata: perl is installed and available on the executable search path.)
## gdata: Unable to load perl libaries needed by read.xls()
## gdata: to support 'XLX' (Excel 97-2004) files.

##

## gdata: Unable to load perl libaries needed by read.xls()
## gdata: to support 'XLSX' (Excel 2007+) files.

##

## gdata: Run the function 'installXLSXsupport()'
## gdata: to automatically download and install the perl
## gdata: libaries needed to support Excel XLS and XLSX formats.

##
## Attaching package: 'gdata'

## The following object is masked from 'package:stats':
##
##     nobs

## The following object is masked from 'package:utils':
##
##     object.size

## The following object is masked from 'package:base':
##
##     startsWith

if(!require(caret)) install.packages("caret", repos = "http://cran.us.r-
project.org")

## Loading required package: caret

## Loading required package: lattice

## Loading required package: ggplot2

if(!require(dplyr)) install.packages("dplyr", repos = "http://cran.us.r-
project.org")

## Loading required package: dplyr

##
## Attaching package: 'dplyr'

## The following objects are masked from 'package:gdata':
##
##     combine, first, last

## The following objects are masked from 'package:stats':
##
##     filter, lag
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

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

```
## Loading required package: recosystem
```

```
library(dplyr)
library(gdata)
temptable <- read.table("rating_complete.csv",
                 header = TRUE, sep = ",")

library(caret)
set.seed(1)
test_index <- createDataPartition(y = temptable$Rating, times = 1, p = 0.1,
                                  list = FALSE)
train_set <- temptable[-test_index,]
test_set <- temptable[test_index,]

test_set <- test_set %>%
  semi_join(train_set, by = "ProductId") %>%
  semi_join(train_set, by = "username")
```

We also have created a partition and with it we also created a few sets: a training set and a testing set, self explanatory on their functions. Also, the testing set will be performing the validation at the end of the coding. Let's see what this database contains:

```
# Analysis
head(temptable)
```

```
##   ï..null username ProductId Rating
## 1       0        0       147    5.0
## 2       0        1      1512    4.5
## 3       0        1       599    4.0
## 4       0        1      2292    4.5
## 5       0        1      1078    3.0
## 6       0        1      1633    4.5
```

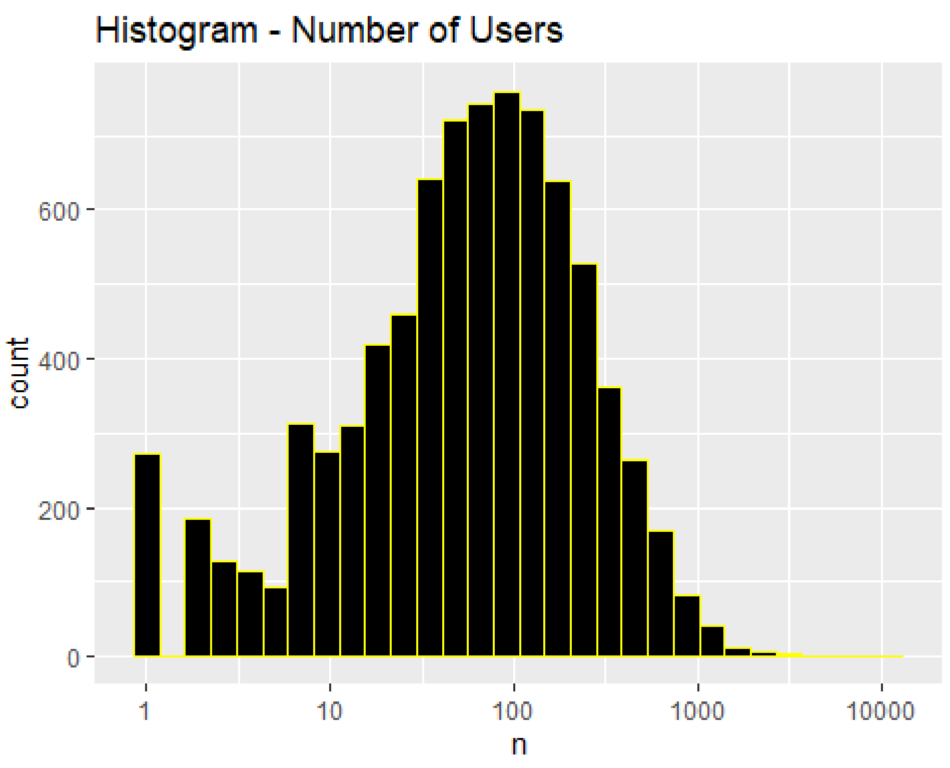
```
str(temptable)
```

```
## 'data.frame':    1048575 obs. of  4 variables:
##  $ ï..null  : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ username : int  0 1 1 1 1 1 1 1 1 1 ...
##  $ ProductId: int  147 1512 599 2292 1078 1633 623 84 2600 2684 ...
##  $ Rating   : num  5 4.5 4 4.5 3 4.5 3.5 2 3 4 ...
```

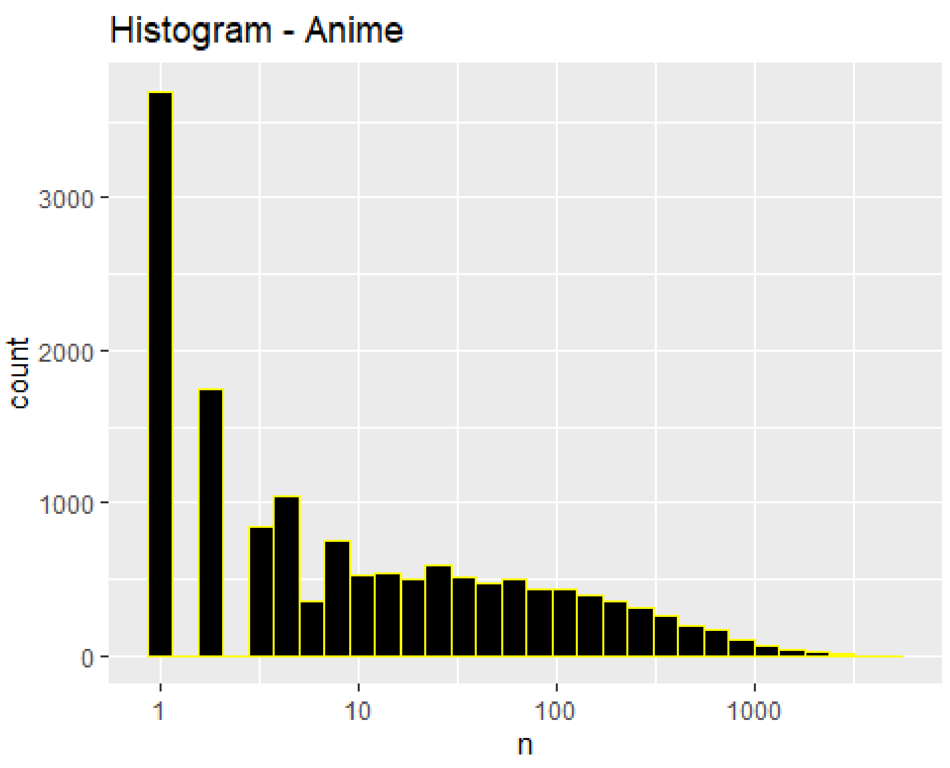
```
mean(temptable$Rating)
```

```
## [1] 3.813309
```

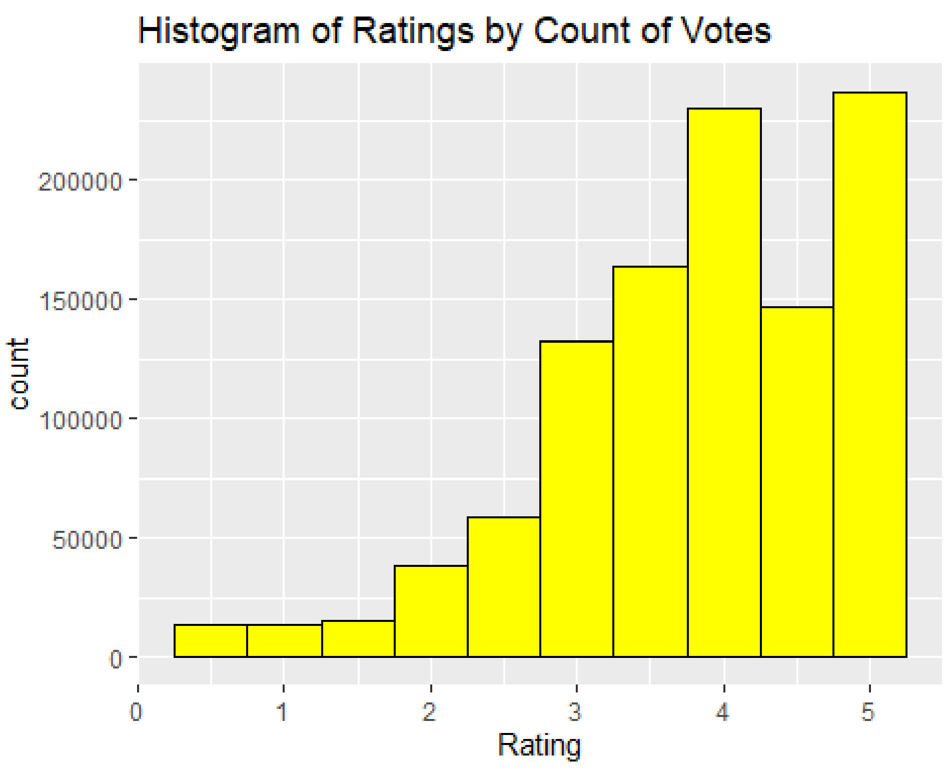
```
temptable %>% group_by(username) %>% summarize(count=n()) %>%
arrange(desc(count))

## # A tibble: 8,276 x 2
##    username count
##       <int> <int>
##  1     9055 11393
##  2     5693  5897
##  3     5546  3505
##  4     3724  3370
##  5     4826  2882
##  6     6852  2320
##  7     7404  2188
##  8     8729  2127
##  9     7561  2108
## 10      532  2096
## # ... with 8,266 more rows

summary(temptable)

##      ï..null     username       ProductId         Rating
##  Min.   :0   Min.   :   0   Min.   :    2   Min.   :0.500
##  1st Qu.:0   1st Qu.:2474   1st Qu.: 1496   1st Qu.:3.000
##  Median :0   Median :4844   Median : 3751   Median :4.000
##  Mean   :0   Mean   :4850   Mean   : 4447   Mean   :3.813
##  3rd Qu.:0   3rd Qu.:7293   3rd Qu.: 6705   3rd Qu.:4.500
##  Max.   :0   Max.   :9575   Max.   :17294   Max.   :5.000

mean(temptable$Rating)

## [1] 3.813309

temptable %>% count(username) %>%
  ggplot(aes(n)) +
  geom_histogram(bins = 30, color = "yellow", fill="black") +
  scale_x_log10() +
  ggtitle("Histogram - Number of Users")
```



```
temptable %>% count(ProductId) %>%
  ggplot(aes(n)) +
  geom_histogram(bins = 30, color = "yellow", fill="black") +
  scale_x_log10() +
  ggtitle("Histogram - Anime")
```



```
ggplot(temptable, aes(Rating)) + geom_histogram(binwidth=0.5, color="black",
fill="yellow") + ggtitle("Histogram of Ratings by Count of Votes")
```



2. Analysis

Interesting information. We can observe a high distribution on the right side of the ratings on the table "Histogram of Ratings by Count of Votes", with a mean of 3.81 on ratings (this will be useful later). Also, we see the majority of the users being the ones that have rated 1o movies or more, and that we have a total of 9575 users that have rated 17294 movies 1,048,575. Our test set will have 104,362 observations (10% of total) and our training set 943,715 (90% of total). Let's begin with our first model. Here is the code:

```
#Define function for RSME
RMSE <- function(Realratings, Predictedratings){
  sqrt(mean((Realratings - Predictedratings)^2))
}


#Ratings average
murating <- mean(temptable$Rating)
murating

## [1] 3.813309

#First Model
FirstRMSE <- RMSE(test_set$Rating, murating)
FirstRMSE

## [1] 1.002447

Comparativetable <- tibble(Method = "First Model = Mu Rating", RMSE =
FirstRMSE)
Comparativetable %>% knitr::kable()
```

Method	RMSE
First Model = Mu Rating	1.002447

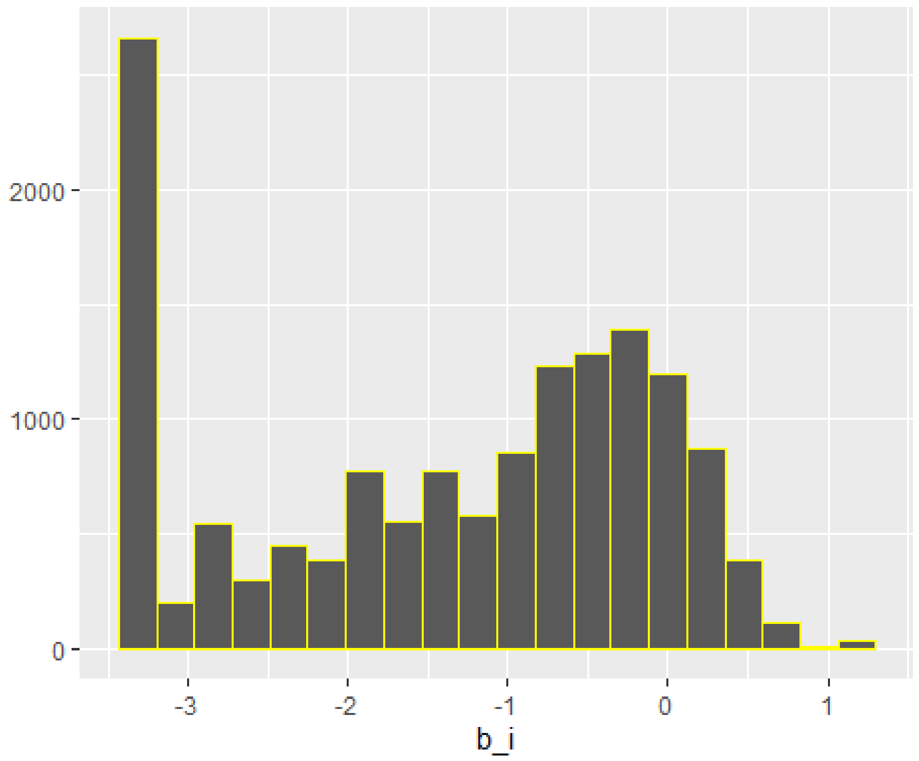
3. Results

First, we define our RSME function, this is the number we want to look at, and want to see improving (decreasing) with each model. This is the square root of the averages of the differences of the Real Ratings on the data set vs what we want to predict as a rating. We define **murating** as the mean of whole set (we know is 3.81), then do the RSME of the Rating column of the test set vs our murating.

We have seen we obtained a RSME of 1.00, perfect for our analysis (to calculate % reduction on RSME). This will be our RSME baseline.

We saw that there are a lot of ratings for only 17K movies, so we can assume that there will be movies in the right tail that will have only a few ratings and thus can be looked as outlier. We can say this is an Anime bias on our Anime. Let's define a new mean from the training set, and lets also calculate the bias, **b_i**, defined as the mean of the difference of each individual rating vs this new mu. Also we look at prodavgs, a new data frame with the result of each b_i. Next, we define our Predicted ratings numbers now with the movie bias on it and compare with our previous RSME.

```
#Second Model Anime Interaction Bias
newmu <- mean(train_set$Rating)
newmu

## [1] 3.813584

prodavgs <- train_set %>%
  group_by(ProductId) %>%
  summarize(b_i = mean(Rating - newmu))


str(prodavgs)

## tibble[,2] [14,587 x 2] (S3: tbl_df/tbl/data.frame)
##  $ ProductId: int [1:14587] 2 3 4 5 6 7 8 10 11 12 ...
##  $ b_i      : num [1:14587] -0.1553 -0.5853 -0.1288 0.091 0.0957 ...

head(prodavgs)

## # A tibble: 6 x 2
##   ProductId     b_i
##       <int>   <dbl>
## 1         2 -0.155
## 2         3 -0.585
## 3         4 -0.129
## 4         5  0.0910
## 5         6  0.0957
## 6         7 -0.341

qplot(b_i, data = prodavgs, bins = 20, color = I("yellow"))
```

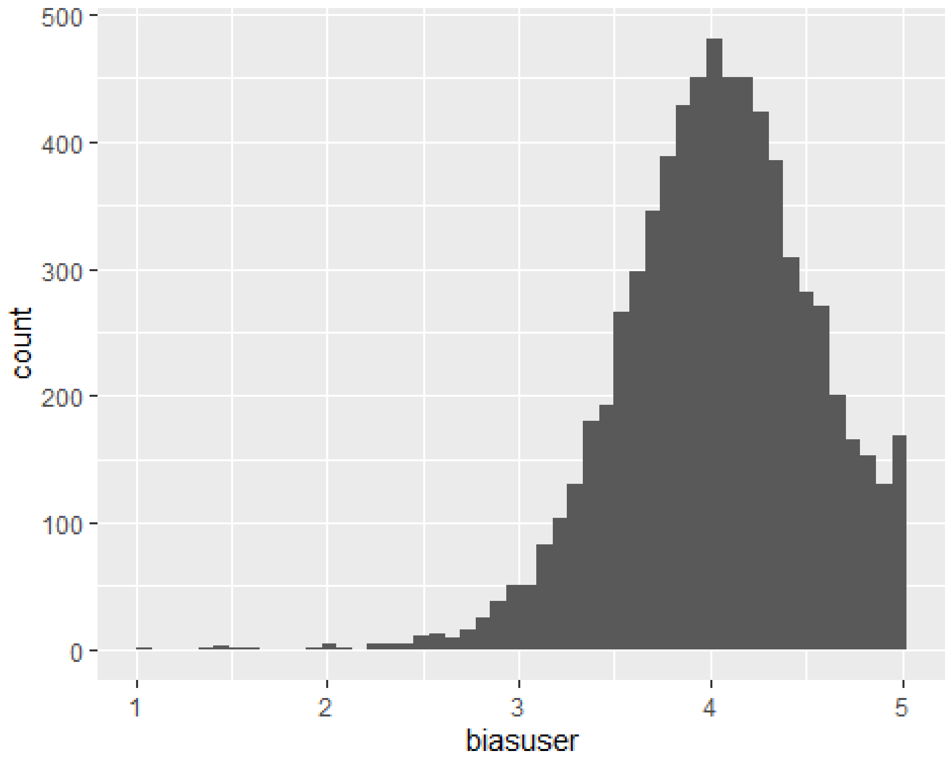


```
Predratings <- newmu + test_set %>%
  left_join(prodavgs, by = 'ProductId') %>%
  pull(b_i)

head(Predratings)

## [1] 3.808989 4.339014 3.690476 4.086053 3.271255 3.976852

SecondRMSE <- RMSE(test_set$Rating, Predratings)
SecondRMSE

## [1] 0.8981843

Comparativetable <- bind_rows(Comparativetable, tibble(Method = "Second Model
= LEAST SQUARE Product (Anime) Bias", RMSE = SecondRMSE))
Comparativetable %>% knitr::kable()
```

Method	RMSE
First Model = Mu Rating	1.0024473
Second Model = LEAST SQUARE Product (Anime) Bias	0.8981843

0.8925, a great reduction of 10% on the RSME. We can still do better, lets add also a rater bias, there are a lot of usernames that are rating a lot of movies, so they could also have a bias. Let's focus on the raters of 10 animes or more. We will follow the same logic as before, creating **userinteraction as our new set**, and **Predictedratings** the new data with both Anime and rater biases.

```r
#Third Model User Interaction Bias
train_set %>%
  group_by(username) %>%
  filter(n()>=10) %>%
  summarize(biasuser = mean(Rating)) %>%
  ggplot(aes(biasuser)) +
  geom_histogram(bins = 50)
```



```r
userinteraction <- train_set %>%
  left_join(prodavgs, by='ProductId') %>%
  group_by(username) %>%
  summarize(biasuser = mean(Rating - newmu - b_i))

head(userinteraction)
```

```
## # A tibble: 6 x 2
##   username biasuser
##      <int>    <dbl>
## 1        0   0.988
## 2        1  -0.0142
## 3        2  -0.348
## 4        3   0.621
## 5        4  -0.576
## 6        5   0.953
```

```r
Predictedratings <- test_set %>%
  left_join(prodavgs, by='ProductId') %>%
  left_join(userinteraction, by = 'username') %>%
  mutate(predictor = newmu + b_i + biasuser) %>%
  pull(predictor)

head(Predictedratings)
```

```
## [1] 3.794822 4.324847 3.676309 4.071886 3.257088 3.962685
```

```
ThirdRMSE <- RMSE(Predictedratings, test_set$Rating)
ThirdRMSE
```

```
## [1] 0.7641973
```

```
Comparativetable <- bind_rows(Comparativetable, tibble(Method = "Third Model
= LEAST SQUARE Anime + UserId bias", RMSE = ThirdRMSE))
Comparativetable %>% knitr::kable()
```

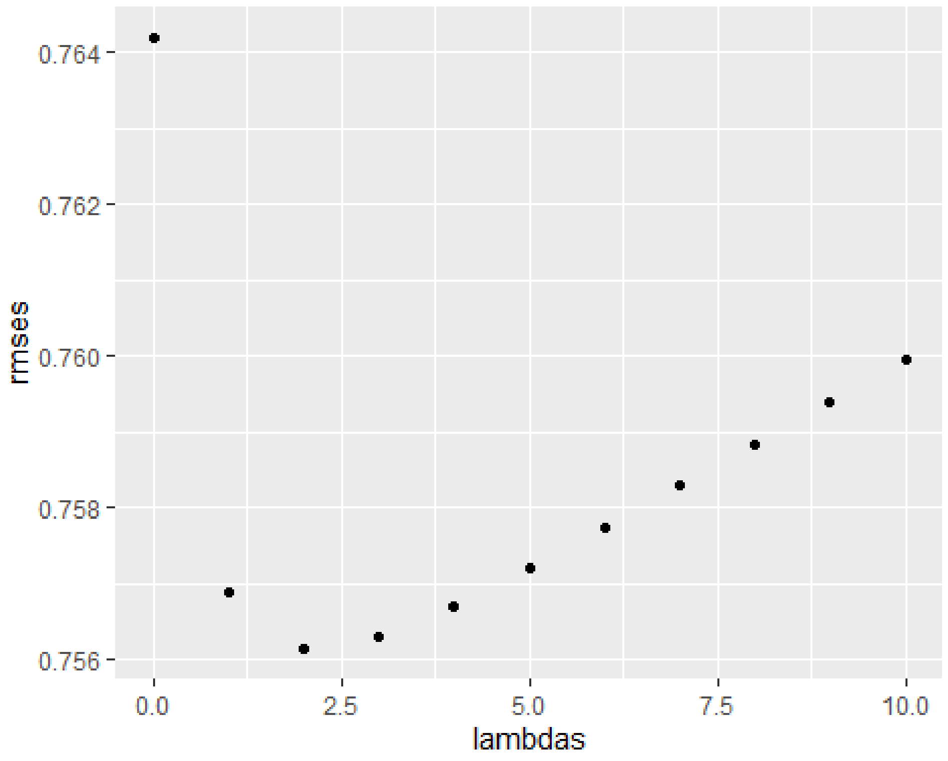
Method	RMSE
First Model = Mu Rating	1.0024473
Second Model = LEAST SQUARE Product (Anime) Bias	0.8981843
Third Model = LEAST SQUARE Anime + UserId bias	0.7641973

Our RSME is 0.76, a 24% reduction. We now will use regularization. Regularization permits us to penalize large estimates that are formed using small sample sizes.The idea is that we are constraining the total variability of the effect sizes of the model. We will also look at penalties, defined by lambda, if the number of ratings is large, lambda is ignored, but if ratings set is small, estimate of b_i(lambda) will make it towards 0. This is the penalized estimates version of the model, compared to the previous least square estimates.

```
#4th model
lambda <- 2
mu <- mean(train_set$Rating)
prod_reg_avgs <- train_set %>%
  group_by(ProductId) %>%
  summarize(b_i = sum(Rating - mu)/(n()+lambda), n_i = n())

predicted_ratings <- test_set %>%
  left_join(prod_reg_avgs, by = "ProductId") %>%
  mutate(pred = mu + b_i) %>%
  pull(pred)
RMSE(predicted_ratings, test_set$Rating)
```

```
## [1] 0.8989743
```

```
lambdas <- seq(0, 10, 1)

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

  mu <- mean(train_set$Rating)
  b_i <- train_set %>%
    group_by(ProductId) %>%
    summarize(b_i = sum(Rating - mu)/(n()+l))

  b_u <- train_set %>%
    left_join(b_i, by="ProductId") %>%
    group_by(username) %>%
    summarize(b_u = sum(Rating - b_i - mu)/(n()+l))

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

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

qplot(lambdas, rmses)
```



```
lambdas[which.min(rmses)]

## [1] 2
```

```
FourthRMSE <- min(rmses)
FourthRMSE

## [1] 0.7561464

Comparativetable <- bind_rows(Comparativetable, tibble(Method = "Fourth Model
= PENALTY ESTIMATE Regularized Anime + User Id", RMSE = FourthRMSE))
Comparativetable %>% knitr::kable()
```

Method	RMSE
First Model = Mu Rating	1.0024473
Second Model = LEAST SQUARE Product (Anime) Bias	0.8981843
Third Model = LEAST SQUARE Anime + UserId bias	0.7641973
Fourth Model = PENALTY ESTIMATE Regularized Anime + User Id	0.7561464

We obtain a lambda of 2, which we use in the code as the optimal. Our RSME for this fourth model is 0.75, which is a 25% improvement. Now let's validate in the testing set this model and see the result.

```
#Validation on Validation Set
library(recosystem)
set.seed(1, sample.kind="Rounding")

## Warning in set.seed(1, sample.kind = "Rounding"): non-uniform 'Rounding'
sampler
## used

tr_reco <- with(train_set, data_memory(user_index = username, item_index =
ProductId, rating = Rating))
validation_reco <- with(test_set, data_memory(user_index = username,
item_index = ProductId, rating = Rating))
r <- Reco()

r$train(tr_reco)

## iter      tr_rmse          obj
##    0       0.9848   1.6344e+06
##    1       0.7763   1.2846e+06
##    2       0.7659   1.2700e+06
##    3       0.7579   1.2633e+06
##    4       0.7470   1.2557e+06
##    5       0.7379   1.2481e+06
##    6       0.7318   1.2429e+06
##    7       0.7275   1.2385e+06
##    8       0.7245   1.2358e+06
##    9       0.7223   1.2339e+06
##   10       0.7205   1.2324e+06
##   11       0.7191   1.2311e+06
##   12       0.7179   1.2300e+06
##   13       0.7169   1.2291e+06
##   14       0.7161   1.2282e+06
##   15       0.7154   1.2279e+06
##   16       0.7148   1.2276e+06
##   17       0.7142   1.2265e+06
##   18       0.7138   1.2263e+06
##   19       0.7133   1.2263e+06
```

```
final_reco <- r$predict(validation_reco, out_memory())
Validated_RMSE <- RMSE(final_reco, test_set$Rating)
Validated_RMSE
```

```
## [1] 0.7425802
```

```
Comparativetable <- bind_rows(Comparativetable, tibble(Method = "Fifth Model
= OPTIMIZED MODEL on Validation Set", RMSE = Validated_RMSE))
Comparativetable %>% knitr::kable()
```

Method	RMSE
First Model = Mu Rating	1.0024473
Second Model = LEAST SQUARE Product (Anime) Bias	0.8981843
Third Model = LEAST SQUARE Anime + UserId bias	0.7641973
Fourth Model = PENALTY ESTIMATE Regularized Anime + User Id	0.7561464
Fifth Model = OPTIMIZED MODEL on Validation Set	0.7425802

0.74, an improvement as well from our testing set, but shows a validation of this model on another set.

4. Conclusion

These shows how using 2 types of analysis, least squares and penalized estimates can give you a great RSME reduction and a trusting model. Now let's validate in the testing set this model and see the result. First model, the mean method, gave us a RSME of 1. We looked at Anime and Rater interaction (columns ProductId and username) and used least squares estimates which gave us RSMEs of 0.89 and 0.76, and finalized with a 0.75 with the regularized model using penalized estimates (lambda) and showed the set with a RSME of 0.74, which validates the model reduction in 25% approximately.