HarvardX Capstone CYO Project

Ernest Kollieguwor

7/28/2021

Preface

The capstone project of HarvardX's Data Science Professional Certificate program on the Edx's website served the basis for this report. The R Markdown code used to generate the report and its PDF version are available on GitHub. HTML version may also be available on RPubs.

Introduction

A user is able to predict the rating or other preferences of a given item using a subclass information filtering system called a "Recommendation System". Customers rating is used by companies with huge customers group to predict their rating or preferences of their products. Similar to the MovieLens project, people can check out personalized recommendations and find out books that is good for them. This dataset contain 10,000 books and 50,000+ users. Ratings are 1 - 5 and each users rated at least 2 books. The dataset comes from a free social cataloging website, Goodreads, that allows individuals to freely search its database of books, annotations, reviews and ratings.

Goal of the Project

I hope to perform data analytics on similar dataset like the Movielens, with the intend to show a comparative results derived from methods for the Root Mean Squared Error (RMSE) and the Mean Absolute Error (MAE), and probably achieve same RMSE target for the Movielens

Data set

The Goodbooks Data set will be used for this project. The Github User Content collected this data set and it can be found at this web site (https://raw.githubusercontent.com/zygmuntz/goodbooks-10k/master/ratings.csv).

Loading the Data set

The course structure provided experience on coding, which will be used to download and re-define the Goodbooks data into an cyo set, as I defined it and 10% validation set. The cyo data set as I defines it, will be further split into a training set and testing set and the final evaluation will be made on the validation set.

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

```
## Loading required package: tidyverse

## ── Attaching packages ─────────────────────────────────────── tidyverse
1.3.1 ──

## ✓ ggplot2 3.3.5     ✓ purrr   0.3.4
## ✓ tibble  3.1.2     ✓ dplyr   1.0.7
## ✓ tidyr   1.1.3     ✓ stringr 1.4.0
## ✓ readr   1.4.0     ✓ forcats 0.5.1

## ── Conflicts ────────────────────────────────────────
tidyverse_conflicts() ──
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()

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

## Loading required package: caret

## Loading required package: lattice

##
## Attaching package: 'caret'

## The following object is masked from 'package:purrr':
##
##     lift

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

## Loading required package: data.table

##
## Attaching package: 'data.table'

## The following objects are masked from 'package:dplyr':
##
##     between, first, last

## The following object is masked from 'package:purrr':
##
##     transpose

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

## Loading required package: recosystem

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

```
## Loading required package: ggthemes

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

## Loading required package: scales

##
## Attaching package: 'scales'

## The following object is masked from 'package:purrr':
##
##     discard

## The following object is masked from 'package:readr':
##
##     col_factor

library(tidyverse)
library(caret)
library(data.table)
library(recosystem)
library(knitr)
library(ggthemes)
library(scales)
library(lubridate)

##
## Attaching package: 'lubridate'

## The following objects are masked from 'package:data.table':
##
##     hour, isoweek, mday, minute, month, quarter, second, wday, week,
##     yday, year

## The following objects are masked from 'package:base':
##
##     date, intersect, setdiff, union

library(tinytex)
library(rmarkdown)
library(Matrix)

##
## Attaching package: 'Matrix'

## The following objects are masked from 'package:tidyr':
##
##     expand, pack, unpack

library(recommenderlab)

## Loading required package: arules
```

```
##
## Attaching package: 'arules'

## The following object is masked from 'package:dplyr':
##
##     recode

## The following objects are masked from 'package:base':
##
##     abbreviate, write

## Loading required package: proxy

##
## Attaching package: 'proxy'

## The following object is masked from 'package:Matrix':
##
##     as.matrix

## The following objects are masked from 'package:stats':
##
##     as.dist, dist

## The following object is masked from 'package:base':
##
##     as.matrix

## Loading required package: registry

## Registered S3 methods overwritten by 'registry':
##   method               from
##   print.registry_field proxy
##   print.registry_entry proxy

##
## Attaching package: 'recommenderlab'

## The following objects are masked from 'package:caret':
##
##     MAE, RMSE

library(kableExtra)

##
## Attaching package: 'kableExtra'

## The following object is masked from 'package:dplyr':
##
##     group_rows

library(readr)
```

```r
book_ratings <-
read.csv("https://raw.githubusercontent.com/zygmuntz/goodbooks-
10k/master/ratings.csv",
                         sep = ",", header = T, stringsAsFactors = F)
book_titles <-
read.csv("https://raw.githubusercontent.com/zygmuntz/goodbooks-
10k/master/books.csv",
                        sep = ",", header = T, stringsAsFactors = F) %>%
select(book_id, title)
ratings <- book_ratings
books <- book_titles

# tables dimensions

dim(books)

## [1] 10000     2

head(books)

##   book_id                                                    title
## 1       1                  The Hunger Games (The Hunger Games, #1)
## 2       2 Harry Potter and the Sorcerer's Stone (Harry Potter, #1)
## 3       3                                  Twilight (Twilight, #1)
## 4       4                                    To Kill a Mockingbird
## 5       5                                         The Great Gatsby
## 6       6                                  The Fault in Our Stars

dim(ratings)

## [1] 5976479       3

head(ratings)

##   user_id book_id rating
## 1       1     258      5
## 2       2    4081      4
## 3       2     260      5
## 4       2    9296      5
## 5       2    2318      3
## 6       2      26      4

# The GoodBooks Rating > 5M dataset:
# https://raw.githubusercontent.com/zygmuntz/goodbooks-10k/master/ratings.csv

# Joining the ratings and books datasets

bookcrossing <- left_join(ratings, books, by = "book_id")
```

```r
# I'm using later version of r; so, i will set seed as follow.

set.seed(1, sample.kind="Rounding")

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

# Creating a validation set. Validation set will be 10% of MovieLens data.

test_index <- createDataPartition(y = bookcrossing$rating, times = 1, p =
0.1, list = FALSE)
cyo <- bookcrossing[-test_index,]
temp <- bookcrossing[test_index,]

# Make sure user_id and book_id in validation set are also in edx set

validation <- temp %>%
  semi_join(cyo, by = "user_id") %>%
  semi_join(cyo, by = "book_id")

# Add rows removed from validation set back into edx set

removed <- anti_join(temp, validation)

## Joining, by = c("user_id", "book_id", "rating", "title")

cyo <- rbind(cyo, removed)

rm(test_index, ratings, bookcrossing, temp, removed)

# Partitioning the data set into a train_set and a test_set

set.seed(1, sample.kind="Rounding")

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

test_index <- createDataPartition(y = cyo$rating, times = 1, p = 0.2, list =
FALSE)
train_set <- cyo[-test_index,]
temp <- cyo[test_index,]

# Matching user_id and book_id in both train and test sets

test_set <- temp %>%
  semi_join(train_set, by = "user_id") %>%
  semi_join(train_set, by = "book_id")
```

```
# Adding back rows into train set

removed <- anti_join(temp, test_set)

## Joining, by = c("user_id", "book_id", "rating", "title")

train_set <- rbind(train_set, removed)

rm(test_index, temp, removed)

# Number of rows and columns in the cyo dataset?

nrow(cyo)

## [1] 5378830

ncol(cyo)

## [1] 4

head(cyo)

##   user_id book_id rating
## 2       2    4081      4
## 3       2     260      5
## 4       2    9296      5
## 5       2    2318      3
## 6       2      26      4
## 7       2     315      3
##                                                                  title
## 2                                                I am Charlotte Simmons
## 3                                How to Win Friends and Influence People
## 4           The Drama of the Gifted Child: The Search for the True Self
## 5 The Millionaire Next Door: The Surprising Secrets of Americas Wealthy
## 6                                The Da Vinci Code (Robert Langdon, #2)
## 7                                                   Who Moved My Cheese?

# Number of zeros given as ratings in the cyo dataset?

cyo %>% filter(rating == 0) %>% tally()

##   n
## 1 0

# Average rating for the cyo is ~ 4 (i.e. 3.919887) Let's see the number of
threes and foursgiven as ratings.

mean(cyo$rating)

## [1] 3.919887

cyo %>% filter(rating == 3) %>% tally()
```

```
##         n
## 1 1233894
```

```
cyo %>% filter(rating == 4) %>% tally()
```

```
##         n
## 1 1925116
```

```
# Different books in the cyo dataset.
```

```
n_distinct(cyo$book_id)
```

```
## [1] 10000
```

```
# Different titles in the cyo dataset.
```

```
n_distinct(cyo$title)
```

```
## [1] 9964
```

```
# Different users in the cyo data set?
```

```
n_distinct(cyo$user_id)
```

```
## [1] 53424
```

```
# Different rating in the cyo data set?
```

```
n_distinct(cyo$rating)
```

```
## [1] 5
```

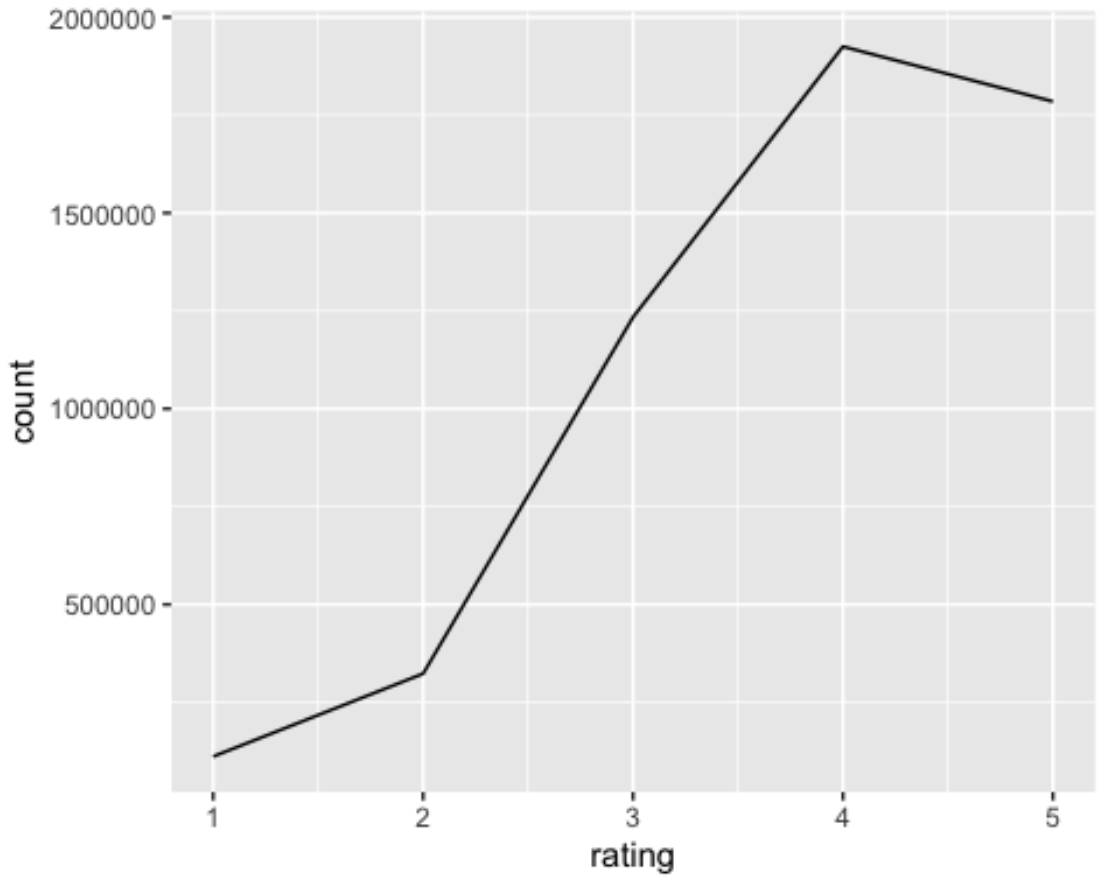
```
# Detecting the structure of the cyo data set.
```

```
cyo %>% group_by(book_id) %>%
  summarise(n=n()) %>%
  head()
```

```
## # A tibble: 6 x 2
##   book_id     n
##     <int> <int>
## 1       1 20564
## 2       2 19696
## 3       3 15215
## 4       4 17233
## 5       5 14966
## 6       6 10117
```

```
cyo %>% group_by(title) %>%
  summarise(n=n()) %>%
  head()
```

```
## # A tibble: 6 x 2
##   title                                                      n
##   <chr>                                                  <int>
## 1 " Angels (Walsh Family, #3)"                             240
## 2 "'Salem's Lot"                                          3982
## 3 "'Tis (Frank McCourt, #2)"                               629
## 4 "\"103"   حكايـات فـرغلي الـمستكـاوي \"حكايـتى مع كفـر الـسحلاويـة
## 5 "#GIRLBOSS"                                              177
## 6 "1,000 Places to See Before You Die"                     364

# books in ascending order

tibble(count = str_count(cyo$title, fixed("|")), title = cyo$title) %>%
  group_by(count, title) %>%
  summarise(n = n()) %>%
  arrange(count) %>%
  head()

## `summarise()` has grouped output by 'count'. You can override using the
`.groups` argument.

## # A tibble: 6 x 3
## # Groups:   count [1]
##   count title                                                  n
##   <int> <chr>                                              <int>
## 1     0 " Angels (Walsh Family, #3)"                         240
## 2     0 "'Salem's Lot"                                      3982
## 3     0 "'Tis (Frank McCourt, #2)"                           629
## 4     0 "\"103"   حكايـات فـرغلي الـمستكـاوي \"حكايـتى مع كفـر الـسحلاويـة
## 5     0 "#GIRLBOSS"                                          177
## 6     0 "1,000 Places to See Before You Die"                 364

# In general, half star ratings are less common than whole star ratings

cyo %>%
  group_by(rating) %>%
  summarize(count = n()) %>%
  ggplot(aes(x = rating, y = count)) +
  geom_line()
```




```
# Movie with the greatest number of ratings?

cyo %>% group_by(book_id, title) %>%
  summarize(count = n()) %>%
  arrange(desc(count))

## `summarise()` has grouped output by 'book_id'. You can override using the
`.groups` argument.

## # A tibble: 10,000 x 3
## # Groups:   book_id [10,000]
##    book_id title
count
##      <int> <chr>
<int>
##  1       1 The Hunger Games (The Hunger Games, #1)
20564
##  2       2 Harry Potter and the Sorcerer's Stone (Harry Potter, #1)
19696
##  3       4 To Kill a Mockingbird
17233
##  4       3 Twilight (Twilight, #1)
15215
```

```
##  5       5 The Great Gatsby
14966
##  6      17 Catching Fire (The Hunger Games, #2)
14869
##  7      20 Mockingjay (The Hunger Games, #3)
14370
##  8      18 Harry Potter and the Prisoner of Azkaban (Harry Potter, #3)
14285
##  9      23 Harry Potter and the Chamber of Secrets (Harry Potter, #2)
14087
## 10       7 The Hobbit
14019
## # … with 9,990 more rows
```

```
# The ten most given ratings in order from most to least?

cyo %>% group_by(rating) %>% summarize(count = n()) %>% top_n(10) %>%
  arrange(desc(count))
```

```
## Selecting by count
```

```
## # A tibble: 5 x 2
##   rating   count
##    <int>   <int>
## 1      4 1925116
## 2      5 1784783
## 3      3 1233894
## 4      2  323309
## 5      1  111728
```

Method and Evaluation

Define Mean Absolute Error (MAE)

#The mean absolute error is the average of absolute differences between the predicted value and the true # value. The metric is linear, which means that all errors are equally weighted. Thus, when predicting #ratings in a scale of 1 to 5, the MAE assumes that an error of 2 is twice as bad as an error of 1.

```
MAE <- function(true_ratings, predicted_ratings){
  mean(abs(true_ratings - predicted_ratings))
}
```

Define Root Mean Squared Error (RMSE)

The Root Mean Squared Error, RMSE, is the square root of the MSE, which is not consider here. It is the

typical metric to evaluate recommendation systems, and is defined by the formula:

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

Building Model

Model 1: Just the average of the data set for all books and users,

The simplest model assumes a random distribution of error from book to book variations, when predicting that all users will rate all books the same.Considering statistics theory, the mean, which is just the average of all observed ratings, minimizes the RMSE, as described in the formula below. $\hat{Y}_{u,i}=\mu+\epsilon_{i,u}$

```
mu <- mean(train_set$rating)
rmse1 <- RMSE(test_set$rating, mu)
```

Let's see how prediction is at this moment by replicating the x value at 2 and halt times of the test set up to maximum and counting the number of rows in the predicted test set and then predicting the RMSE of its rating

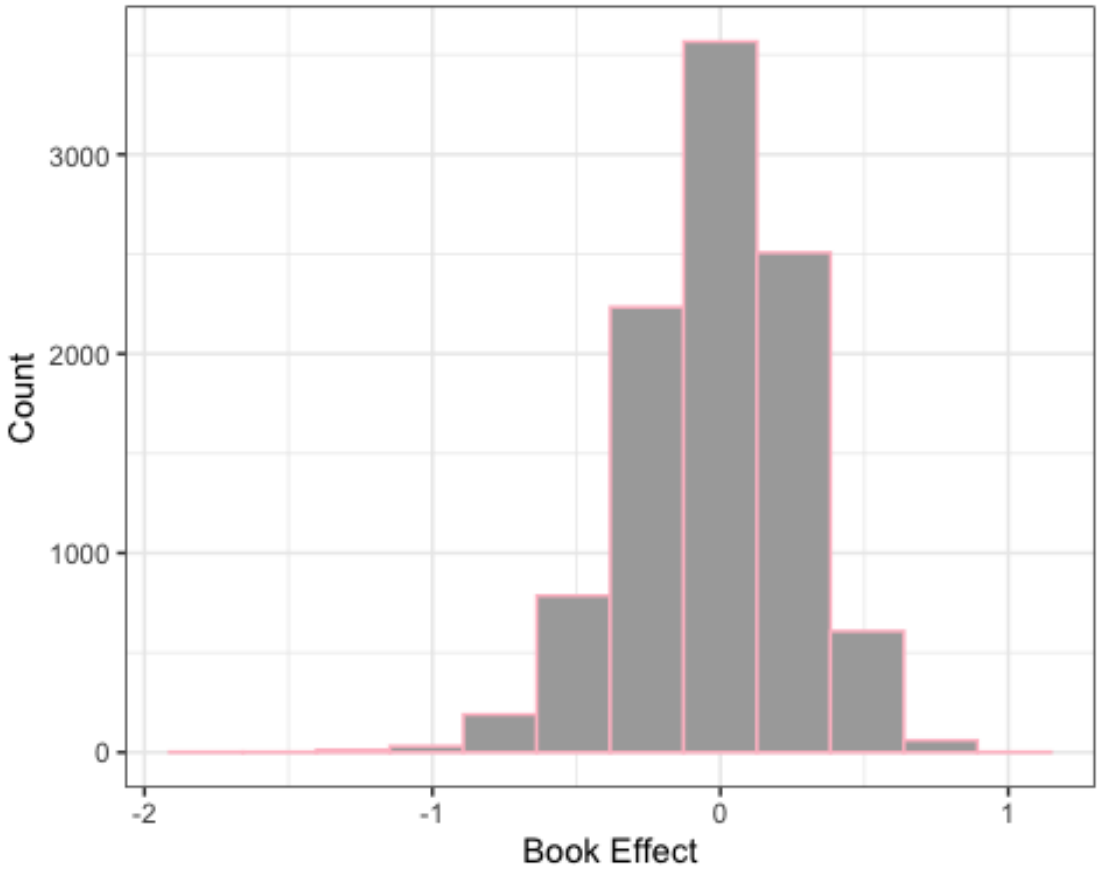
```
predictions <- rep(2.5, nrow(test_set))
RMSE(test_set$rating, predictions)

## [1] 1.731573

# 6 significant digits
options(digits = 6)

# creating a table to store the rmse results for every model's result along
the way.

naive_rmse <- RMSE(test_set$rating, mu)
results <- tibble(Method = "Mean", RMSE = naive_rmse, MAE = naive_rmse)
results

## # A tibble: 1 x 3
##   Method  RMSE   MAE
```

```
##   <chr>  <dbl> <dbl>
## 1 Mean   0.991 0.991
```

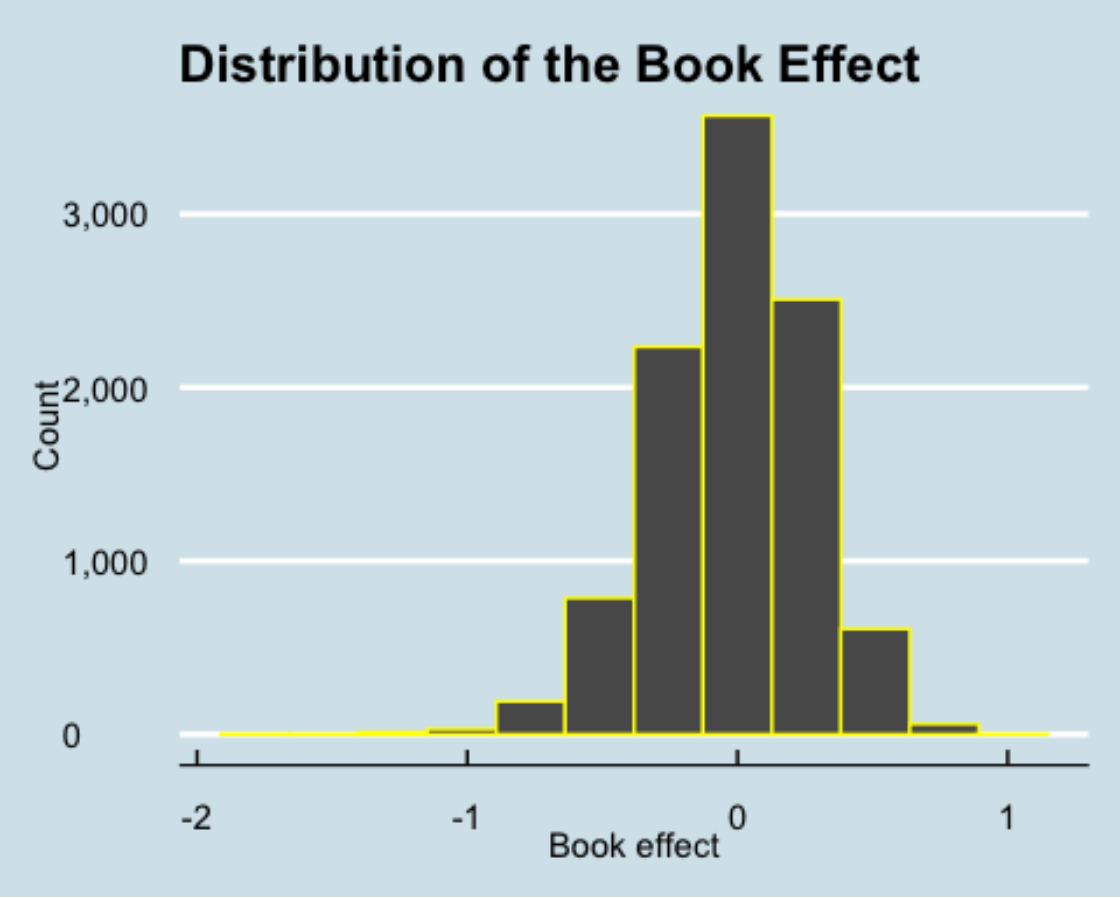
Model 2: the book effect on ratings

From exploratory data analysis, it was observed that some books are more popular than others and receive higher ratings. Considering the book effect, this model will be improved by adding the term bi to the formula used to determine the average of all books like this;
$Y_{u,i} = \mu + b_i + \epsilon_{u,i}$

```
bi <- train_set %>%
  group_by(book_id) %>%
  summarize(b_i = mean(rating - mu))

# Visual description of the book effect

bi %>% ggplot(aes(b_i)) +
  geom_histogram(color = "pink", fill = "darkgrey", bins = 12) +
  xlab("Book Effect") +
  ylab("Count") +
  theme_bw()
```



```
# Normal distribution for the book effect

bi %>% ggplot(aes(x = b_i)) +
  geom_histogram(bins=12, col = I("yellow")) +
  ggtitle("Distribution of the Book Effect") +
  xlab("Book effect") +
  ylab("Count") +
  scale_y_continuous(labels = comma) +
  theme_economist()
```



Predicting the rmse of the book effect on the test set considering the mean as well and then using left_join to return all row column by book_id.

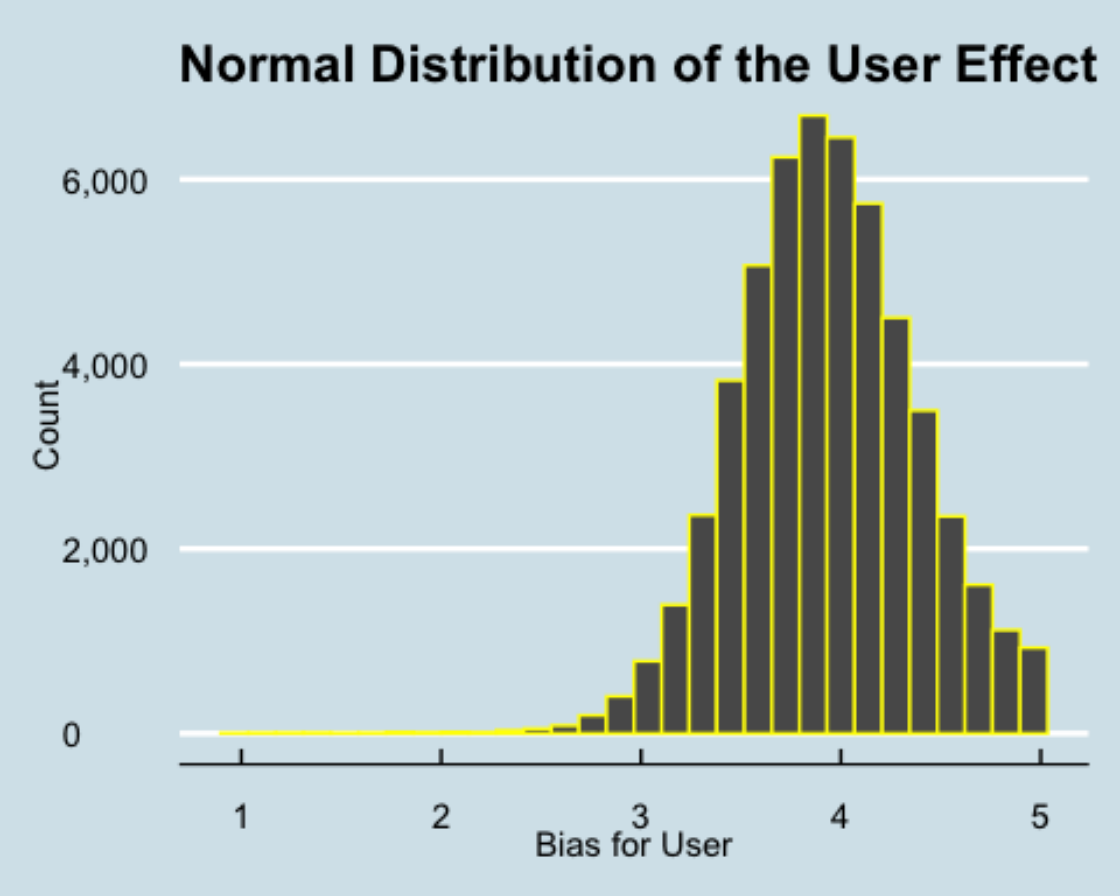
```
predicted_ratings <- mu + test_set %>%
  left_join(bi, by = "book_id") %>%
  .$b_i
book_effect_rmse <- RMSE(predicted_ratings, test_set$rating)
results <- bind_rows(results, tibble(Method = "The Book Effect", RMSE =
book_effect_rmse,
                                     MAE = book_effect_rmse))
results
```

```
## # A tibble: 2 x 3
##   Method           RMSE   MAE
##   <chr>           <dbl> <dbl>
## 1 Mean            0.991 0.991
## 2 The Book Effect 0.954 0.954
```

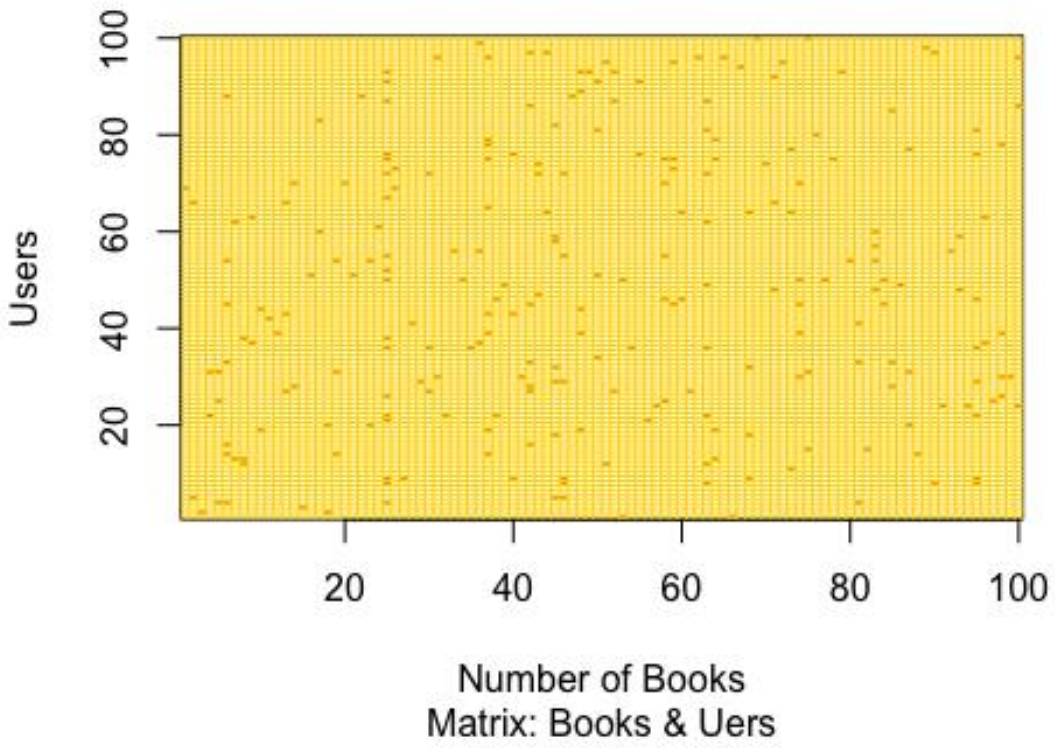
Model 3: the user's specific effect on ratings.

Considering the user's effect, this model can be improved by adding the term "bu" to the formula used in previous model like this; Yu,i = μ + bi + bu + ϵu,i Here the user effect model is built on the train_set by returning all rows and columns from the book effectby bookid and then taking the mean of the rating excluding the overall average and the book effect

```
bu <- train_set %>%
  left_join(bi, by = "book_id") %>%
  group_by(user_id) %>%
  summarize(b_u = mean(rating - mu - b_i))

# Normal distribution for the user effect

train_set %>%
  group_by(user_id) %>%
  summarize(b_u = mean(rating)) %>%
  filter(n()>=100) %>%
  ggplot(aes(b_u)) +
  geom_histogram(color = "yellow") +
  ggtitle("Normal Distribution of the User Effect") +
  xlab("Bias for User") +
  ylab("Count") +
  scale_y_continuous(labels = comma) +
  theme_economist()

## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```




```r
# plotting the book and user matrix

users <- sample(unique(cyo$user_id), 100)
cyo %>% filter(user_id %in% users) %>%
  select(user_id, book_id, rating) %>%
  mutate(rating = 1) %>%
  spread(book_id, rating) %>%
  select(sample(ncol(.), 100)) %>%
  as.matrix() %>% t(.) %>%
  image(1:100, 1:100,. , xlab="Number of Books", ylab="Users") %>%
  abline(h=0:100+0.5, v=0:100+0.5, col = "gold") %>%
  title("Matrix: Books & Uers")
```




```
# Predicting the rmse of the user effect

predicted_ratings <- test_set %>%
  left_join(bi, by = "book_id") %>%
  left_join(bu, by = "user_id") %>%
  mutate(pred = mu + b_i + b_u) %>%
  .$pred
user_effect_rmse <- RMSE(predicted_ratings, test_set$rating)
results <- bind_rows(results, tibble(Method = "The Book & User Effect", RMSE
= user_effect_rmse,
                                     MAE = user_effect_rmse))
results

## # A tibble: 3 x 3
##   Method                  RMSE   MAE
##   <chr>                  <dbl> <dbl>
## 1 Mean                   0.991 0.991
## 2 The Book Effect        0.954 0.954
## 3 The Book & User Effect 0.858 0.858

# different book titles

titles <- train_set %>%
```

```
  select(book_id, title) %>%
  distinct()

# Best unknown books rated by b_i

bi %>%
  inner_join(titles, by = "book_id") %>%
  arrange(-b_i) %>%
  select(title) %>%
  head()
```

```
## # A tibble: 6 x 1
##   title
##   <chr>
## 1 The Complete Calvin and Hobbes
## 2 The Revenge of the Baby-Sat
## 3 One Piece, Volume 38: Rocketman!! (One Piece, #38)
## 4 The Indispensable Calvin and Hobbes
## 5 It's a Magical World: A Calvin and Hobbes Collection
## 6 ESV Study Bible
```

```
# Worst unknown books rated by b_i

bi %>%
  inner_join(titles, by = "book_id") %>%
  arrange(b_i) %>%
  select(title) %>%
  head()
```

```
## # A tibble: 6 x 1
##   title
##   <chr>
## 1 One Night at the Call Center
## 2 Half Girlfriend
## 3 Of Course I Love You...! Till I Find Someone Better...
## 4 Can Love Happen Twice?
## 5 Revolution 2020: Love, Corruption, Ambition
## 6 The 3 Mistakes of My Life
```

```
# Number of ratings for 10 best movies:

train_set %>%
  left_join(bi, by = "book_id") %>%
  arrange(desc(b_i)) %>%
  group_by(title) %>%
  summarise(n = n()) %>%
  slice(1:10)
```

```
## # A tibble: 10 x 2
##    title
n
```

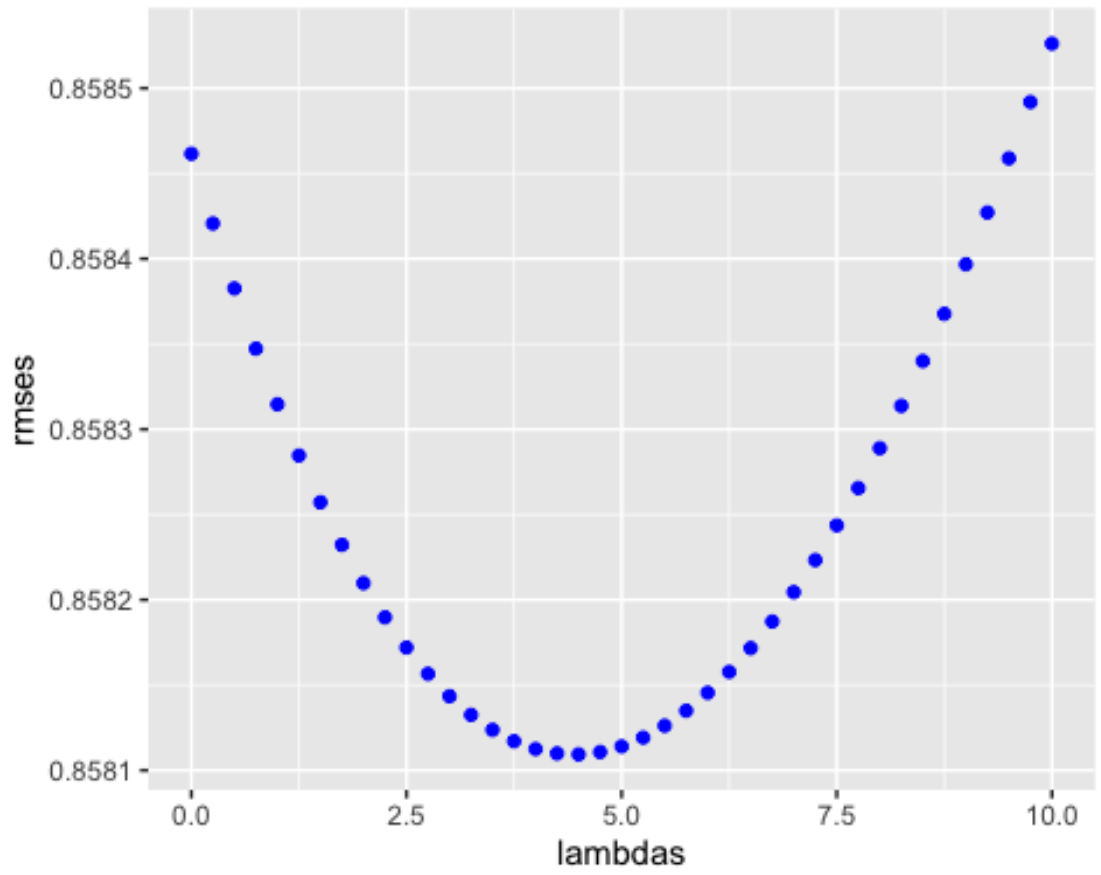
```
##    <chr>
<int>
##  1 " Angels (Walsh Family, #3)"
194
##  2 "'Salem's Lot"
3195
##  3 "'Tis (Frank McCourt, #2)"
517
##  4 "\"حكايـات فـرغلي الـمستكاوي \"حكايـتى مع كفـر الـسحلاويـة"
       86
##  5 "#GIRLBOSS"
141
##  6 "1,000 Places to See Before You Die"
284
##  7 "1/4 جرام"
   137
##  8 "10% Happier: How I Tamed the Voice in My Head, Reduced Stress Without…
211
##  9 "100 Bullets, Vol. 1: First Shot, Last Call"
125
## 10 "100 Love Sonnets"
93

train_set %>% count(book_id) %>%
  left_join(bi, by="book_id") %>%
  arrange(desc(b_i)) %>%
  slice(1:10) %>%
  pull(n)

##  [1] 356 127  52 155 274  58 179 178 454 102
```

Model 4:

Regularizing the book and user effects on rating using the best parameters from lanbdas to penalize or reduce noisy data. Here,three sets of lambdas are defined to tune lambdas beforehand.

```
lambdas <- seq(0, 10, 0.25)
rmses <- sapply(lambdas, function(x){
  b_i <- train_set %>%
    group_by(book_id) %>%
    summarize(b_i = sum(rating - mu)/(n()+x))
  b_u <- train_set %>%
    left_join(b_i, by = "book_id") %>%
    group_by(user_id) %>%
    summarize(b_u = sum(rating - b_i - mu)/(n()+x))
  predicted_ratings <- test_set %>%
    left_join(b_i, by = "book_id") %>%
    left_join(b_u, by = "user_id") %>%
```

```
    mutate(pred = mu + b_i + b_u) %>%
    .$pred
  return(RMSE(predicted_ratings, test_set$rating))
})

# plotting lambdas vs. RMSE

qplot(lambdas, rmses, color = I("blue"))
```




```
# Picking lambdas with the lowest RMSE to be used for regularizing the book
and user effects.

lamb <- lambdas[which.min(rmses)]
lamb

## [1] 4.5

# Predicting the rmse from a regularized book and user effects

mu <- mean(train_set$rating)

b_i <- train_set %>%
  group_by(book_id) %>%
```

```
  summarize(b_i = sum(rating - mu)/(n()+lamb))

b_u <- train_set %>%
  left_join(b_i, by="book_id") %>%
  group_by(user_id) %>%
  summarize(b_u = sum(rating - b_i - mu)/(n()+lamb))

# Prediction

mu_reg <- test_set %>%
  left_join(b_i, by = "book_id") %>%
  left_join(b_u, by = "user_id") %>%
  mutate(pred = mu + b_i + b_u) %>%
  .$pred

# Update the result table
results <- bind_rows(results,
                     tibble(Method = "Regularization: The Book & User
Effects",
                            RMSE = RMSE(test_set$rating, mu_reg),
                            MAE  = MAE(test_set$rating, mu_reg)))
results
```

```
## # A tibble: 4 x 3
##   Method                                    RMSE   MAE
##   <chr>                                    <dbl> <dbl>
## 1 Mean                                     0.991 0.991
## 2 The Book Effect                          0.954 0.954
## 3 The Book & User Effect                   0.858 0.858
## 4 Regularization: The Book & User Effects 0.858 0.672
```

Model 5: matrix factorization - alternatively using the recosystem for tuning due to memory gap. Matrix Factorization - the alternative Recosystem will be used instead due to the memory gap on commercial computer currently in use. Here, the best tuning parameters is used from an R suggested class object called Reco(). The train() method allows for a set of parameters inside the function and then, the $predict() is used for predicted values.

```
set.seed(1, sample.kind="Rounding")
```

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

```
train_reco <- with(train_set, data_memory(user_index = user_id, item_index =
book_id, rating = rating))
test_reco <- with(test_set, data_memory(user_index = user_id, item_index =
book_id, rating = rating))
rec <- Reco()
```

```r
alt_reco <- rec$tune(train_reco, opts = list(dim = c(20, 30),
                                             lrate = c(0.01, 0.1),
                                             costp_l1 = c(0.01, 0.1),
                                             costq_l1 = c(0.01, 0.1),
                                             nthread = 4,
                                             niter = 10))

rec$train(train_reco, opts = c(alt_reco$min, nthread = 4, niter = 40))
```

```
## iter      tr_rmse          obj
##    0       1.0216   6.7217e+06
##    1       0.8848   5.5821e+06
##    2       0.8717   5.4806e+06
##    3       0.8587   5.3932e+06
##    4       0.8434   5.2980e+06
##    5       0.8275   5.2043e+06
##    6       0.8121   5.1164e+06
##    7       0.7977   5.0389e+06
##    8       0.7838   4.9653e+06
##    9       0.7709   4.9006e+06
##   10       0.7590   4.8423e+06
##   11       0.7485   4.7937e+06
##   12       0.7384   4.7466e+06
##   13       0.7294   4.7068e+06
##   14       0.7212   4.6704e+06
##   15       0.7136   4.6387e+06
##   16       0.7067   4.6093e+06
##   17       0.7006   4.5851e+06
##   18       0.6950   4.5617e+06
##   19       0.6896   4.5408e+06
##   20       0.6847   4.5209e+06
##   21       0.6803   4.5035e+06
##   22       0.6765   4.4889e+06
##   23       0.6727   4.4752e+06
##   24       0.6691   4.4617e+06
##   25       0.6660   4.4499e+06
##   26       0.6629   4.4382e+06
##   27       0.6602   4.4279e+06
##   28       0.6575   4.4176e+06
##   29       0.6552   4.4089e+06
##   30       0.6529   4.4017e+06
##   31       0.6506   4.3929e+06
##   32       0.6486   4.3852e+06
##   33       0.6467   4.3778e+06
##   34       0.6451   4.3727e+06
##   35       0.6434   4.3663e+06
##   36       0.6417   4.3596e+06
##   37       0.6401   4.3545e+06
##   38       0.6388   4.3493e+06
##   39       0.6374   4.3442e+06
```

```
results_alt_reco <- rec$predict(test_reco, out_memory())

mat_factor_rmse <- RMSE(results_alt_reco, test_set$rating)
results <- bind_rows(results, tibble(Method = "Matrix factorization - the
recosystem",
                                     RMSE = mat_factor_rmse, MAE =
mat_factor_rmse))
results
```

```
## # A tibble: 5 x 3
##   Method                                    RMSE   MAE
##   <chr>                                    <dbl> <dbl>
## 1 Mean                                     0.991 0.991
## 2 The Book Effect                          0.954 0.954
## 3 The Book & User Effect                   0.858 0.858
## 4 Regularization: The Book & User Effects 0.858 0.672
## 5 Matrix factorization - the recosystem   0.834 0.834
```

The Validation set

Finalizing rmse prediction on the validation set. The lowest thus far, has been obtained on the fourth of four models using matrix factorization with the recosystem. Finally, the cyo data set will be used to train result fromm the fourth model, while the validation set will be used to test for accuracy.

```
set.seed(1, sample.kind="Rounding")
```

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

```
cyo_reco_sys <- with(cyo, data_memory(user_index = user_id, item_index =
book_id, rating = rating))
valid_reco <- with(validation, data_memory(user_index = user_id, item_index =
book_id, rating = rating))
rec <- Reco()

alt_reco <- rec$tune(cyo_reco_sys, opts = list(dim = c(20, 30),
                                               lrate = c(0.01, 0.1),
                                               costp_l2 = c(0.01, 0.1),
                                               costq_l2 = c(0.01, 0.1),
                                               nthread = 4,
                                               niter = 10))

rec$train(cyo_reco_sys, opts = c(alt_reco$min, nthread = 4, niter = 40))
```

```
## iter      tr_rmse          obj
##    0       1.0266   8.1662e+06
##    1       0.8706   6.2135e+06
```

```
##    2       0.8584   5.9528e+06
##    3       0.8398   5.7097e+06
##    4       0.8195   5.4877e+06
##    5       0.8025   5.3187e+06
##    6       0.7872   5.1765e+06
##    7       0.7735   5.0580e+06
##    8       0.7611   4.9573e+06
##    9       0.7504   4.8724e+06
##   10       0.7410   4.8010e+06
##   11       0.7329   4.7384e+06
##   12       0.7258   4.6852e+06
##   13       0.7196   4.6397e+06
##   14       0.7142   4.6007e+06
##   15       0.7095   4.5667e+06
##   16       0.7053   4.5357e+06
##   17       0.7016   4.5095e+06
##   18       0.6983   4.4854e+06
##   19       0.6953   4.4646e+06
##   20       0.6925   4.4457e+06
##   21       0.6900   4.4289e+06
##   22       0.6877   4.4127e+06
##   23       0.6856   4.3981e+06
##   24       0.6835   4.3837e+06
##   25       0.6818   4.3735e+06
##   26       0.6803   4.3627e+06
##   27       0.6786   4.3517e+06
##   28       0.6771   4.3425e+06
##   29       0.6757   4.3339e+06
##   30       0.6745   4.3255e+06
##   31       0.6732   4.3180e+06
##   32       0.6721   4.3102e+06
##   33       0.6710   4.3043e+06
##   34       0.6700   4.2984e+06
##   35       0.6689   4.2915e+06
##   36       0.6680   4.2866e+06
##   37       0.6671   4.2812e+06
##   38       0.6662   4.2757e+06
##   39       0.6654   4.2716e+06
```

```
valid_reco <- rec$predict(valid_reco, out_memory())

valid_final_rmse <- RMSE(valid_reco, validation$rating)
results <- bind_rows(results, tibble(Method = "Final validation: Matrix
factorization - the recosystem",
                                     RMSE = valid_final_rmse, MAE =
valid_final_rmse))
results
```

```
## # A tibble: 6 x 3
##   Method                                                RMSE   MAE
```

```
##   <chr>                                                  <dbl> <dbl>
## 1 Mean                                                   0.991 0.991
## 2 The Book Effect                                        0.954 0.954
## 3 The Book & User Effect                                 0.858 0.858
## 4 Regularization: The Book & User Effects                0.858 0.672
## 5 Matrix factorization - the recosystem                  0.834 0.834
## 6 Final validation: Matrix factorization - the recosystem 0.815 0.815
```

Conclusion

Comparing the Mean Absolute Error to the Root Mean Squared Error using a naive approach has been implemented together with the book effect and user-book effect taken as second and third models respectively. Although, there was no RMSE target set for this project, the lowest error of 0.815 for both the RMSE and the MAE. Note that both results are the same except for the regularization model. Also note that the RMSE results are the same for the Book and User effects and Regularization. It is only the Regularization model that shows my expectation for this project. All other models including the validation set give same results, and I am not sure if it is the dataset or the model that cause this. However, I still look forward to see this same dataset to be used with other methods and/or dataset as well to present analysis in a comparative results for the MAE, the RMS and the RMSE itself.