HarvardX:PH125.9x - Movie Lens Project

Sneha Kumar

In this project, we will be using machine learning to develop a movie recommendation system for the movielens data (from the dslabs package). The algorithm's effectiveness will be tested using RMSE (Root Mean Square Error)

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

First we will create our own edx dataset as well as validation dataset that will be used to test our algorithm in the end.

Note: this process could take a couple of minutes

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

# MovieLens 10M dataset:
# https://grouplens.org/datasets/movielens/10m/
# http://files.grouplens.org/datasets/movielens/ml-10m.zip

dl <- tempfile()
download.file("http://files.grouplens.org/datasets/movielens/ml-10m.zip", dl)

ratings <- fread(text = gsub("::", "\t", readLines(unzip(dl, "ml-10M100K/ratings.dat"))),
                 col.names = c("userId", "movieId", "rating", "timestamp"))

movies <- str_split_fixed(readLines(unzip(dl, "ml-10M100K/movies.dat")), "\\::", 3)
colnames(movies) <- c("movieId", "title", "genres")

movies <- as.data.frame(movies) %>% mutate(movieId = as.numeric(movieId),
                                           title = as.character(title),
                                           genres = as.character(genres))

movielens <- left_join(ratings, movies, by = "movieId")

# Validation set will be 10% of MovieLens data
set.seed(1, sample.kind="Rounding") # if using R 3.5 or earlier, use `set.seed(1)`
test_index <- createDataPartition(y = movielens$rating, times = 1, p = 0.1, list = FALSE)
edx <- movielens[-test_index,]
temp <- movielens[test_index,]

# Make sure userId and movieId in validation set are also in edx set
```

```
validation <- temp %>%
  semi_join(edx, by = "movieId") %>%
  semi_join(edx, by = "userId")

# Add rows removed from validation set back into edx set
removed <- anti_join(temp, validation)
edx <- rbind(edx, removed)

rm(dl, ratings, movies, test_index, temp, movielens, removed)
```

Exploratory Analysis

Now that we have our datasets, we will first explore our data and gain some insights.

These functions tell us the structure of the data and summary statistics of each of the columns

```
head(edx)
```

```
##    userId movieId rating timestamp                         title
## 1:      1     122      5 838985046              Boomerang (1992)
## 2:      1     185      5 838983525               Net, The (1995)
## 3:      1     292      5 838983421               Outbreak (1995)
## 4:      1     316      5 838983392               Stargate (1994)
## 5:      1     329      5 838983392 Star Trek: Generations (1994)
## 6:      1     355      5 838984474       Flintstones, The (1994)
##                           genres
## 1:                Comedy|Romance
## 2:         Action|Crime|Thriller
## 3:  Action|Drama|Sci-Fi|Thriller
## 4:       Action|Adventure|Sci-Fi
## 5: Action|Adventure|Drama|Sci-Fi
## 6:       Children|Comedy|Fantasy
```

```
str(edx)
```

```
## Classes 'data.table' and 'data.frame':   9000055 obs. of  6 variables:
##  $ userId   : int  1 1 1 1 1 1 1 1 1 1 ...
##  $ movieId  : num  122 185 292 316 329 355 356 362 364 370 ...
##  $ rating   : num  5 5 5 5 5 5 5 5 5 5 ...
##  $ timestamp: int  838985046 838983525 838983421 838983392 838983392 838984474 838983653 838984885 8
##  $ title    : chr  "Boomerang (1992)" "Net, The (1995)" "Outbreak (1995)" "Stargate (1994)" ...
##  $ genres   : chr  "Comedy|Romance" "Action|Crime|Thriller" "Action|Drama|Sci-Fi|Thriller" "Action|A
##  - attr(*, ".internal.selfref")=<externalptr>
```

```
summary(edx)
```

```
##      userId         movieId          rating        timestamp
##  Min.   :    1   Min.   :    1   Min.   :0.500   Min.   :7.897e+08
##  1st Qu.:18124   1st Qu.:  648   1st Qu.:3.000   1st Qu.:9.468e+08
##  Median :35738   Median : 1834   Median :4.000   Median :1.035e+09
##  Mean   :35870   Mean   : 4122   Mean   :3.512   Mean   :1.033e+09
##  3rd Qu.:53607   3rd Qu.: 3626   3rd Qu.:4.000   3rd Qu.:1.127e+09
##  Max.   :71567   Max.   :65133   Max.   :5.000   Max.   :1.231e+09
##     title              genres
##  Length:9000055     Length:9000055
##  Class :character   Class :character
##  Mode  :character   Mode  :character
##
##
##
```

The data contains 10,677 unique movies and 69,878 unique users.

```
n_distinct(edx$movieId)
```

```
## [1] 10677
```

```
n_distinct(edx$userId)
```

```
## [1] 69878
```

These are the most popular movies.

```
edx%>%
  group_by(title)%>%
  summarize(count=n())%>%
  arrange(desc(count))
```

```
## # A tibble: 10,676 x 2
##    title                                                          count
##    <chr>                                                          <int>
##  1 Pulp Fiction (1994)                                            31362
##  2 Forrest Gump (1994)                                            31079
##  3 Silence of the Lambs, The (1991)                               30382
##  4 Jurassic Park (1993)                                           29360
##  5 Shawshank Redemption, The (1994)                               28015
##  6 Braveheart (1995)                                              26212
##  7 Fugitive, The (1993)                                           25998
##  8 Terminator 2: Judgment Day (1991)                              25984
##  9 Star Wars: Episode IV - A New Hope (a.k.a. Star Wars) (1977) 25672
## 10 Apollo 13 (1995)                                               24284
## # ... with 10,666 more rows
```

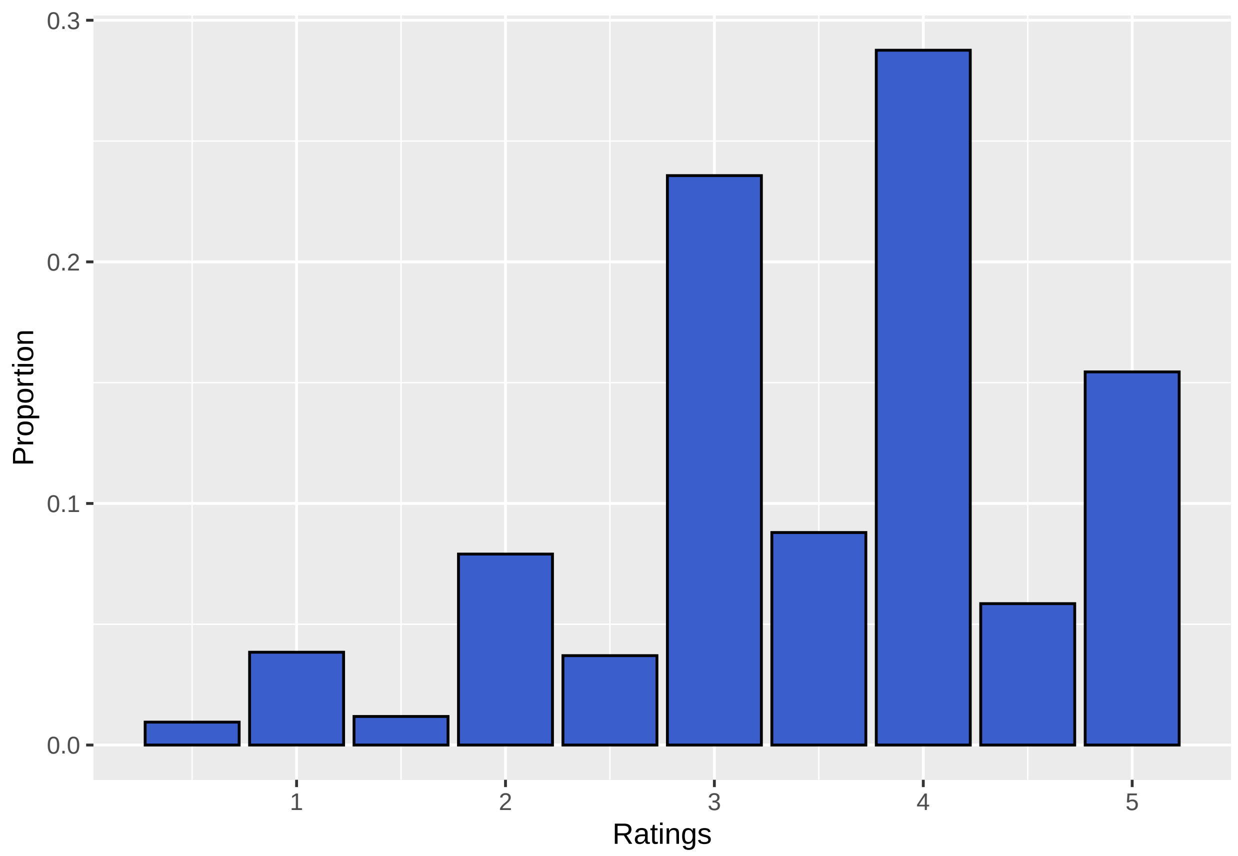
We can also see the most popular movie genres

```
edx%>%
  group_by(genres)%>%
  summarize(count=n())%>%
  arrange(desc(count))
```

```
## # A tibble: 797 x 2
##    genres                     count
##    <chr>                      <int>
##  1 Drama                     733296
##  2 Comedy                    700889
##  3 Comedy|Romance            365468
##  4 Comedy|Drama              323637
##  5 Comedy|Drama|Romance      261425
##  6 Drama|Romance             259355
##  7 Action|Adventure|Sci-Fi   219938
##  8 Action|Adventure|Thriller 149091
##  9 Drama|Thriller            145373
## 10 Crime|Drama               137387
## # ... with 787 more rows
```

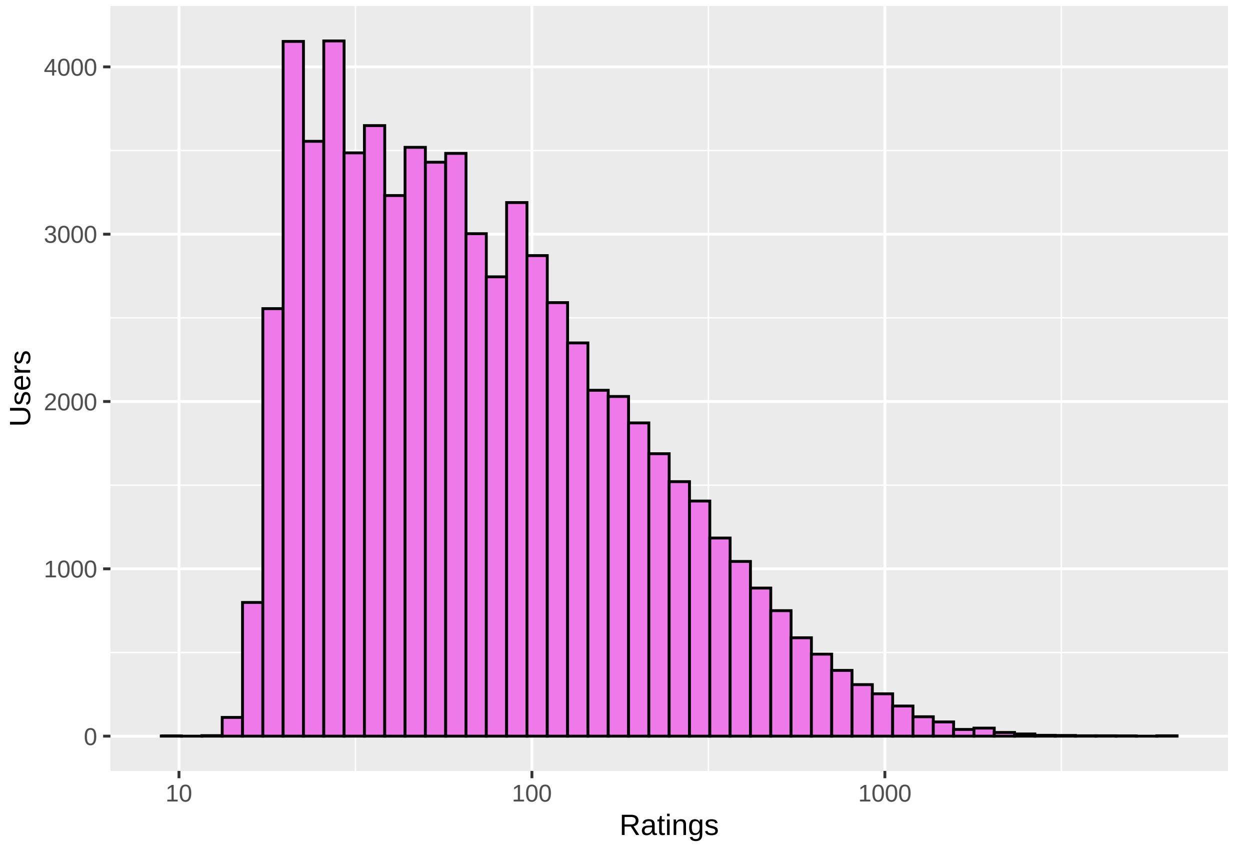
If we plot the ratings distributions (0.5-5.0), we see that whole numbers are generally used more often.

```
edx%>%
  ggplot(aes(rating,..prop..))+
  geom_bar(color="black",fill="royalblue3")+
  labs(x="Ratings",y="Proportion")
```



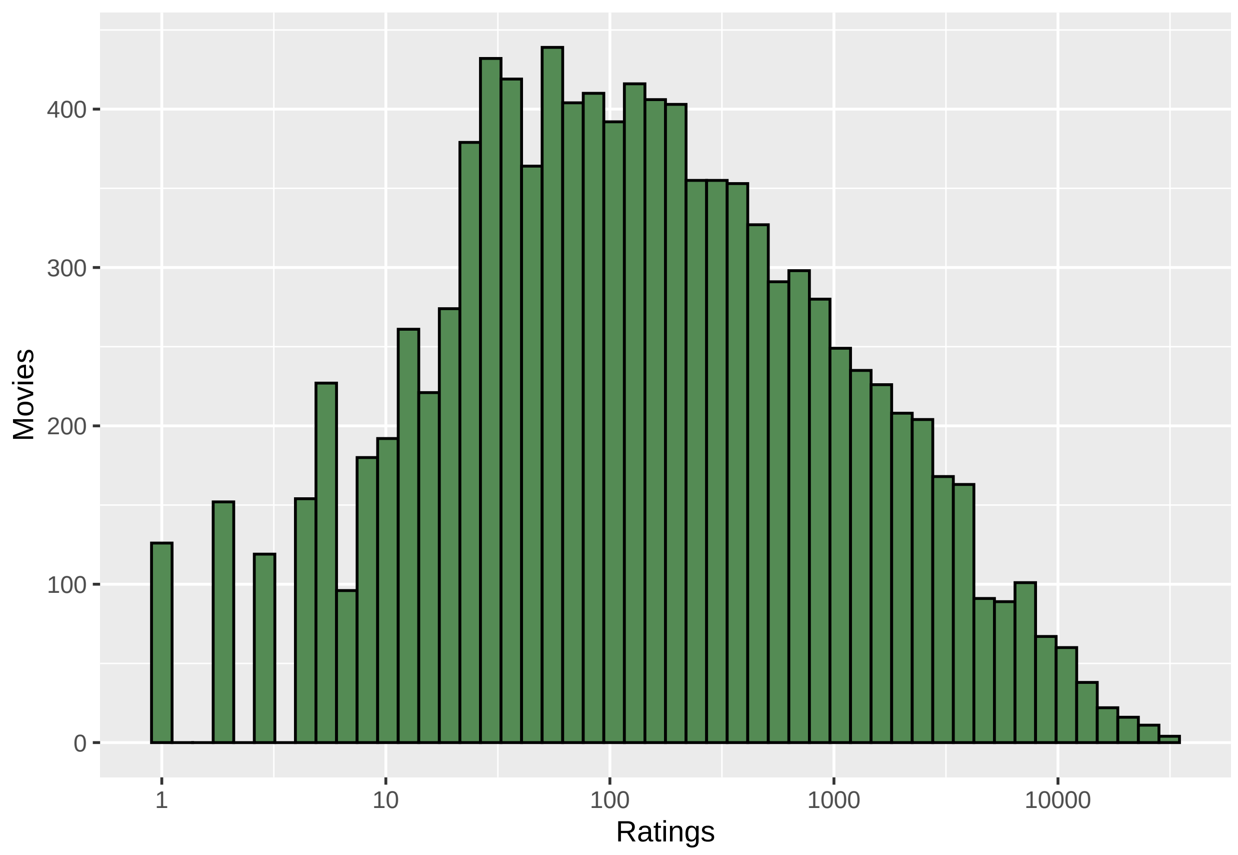
Ratings vs. Users has a distribution skewed the right. Hence, there is a user bias present in the movie ratings i.e. some users rate more movies than other users.

```r
edx%>%
  group_by(userId)%>%
  summarize(count=n())%>%
  ggplot(aes(count))+
  geom_histogram(color="black",fill="orchid2",bins=50)+
  scale_x_log10()+
  labs(x="Ratings",y="Users")
```



Ratings vs. Movies can also be graphed in the same manner. We also see a movie bias i.e. some movies get more ratings than others.

```r
edx%>%
  group_by(movieId)%>%
  summarize(count=n())%>%
  ggplot(aes(count))+
  geom_histogram(color="black",fill="palegreen4",bins=50)+
  scale_x_log10()+
  labs(x="Ratings",y="Movies")
```



Data Modeling

Create training and testing sets

Before we start modeling, we will first create our training and testing datasets

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

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

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

test <- temp%>%
  semi_join(train,by="movieId")%>%
  semi_join(train,by="userId")
train <- rbind(train, anti_join(temp,test))
```

```
## Joining, by = c("userId", "movieId", "rating", "timestamp", "title", "genres")
```

```
rm(test_index, temp)
```

RMSE function

Let's create a function that will calculate the RMSE, given our predicted and actual data.

```
rmse<-function(actual,predicted){
  sqrt(mean((actual-predicted)^2))
}
```

Models

1. Mean Ratings The simplest way is to use the average movie ratings

```
mu_hat<-mean(train$rating)
```

It has an RMSE of 1.06

```
mean_rmse<-rmse(test$rating,mu_hat)
```

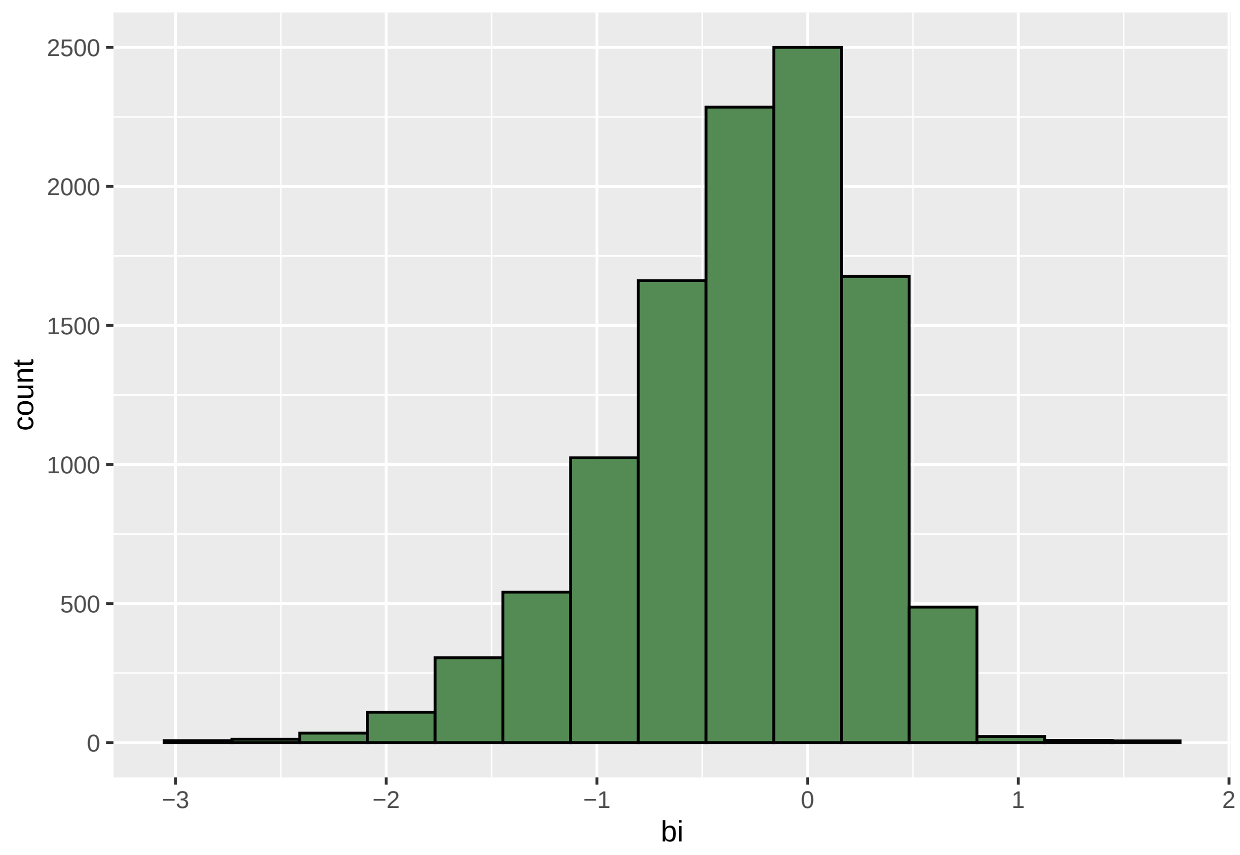
We will be storing this results in a table for easier reference

```
results<-tibble(Method="Model 1: Mean",RMSE=mean_rmse)
```

2. Movie Bias As we can see here once again, we see that the data has a movie bias

```
bi<-train%>%
  group_by(movieId)%>%
  summarize(bi=mean(rating-mu_hat))

#Distribution
bi %>%
  ggplot(aes(bi)) +
  geom_histogram(color = "black", fill = "palegreen4",bins=15)
```



So, we can improve our model using movies.

```
pred1<-mu_hat+test %>%
  left_join(bi, by = "movieId") %>%
  pull(bi)
```

This has reduced our RMSE to 0.94

```
#RMSE
mov_bias_rmse<-rmse(test$rating,pred1)

#Storing results into a tibble
results<-bind_rows(results,tibble(Method="Model 2: Mean & Movie Bias",RMSE=mov_bias_rmse))
```

3. User Bias Next, as seen earlier, our data also containts user bias. This can be further added to improve our model

```
bu<-train%>%
  left_join(bi,by ="movieId")%>%
  group_by(userId)%>%
  summarize(bu=mean(rating-mu_hat-bi))

#Predicted Ratings
pred2<-test%>%
  left_join(bi,by="movieId")%>%
  left_join(bu,by="userId")%>%
  mutate(pred=mu_hat+bi+bu)%>%
  pull(pred)
```

Our RMSE has also reduced to 0.86

```
#RMSE
user_bias_rmse<-rmse(test$rating,pred2)

#Storing results into a tibble
results<-bind_rows(results,tibble(Method="Model 3: Mean, Movie Bias, & User Bias",RMSE=user_bias_rmse))
```

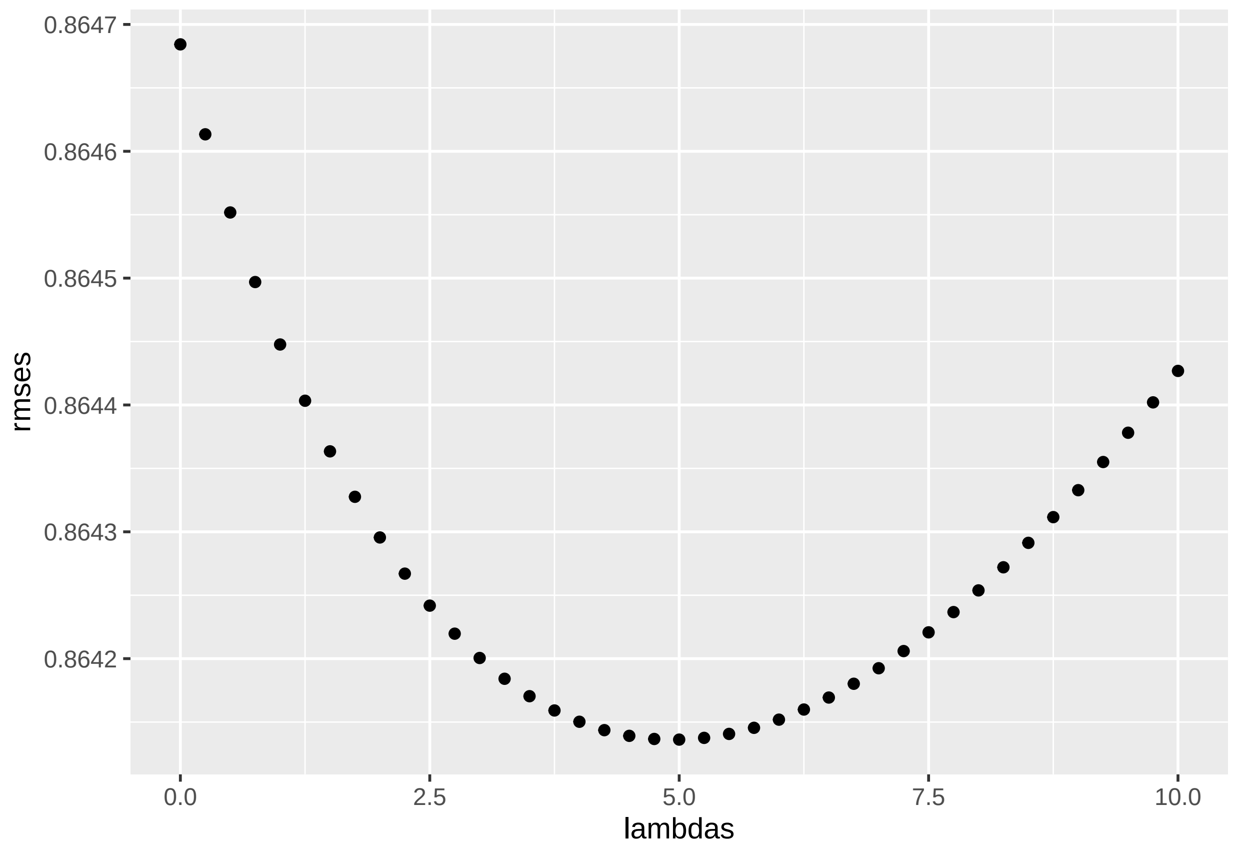
4. Regularization (Movie & Users) Some of our datas are noisy i.e. some movies have more ratings than others. To balance this out we can use regularization.

```
lambdas<-seq(0,10,0.25)

#RMSE
rmses <- sapply(lambdas, function(x){
  bi<-train%>%
    group_by(movieId)%>%
    summarize(bi=sum(rating-mu_hat)/(n()+x))
  bu<-train%>%
    left_join(bi,by="movieId")%>%
    group_by(userId)%>%
    summarize(bu=sum(rating-bi-mu_hat)/(n()+x))
  pred3<-test%>%
    left_join(bi,by="movieId")%>%
    left_join(bu,by="userId")%>%
    mutate(pred=mu_hat+bi+bu) %>%
    pull(pred)
  return(rmse(pred3, test$rating))
})
```

We can plot our rmses and find the lowest lambda

```
# Plot
qplot(lambdas,rmses)
```



```
#Choose the best lambda
lambda <- lambdas[which.min(rmses)]
```

We can calculate our new RMSE using the lowest value in rmses. This has still reduced our RMSE by a little

```
#RMSE
reg_rmse=min(rmses)

#Storing results into a tibble
results<-bind_rows(results,tibble(Method="Model 4: Regularization",RMSE=reg_rmse))
```

5. Matrix Factorization (recosystem) We can also use the recosystem package to perform matrix factorization. Let's see if this further reduces our RMSE.

We will first convert our data into recosystem format

```
library(recosystem)
set.seed(1, sample.kind="Rounding")
train_rec<-with(train,data_memory(user_index=userId,item_index=movieId,rating=rating))
test_rec<-with(test,data_memory(user_index=userId,item_index=movieId,rating=rating))
rec <- Reco()

#Tuning
tune_reco<-rec$tune(train_rec,opts=list(dim=c(20, 30),
```

```r
                                          costp_l2=c(0.01, 0.1),
                                          costq_l2=c(0.01, 0.1),
                                          lrate=c(0.01, 0.1),
                                          nthread=4,
                                          niter=10))
#Training
rec$train(train_rec,opts=c(tune_reco$min,nthread = 4,niter = 30))

#Predicted Ratings
pred4<-rec$predict(test_rec,out_memory())
```

We get our new RMSE of 0.78, which is a lot less than what we had earlier

```r
#RMSE
mat_fact_rmse<-rmse(test$rating,pred4)

#Storing results into a tibble
results<-bind_rows(results,tibble(Method="Model 5: Matrix Factorization",RMSE=mat_fact_rmse))
```

Here is a summary of our models

```r
results%>%knitr::kable()
```

Method	RMSE
Model 1: Mean	1.0600537
Model 2: Mean & Movie Bias	0.9429615
Model 3: Mean, Movie Bias, & User Bias	0.8646843
Model 4: Regularization	0.8641362
Model 5: Matrix Factorization	0.7845422

Final Validation

Now we will use our model with the lowest RMSE and run it with our validation dataset

```r
set.seed(1, sample.kind="Rounding")
edx_reco<- with(edx,data_memory(user_index=userId,item_index=movieId,rating=rating))
val_rec<-with(validation, data_memory(user_index=userId,item_index=movieId,rating=rating))
r<-Reco()

par_reco <- r$tune(edx_reco,opts=list(dim=c(20,30),
                                      costp_l2=c(0.01, 0.1),
                                      costq_l2=c(0.01, 0.1),
                                      lrate=c(0.01, 0.1),
                                      nthread=4,
                                      niter=10))

r$train(edx_reco, opts = c(par_reco$min, nthread = 4, niter = 30))
final_rec<- r$predict(val_rec, out_memory())

#Final RMSE
```

```r
final_rmse<-rmse(validation$rating,final_rec)

#Store final value to tibble
results<-bind_rows(results,tibble(Method ="Final validation: Matrix factorization ",RMSE=final_rmse))

results%>%knitr::kable()
```

Our final RMSE is 0.7811. This is the lowest RMSE we have achieved using matrix factorization.