Capstone Recommender Project

John Hopkins

November 27th, 2019

Abstract

This document presents several approaches to implementing a recommender system utilizing the Movielens dataset. The relative effectiveness of these models is compared as to determine the best predictive solution.

Contents

Summary

A Recommender System refers to a system that is capable of predicting the future preference of a set of items for a user, and recommend the top items. [1]

The Movielens [2] dataset will be used as the basis of our search for the best recommender system.

Several techniques will be presented to refine the predictive results.

- Just the average
- Movie Effect Model on validation set
- Movie + User Effects Model4
- Regularized Movie + User Effect Model
- Recosystem - Regularized Movie and User

A comparison of the RMSE results of each methods is presented at the conclusion of this document.

Setup

Movies and Ratings data are read and used to create a training and testing dataset.

Note that the userId and movieId in testing are also in training set.

A master list of all movie titles is created and saved for futher reference.

```
library(dslabs)
library(tidyverse)
library(caret)
library(devtools)
library(dplyr)
library(data.table)
library(gridExtra)
library(recosystem)
```

```
library(rrecsys)
library(slimrec)

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(levels(movieId))[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")
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)
#
# # connect movieId to movie title
movie_titles <- movielens %>%
 dplyr::select(movieId, title) %>%
 distinct()

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

Review

The content of the data is examined as to the number of distinct movies and user and minimal and maximum ratings.

The top 5 movies are presented as well as a heatmap showing which movies/user combinations have actually submitted a rating.

Counts by movie and rating are also displayed.

```
edx %>% summarize(
  n_users=n_distinct(userId),# unique users from edx
  n_movies=n_distinct(movieId),# unique movies from edx
```

```
  min_rating=min(rating),  # the lowest rating
  max_rating=max(rating) # the highest rating
)
```

```
##   n_users n_movies min_rating max_rating
## 1   69878    10677        0.5          5
```

```
# matrix for top 5 movies
top5 <- edx %>%
  count(movieId) %>%
  top_n(5, n) %>%
  .$movieId

top5.tbl <- edx %>%
  filter(movieId %in% top5) %>%
  filter(userId %in% c(13:20)) %>%
  dplyr::select(userId, title, rating) %>%
  mutate(title = str_remove(title, ", The"),
         title = str_remove(title, ":.*")) %>%
  spread(title, rating)

top5.tbl[,1:4] %>% knitr::kable( caption = 'Top 5 Movies')
```

Table 1: Top 5 Movies

userId	Forrest Gump (1994)	Jurassic Park (1993)	Pulp Fiction (1994)
13	NA	NA	4
16	NA	3	NA
17	NA	NA	NA
18	NA	3	5
19	4	1	NA

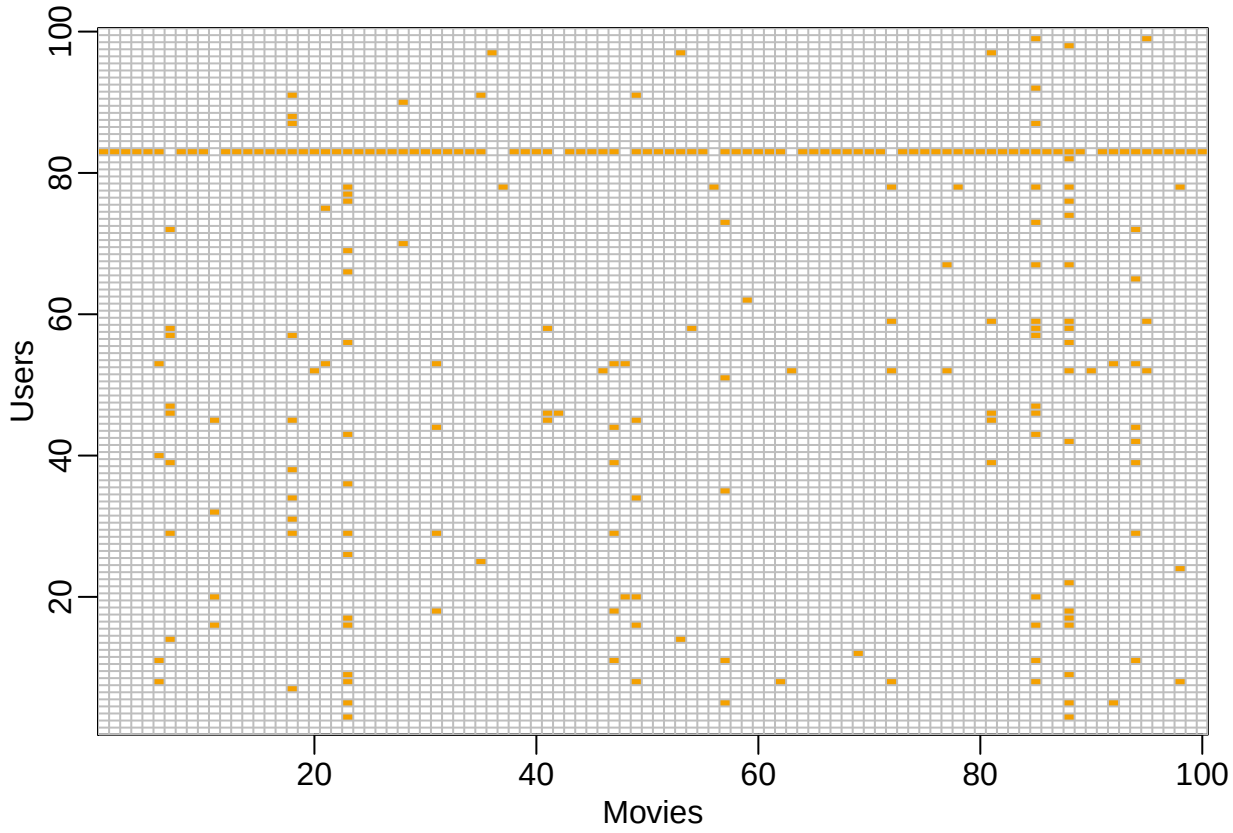
```
top5.tbl[,c(1,5:6)] %>% knitr::kable( caption = 'Top 5 Movies')
```

Table 2: Top 5 Movies

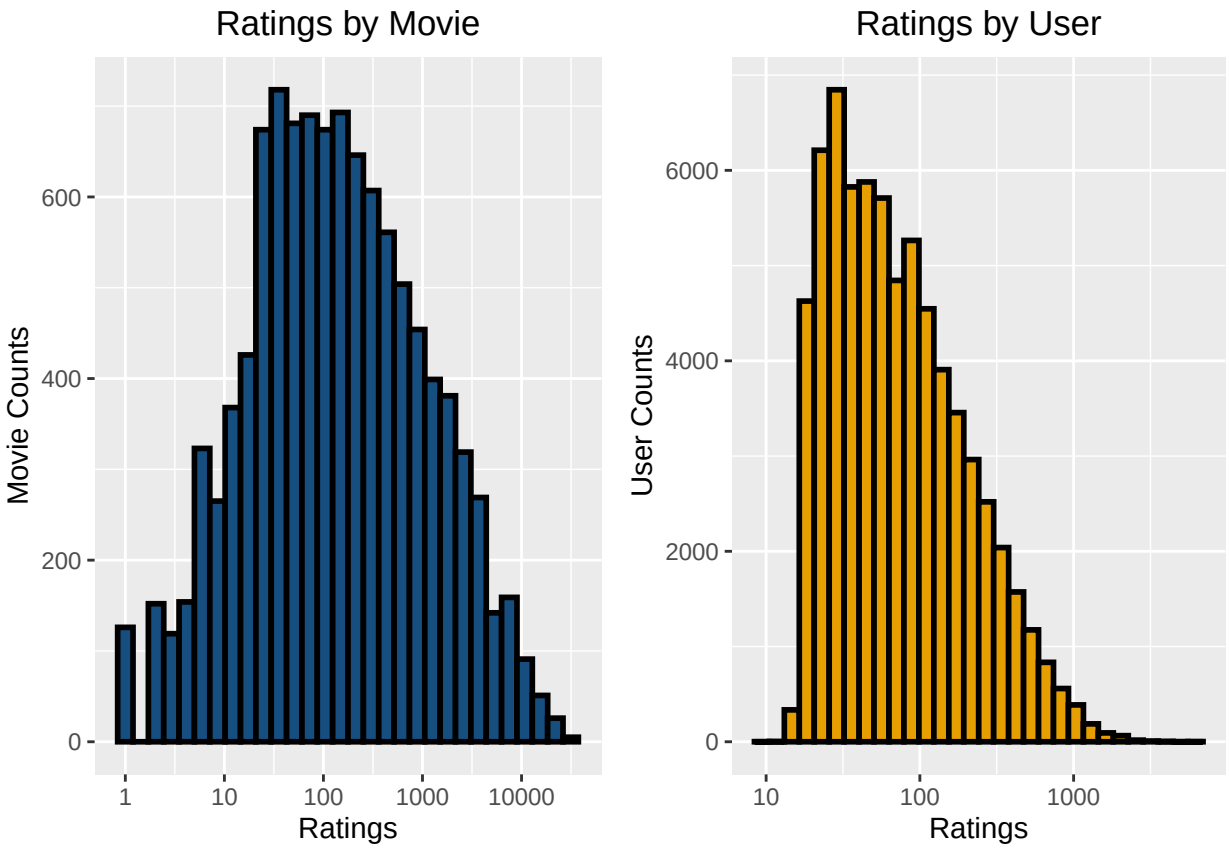
userId	Shawshank Redemption (1994)	Silence of the Lambs (1991)
13	NA	NA
16	NA	NA
17	NA	5
18	4.5	5
19	4.0	NA

```
# matrix for a random sample of 100 movies and 100 users with yellow
# indicating a user/movie combination for which we have a rating.

users <- sample(unique(edx$userId), 100)
rafalib::mypar()
edx %>% filter(userId %in% users) %>%
  dplyr::select(userId, movieId, rating) %>%
  mutate(rating = 1) %>%
```

```
  spread(movieId, rating) %>%
  dplyr::select(sample(ncol(.), 100)) %>%
  as.matrix() %>% t(.) %>%
  image(1:100, 1:100,. , xlab="Movies", ylab="Users")
abline(h=0:100+0.5, v=0:100+0.5, col = "grey")
```




```
# plot count rating by movie
RM.pLot <- edx %>%
  count(movieId) %>%
  ggplot(aes(n)) +
  geom_histogram(bins = 30, color = "black", fill='#164E80', size=1) +
  scale_x_log10() +
  labs(title="Ratings by Movie",x="Ratings", y = "Movie Counts")+
  theme( plot.title = element_text(hjust = 0.5))

# plot count rating by user
RU.pLot <- edx %>%
  count(userId) %>%
  ggplot(aes(n)) +
  geom_histogram(bins = 30, color = "black", fill='#E69F00', size=1) +
  scale_x_log10() +
  labs(title="Ratings by User",x="Ratings", y = "User Counts")+
  theme( plot.title = element_text(hjust = 0.5))

grid.arrange(RM.pLot, RU.pLot, ncol=2)
```




```
# to save space
rm(top5.tbl, top5, users, RM.pLot, RU.pLot)
```

Simple Average

A simple average using the mean is first calculated. This doesn't take into acount the impact of individual movies or users, but serves as a basis from which more qualified methods will be built upon.

```
# RMSE compute root mean square error (RMSE)
RMSE <- function(true_ratings, predicted_ratings){
  sqrt(mean((true_ratings - predicted_ratings)^2))
}

mu <- mean(edx$rating) # compute mean rating
mu
```

```
## [1] 3.512465
```

```
model_1_rmse <- RMSE(validation$rating, mu)
model_1_rmse
```

```
## [1] 1.061202
```

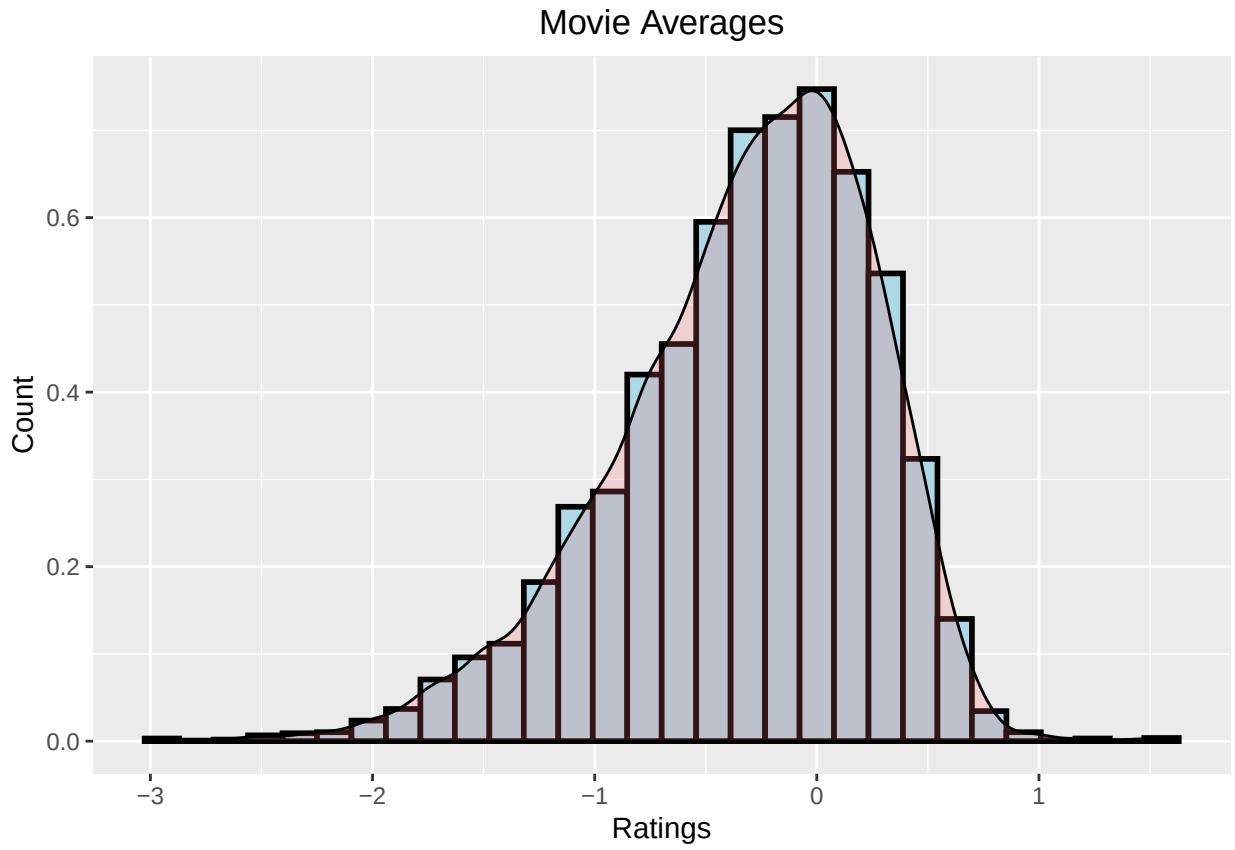
```
rsys_rmse_results <- tibble(method = "Just the average",
                            RMSE = model_1_rmse)
```

Movie Effect

The Movie effect is added to the simple average previously calculated. This has the effect of accounting for the average influence which popular movies have on preferences.

```
movie_avgs <- edx %>%
  group_by(movieId) %>%
  summarize(b_i = mean(rating - mu))

ggplot(movie_avgs, aes(x=b_i)) +
  geom_histogram(aes(y = ..density..), colour="black", fill="lightblue", size=1)+
  geom_density(alpha=.2, fill="#FF6666") +
  labs(title="Movie Averages",x="Ratings", y = "Count")+
  theme( plot.title = element_text(hjust = 0.5))
```

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




```
# create a results table with this and prior approaches
predicted_ratings <- mu + validation %>%
  left_join(movie_avgs, by='movieId') %>%
  .$b_i

model_2_rmse <- RMSE(predicted_ratings, validation$rating)
model_2_rmse
```

```
## [1] 0.9439087
```

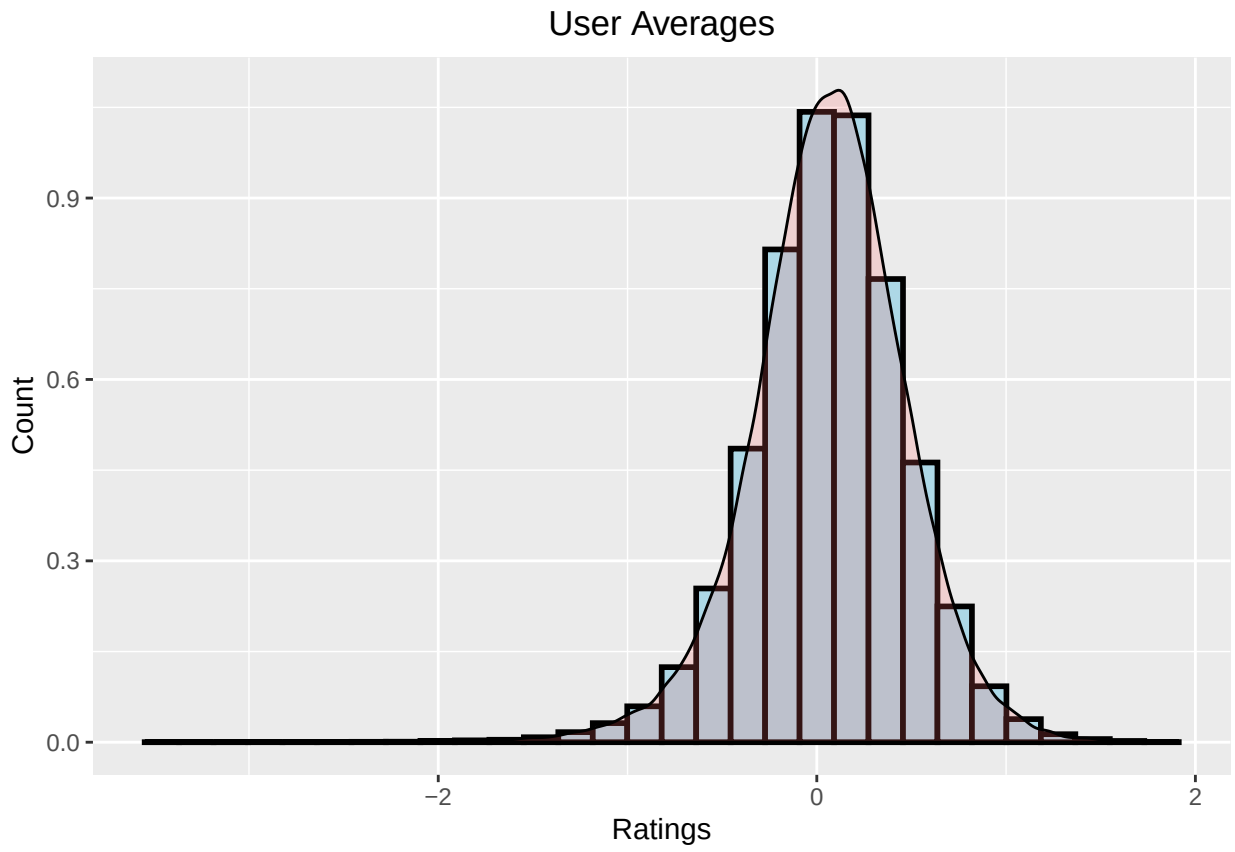
```
rsys_rmse_results <- bind_rows(rsys_rmse_results,
                               tibble(method="Movie Effect Model",
                                      RMSE = model_2_rmse))
```

User Effect

The impact of users is then added to the movie effect, taking into account both movie and user effects. This has the effect of improving RSME.

```
user_avgs <- edx %>%
  left_join(movie_avgs, by='movieId') %>%
  group_by(userId) %>%
  summarize(b_u = mean(rating - mu - b_i))

ggplot(user_avgs, aes(x=b_u)) +
  geom_histogram(aes(y = ..density..), colour="black", fill="lightblue", size=1)+
  geom_density(alpha=.2, fill="#FF6666") +
  labs(title="User Averages",x="Ratings", y = "Count")+
  theme( plot.title = element_text(hjust = 0.5))
```

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




```
predicted_ratings <- edx %>%
  left_join(movie_avgs, by='movieId') %>%
  left_join(user_avgs, by='userId') %>%
  mutate(pred = mu + b_i + b_u) %>%
  .$pred
```

```
model_3_rmse <- RMSE(predicted_ratings, edx$rating)
model_3_rmse
```

```
## [1] 0.8567039
```

```
rsys_rmse_results <- bind_rows(rsys_rmse_results,
                               tibble(method="Movie + User Effects Model",
                               RMSE = model_3_rmse))
```

Regularization

Regularization is used to further refine the effects of movies and users.

Cross-validation is employed to pick a lambda which reflects the optimal best fit.

```
# top 10 best movies based on b_i
movie_avgs %>% left_join(movie_titles, by="movieId") %>%
  arrange(desc(b_i)) %>%
  dplyr::select(title, b_i) %>%
  slice(1:10) %>%
  knitr::kable()
```

title	b_i
Hellhounds on My Trail (1999)	1.487535
Satan's Tango (SÃ Â¡tÃ Â¡ntangÃ Â³) (1994)	1.487535
Shadows of Forgotten Ancestors (1964)	1.487535
Fighting Elegy (Kenka erejii) (1966)	1.487535
Sun Alley (Sonnenallee) (1999)	1.487535
Blue Light, The (Das Blaue Licht) (1932)	1.487535
Who's Singin' Over There? (a.k.a. Who Sings Over There) (Ko to tamo peva) (1980)	1.237535
Human Condition II, The (Ningen no joken II) (1959)	1.237535
Human Condition III, The (Ningen no joken III) (1961)	1.237535
Constantine's Sword (2007)	1.237535

```
# top 10 worse movies based on b_i
movie_avgs %>% left_join(movie_titles, by="movieId") %>%
  arrange(b_i) %>%
  dplyr::select(title, b_i) %>%
  slice(1:10) %>%
  knitr::kable()
```

title	b_i
Besotted (2001)	-3.012465
Hi-Line, The (1999)	-3.012465
Accused (Anklaget) (2005)	-3.012465
Confessions of a Superhero (2007)	-3.012465
War of the Worlds 2: The Next Wave (2008)	-3.012465
SuperBabies: Baby Geniuses 2 (2004)	-2.717822
Hip Hop Witch, Da (2000)	-2.691037
Disaster Movie (2008)	-2.653090
From Justin to Kelly (2003)	-2.610455
Criminals (1996)	-2.512465

```
# add number of rating of the "best" obscure movies
edx %>%
  count(movieId) %>%
  left_join(movie_avgs) %>%
  left_join(movie_titles, by="movieId") %>%
  arrange(desc(b_i)) %>%
  dplyr::select(title, b_i, n) %>%
  slice(1:10) %>%
  knitr::kable()
```

```
## Joining, by = "movieId"
```

title	b_i	n
Hellhounds on My Trail (1999)	1.487535	1
Satan's Tango (SÃ Â¡tÃ Â¡ntangÃ Â³) (1994)	1.48	7535 2
Shadows of Forgotten Ancestors (1964)	1.487535	1
Fighting Elegy (Kenka erejii) (1966)	1.487535	1
Sun Alley (Sonnenallee) (1999)	1.487535	1
Blue Light, The (Das Blaue Licht) (1932)	1.487535	1
Who's Singin' Over There? (a.k.a. Who Sings Over There) (Ko to tamo peva) (1980)	1.237535	4
Human Condition II, The (Ningen no joken II) (1959)	1.237535	4
Human Condition III, The (Ningen no joken III) (1961)	1.237535	4
Constantine's Sword (2007)	1.237535	2

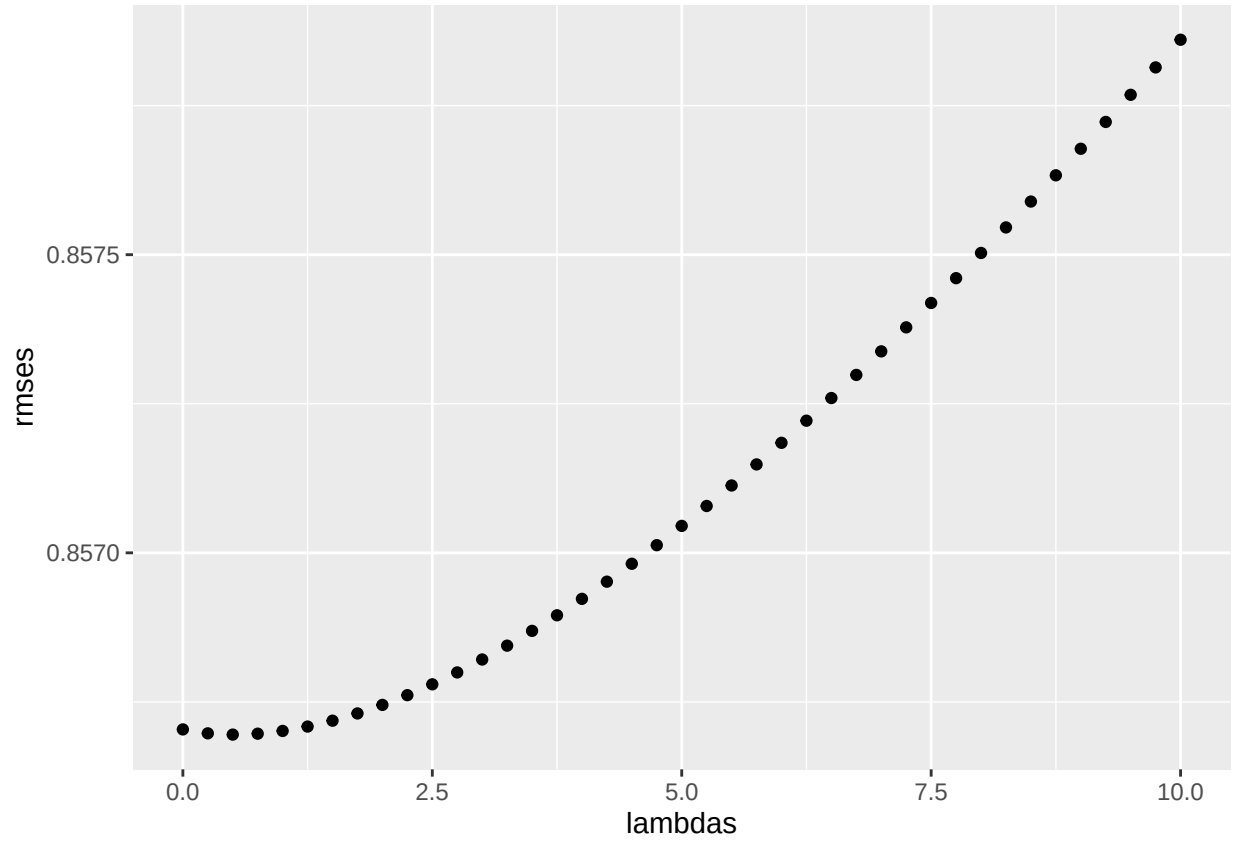
```
edx %>%
  count(movieId) %>%
  left_join(movie_avgs) %>%
  left_join(movie_titles, by="movieId") %>%
  arrange(b_i) %>%
  dplyr::select(title, b_i, n) %>%
  slice(1:10) %>%
  knitr::kable()
```

```
## Joining, by = "movieId"
```

title	b_i	n
Besotted (2001)	-3.012465	2
Hi-Line, The (1999)	-3.012465	1
Accused (Anklaget) (2005)	-3.012465	1
Confessions of a Superhero (2007)	-3.012465	1
War of the Worlds 2: The Next Wave (2008)	-3.012465	2
SuperBabies: Baby Geniuses 2 (2004)	-2.717822	56
Hip Hop Witch, Da (2000)	-2.691037	14
Disaster Movie (2008)	-2.653090	32
From Justin to Kelly (2003)	-2.610455	199
Criminals (1996)	-2.512465	2

```
# use cross-validation to pick a lambda:

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

```
rmses <- sapply(lambdas, function(l){
  mu <- mean(edx$rating)
  b_i <- edx %>%
    group_by(movieId) %>%
    summarize(b_i = sum(rating - mu)/(n() + l))

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

  predicted_ratings <-
    edx %>%
    left_join(b_i, by = "movieId") %>%
    left_join(b_u, by = "userId") %>%
    mutate(pred = mu + b_i + b_u) %>%
    .$pred
  return(RMSE(predicted_ratings, edx$rating))
})

qplot(lambdas, rmses)
```




```
lambda <- lambdas[which.min(rmses)]
lambda
```

```
## [1] 0.5
```

```
model_4_rmse <- min(rmses)
model_4_rmse
```

```
## [1] 0.8566952
```

```
rsys_rmse_results <- bind_rows(rsys_rmse_results,
                               tibble(method="Regularized Movie + User Effect Model",
                               RMSE = model_4_rmse))
```

Recosystem

The recosystem is an R wrapper of the LIBMF library developed by Yu-Chin Juan, Wei-Sheng Chin, Yong Zhuang, Bo-Wen Yuan, Meng-Yuan Yang, and Chih-Jen Lin [3]

LIBMF is a high-performance C++ library for large scale matrix factorization. LIBMF itself is a parallelized library, meaning that users can take advantage of multicore CPUs to speed up the computation. It also utilizes some advanced CPU features to further improve the performance. (Chin, Yuan, et al. 2015)

We will use the regularized rating as input into the recosystem.

```
mu <- mean(edx$rating)

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

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

# compute residuals on edx set
edx <- edx %>%
  left_join(b_i, by = "movieId") %>%
  left_join(b_u, by = "userId") %>%
  mutate(res = rating - mu - b_i - b_u)

# compute residuals on validation set
validation <- validation %>%
  left_join(b_i, by = "movieId") %>%
  left_join(b_u, by = "userId") %>%
  mutate(res = rating - mu - b_i - b_u)

# create data saved on disk in 3 columns with no headers
edx_data <- data_memory(user_index = edx$userId,
                               item_index = edx$movieId,
                               rating = edx$res,
                               index1 = T)

validation_data <- data_memory(user_index = validation$userId,
                               item_index = validation$movieId,
                               index1 = T)

# create a model object
recommender <- Reco()
```

```r
# This is a randomized algorithm
set.seed(1)

# call the `$tune()` method to select best tuning parameters
res = recommender$tune(
  edx_data,
  opts = list(dim = c(10, 20, 30),
              costp_l1 = 0, costq_l1 = 0,
              lrate = c(0.05, 0.1, 0.2), nthread = 3)
)

# show best tuning parameters
print(res$min)
```

```
## $dim
## [1] 30
##
## $costp_l1
## [1] 0
##
## $costp_l2
## [1] 0.01
##
## $costq_l1
## [1] 0
##
## $costq_l2
## [1] 0.1
##
## $lrate
## [1] 0.1
##
## $loss_fun
## [1] 0.7957179
```

```r
# Train the model by calling the `$train()` method
# some parameters coming from the result of `$tune()`
# This is a randomized algorithm
set.seed(1)
suppressWarnings(recommender$train(edx_data,
                                   opts = c(dim = res$min$dim, costp_l1 = res$min$costp_l1,
                                            costp_l2 = res$min$costp_l2,
                                            costq_l1 = res$min$costq_l1,
                                            costq_l2 = res$min$costq_l2,
                                            lrate = res$min$lrate,
                                            verbose = FALSE)))

# use the `$predict()` method to compute predicted values
# return predicted values in memory
predicted_ratings <- recommender$predict(validation_data, out_memory()) +
  mu + validation$b_i + validation$b_u

# ceiling rating at 5
ind <- which(predicted_ratings > 5)
```

```
predicted_ratings[ind] <- 5

# floor rating at 0.50
ind <- which(predicted_ratings < 0.5)
predicted_ratings[ind] <- 0.5

model_5_rmse <- RMSE(validation$rating, predicted_ratings)
model_5_rmse
```

```
## [1] 0.7866883
```

```
rsys_rmse_results <- bind_rows(rsys_rmse_results,
                               tibble(method="Recosystem - Regularized Movie and User",
                               RMSE = model_5_rmse))
```

Compare Models

The results show the Recosystem as the clear winner. However, using this method somes at a cost of more processing time. In part, this is mediated by the use of multiple threads where supported by the hardware.

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

method	RMSE
Just the average	1.0612018
Movie Effect Model	0.9439087
Movie + User Effects Model	0.8567039
Regularized Movie + User Effect Model	0.8566952
Recosystem - Regularized Movie and User	0.7866883

Sources

[1] Recommender System - Rishabh Mall https://towardsdatascience.com/recommender-system-a1e4595fc0f0

[2] GroupLens Research has collected and made available rating data sets from the MovieLens web site (http://movielens.org).

[3] recosystem Package (http://www.csie.ntu.edu.tw/~cjlin/libmf/), an open source library for recommender system using parallel marix factorization. (Chin, Yuan, et al. 2015)