Movies Recommendation System

Marcello Riderelli Belli

22/5/2020

Introduction

This work is using MovieLens 10M dataset:

https://grouplens.org/datasets/movielens/10m/

http://files.grouplens.org/datasets/movielens/ml-10m.zip

The aim of this work is to build the basic structure of a movie recommendation system. We'll start from importing and exploring data, then we'll introduce the model and apply it to show the issues related to the high variability of sample sizes due to the fact that only blockbuster movies have a lot of ratings while the number of ratings by user is very very variable. This will lead us to introduce a tuning parameter in the model and apply cross validation to optimize its value.

We'll reach the final result with a simple way to reduce the computation time which is one big issue with such the dimension of this data set.

Now let's start.

Import

The following code will import data in a data frame called "movielens" and save it in a file.

Please use getwd() command to know your working directory and create in it "R" and "R/Data" subdirectories to use the code as is

```
if (sum(dir("~/R/Data/") == "ML10M.Rdata") == 0) {

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")

save(movielens, file="~/R/Data/ML10M.Rdata")

}
```

Tidy and Transform

Now we can create edx set, which will be our training set, and validation set, our test set. Validation set will be 10% of MovieLens data

```
load(file = "~/R/Data/ML10M.RData")
paste("How many NA in movielens?", sum(is.na(movielens)))
```

```
## [1] "How many NA in movielens? 0"
```

```
options(digits=5)

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)
```

Finally let's clean Global Environment

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

Explore

We have a huge training set

```
dim(edx)
```

```
## [1] 9000055       6
```

```
head(edx) %>% knitr::kable()
```

	userId	movieId	rating	timestamp	title	genres
1	1	122	5	838985046	Boomerang (1992)	Comedy|Romance
2	1	185	5	838983525	Net, The (1995)	Action|Crime|Thriller
4	1	292	5	838983421	Outbreak (1995)	Action|Drama|Sci-Fi|Thriller
5	1	316	5	838983392	Stargate (1994)	Action|Adventure|Sci-Fi

	userId	movieId	rating	timestamp	title	genres
6	1	329	5	838983392	Star Trek: Generations (1994)	Action|Adventure|Drama|Sci-Fi
7	1	355	5	838984474	Flintstones, The (1994)	Children|Comedy|Fantasy

How many movies and users?

```
data.frame(Movies = length(unique(edx$movieId)), Users = length(unique(edx$userId))) %>% knitr::kable()
```

Movies	Users
10677	69878

Let's give a look at ratings by genres

```
edx %>%
  group_by(genres) %>%
  summarize(Num_Ratings = length(rating)) %>%
  summarize(Drama = sum(Num_Ratings[str_detect(genres,"Drama")]),
            Comedy = sum(Num_Ratings[str_detect(genres,"Comedy")]),
            Thriller = sum(Num_Ratings[str_detect(genres,"Thriller")]),
            Romance = sum(Num_Ratings[str_detect(genres,"Romance")])) %>%
  knitr::kable()
```

Drama	Comedy	Thriller	Romance
3910127	3540930	2325899	1712100

Most rated movies

```
edx %>%
  group_by(movieId) %>%
  summarize(Title = first(title),
            Num_Ratings = length(rating)) %>%
  arrange(desc(Num_Ratings)) %>%
  slice(1:20) %>%
  knitr::kable()
```

movieId	Title	Num_Ratings
296	Pulp Fiction (1994)	31362
356	Forrest Gump (1994)	31079
593	Silence of the Lambs, The (1991)	30382
480	Jurassic Park (1993)	29360
318	Shawshank Redemption, The (1994)	28015
110	Braveheart (1995)	26212
457	Fugitive, The (1993)	25998
589	Terminator 2: Judgment Day (1991)	25984
260	Star Wars: Episode IV - A New Hope (a.k.a. Star Wars) (1977)	25672
150	Apollo 13 (1995)	24284
592	Batman (1989)	24277
1	Toy Story (1995)	23790
780	Independence Day (a.k.a. ID4) (1996)	23449
590	Dances with Wolves (1990)	23367

movieId	Title	Num_Ratings
527	Schindler's List (1993)	23193
380	True Lies (1994)	22823
1210	Star Wars: Episode VI - Return of the Jedi (1983)	22584
32	12 Monkeys (Twelve Monkeys) (1995)	21891
50	Usual Suspects, The (1995)	21648
608	Fargo (1996)	21395

Most frequent ratings

```
edx %>%
  group_by(rating) %>%
  summarize(Occurance = n()) %>%
  arrange(desc(Occurance)) %>%
  knitr::kable()
```

rating	Occurance
4.0	2588430
3.0	2121240
5.0	1390114
3.5	791624
2.0	711422
4.5	526736
1.0	345679
2.5	333010
1.5	106426
0.5	85374

Full star ratings are more frequent than half star

```
edx %>%
  group_by(rating) %>%
  summarize(Occurance = n()) %>%
  arrange(desc(Occurance)) %>%
  summarize(Full_Star = sum(Occurance[(rating - round(rating)) == 0]),
            Half_Star = sum(Occurance[(rating - round(rating)) == 0.5])) %>%
  knitr::kable()
```

Full_Star	Half_Star
7156885	945120

Model

The model we use is linear, we start estimating the mean rating of all movies across all users, then considering that there are good movies and bad movies, we'll add a movie effect term and finally considering that users can be more or less "friendly" when rating but also and especially that users typically tend to rate movies that they liked and to skip rating when not, we'll add a user effect term.

The complete model is

$Y_{u,i} = \mu + b_i + b_u + \epsilon_{u,i}$

where μ is calculated as average rating of all movies across all users, which is the least square estimate of μ, that is the estimate that minimizes the root mean square error (RMSE)

b_i and b_u will be calculated as the average rating of movie "i" and users "u" respectively

Lets define our loss function: the root mean square error

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

As said, in the most basic model, the average of all the ratings is used as the estimate of μ

```
mu_hat <- mean(edx$rating)
naive_rmse <- RMSE(validation$rating, mu_hat)

rmse_results <- data_frame(Method = "Full Set - Just the average",
                           RMSE = naive_rmse)

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

Method	RMSE
Full Set - Just the average	1.0612

Let's add that some movies are rated better or worse than others

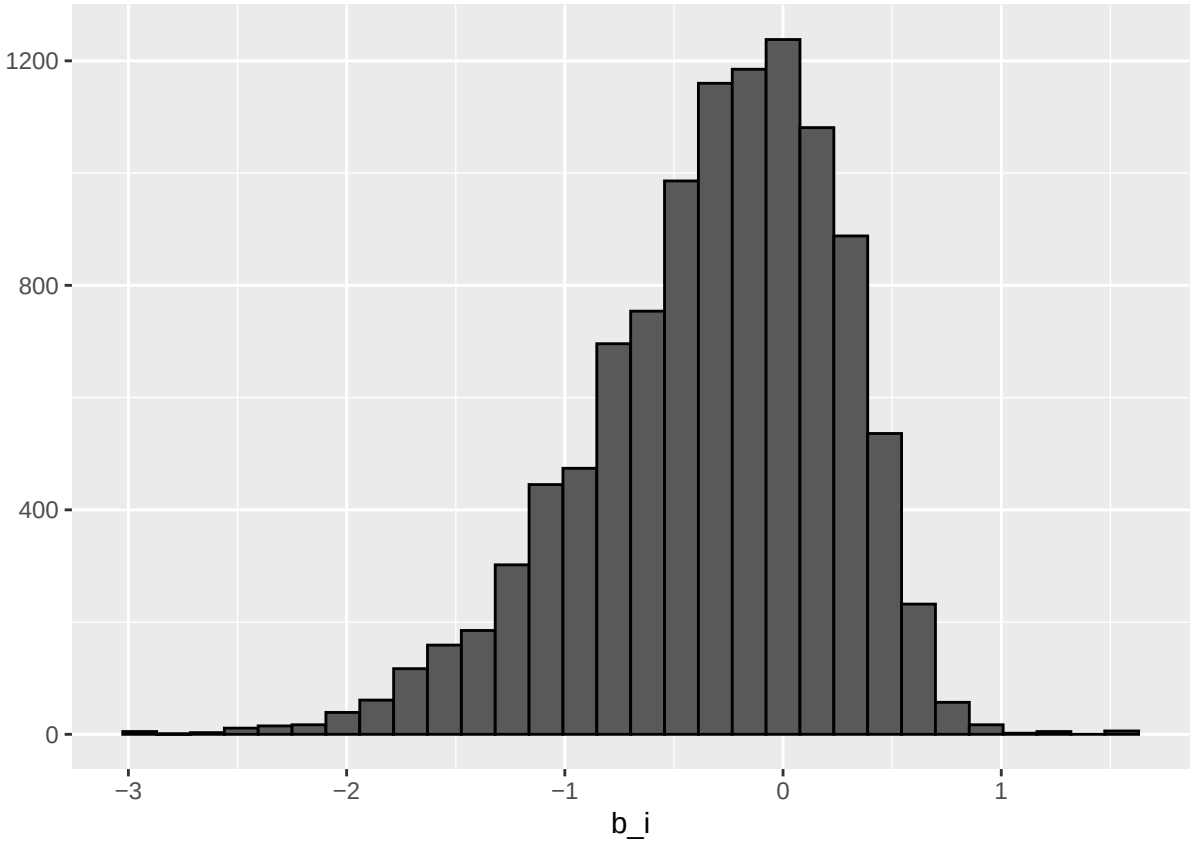
```
mu <- mean(edx$rating)
b_i <-
  edx %>%
  group_by(movieId) %>%
  summarize(b_i = mean(rating - mu))

predicted_ratings <- validation %>%
  left_join(b_i, by='movieId') %>%
  mutate(pred = mu + b_i) %>%
  .$pred

model_1_rmse <- RMSE(predicted_ratings, validation$rating)
rmse_results <- bind_rows(rmse_results,
                          data_frame(Method="Full Set - Movie Effect Model",
                                     RMSE = model_1_rmse ))

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

Method	RMSE
Full Set - Just the average	1.06120
Full Set - Movie Effect Model	0.94391

The following histogram shows the distribution and hence the variability respect to the average, of ratings by movie. We can see that the bigger portion of movies has ratings below the average.

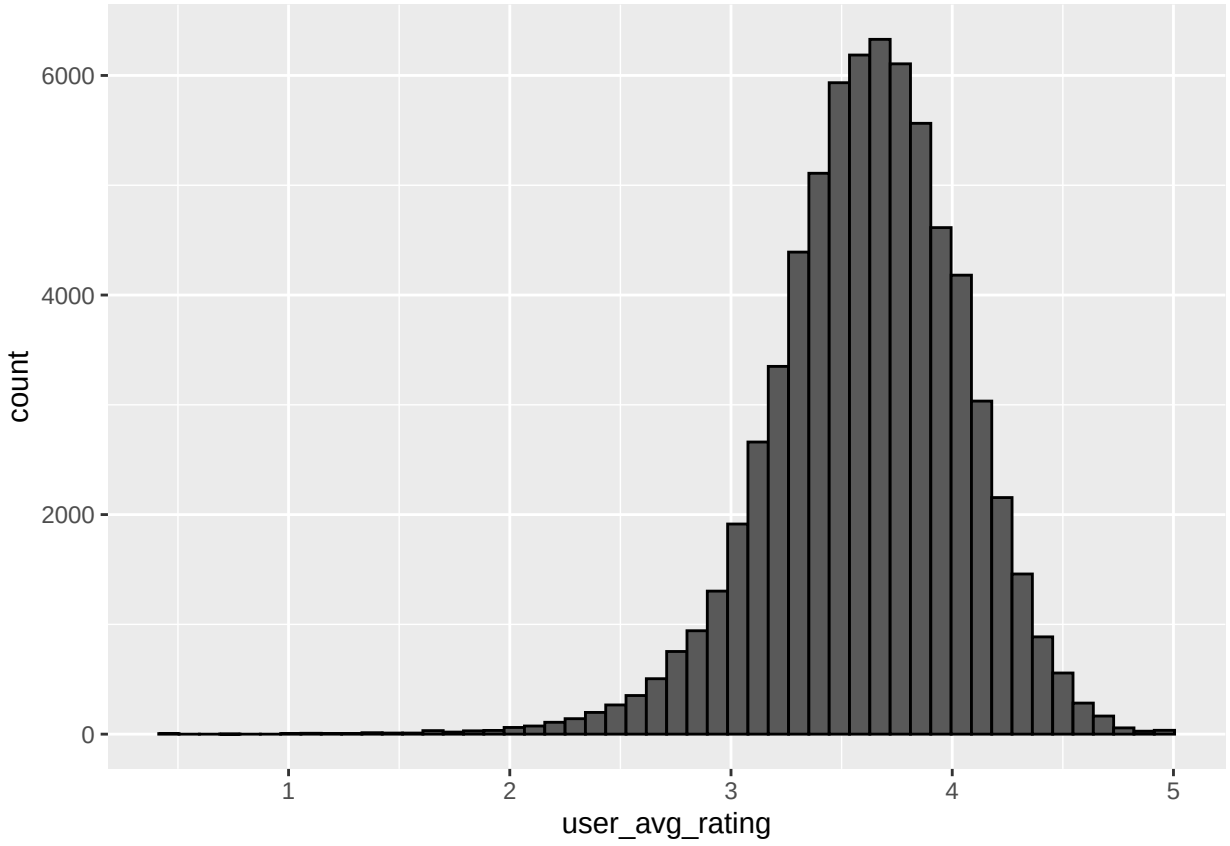
```
b_i %>% qplot(b_i,
              geom ="histogram",
              bins = 30,
```

```
            data = .,
            color = I("black"))
```



Now let's give a look to the distribution of the average rating of each user

```
edx %>%
  group_by(userId) %>%
  summarize(user_avg_rating = mean(rating)) %>%
  ggplot(aes(user_avg_rating)) +
  geom_histogram(bins = 50, color = "black")
```



Probably users rate movies more frequently when they like it.

Let's add the user effect

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

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

model_2_rmse <- RMSE(predicted_ratings, validation$rating)
rmse_results <- bind_rows(rmse_results,
                          data_frame(Method="Full Set - Movie & User Effects Model",
                                     RMSE = model_2_rmse ))
rmse_results %>% knitr::kable()
```

Method	RMSE
Full Set - Just the average	1.06120
Full Set - Movie Effect Model	0.94391
Full Set - Movie & User Effects Model	0.86535

Let's give a closer look at our model, we'll recommend movies with higher b_i So let's check which movies have the highest b_i ...

```
movie_titles <- edx %>%
  select(movieId, title) %>%
  distinct()
```

Top 10 best movies

```
b_i %>%
  left_join(movie_titles, by="movieId") %>%
  arrange(desc(b_i)) %>%
  select(title, b_i) %>%
  slice(1:10) %>%
  knitr::kable()
```

title	b_i
Hellhounds on My Trail (1999)	1.4875
Satan's Tango (SÃ¡tÃ¡ntangÃ³) (1994)	1.4875
Shadows of Forgotten Ancestors (1964)	1.4875
Fighting Elegy (Kenka erejii) (1966)	1.4875
Sun Alley (Sonnenallee) (1999)	1.4875
Blue Light, The (Das Blaue Licht) (1932)	1.4875
Who's Singin' Over There? (a.k.a. Who Sings Over There) (Ko to tamo peva) (1980)	1.2375
Human Condition II, The (Ningen no joken II) (1959)	1.2375
Human Condition III, The (Ningen no joken III) (1961)	1.2375
Constantine's Sword (2007)	1.2375

... and which one will have the lowest b_i

Top 10 worst movies

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

title	b_i
Besotted (2001)	-3.0125
Hi-Line, The (1999)	-3.0125
Accused (Anklaget) (2005)	-3.0125
Confessions of a Superhero (2007)	-3.0125
War of the Worlds 2: The Next Wave (2008)	-3.0125
SuperBabies: Baby Geniuses 2 (2004)	-2.7178
Hip Hop Witch, Da (2000)	-2.6910
Disaster Movie (2008)	-2.6531
From Justin to Kelly (2003)	-2.6105
Criminals (1996)	-2.5125

There is something weird: we don't recognize any movie among the best or the worst.

The point is, when calculating b_i or b_u we are grouping and averaging without considering the sample size,

that is how many ratings the movie had or how many ratings the user did.

How many ratings the best movies had?

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

title	b_i	n
Hellhounds on My Trail (1999)	1.4875	1
Satan's Tango (SÃ¡tÃ¡ntangÃ³) (1994)	1.4875	2
Shadows of Forgotten Ancestors (1964)	1.4875	1
Fighting Elegy (Kenka erejii) (1966)	1.4875	1
Sun Alley (Sonnenallee) (1999)	1.4875	1
Blue Light, The (Das Blaue Licht) (1932)	1.4875	1
Who's Singin' Over There? (a.k.a. Who Sings Over There) (Ko to tamo peva) (1980)	1.2375	4
Human Condition II, The (Ningen no joken II) (1959)	1.2375	4
Human Condition III, The (Ningen no joken III) (1961)	1.2375	4
Constantine's Sword (2007)	1.2375	2

How many ratings the worst movies had?

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

title	b_i	n
Besotted (2001)	-3.0125	2
Hi-Line, The (1999)	-3.0125	1
Accused (Anklaget) (2005)	-3.0125	1
Confessions of a Superhero (2007)	-3.0125	1
War of the Worlds 2: The Next Wave (2008)	-3.0125	2
SuperBabies: Baby Geniuses 2 (2004)	-2.7178	56
Hip Hop Witch, Da (2000)	-2.6910	14
Disaster Movie (2008)	-2.6531	32
From Justin to Kelly (2003)	-2.6105	199
Criminals (1996)	-2.5125	2

Regularization

First of all we need a more manageable dataset. Let's take ramdomly 300,000 samples from the edx set and create new training and test sets

```
edx_subset <- sample(1:nrow(edx), 300000, replace = FALSE)
edx_1 <- edx[edx_subset, ]

tst_idx <- createDataPartition(y = edx_1$rating,
                               times = 1,
                               p = 0.2,
                               list = FALSE)
trn_set <- edx_1[-tst_idx, ]
tst_set <- edx_1[tst_idx, ]

tst_set <- tst_set %>%
  semi_join(trn_set, by = "movieId") %>%
  semi_join(trn_set, by = "userId")
```

Now we remake the calculations of the same methods above using the new small dataset

```
mu_hat <- mean(trn_set$rating)
naive_rmse <- RMSE(tst_set$rating, mu_hat)
rmse_results <- data_frame(method = "Small Set - Just the average", RMSE = naive_rmse)

# Movie Effect
mu <- mean(trn_set$rating)

b_i <- trn_set %>%
  group_by(movieId) %>%
  summarize(b_i = mean(rating - mu))

predicted_ratings <- mu + tst_set %>%
  left_join(b_i, by='movieId') %>%
  .$b_i

model_1_rmse <- RMSE(predicted_ratings, tst_set$rating)
rmse_results <- bind_rows(rmse_results,
                          data_frame(method="Small Set - Movie Effect Model",
                                     RMSE = model_1_rmse ))

# Movie and User effect
b_u <- trn_set %>%
  left_join(b_i, by='movieId') %>%
  group_by(userId) %>%
  summarize(b_u = mean(rating - mu - b_i))

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

model_2_rmse <- RMSE(predicted_ratings, tst_set$rating)
rmse_results <- bind_rows(rmse_results,
                          data_frame(method="Small Set - Movie & User Effects Model",
                                     RMSE = model_2_rmse ))

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

method	RMSE
Small Set - Just the average	1.05508
Small Set - Movie Effect Model	0.95225
Small Set - Movie & User Effects Model	0.97088

We have a more manageable dataset but not considering the number of rating yet, in fact ...

```
movie_titles <- edx_1 %>%
  select(movieId, title) %>%
  distinct()
```

Top 10 best movies

```
edx_1 %>%
  dplyr::count(movieId) %>%
  left_join(b_i) %>%
  left_join(movie_titles, by="movieId") %>%
  arrange(desc(b_i)) %>%
  select(title, b_i, n) %>%
  slice(1:10) %>%
  knitr::kable()
```

title	b_i	n
Nobody Loves Me (Keiner liebt mich) (1994)	1.4849	1
Dangerous Game (1993)	1.4849	1
I Don't Want to Talk About It (De eso no se habla) (1993)	1.4849	1
Asfour Stah (1990)	1.4849	2
Tarantella (1995)	1.4849	1
They Bite (1996)	1.4849	1
Hedd Wyn (1992)	1.4849	1
Small Faces (1996)	1.4849	2
Sweet Nothing (1996)	1.4849	2
Normal Life (1996)	1.4849	1

Top 10 worst movies

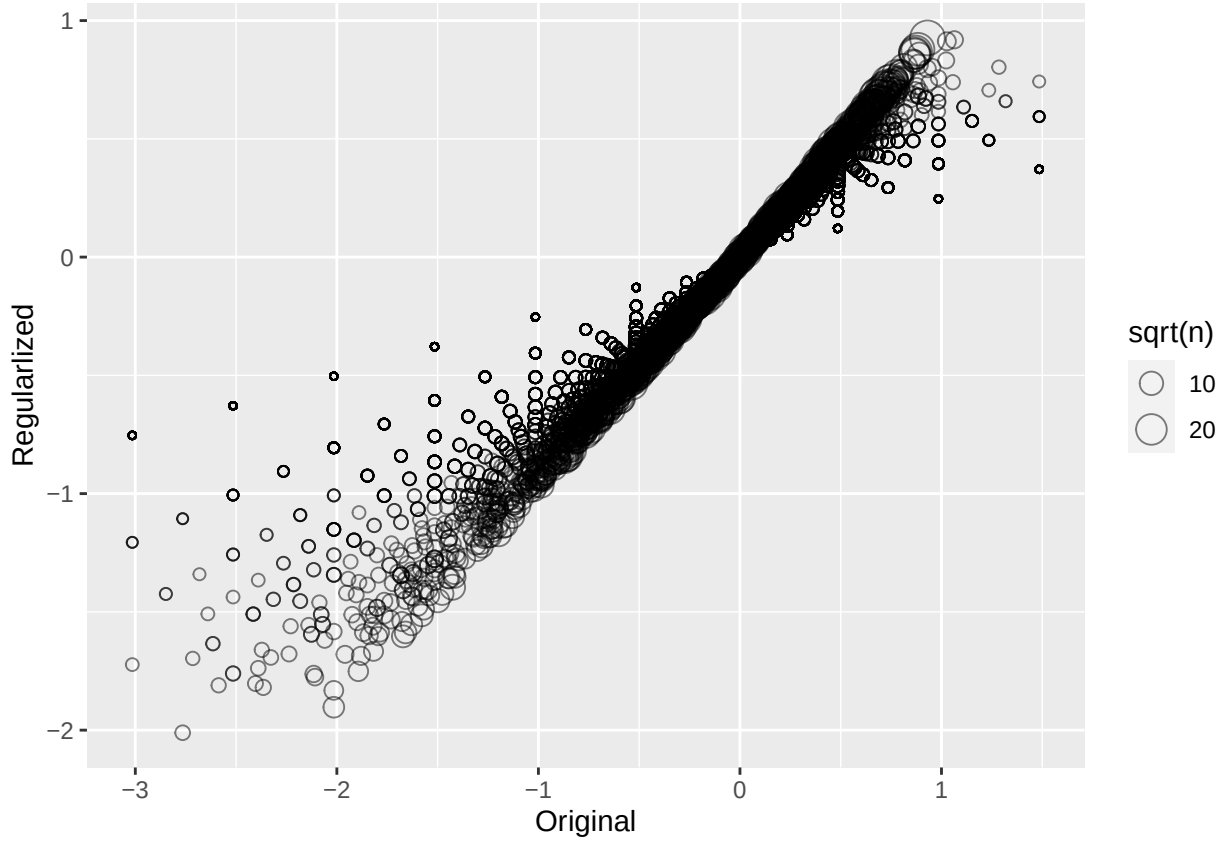
```
edx_1 %>%
  dplyr::count(movieId) %>%
  left_join(b_i) %>%
  left_join(movie_titles, by="movieId") %>%
  arrange(b_i) %>%
  select(title, b_i, n) %>%
  slice(1:10) %>%
  knitr::kable()
```

title	b_i	n
No Looking Back (1998)	-3.0151	1
Goodbye Lover (1999)	-3.0151	3
Simon Sez (1999)	-3.0151	1
My Tutor (1983)	-3.0151	1
Taffin (1988)	-3.0151	1
American Virgin (2000)	-3.0151	1

title	b_i	n
Turn It Up (2000)	-3.0151	1
Slumber Party Massacre III (1990)	-3.0151	1
Wendigo (2001)	-3.0151	3
Burial Ground (a.k.a. Zombie Horror) (a.k.a. Zombie 3) (Notti del Terrore, Le) (1981)	-3.0151	2

We need to add a penalization term for small sample size when calculating the averages. Let's try a possible value for regularization, just to see how it works. Looking at the plot the action of the tuning parameter λ is the shirinking of high estimates when the sample size is small

```
lambda <- 3

b_i_reg <- trn_set %>%
  group_by(movieId) %>%
  summarize(b_i = sum(rating - mu)/(n()+lambda), n_i = n())

data_frame(Original = b_i$b_i,
           Regularlized = b_i_reg$b_i,
           n = b_i_reg$n_i) %>%
  ggplot(aes(Original, Regularlized, size=sqrt(n))) +
  geom_point(shape=1, alpha=0.5)
```



Now we give a look to the ranking after this non optimal regularization. It's just a trial.

Top 10 best movies after regularization

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

title	b_i	n
Shawshank Redemption, The (1994)	0.92676	753
My Man Godfrey (1936)	0.91878	19
General, The (1927)	0.91250	24
Usual Suspects, The (1995)	0.87814	582
Schindler's List (1993)	0.86697	569
Godfather, The (1972)	0.86159	467
One Flew Over the Cuckoo's Nest (1975)	0.86037	358
Sunset Blvd. (a.k.a. Sunset Boulevard) (1950)	0.85898	84
Palm Beach Story, The (1942)	0.83148	13
Third Man, The (1949)	0.83025	84

Top 10 worst movies after regularization

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

title	b_i	n
House of the Dead, The (2003)	-2.0110	8
Battlefield Earth (2000)	-1.9031	51
Jaws 3-D (1983)	-1.8319	30
Glitter (2001)	-1.8193	10
FearDotCom (a.k.a. Fear.com) (a.k.a. Fear Dot Com) (2002)	-1.8106	7
Meatballs III (1987)	-1.8030	9
Baby Geniuses (1999)	-1.7759	16
Psycho III (1986)	-1.7626	15
PokÃ©mon 3: The Movie (2001)	-1.7606	7
Kangaroo Jack (2003)	-1.7606	7

Let's apply the movie effect to the small dataset after regularization

```
predicted_ratings <- tst_set %>%
  left_join(b_i_reg, by='movieId') %>%
  mutate(pred = mu + b_i) %>%
  .$pred
```

```
model_3_rmse <- RMSE(predicted_ratings, tst_set$rating)
rmse_results <- bind_rows(rmse_results,
                          data_frame(method="Small Set - Regularized Movie Effect Model",
                                     RMSE = model_3_rmse ))
rmse_results %>% knitr::kable()
```

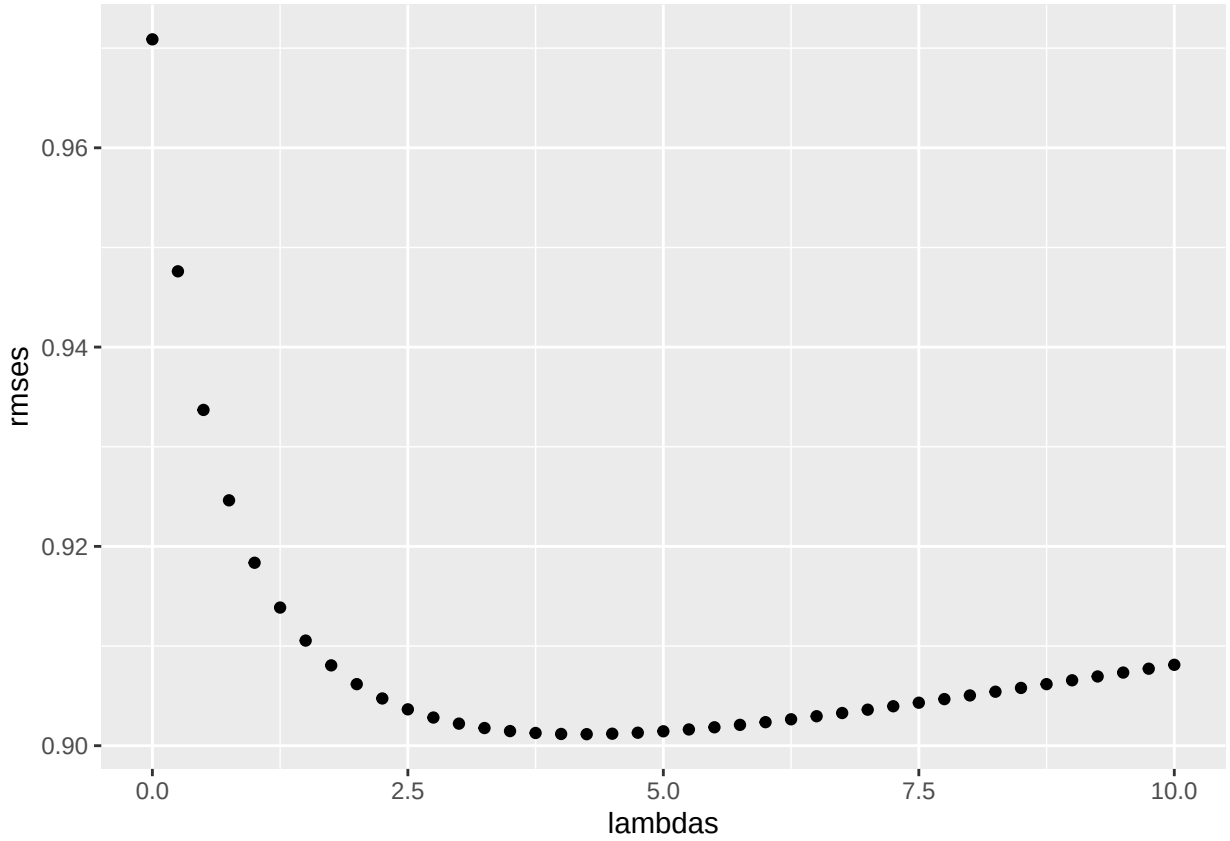
method	RMSE
Small Set - Just the average	1.05508
Small Set - Movie Effect Model	0.95225
Small Set - Movie & User Effects Model	0.97088
Small Set - Regularized Movie Effect Model	0.94695

It works but the trial value is probably not optimal so let's use cross validation to choose a better λ. The use of a small dataset is really good to reduce the computation time

```
lambdas <- seq(0, 10, 0.25)
mu <- mean(trn_set$rating)

reg_fun <- function(l) {

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

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

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

  return(RMSE(predicted_ratings, tst_set$rating))
}

rmses <- sapply(lambdas, reg_fun)
```

Let's see if there is a clear minimum

```
qplot(lambdas, rmses)
```




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

```
## [1] 4.25
```

yes there is and the updated table of results is

```
rmse_results <- bind_rows(rmse_results,
                          data_frame(method="Small Set - Regularized Movie & User Effect Model",
                                     RMSE = min(rmses)))
rmse_results %>% knitr::kable()
```

method	RMSE
Small Set - Just the average	1.05508
Small Set - Movie Effect Model	0.95225
Small Set - Movie & User Effects Model	0.97088
Small Set - Regularized Movie Effect Model	0.94695
Small Set - Regularized Movie & User Effect Model	0.90116

We finally apply the same value of λ but we use the full dataset

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

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

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

predicted_ratings <-
  validation %>%
  left_join(b_i_reg, by = "movieId") %>%
  left_join(b_u_reg, by = "userId") %>%
  mutate(pred = mu + b_i + b_u) %>%
  .$pred

model_4_rmse <- RMSE(predicted_ratings, validation$rating)
rmse_results <- bind_rows(rmse_results,
                          data_frame(method="Full Set - Regularized Movie & User Effect Model",
                                     RMSE = model_4_rmse ))
rmse_results %>% knitr::kable()
```

method	RMSE
Small Set - Just the average	1.05508
Small Set - Movie Effect Model	0.95225
Small Set - Movie & User Effects Model	0.97088
Small Set - Regularized Movie Effect Model	0.94695
Small Set - Regularized Movie & User Effect Model	0.90116
Full Set - Regularized Movie & User Effect Model	0.86483

Let's finally check what are the best and worst movies

```
movie_titles <- edx %>%
  select(movieId, title) %>%
  distinct()
```

Top 10 best movies after regularization

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

title	b_i	n
Shawshank Redemption, The (1994)	0.94252	28015
Godfather, The (1972)	0.90268	17747
Usual Suspects, The (1995)	0.85322	21648
Schindler's List (1993)	0.85087	23193
Casablanca (1942)	0.80765	11232
Rear Window (1954)	0.80575	7935
Sunset Blvd. (a.k.a. Sunset Boulevard) (1950)	0.80225	2922
Third Man, The (1949)	0.79782	2967
Double Indemnity (1944)	0.79678	2154

title	b_i	n
Paths of Glory (1957)	0.79411	1571

Top 10 worst movies after regularization

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

title	b_i	n
From Justin to Kelly (2003)	-2.5559	199
SuperBabies: Baby Geniuses 2 (2004)	-2.5261	56
PokÃ©mon Heroes (2003)	-2.4085	137
Disaster Movie (2008)	-2.3420	32
Glitter (2001)	-2.3080	339
Gigli (2003)	-2.2881	313
Pokemon 4 Ever (a.k.a. PokÃ©mon 4: The Movie) (2002)	-2.2862	202
Carnosaur 3: Primal Species (1996)	-2.2816	68
Barney's Great Adventure (1998)	-2.2784	208
Yu-Gi-Oh! (2004)	-2.1661	80

The final RMSE is below the target (0.86490) and the movie ranking now makes sense and is consistent with all the considerations made along the report.