MovieLens Project

Antonio Caputo

22/9/2019

Executive Summary

For this project, we will be creating a movie recommendation system using the MovieLens dataset. Recommendation systems use ratings that *users* have given *items* to make specific recommendations. Companies that sell many products to many customers and permit these customers to rate their products. Items for which a high rating is predicted for a given user are then recommended to that user.

We will use the 10M version of the MovieLens dataset to make the computation a little easier.

We will use the following code to generate our datasets:

```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(matrixStats)
library(tidyverse)
library(caret)
library(data.table)
library(dplyr)
library(tinytex)
library(lubridate)

# 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(levels(movieId))[movieId],
    title = as.character(title), genres = as.character(genres))

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

We can see this table format:

```r
print(movielens[1:10, ])
```

```
##    userId movieId rating timestamp                         title
## 1       1     122      5 838985046              Boomerang (1992)
## 2       1     185      5 838983525               Net, The (1995)
## 3       1     231      5 838983392          Dumb & Dumber (1994)
## 4       1     292      5 838983421               Outbreak (1995)
## 5       1     316      5 838983392               Stargate (1994)
## 6       1     329      5 838983392 Star Trek: Generations (1994)
## 7       1     355      5 838984474       Flintstones, The (1994)
## 8       1     356      5 838983653           Forrest Gump (1994)
## 9       1     362      5 838984885       Jungle Book, The (1994)
## 10      1     364      5 838983707         Lion King, The (1994)
##                                           genres
## 1                                 Comedy|Romance
## 2                          Action|Crime|Thriller
## 3                                         Comedy
## 4                   Action|Drama|Sci-Fi|Thriller
## 5                        Action|Adventure|Sci-Fi
## 6                  Action|Adventure|Drama|Sci-Fi
## 7                        Children|Comedy|Fantasy
## 8                       Comedy|Drama|Romance|War
## 9                     Adventure|Children|Romance
## 10 Adventure|Animation|Children|Drama|Musical
```

So we can think of these data as a very large matrix, with users on the rows and movies on the columns, with many empty cells.

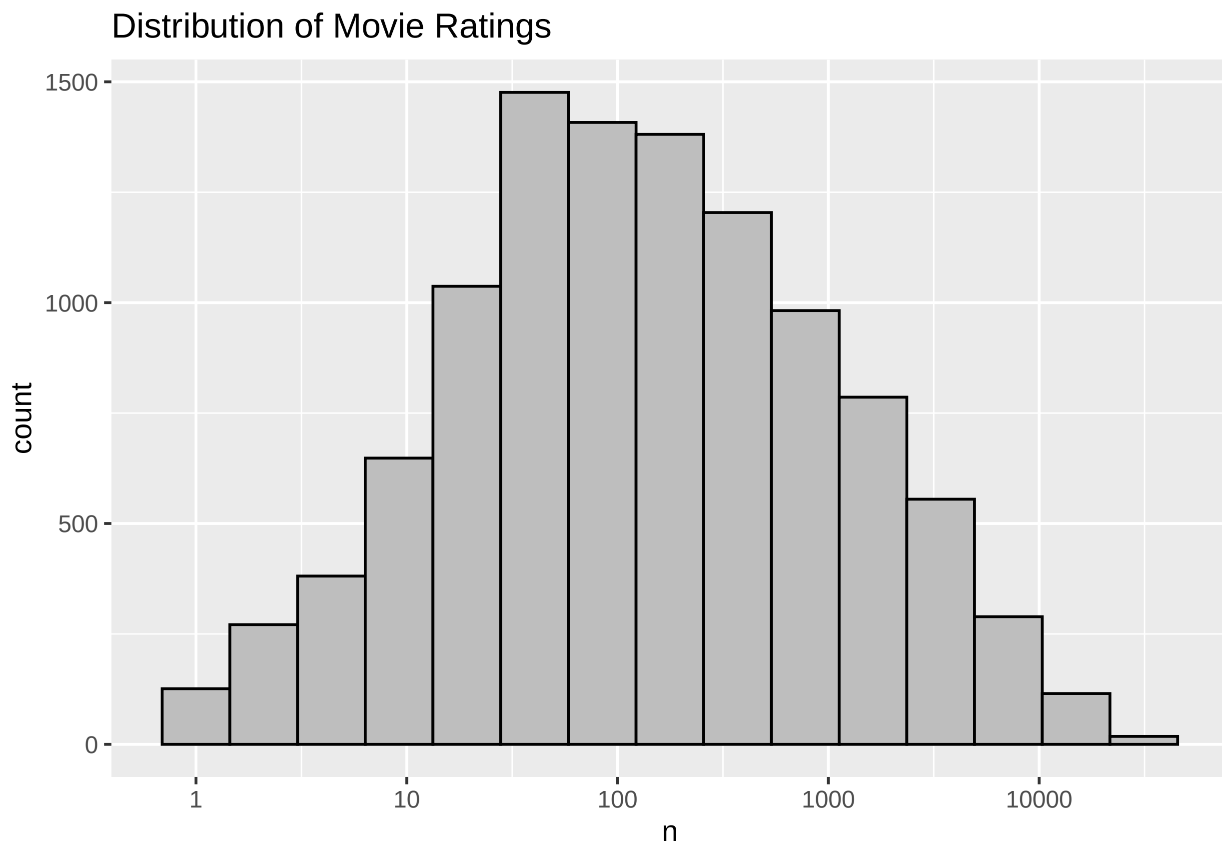
We can think of the task of a recommendation system as filling in the NAs in the table above.

We need to build an algorithm with data we have collected that will then be applied outside our control, as users look for movie recommendations. So let's create a test set to assess the accuracy of the models we implement.

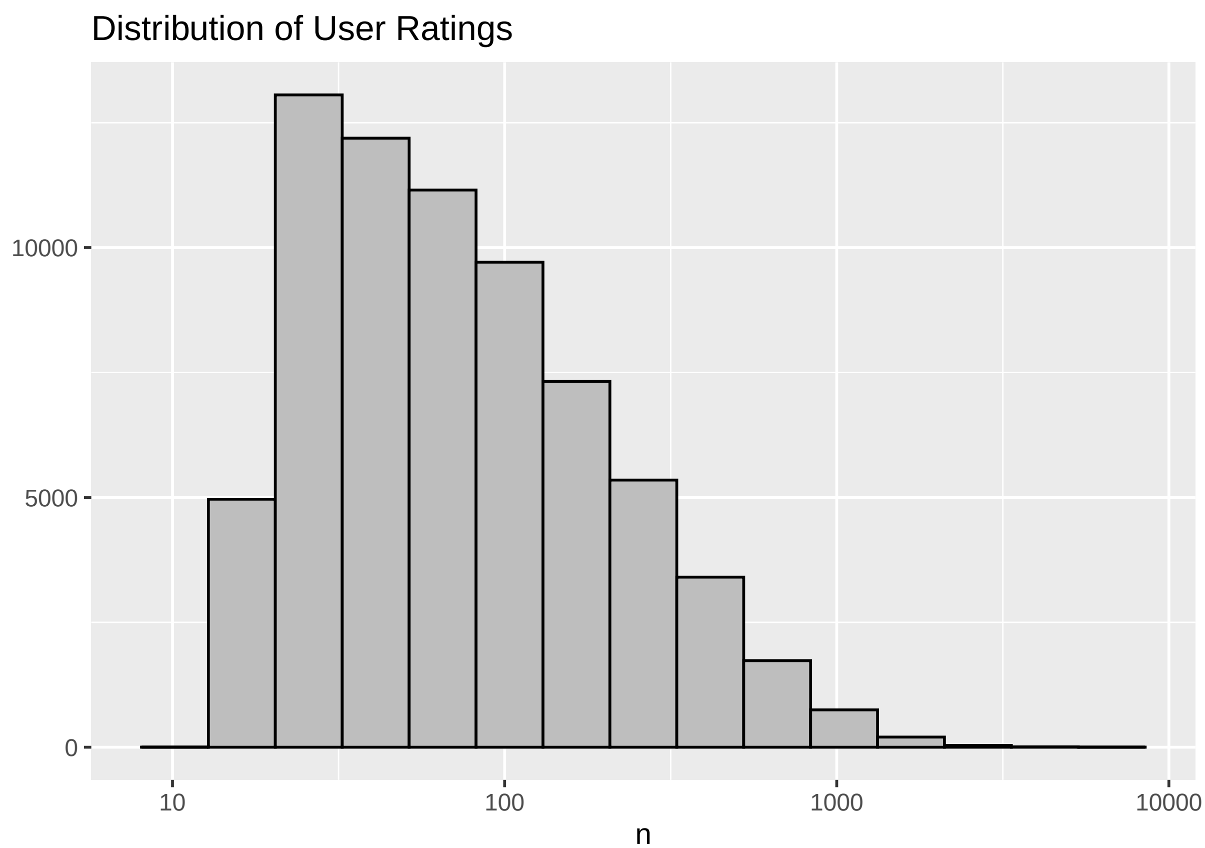
```r
# Validation set will be 10% of MovieLens data

set.seed(1, sample.kind = "Rounding")
# if using R 3.5 or earlier, use `set.seed(1)` instead
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)
```

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

Let's look at some of the general properties of the data to better understand the challenges. The first thing we notice is that some movies get rated more than others. Here is the distribution:



Our second observation is that some users are more active than others at rating movies:



Methods/Analysis

We will develop our algorithm using the *edx* set. For a final test of your algorithm, we will predict movie ratings in the validation set as if they were unknown. RMSE will be used to evaluate how close our predictions are to the true values in the validation set. We define $y_{u,i}$ as the rating for movie i by user u and denote our prediction with $\hat{y}_{u,i}$. The RMSE is then defined as: RMSE =

$$RMSE = \sqrt{\frac{1}{N}\sum_{u,i}(\hat{y}_{u,i} - y_{u,i})^2}$$

with N being the number of user/movie combinations and the sum occurring over all these combinations.

Let's write a function that computes the RMSE for vectors of ratings and their corresponding predictors:

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

Our algorithm has to take accouting of different factors. We start from a very basic model and will add more factors going haed. Our simplest model is:

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

We know that the estimate that minimizes the RMSE is the least squares estimate of *??* and, in this case, is the average of all ratings.

```r
mu_hat <- mean(edx$rating)
mu_hat
```

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

If we predict all unknown ratings with ˆ?? we obtain the following RMSE:

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

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

Now,we will add more informations and effects.

Movie Effects

We know from experience that some movies are just generally rated higher than others. This intuition, that different movies are rated differently, is confirmed by data. We can model by adding the term b_i to represent average ranking for movie i:

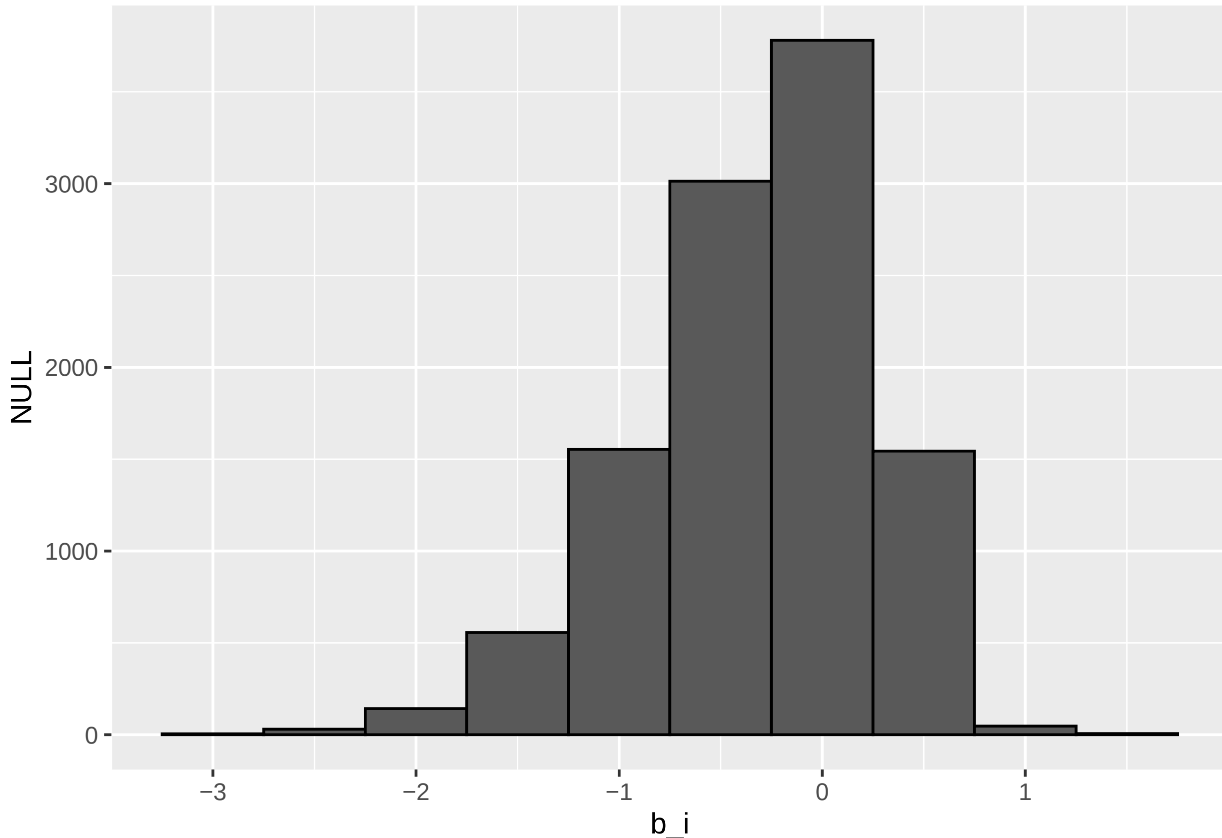
$$Y_{u,i} = \mu + b_i + \epsilon_{u,i}$$

we know that the least square estimate $\hat{b}_i$ is just the average of $Y_{u,i}$???$\hat{\mu}$ for each movie i. So we can compute them this way:

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

We can see that these estimates vary substantially:

```r
movie_avgs %>% qplot(b_i, geom = "histogram", bins = 10, data = .,
    color = I("black"))
```



Let's see how much our prediction improves:

```
predicted_ratings <- mu + validation %>% left_join(movie_avgs,
    by = "movieId") %>% pull(b_i)

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

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

User Effects

There is substantial variability across users as well: some users are very cranky and others love every movie. This implies that a further improvement to our model may be:

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

where b_u is a user-specific effect.

we will compute an approximation by computing $\hat{\mu}$ and $\hat{b}_i$ and estimating $\hat{b}_u$ as the average of $Y_{u,i}???\hat{\mu} - \hat{b}_i$:

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

We can now construct predictors and see how much the RMSE improves:

```r
predicted_ratings <- validation %>% left_join(movie_avgs, by = "movieId") %>%
    left_join(user_avgs, by = "userId") %>% mutate(pred = mu +
    b_i + b_u) %>% pull(pred)

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

```
## [1] 0.8653488
```

Regularization

Despite the large movie to movie variation, our improvement in RMSE was very low. Tis is becouse some movies are rated by very few users, in most cases just 1, so wa have more uncertainty. Therefore, larger estimates of b_i, negative or positive, are more likely. These are noisy estimates that we should not trust, especially when it comes to prediction. Large errors can increase our RMSE, so we would rather be conservative when unsure. For this, we introduce the concept of regularization, to penalize large estimates that are formed using small sample sizes.

Results

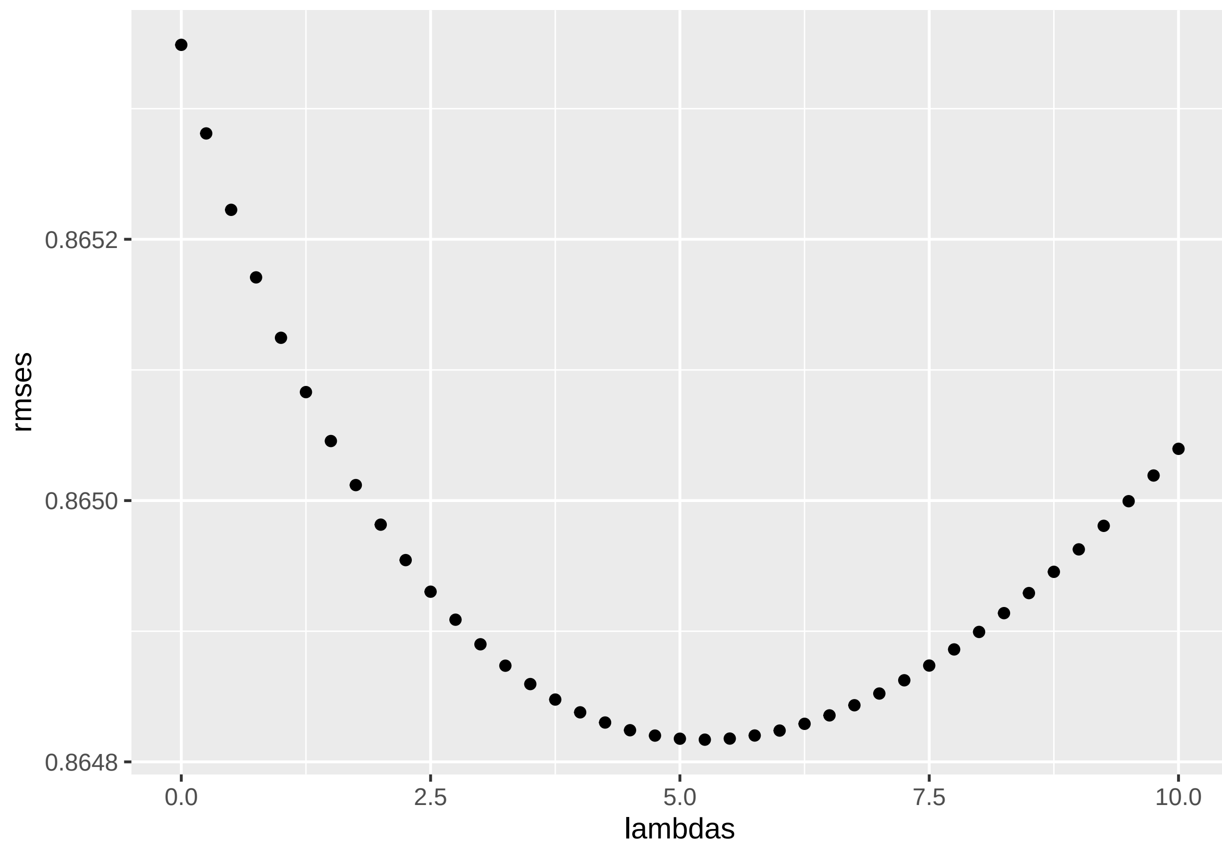
The general idea of penalized regression is to control the total variability of the effects.

Definitely, We are minimizing:

$$\frac{1}{N}\sum_{u,i}(y_{u,i} - \mu - b_i - b_u) + \lambda\left(\sum_i b_i^2 + \sum_i b_u^2\right)$$

The estimates that minimize this can be found using cross-validationto pick a λ:

```r
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 <- validation %>% left_join(b_i, by = "movieId") %>%
        left_join(b_u, by = "userId") %>% mutate(pred = mu +
        b_i + b_u) %>% pull(pred)

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

qplot(lambdas, rmses)
```




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

```
## [1] 5.25
```

In the end we obtain:

```
min(rmses)
```

```
## [1] 0.864817
```

So we improved our system a little bit more.

We can see the difference beetwin all model in the table below:

```
options(digits = 4)
rmse_results <- tibble(method = "Just the average", RMSE = naive_rmse)
rmse_results <- bind_rows(rmse_results, tibble(method = "Movie + User Effects Model",
    RMSE = model_2_rmse))
rmse_results <- bind_rows(rmse_results, tibble(method = "Regularized Movie + User Effect Model",
    RMSE = min(rmses)))

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

method	RMSE
Just the average	1.0612
Movie + User Effects Model	0.8653
Regularized Movie + User Effect Model	0.8648

Conclusion

In conclusion we reached a RMSE of 0.8648 less than 0.8649, a good result in confront of the start point we saw with the basic model (RMSE= 1.0312). We reached this result adding different bias and regularization. Nevertheless our model does not take into account important source of variation related to the fact that groups of movie have similar rating patterns and groups of users ghave similar patterns as well. We could discover these patterns by studying the residuals and see that there is structure in the data. We could model this structure in the future with actor analysis, singular value decomposition (SVD) and principal component analysis (PCA).