MovieLens Project for edX Data Science 125x.9

Tasha Vincent

January 26, 2019

Introduction

The purpose of this project is to replicate and improve upon the results of the MovieLens recommendation system introduced in edX Data Science 125x.8. The dataset used was as described in the setup instructions for the MovieLens project. In addition to modeling the effects of movie-specific and user-specific ratings, I also explored the effect of genre on the ratings. This involved adding additional packages such as tidyr, and stringr, to manipulate the genres into a single variable that could be analyzed.

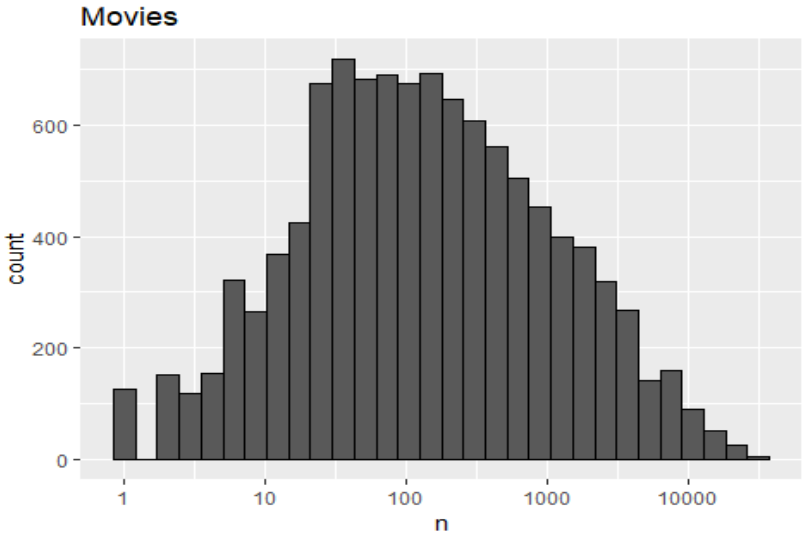
The quality of the model was evaluated based on RMSE rating, with a target result of RMSE <= 0.87750. Results were summarized separately as each term was added to the model.

Methods and Analysis

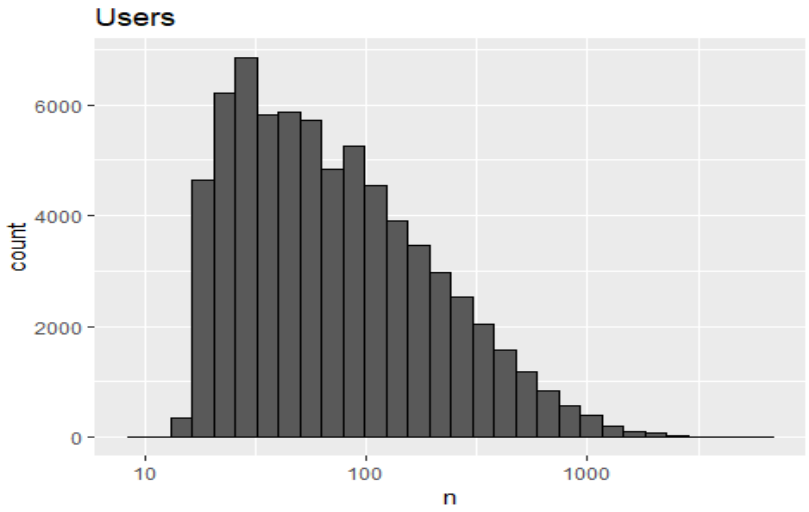
Per the instructions for this project, the MovieLens dataset was accessed and manipulated to retrieve a subset of the data. I partitioned the data to create a subset of 10% of the data to validate the algorithm developed in the course of the project. To ensure that no movies with only 1 rating ended up in the test data, I made a list of single-rating movies to keep them out of validation set, then split the data, ensuring as well that both the training (edx) set and the validation set contained the same users and movies.

Next I explored the data's contents and dimensions to generate hypothetical effects on predicted ratings. I created a visualization of the count of ratings per movie to see if there was a pattern.

The single-rating movies stand out at the left.



I then looked at the number of ratings per user. It's clear from the graph that many users only evaluate one to two movies, but the majority of people rate multiple movies. A few superstars rate thousands of them.



Evaluating Results

Since we are using residual means of the squared estimate (RMSE) to measure the quality of the algorithm, I created a function to evaluate whether the predictions generated are close to the actual ratings for a given movie and user.

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

Models

Initially, I started with a basic model that predicts that any movie will get the average rating, `mu`.

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

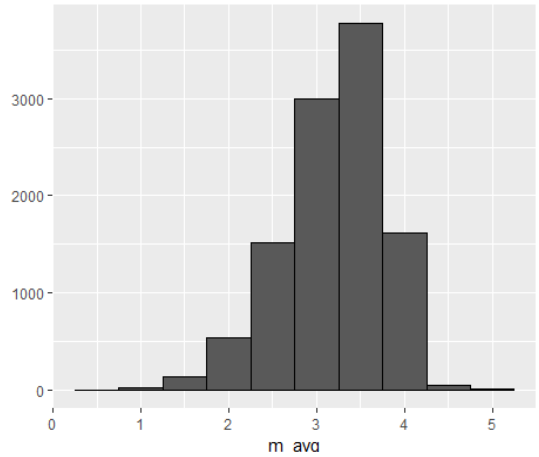
I then applied the function that I created to determine how well this model performs.

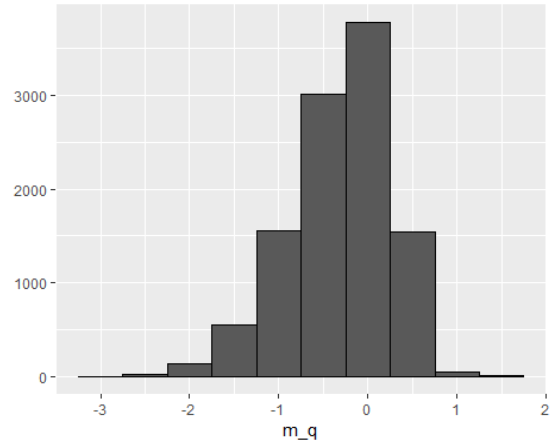
```
## [1] 1.061202
```

To validate that this simple model it is better than guessing, I plugged in various values of ratings. The result is a larger RMSE, so mu is better than guessing.

So that I may store and easily compare and report results from various models, I created a data frame, and added the first results.

Movie Effect





I hypothesized that the inherent quality of the movie should be a big factor in determining the ratings awarded by users.to confirm that there is a distribution of ratings by movie, I created a visualization showing the number of ratings within each band.

The ratings appear to approximate a normal distribution, so a movie-specific weighting is appropriate. I then created a term for movie quality, m_q defined as the difference in average rating for a given movie from the average rating of all movies, mu.

To confirm that this was on the right track, I created a visualization to verify that the distribution of the movie quality term matches the distribution of average movie ratings overall.

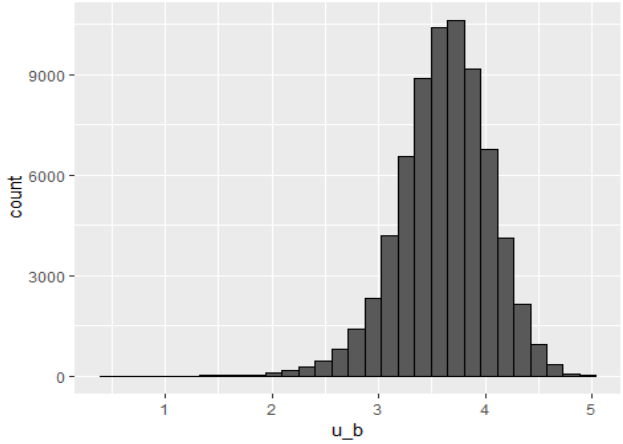
I added the new term to the model to see how much the prediction improves using `predictions = mu + m_q` and evaluated the model.

Model	RMSE
Simple average	1.0612018
Movie Effect Model	0.9439087

Users

Next I hypothesized that there is a 'set point' of ratings by user; in other words, some users may love movies, or just be favorably predisposed to award higher ratings, and others may never rate above a 3. I created a graph to see how much a given user varies from the average rating for all users.

Although there is clearly a wide range of average ratings per user, there does not seem to be any more distribution if user only rated 10, 5 or even 1 film. The filter does not change the distribution.



I defined the user bias per movie (u_b) as the difference between the user's average rating and the overall average and the movie-specific average.

Next I added the user term to the model to see how much the prediction improves using `predictions = mu + m_q + u_b` and evaluated the model.

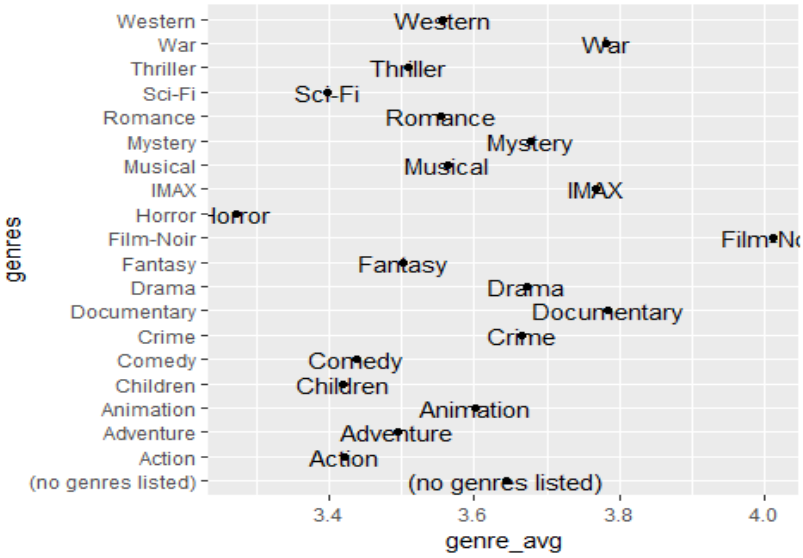
Model	RMSE
Simple average	1.0612018
Movie Effect Model	0.9439087
Movie + User Effects Model	0.8653488

Genre

Next I hypothesized that movies of a given the genre would be rated differently, so I created a list of ratings per genre for each movie.

```
## # A tibble: 20 x 2
##    genres               `n()`
##    <chr>                <int>
##  1 (no genres listed)       7
##  2 Action             2560545
##  3 Adventure          1908892
##  4 Animation           467168
##  5 Children            737994
##  6 Comedy             3540930
##  7 Crime              1327715
##  8 Documentary          93066
##  9 Drama              3910127
## 10 Fantasy             925637
## 11 Film-Noir           118541
## 12 Horror              691485
## 13 IMAX                  8181
## 14 Musical             433080
## 15 Mystery             568332
## 16 Romance            1712100
## 17 Sci-Fi             1341183
## 18 Thriller           2325899
## 19 War                 511147
## 20 Western             189394
```

Now that I had a clean list of ratings per genre, I plotted mean ratings per genre to test the hypothesis that there is a genre effect.



Clearly the genre has some predictive value, as the averages for Horror are low compared to War, IMAX, Documentary and Film-Noir.

I defined the genre bias (`g_b`) as the difference between the overall mean and the genre-specific mean.

I added the genre term to the model `predictions = mu + m_q + u_b + g_b` and evaluated the results using the validation data.

Model	RMSE
Simple average	1.0612018
Movie Effect Model	0.9439087
Movie + User Effects Model	0.8653488
Movie + User + Genre Effects Model	0.8631334

That was a lot of work for a pretty tiny improvement!

Results vs. Wrong Turns

I took some wrong turns in the course of this project, which taught me a few lessons.

First, never underestimate the value of the escape character. Initially when I applied the separate_rows function, I had separated values based on the pipe character alone (`separate_rows(edx, genres, sep="|")`), which led to 25 genres including *Sci, Fi, Film, Noir, No, genre,* and *listed*. To address these problems, I loaded the string_r packaged and applied the remove function to remove hyphens and spaces from the genres column. However, I was still getting errors in the results. It turns out that the replace function and the remove function only apply to the first instance of the search term in any given row, so films like Blade Runner that were both Film-Noir and Sci-Fi we're still throwing errors in the data. I addressed this problem by running the function twice.

Second, I ran into several challenges with R being unable to allocate sufficient memory to complete certain tasks. I found that if I created a new column, rather than trying to replace a value in an existing column, that the command would often complete rather than

throwing a memory exception. I also learned that if I shut down every other application on my computer so that R was the only application running, the command would often incomplete. When neither of those approaches worked, I shut down the R session entirely and started a new session with the command being the first thing I ran in the new session.

Third, I wasted a lot of time splitting the edx set into a train_set and test_set. I thought that this would be helpful since it would reduce the amount of data to process while training the model, but I had not taken sufficient care to ensure that the same movies and users existed in the training set, the test set, and the validation set. Ultimately, I ended up with a model that worked well on the test and train data subsets of edx, but failed to generate results on the validation data because of movies with a single rating appearing only in the validation set. I created a list of movie_ids with a single rating, and removed those from the validation set using anti_join. Once I removed the complication of the test and train data subsets of edx, it was easy to apply the model that I had developed to the full edx and validation sets.

Conclusion and Next Steps

Given that the target of RMSE <= 0.87750 was achieved with the movie and user effects model, the amount of effort required to further improve it by adding genre yielded minimal improvements. It's possible that this Improvement will not be realized in the larger data set, or may in fact get worse. To create a more robust model for genre effect, I would next explore the nearest neighbors (knn) and other methods to generate predictions for genre. I would also explore whether time plays a role; for example if people are rating the 20th anniversary release of a film as opposed to its original.