Harvard Data Science Capstone: Movielens

James Hope

1/24/2019

Executive Summary

We wish to accurately predict a rating for a movie, given a user. Using the Movielens data set we construct a naiive model that predicts movie ratings with a RMSE of 1.0605613. We note that the data shows a movie and user bias and account for this bias in our model to achieve a final RMSE of 0.8439865.

Import

First, we download and read in the data using the sample code provided by EdX.

```r
head(edx)
```

```
##   userId movieId rating timestamp                         title
## 1      1     122      5 838985046              Boomerang (1992)
## 2      1     185      5 838983525               Net, The (1995)
## 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)
##                           genres
## 1                 Comedy|Romance
## 2          Action|Crime|Thriller
## 4  Action|Drama|Sci-Fi|Thriller
## 5        Action|Adventure|Sci-Fi
## 6 Action|Adventure|Drama|Sci-Fi
## 7        Children|Comedy|Fantasy
```

Discovery

We run some simple tests to get a better understanding of the data and its structure.

```r
# Inspect the dimensions
dim(edx)
```

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

```r
# Number of 3* ratings
length(which(edx$rating==3))
```

```
## [1] 2121240
```

```r
# Number of unique movies
length(unique(edx$movieId))
```

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

```r
# Number of unique users
length(unique(edx$userId))
```

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

```r
# Identify the most popular genres
drama <- edx %>% filter(str_detect(genres,"Drama"))
drama <- drama %>% group_by(movieId)
length(drama$rating)
```

```
## [1] 3910127
```

```r
comedy <- edx %>% filter(str_detect(genres,"Comedy"))
comedy <- comedy %>% group_by(movieId)
length(comedy$rating)
```

```
## [1] 3540930
```

```r
Thriller <- edx %>% filter(str_detect(genres,"Thriller"))
Thriller <- Thriller %>% group_by(movieId)
length(Thriller$rating)
```

```
## [1] 2325899
```

```r
romance <- edx %>% filter(str_detect(genres,"Romance"))
romance <- romance %>% group_by(movieId)
length(romance$rating)
```

```
## [1] 1712100
```

```r
# Identify the 25 most popular movies
top_movies <- edx %>% group_by(movieId,title) %>% summarize(rating=n()) %>% arrange(desc(rating))
top_movies %>% print(n=25)
```

```
## # A tibble: 10,677 x 3
## # Groups:   movieId [10,677]
##    movieId title                                                        rating
##      <dbl> <chr>                                                         <int>
##  1     296 Pulp Fiction (1994)                                           31362
##  2     356 Forrest Gump (1994)                                           31079
##  3     593 Silence of the Lambs, The (1991)                              30382
##  4     480 Jurassic Park (1993)                                          29360
##  5     318 Shawshank Redemption, The (1994)                              28015
##  6     110 Braveheart (1995)                                             26212
##  7     457 Fugitive, The (1993)                                          25998
##  8     589 Terminator 2: Judgment Day (1991)                             25984
##  9     260 Star Wars: Episode IV - A New Hope (a.k.a. Star Wars) (1~     25672
## 10     150 Apollo 13 (1995)                                              24284
## 11     592 Batman (1989)                                                 24277
## 12       1 Toy Story (1995)                                              23790
## 13     780 Independence Day (a.k.a. ID4) (1996)                          23449
## 14     590 Dances with Wolves (1990)                                     23367
## 15     527 Schindler's List (1993)                                       23193
## 16     380 True Lies (1994)                                              22823
## 17    1210 Star Wars: Episode VI - Return of the Jedi (1983)             22584
## 18      32 12 Monkeys (Twelve Monkeys) (1995)                            21891
## 19      50 Usual Suspects, The (1995)                                    21648
## 20     608 Fargo (1996)                                                  21395
## 21     377 Speed (1994)                                                  21361
## 22     588 Aladdin (1992)                                                21173
## 23    2571 Matrix, The (1999)                                            20908
## 24    1196 Star Wars: Episode V - The Empire Strikes Back (1980)      20729
## 25      47 Seven (a.k.a. Se7en) (1995)                                20311
## # ... with 1.065e+04 more rows
```

```r
# Identify the most popoular ratings
length(which(edx$rating==1))
```

```
## [1] 345679
```

```r
length(which(edx$rating==1.5))
```

```
## [1] 106426
```

```r
length(which(edx$rating==2))
```

```
## [1] 711422
```

```r
length(which(edx$rating==2.5))
```

```
## [1] 333010
```

```r
length(which(edx$rating==3))
```

```
## [1] 2121240
```

```r
length(which(edx$rating==3.5))
```

```
## [1] 791624
```

```r
length(which(edx$rating==4))
```

```
## [1] 2588430
```

```r
length(which(edx$rating==4.5))
```

```
## [1] 526736
```

```r
length(which(edx$rating==5))
```

```
## [1] 1390114
```

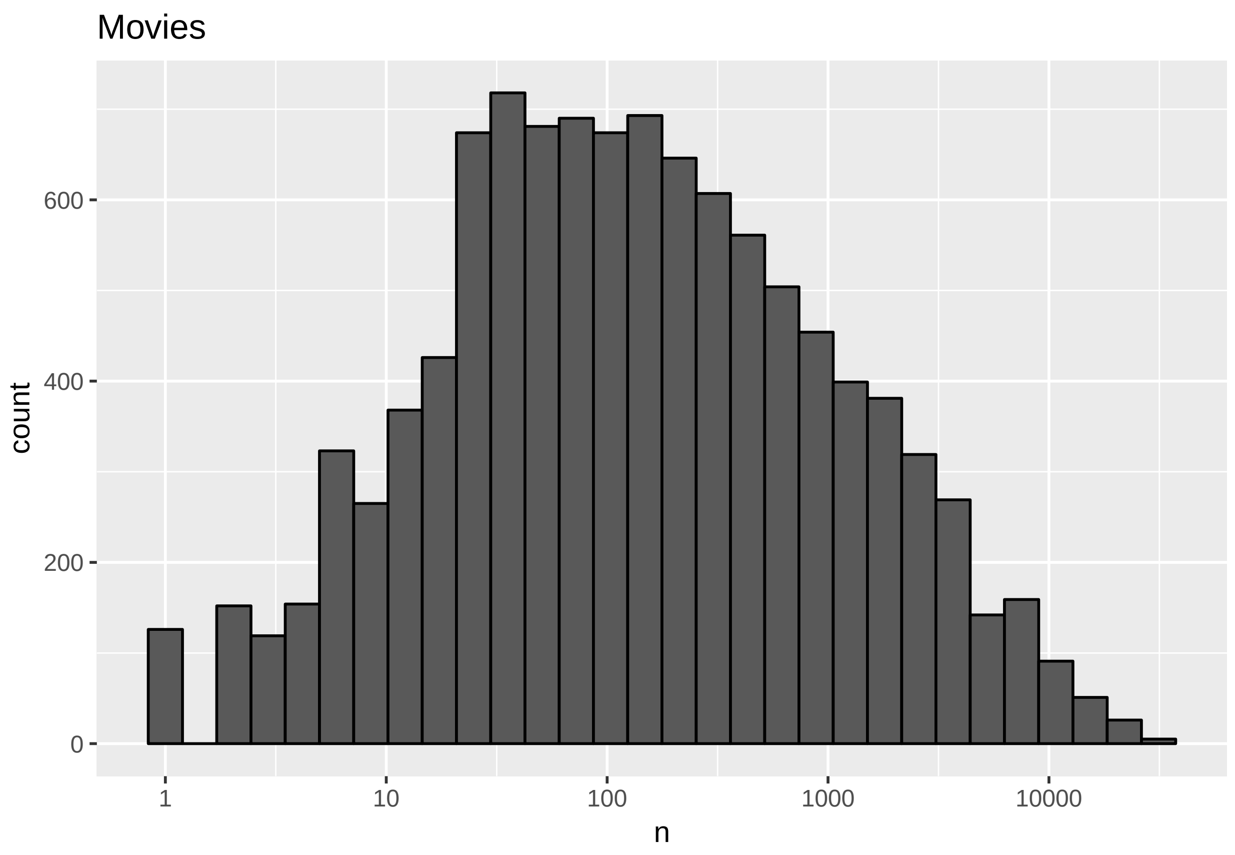
```r
# Note that non-integer ratings are less popular than integer ratings
length(which(edx$rating==1)) + length(which(edx$rating==2)) + length(which(edx$rating==3)) +
  length(which(edx$rating==4)) + length(which(edx$rating==5)) > length(which(edx$rating==2.5)) +
  length(which(edx$rating==3.5)) +length(which(edx$rating==4.5)) + length(which(edx$rating==1.5))
```

```
## [1] TRUE
```

Let's have a look at some of the general properties of the data to better understand the challenge.

The first thing we notice is that some of the movies get rated more than others. Here is the distribution.

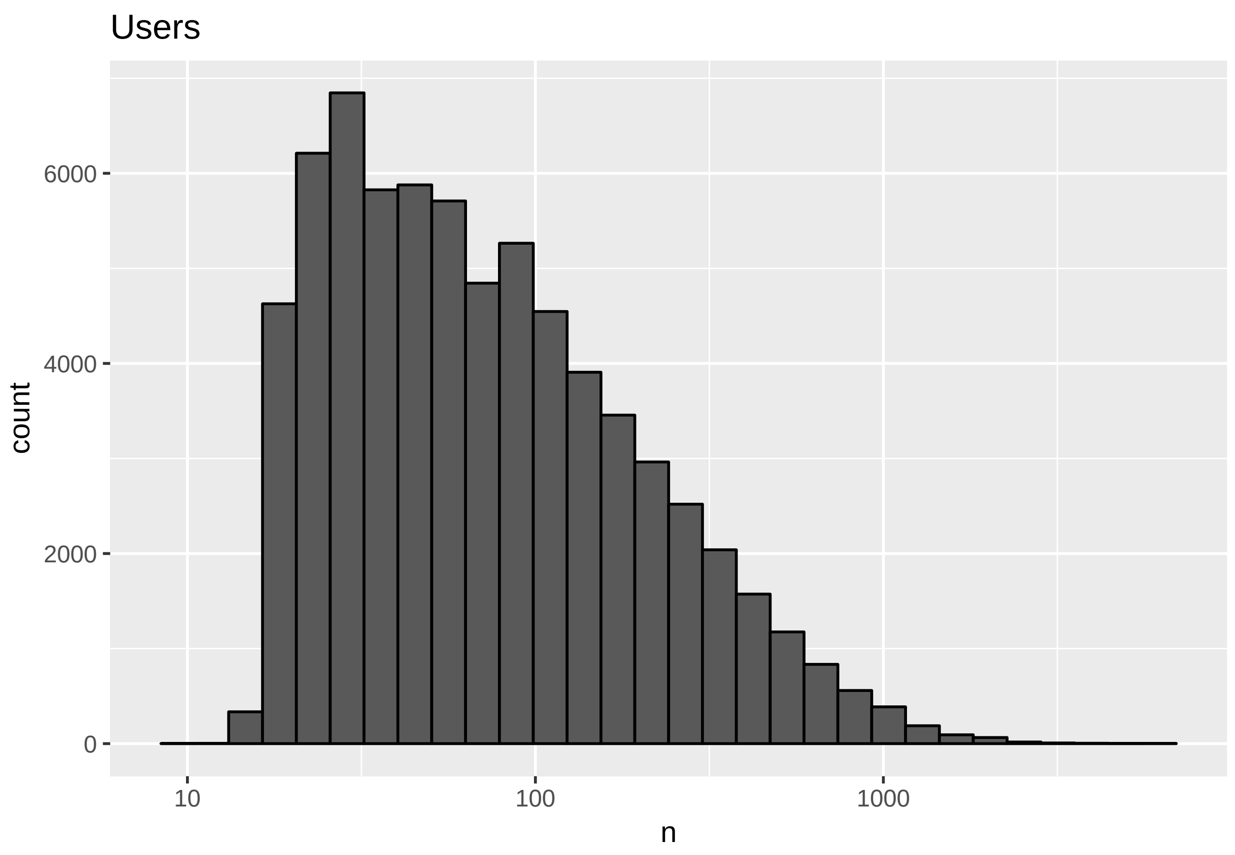
```r
edx %>%
  count(movieId) %>%
  ggplot(aes(n)) +
  geom_histogram(bins = 30, color = "black") +
  scale_x_log10() +
  ggtitle("Movies")
```



This should not surprise us given that there are blockbuster movies watched by millions and artsy, independent movies watched by just a few.

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

```
edx %>%
  count(userId) %>%
  ggplot(aes(n)) +
  geom_histogram(bins = 30, color = "black") +
  scale_x_log10() +
  ggtitle("Users")
```



Model Building

We will build an algorithm with the data we have collected. We begin by creating a test set to assess the accuracy of the models we implement.

```
library(caret)
set.seed(755)
test_index <- createDataPartition(y = edx$rating, times = 1, p = 0.2, list = FALSE)
train_set <- edx[-test_index,]
test_set <- edx[test_index,]
```

To make sure we don't include users and movies in the test set that do not appear in the training set, we remove these entries using the semi_join function:

```
test_set <- test_set %>%
  semi_join(train_set, by = "movieId") %>%
  semi_join(train_set, by = "userId")
```

Next we defined a cost or loss function for this challenge. We will use the root mean square error.

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

A Baseline Model

We start by building the simplest possible recommendation system. We predict the same rating for all users regardless of user. We build a model that assumes the same rating for all movies and users with the differences explained by random variation. We know that the estimate that minimises the RMSE is the least squares estimate of mu, and, in this case, is the average of all the ratings.

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

```
## [1] 3.512527
```

```
naive_rmse <- RMSE(test_set$rating, mu_hat)
naive_rmse
```

```
## [1] 1.060561
```

```
predictions <- rep(2.5, nrow(test_set))
RMSE(test_set$rating, predictions)
```

```
## [1] 1.466085
```

We can build a dataframe to hold our results as we improve the algorithm.

```
rmse_results <- data_frame(method = "Just the average", RMSE = naive_rmse)
```

```
## Warning: `data_frame()` is deprecated, use `tibble()`.
## This warning is displayed once per session.
```

Movie Effects

We can definately do better than this baseline model. We know from intuition that different movies are rated differently. This is confirmed by our data. We can augment our previous model by adding a bias to represent the average ranking for movie i.

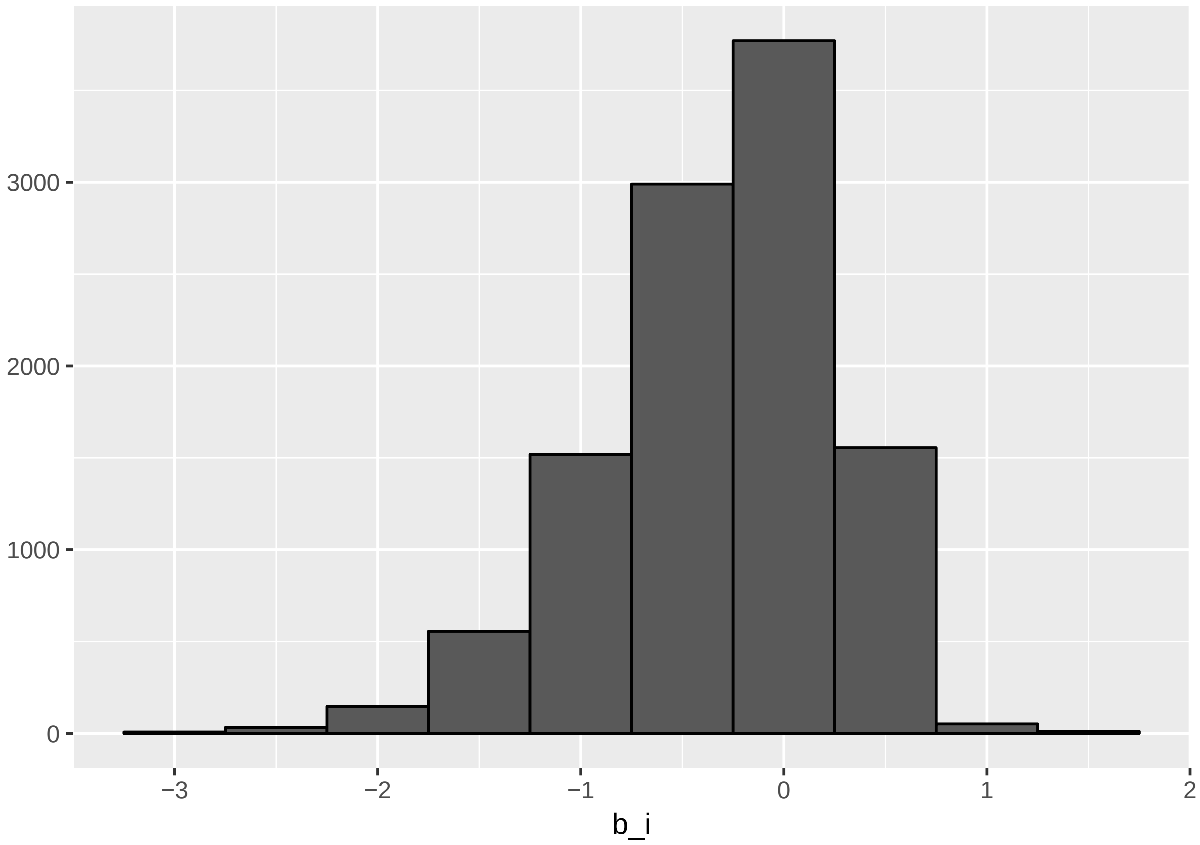
The least squares estimate is just the average of the residual (after mu is subtracted) for each movie i.

```
# Model Movie Effects / Bias

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

movie_avgs
```

```
## # A tibble: 10,641 x 2
##    movieId    b_i
##      <dbl>  <dbl>
##  1       1  0.419
##  2       2 -0.300
##  3       3 -0.358
##  4       4 -0.646
##  5       5 -0.443
##  6       6  0.301
##  7       7 -0.153
##  8       8 -0.384
##  9       9 -0.516
## 10      10 -0.0932
## # ... with 10,631 more rows
```

We can see that the estimates for the movie bias vary substantially.

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



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

model_1_rmse <- RMSE(predicted_ratings, test_set$rating)
rmse_results <- bind_rows(rmse_results,
                          data_frame(method="Movie Effect Model",
                                     RMSE = model_1_rmse ))
rmse_results %>% knitr::kable()
```

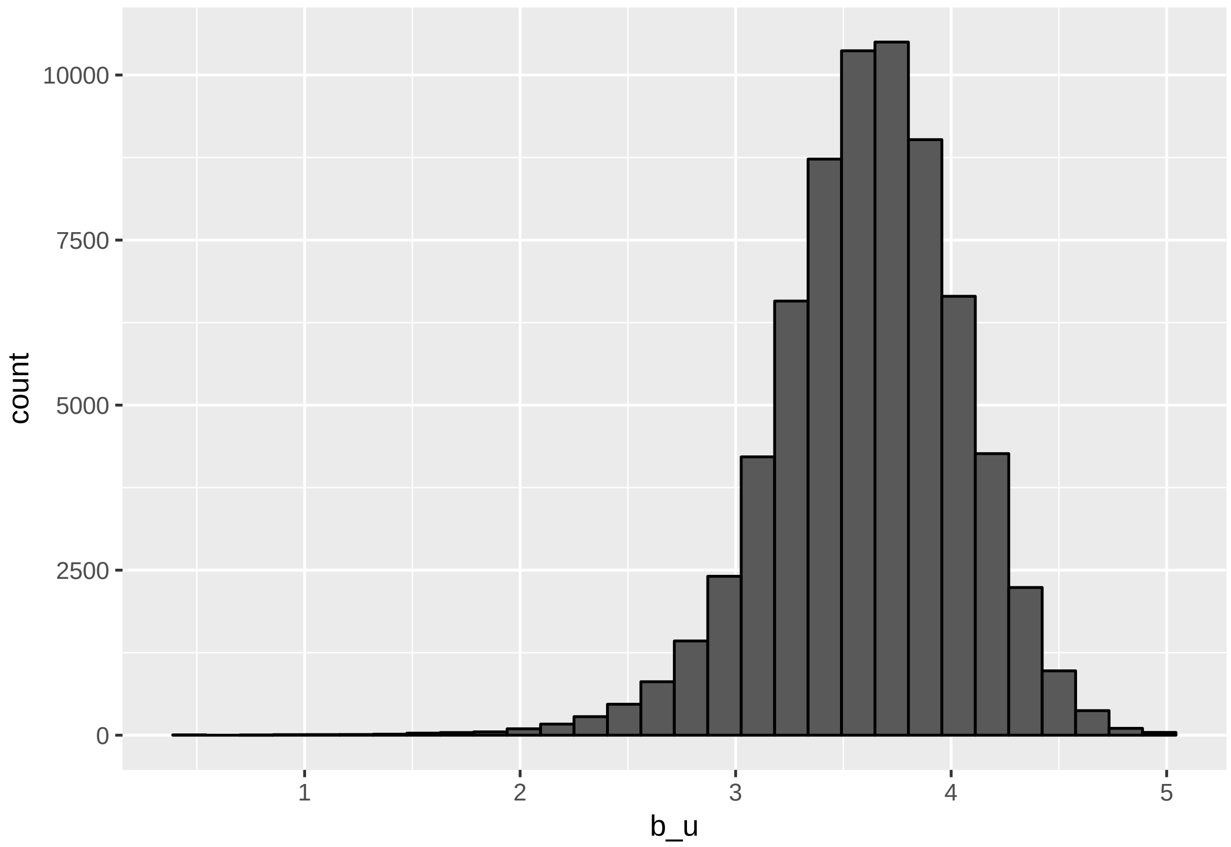
method	RMSE
Just the average	1.0605613
Movie Effect Model	0.9439868

We can see that the prediction RMSE improves under this model.

User Effects

We can also model user effects by computing an average rating for each user. Notice that 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 a bias term for a user-specific effect.

```
# Model User Effects / Bias

train_set %>%
  group_by(userId) %>%
  summarize(b_u = mean(rating)) %>%
  filter(n()>=100) %>%
  ggplot(aes(b_u)) +
  geom_histogram(bins = 30, color = "black")
```



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

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

Final Model Evaluation

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

```
model_2_rmse <- RMSE(predicted_ratings, test_set$rating)
rmse_results <- bind_rows(rmse_results,
                          data_frame(method="Movie + User Effects Model",
                                     RMSE = model_2_rmse ))
rmse_results %>% knitr::kable()
```

method	RMSE
Just the average	1.0605613
Movie Effect Model	0.9439868
Movie + User Effects Model	0.8439865