Capstone: Recommender System

Andrew Farina

6/08/2019

Introduction:

One of the most challenging goals of human decision making is taking large, complex decisions and consolidating those seemingly impossible complex problems into multiple smaller scale decisisons that are more easily solvable. This same process can be applied to machine learning tasks in which we attempt to solve complex decisions using an algorithm that uses multiple rules to solve a series of complex problems. The resulting solutions can be combined to help solve the orignial complex task. This approach was used to determine how users would rate a particular movie.

Dataset:

The dataset for this analysis is the MovieLens 10M dataset. A full description and dataset can be found at the following locations:

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

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

The dataset contains over 10 Million ratings of 10K movies by 71K users. All users rated at least 20 movies. Please see citations at the end of the document for acknowledgement of the creators of this project.

The data for consideration contained randomly selected userIds, ratings on a 5 star scale, a timestamp indicating when the rating was given, the MovieId for each movie, the title of each movie, and the genres of each movie.

The goal of this project is to predict how users would rate specific movies based on their previous movie ratings, how various movies were rated by others, when the movies were produced, and the time difference between when the movies were produced and when the movies were rated. This was accomplished though means based analysis of each of the previously mentioned elements of the dataset. The following analysis produced a root mean squared error (RMSE) of 0.865.

Methods:

Creating the dataset:

```r
dl <- tempfile()
download.file("http://files.grouplens.org/datasets/movielens/ml-10m.zip", dl)

ratings <- read.table(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")
```

Cleaning and Prepare the Dataset:

When thinking through this dataset, four items were considered during cleaning. The first consideration would be the need to convert the timestamp data (when the movie was rated online) into a POSIXct format for analysis. this was a simple mutation. I created the date-time-group ('DTG') feature and removed the timestamp feature:

```
movielens <- movielens %>% mutate(DTG = as.Date.POSIXct(timestamp))
movielens <- movielens[, names(movielens) != "timestamp"]
```

The second consideration was the need to group the movies together so that I could determine if a 'cohort bias' existed. That is, did the year that the movie premiered effect how it was rated. This was a little more challenging but leveraging teh stri_extract_last_regex function from the stringi package, I was able to extract the year that the movie premiered. Unfortunately, the month and day were not known from the data. I used the first of the year as a standard timepoint for all movie premiers. This extraction was placed in the 'movie_yr' feature and the movie title was removed:

```
movie_yr <- movielens$title %>% stri_extract_last_regex('\\d{4}')
movielens <- cbind(movielens, movie_yr)
movielens <- movielens[, names(movielens) != "title"]
movielens$movie_yr <- as.Date(paste(movielens$movie_yr, 1, 1, sep = "-"))
```

The third consideration was the time difference between these two dates. The resulting variable, 'yrs_btw' is a rounded number of years between when the movie premiered (movie_yr) and when the movie was rated (DTG).

```
movielens <- movielens %>% mutate(yrs_btw = round((DTG - movie_yr) / 365))
attributes(movielens$yrs_btw) <- NULL
```

Finally, I wanted to consider the genres to better understand the types of movies viewed. This was a difficult (and time resource intensive) transformation. To accomplish this, I created a function (genres_func) and applied it to the genres column in the movielens dataset.

```
genres_func <- function(data, column) {
  data %>% mutate(Action = ifelse(str_detect(column, coll("Action")), 1, 0),
                  Adventure = ifelse(str_detect(column, coll("Adventure")), 1, 0),
                  Animation = ifelse(str_detect(column, coll("Animation")), 1, 0),
                  Chilren = ifelse(str_detect(column, coll("Children")), 1, 0),
                  Comedy = ifelse(str_detect(column, coll("Comedy")), 1, 0),
                  Crime = ifelse(str_detect(column, coll("Crime")), 1, 0),
                  Documentary = ifelse(str_detect(column, coll("Documentary")), 1, 0),
                  Drama = ifelse(str_detect(column, coll("Drama")), 1, 0),
                  Fantasy = ifelse(str_detect(column, coll("Fantasy")), 1, 0),
                  Film_Noir = ifelse(str_detect(column, coll("Film-Noir")), 1, 0),
                  Horror = ifelse(str_detect(column, coll("Horror")), 1, 0),
                  Musical = ifelse(str_detect(column, coll("Musical")), 1, 0),
                  Mystery = ifelse(str_detect(column, coll("Mystery")), 1, 0),
                  Romance = ifelse(str_detect(column, coll("Romance")), 1, 0),
                  Sci_Fi = ifelse(str_detect(column, coll("Sci-Fi")), 1, 0),
                  Thriller = ifelse(str_detect(column, coll("Thriller")), 1, 0),
                  War = ifelse(str_detect(column, coll("War")), 1, 0),
                  Western = ifelse(str_detect(column, coll("Western")), 1, 0),)}

movielens <- genres_func(movielens, movielens$genres)
movielens <- movielens[, names(movielens) != "genres"]
```

Creating the test set and training set:

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)
```

```
## Joining, by = c("userId", "movieId", "rating", "DTG", "movie_yr", "yrs_btw", "Action", "Adventure",
```

```
edx <- rbind(edx, removed)

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

Data Exploration

To explore this data, I looked at the structure and summaries of the data; the number of movies, users, and genres; and the distribution of the movieId, userId, and rating features.

```
head(edx)
```

```
##   userId movieId rating        DTG   movie_yr yrs_btw Action Adventure
## 1      1     122      5 1996-08-02 1992-01-01       5      0         0
## 2      1     185      5 1996-08-02 1995-01-01       2      1         0
## 4      1     292      5 1996-08-02 1995-01-01       2      1         0
## 5      1     316      5 1996-08-02 1994-01-01       3      1         1
## 6      1     329      5 1996-08-02 1994-01-01       3      1         1
## 7      1     355      5 1996-08-02 1994-01-01       3      0         0
##   Animation Chilren Comedy Crime Documentary Drama Fantasy Film_Noir
## 1         0       0      1     0           0     0       0         0
## 2         0       0      0     1           0     0       0         0
## 4         0       0      0     0           0     1       0         0
## 5         0       0      0     0           0     0       0         0
## 6         0       0      0     0           0     1       0         0
## 7         0       1      1     0           0     0       1         0
##   Horror Musical Mystery Romance Sci_Fi Thriller War Western
## 1      0       0       0       1      0        0   0       0
## 2      0       0       0       0      0        1   0       0
## 4      0       0       0       0      1        1   0       0
## 5      0       0       0       0      1        0   0       0
## 6      0       0       0       0      1        0   0       0
## 7      0       0       0       0      0        0   0       0
```

```r
head(validation)
```

```
##   userId movieId rating        DTG   movie_yr yrs_btw Action Adventure
## 1      1     231      5 1996-08-02 1994-01-01       3      0         0
## 2      1     480      5 1996-08-02 1993-01-01       4      1         1
## 3      1     586      5 1996-08-02 1990-01-01       7      0         0
## 4      2     151      3 1997-07-07 1995-01-01       3      1         0
## 5      2     858      2 1997-07-07 1972-01-01      26      0         0
## 6      2    1544      3 1997-07-07 1997-01-01       1      1         1
##   Animation Chilren Comedy Crime Documentary Drama Fantasy Film_Noir
## 1         0       0      1     0           0     0       0         0
## 2         0       0      0     0           0     0       0         0
## 3         0       1      1     0           0     0       0         0
## 4         0       0      0     0           0     1       0         0
## 5         0       0      0     1           0     1       0         0
## 6         0       0      0     0           0     0       0         0
##   Horror Musical Mystery Romance Sci_Fi Thriller War Western
## 1      0       0       0       0      0        0   0       0
## 2      0       0       0       0      1        1   0       0
## 3      0       0       0       0      0        0   0       0
## 4      0       0       0       1      0        0   1       0
## 5      0       0       0       0      0        0   0       0
## 6      1       0       0       0      1        1   0       0
```

```r
str(edx)
```

```
## 'data.frame':    9000055 obs. of  24 variables:
##  $ userId     : int  1 1 1 1 1 1 1 1 1 1 ...
##  $ movieId    : num  122 185 292 316 329 355 356 362 364 370 ...
##  $ rating     : num  5 5 5 5 5 5 5 5 5 5 ...
##  $ DTG        : Date, format: "1996-08-02" "1996-08-02" ...
##  $ movie_yr   : Date, format: "1992-01-01" "1995-01-01" ...
##  $ yrs_btw    : num  5 2 2 3 3 3 3 3 3 3 ...
##  $ Action     : num  0 1 1 1 1 0 0 0 0 1 ...
##  $ Adventure  : num  0 0 0 1 1 0 0 1 1 0 ...
##  $ Animation  : num  0 0 0 0 0 0 0 0 1 0 ...
##  $ Chilren    : num  0 0 0 0 0 1 0 1 1 0 ...
##  $ Comedy     : num  1 0 0 0 0 1 1 0 0 1 ...
##  $ Crime      : num  0 1 0 0 0 0 0 0 0 0 ...
##  $ Documentary: num  0 0 0 0 0 0 0 0 0 0 ...
##  $ Drama      : num  0 0 1 0 1 0 1 0 1 0 ...
##  $ Fantasy    : num  0 0 0 0 0 1 0 0 0 0 ...
##  $ Film_Noir  : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ Horror     : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ Musical    : num  0 0 0 0 0 0 0 0 1 0 ...
##  $ Mystery    : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ Romance    : num  1 0 0 0 0 0 1 1 0 0 ...
##  $ Sci_Fi     : num  0 0 1 1 1 0 0 0 0 0 ...
##  $ Thriller   : num  0 1 1 0 0 0 0 0 0 0 ...
##  $ War        : num  0 0 0 0 0 0 1 0 0 0 ...
##  $ Western    : num  0 0 0 0 0 0 0 0 0 0 ...
```

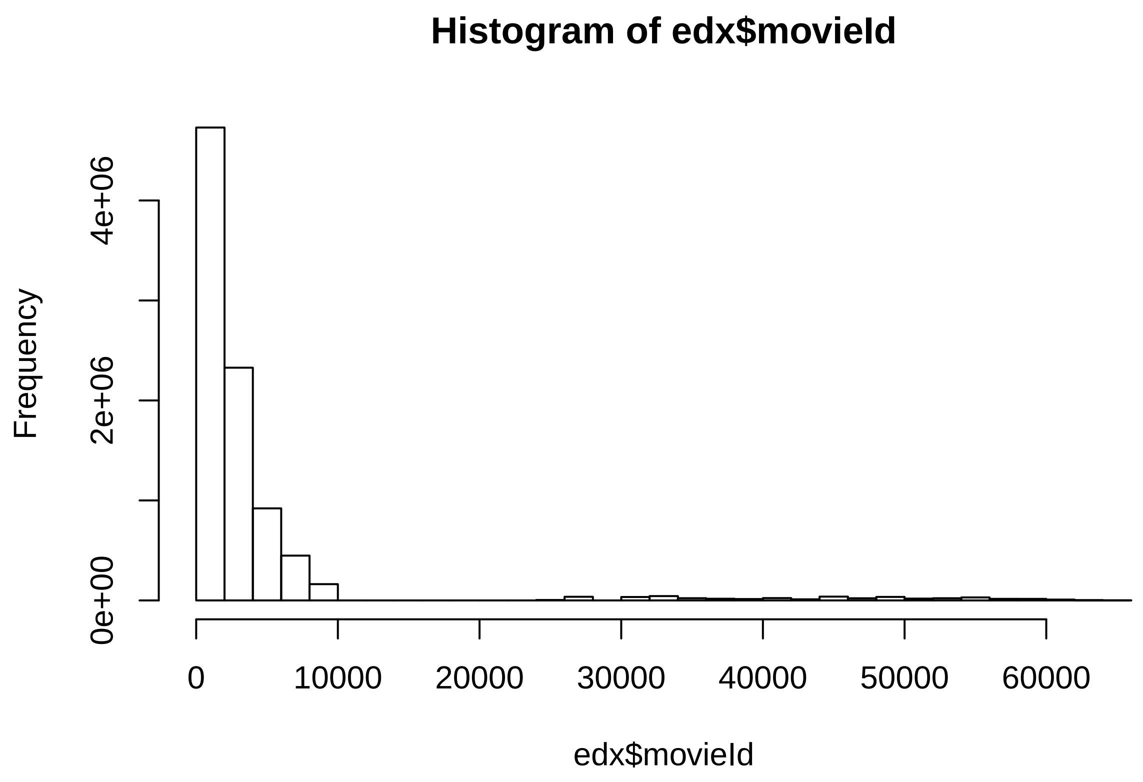
```r
edx %>% summarize(n_users = n_distinct(userId),
                  n_movies = n_distinct(movieId))
```

```
##   n_users n_movies
## 1   69878    10677
```

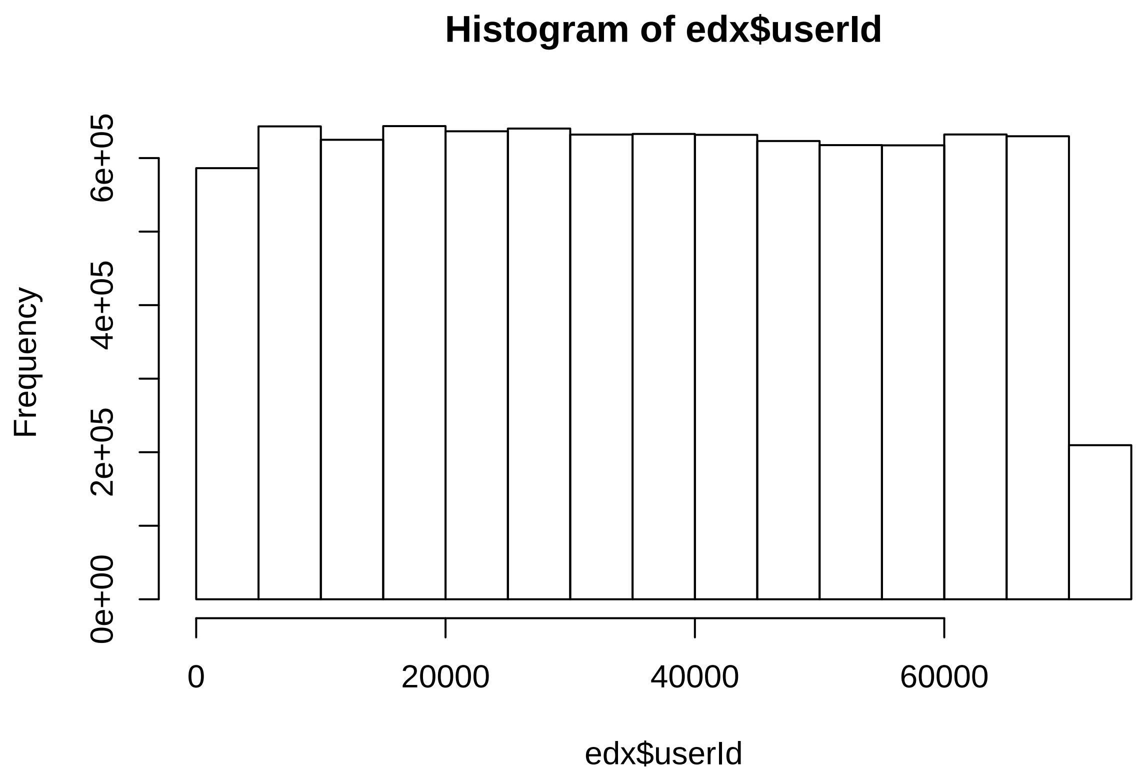
```r
map_dbl(edx[,7:24], sum)
```

```
##      Action   Adventure   Animation     Chilren      Comedy       Crime
##     2560545     1908892      467168      737994     3540930     1327715
## Documentary       Drama     Fantasy   Film_Noir      Horror     Musical
##       93066     3910127      925637      118541      691485      433080
##     Mystery     Romance      Sci_Fi    Thriller         War     Western
##      568332     1712100     1341183     2325899      511147      189394
```

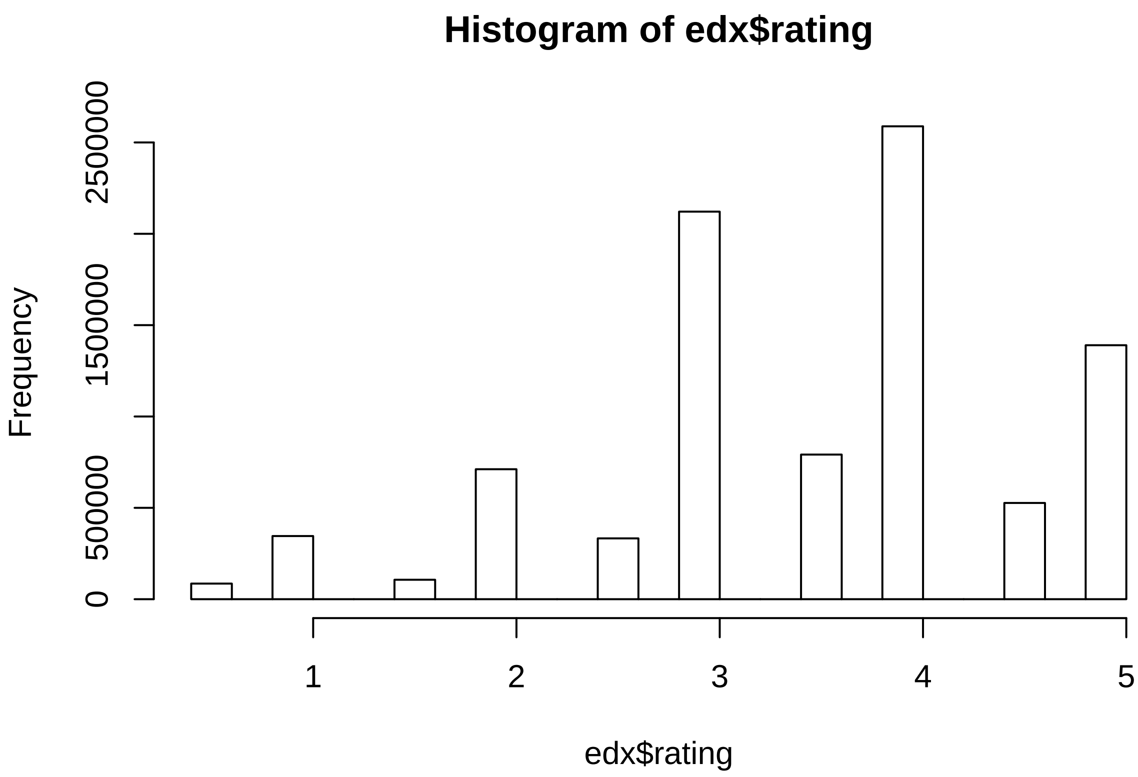
```r
hist(edx$movieId)
```



```r
hist(edx$userId)
```



```r
hist(edx$rating)
```



## Modeling Approach

Once I understood the dataset, I started to model the data to see how much variance we could account for and how we could use mean clustering to reduce the RMSE of our predicted data and the actual ratings. First, I created a function to determine the RMSE from predicted and actual ratings:

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

Results

Model 1: A simple mean

Model 1 only considered the average rating for all movies. This created a starting point and resulted in an RMSE of 1.06

```r
mu_hat <- mean(edx$rating)
naive_rmse <- RMSE(validation$rating, mu_hat)
paste("mu_hat =", round(mu_hat, 2), "| naive_rmse =", round(naive_rmse, 2))
```

```
## [1] "mu_hat = 3.51 | naive_rmse = 1.06"
```

In order to compare the RMSE results, I build a results table:

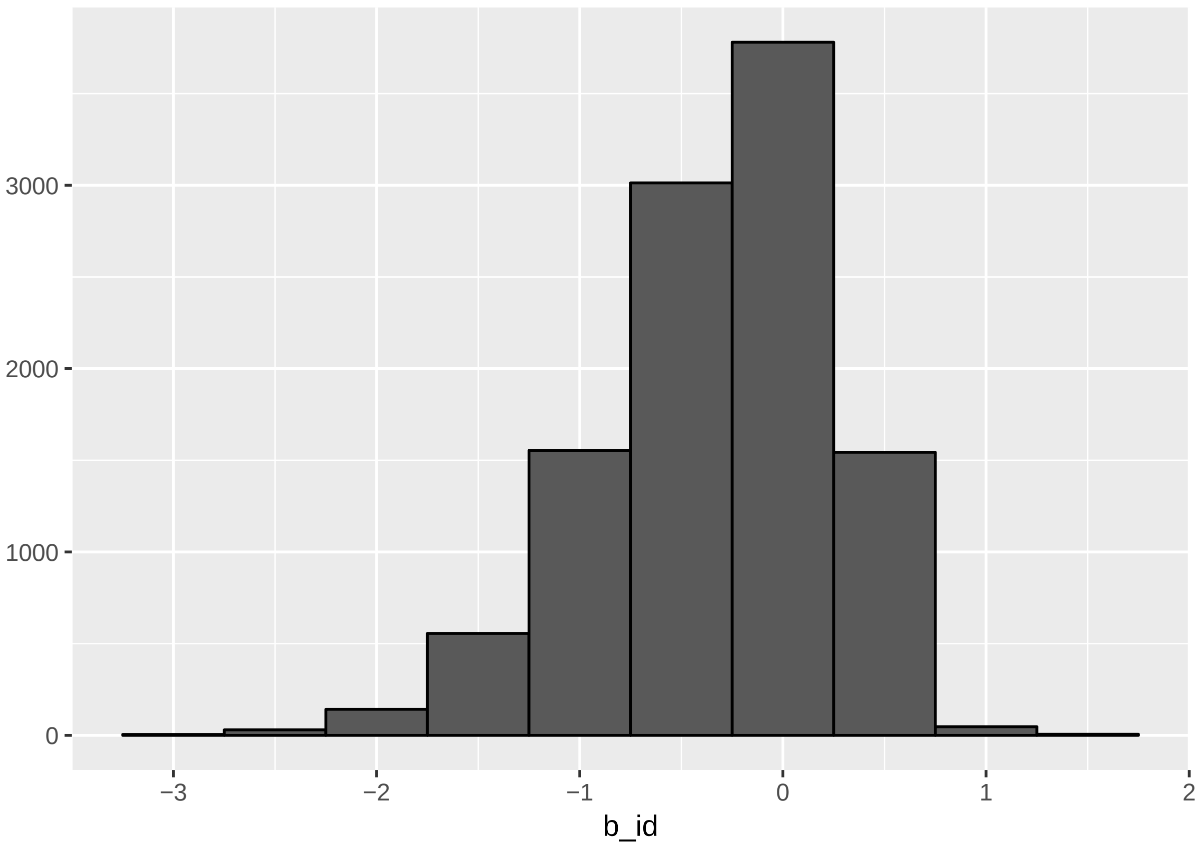
```
rmse_results <- tibble(method = "Just the average",
                           RMSE = naive_rmse)
rmse_results
```

```
## # A tibble: 1 x 2
##   method            RMSE
##   <chr>            <dbl>
## 1 Just the average  1.06
```

Model 2: Movie Effects

Model 2 considered the effect that each movie has. That is, some movies will simply be better movies and will tend to be rated higher than other movies, regardless of who rates them. This model created an average for each of the movies in the dataset used the difference between the movie average and overall average to adjust the predicted outcome. This created an RMSE of 0.944

```
mu <- mean(edx$rating)
movie_avgs <- edx %>%
  group_by(movieId) %>%
  summarize(b_id = mean(rating - mu))
movie_avgs %>% qplot(b_id, geom = "histogram", bins = 10,
                     data = ., color = I("black"))
```



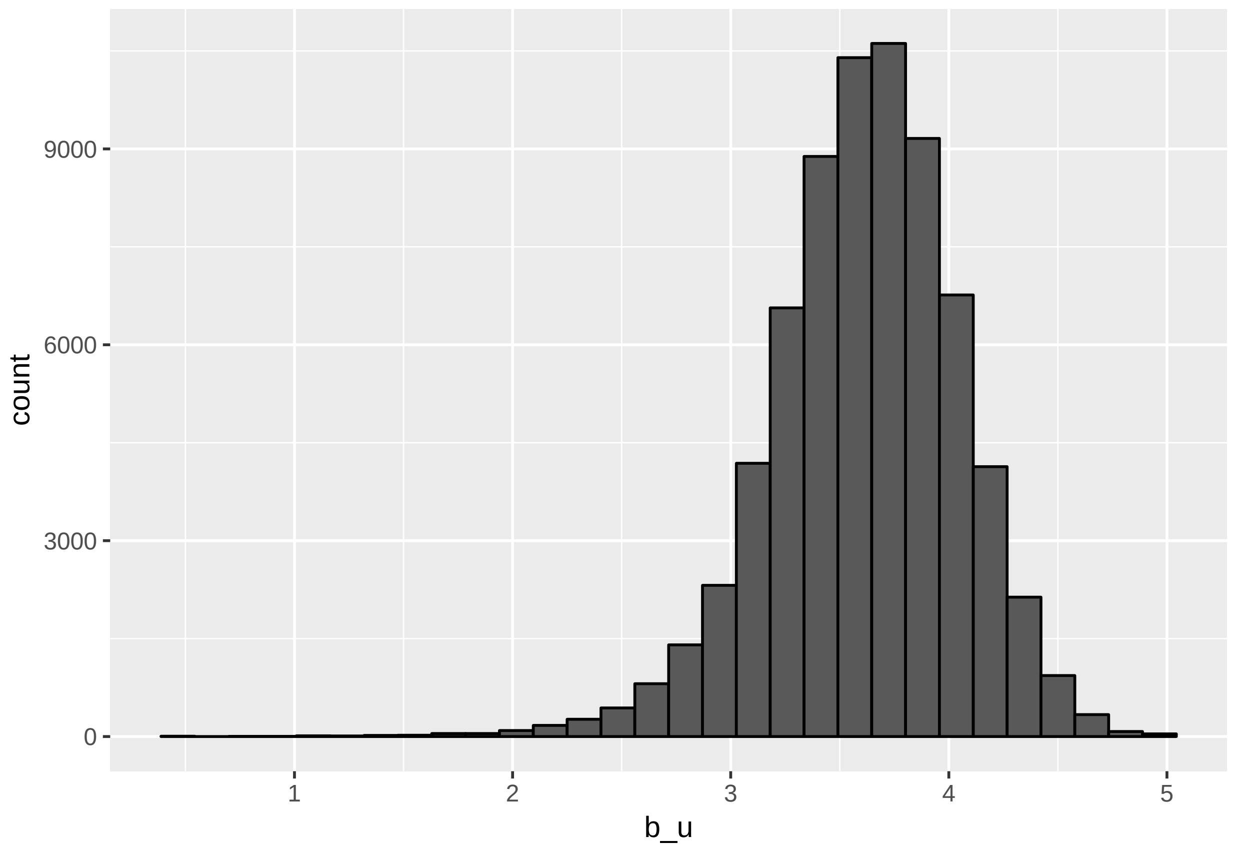
```r
predicted_ratings <- mu + validation %>%
  left_join(movie_avgs, by = "movieId") %>%
  pull(b_id)

model_2_rmse <- RMSE(predicted_ratings, validation$rating)
rmse_results <- bind_rows(rmse_results, tibble(method= "Movie Effect Model",
                                               RMSE= model_2_rmse))
rmse_results
```

```
## # A tibble: 2 x 2
##   method              RMSE
##   <chr>              <dbl>
## 1 Just the average   1.06
## 2 Movie Effect Model 0.944
```

Model 3: Movie plus User Effects

Model 3 considers both the movie averages from Model 2 and additionally considers the rater bias. That is, the actual users may tend to rate all movies higher (or lower) than other users. For this model, the movie rated averages by user were considered and used to adjust the rating above and beyond what the movie averages were. This created an RMSE of 0.865

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



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

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

model_3_rmse <- RMSE(predicted_ratings, validation$rating)
rmse_results <- bind_rows(rmse_results,
                          tibble(method= "Movie + User Effects",
                                 RMSE = model_3_rmse))
rmse_results
```

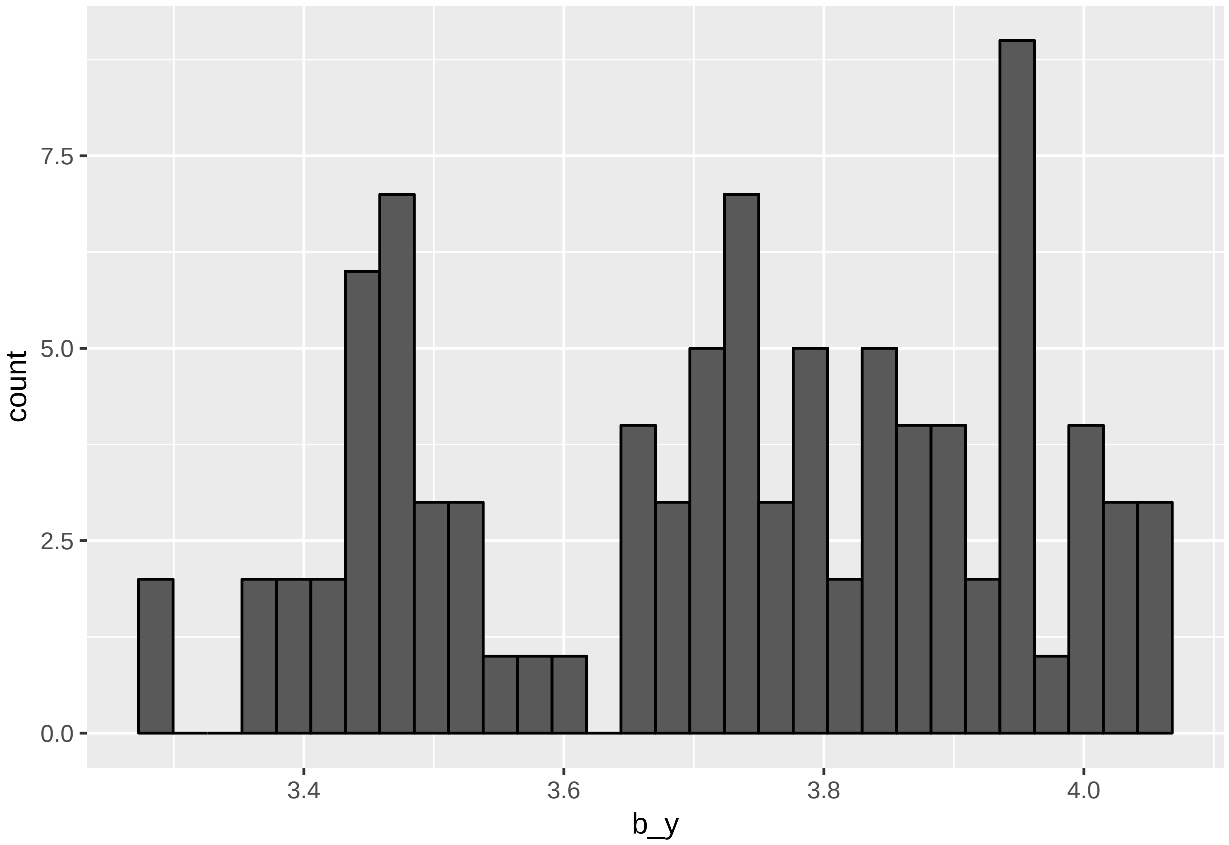
```
## # A tibble: 3 x 2
##   method                RMSE
##   <chr>                <dbl>
## 1 Just the average     1.06
## 2 Movie Effect Model   0.944
## 3 Movie + User Effects 0.865
```

Model 4: Movie Effect plus user effect plus cohort effect

Model 4 considers Model 3 (Movie Effect plus User Effect) and additionally looks at the cohort effect. That is, does the year that the movie came out some how bias the ratings above and beyond what is accounted for by the movies and users. This does not improve the model above and beyond what movie and user effects do (RMSE remains at 0.865).

```
edx %>%
  group_by(movie_yr) %>%
  summarize(b_y = mean(rating)) %>%
  ggplot(aes(b_y)) +
  geom_histogram(color = "black")
```

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



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

predicted_ratings <- validation %>%
  left_join(movie_avgs, by= "movieId") %>%
  left_join(user_avgs, by= "userId") %>%
  left_join(cohort_avgs, by = "movie_yr") %>%
  mutate(pred = mu + b_y + b_id + b_u) %>%
  pull(pred)

model_4_rmse <- RMSE(predicted_ratings, validation$rating)
rmse_results <- bind_rows(rmse_results,
                          tibble(method = "Movie Effect + User Effect + Cohort Effect", RMSE = model_4_
rmse_results
```

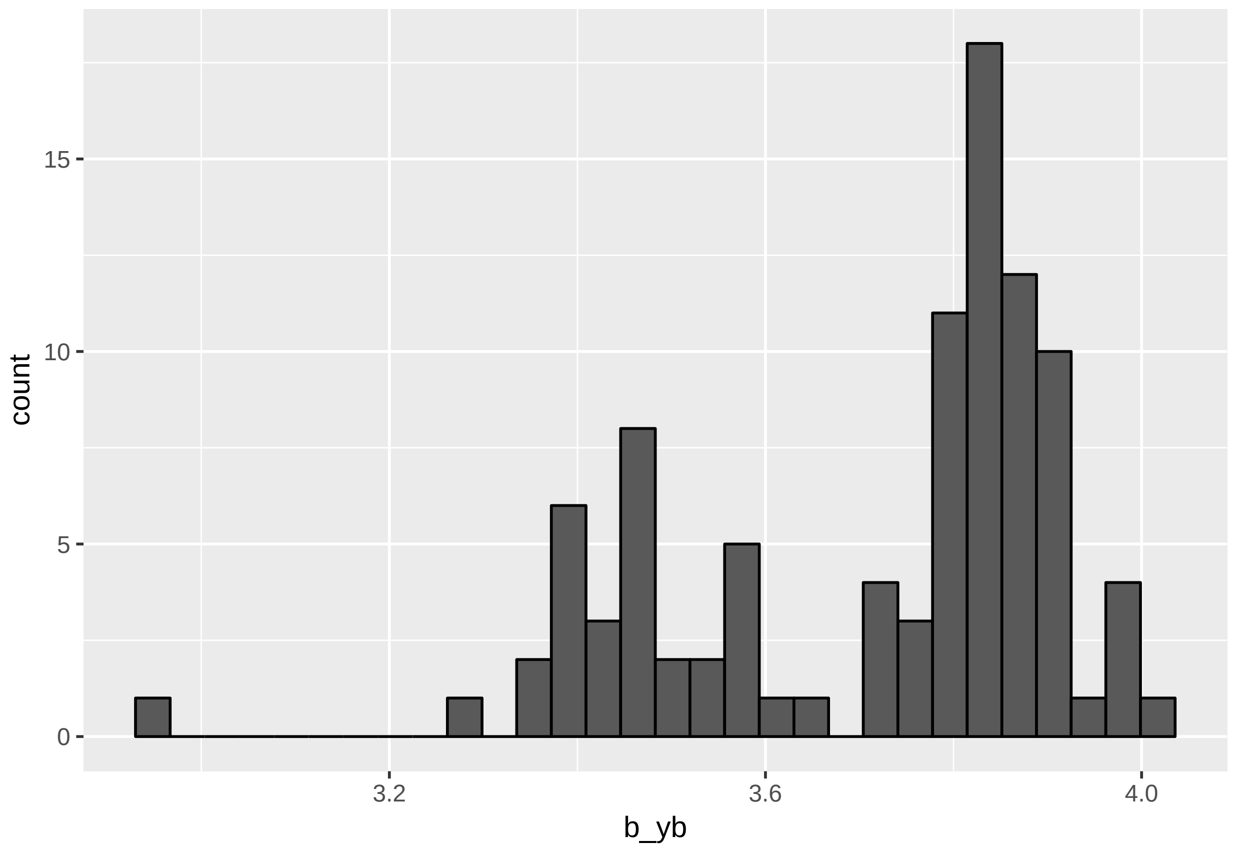
```
## # A tibble: 4 x 2
##   method                                      RMSE
##   <chr>                                      <dbl>
## 1 Just the average                           1.06
## 2 Movie Effect Model                         0.944
## 3 Movie + User Effects                       0.865
## 4 Movie Effect + User Effect + Cohort Effect 0.865
```

Model 5: Movie Effect plus user effect plus cohort effect plus time effect.

Model 5 considers the movie effect, user effect, cohort effect and includes the difference (in year increments) between when the movie was first screened (movie_yr) and when the movie was rated (DTG). My assumption is that the longer time between the ratings, the less bias will be in the data due to marketing of the particular movie. This is also not predictive above and beyond the movie, user and cohort effects account for (RMSE remained at 0.865).

```
edx %>%
  group_by(yrs_btw) %>%
  summarize(b_yb = mean(rating)) %>%
  ggplot(aes(b_yb)) +
  geom_histogram(color = "black")
```

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



```r
yrs_btw_avgs <- edx %>%
  left_join(movie_avgs, by = "movieId") %>%
  left_join(user_avgs, by = "userId") %>%
  left_join(cohort_avgs, by = "movie_yr") %>%
  group_by(yrs_btw) %>%
  summarize(b_yb = mean(rating - mu- b_id - b_u- b_y))

predicted_ratings <- validation %>%
  left_join(movie_avgs, by= "movieId") %>%
  left_join(user_avgs, by= "userId") %>%
  left_join(cohort_avgs, by = "movie_yr") %>%
  left_join(yrs_btw_avgs, by = "yrs_btw") %>%
  mutate(pred = mu + b_id + b_u + b_yb+ b_y) %>%
  pull(pred)

model_5_rmse <- RMSE(predicted_ratings, validation$rating)
rmse_results <- bind_rows(rmse_results,
                          tibble(method = "Movie Effect + User Effect + cohort Effect + yrs_btw Effect"
rmse_results
```

```
## # A tibble: 5 x 2
##   method                                                        RMSE
##   <chr>                                                        <dbl>
## 1 Just the average                                             1.06
## 2 Movie Effect Model                                           0.944
## 3 Movie + User Effects                                         0.865
## 4 Movie Effect + User Effect + Cohort Effect                   0.865
## 5 Movie Effect + User Effect + cohort Effect + yrs_btw Effect 0.865
```

Conclusion:

The approach of means clustering provided an overall RMSE of 0.865. Further attempts to improve the model did not result in any additional reduction in error. Although, for this analysis, the cohort effect and time between premier and rating did not improve the model fit, I kept them in the model as they each provided marginal benefits on their own. It appears that the movie effect is an important predicter in movie ratings. This is a reasonable conclusion as we would expect the actual movie to account for most of the variablity in the ratings. That is to say, if the movie is good, it should be rated higher and if it is not, then we would expect the movie to be rated lower. When we also consider the biases of the indivuals rating the movies, we get an imporoved model that accounts for most of the error in the ratings.

Given more time and resources, additional analysis should focus on the effect of genres: does rater biases change based on the genre of the movie.
*** # Ratings will go into the CSV submission file below:

```r
write.csv(validation %>% select(userId, movieId) %>% mutate(rating = predicted_ratings),
          "submission.csv", na = "", row.names=FALSE)
```

Citations:

F. Maxwell Harper and Joseph A. Konstan. 2015. The MovieLens Datasets: History and Context. ACM Transactions on Interactive Intelligent Systems (TiiS) 5, 4, Article 19 (December 2015), 19 pages. DOI=http://dx.doi.org/10.1145/2827872