MovieLensProject

Kru123

21/01/2019

Introduction

This is a project to design a recommendation system based on the MovieLens dataset. The data for this project is based on the MovieLens 10M data set found at https://grouplens.org/datasets/movielens/10m/ and http://files.grouplens.org/datasets/movielens/ml-10m.zip.

The train set (edx) is 90% of the data and the test set (validation) is 10%. The test set (validation) is created in such a way that all movieId and userId in this set is also in the train set (edx).

In this analysis we look at the effect of the specific user, the movie, the release year of the movie and the genre on the ratings of the movie. In order to look at the release year and the genre effect further data manipulation is necessary.

The goal of the project is to build a model to predict the rating values in the validation set. Possible effects that can create bias are: + user effect - some users score movies lower than others + movie effect - some movies are scored higher or lower than average + genre effect - some genres are scored higher than others + time effect - there could possibly be an effect based on when the movie was released.

Key steps that were performed are: 1. Data exploration - this showed the need to extract the release date from the title field and also that the genres are pipe delimited values with more than one genre assigned to a movie. 2. Data cleaning - this was done to extract the release year and to get the genre in a format that can be used. 3. Build models, test and log the results.

Methods/Analysis

1. Data Exploration

The size of the edx data set:

```r
dim(edx)
```

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

The size of the validation set:

```r
dim(validation)
```

```
## [1] 999999      6
```

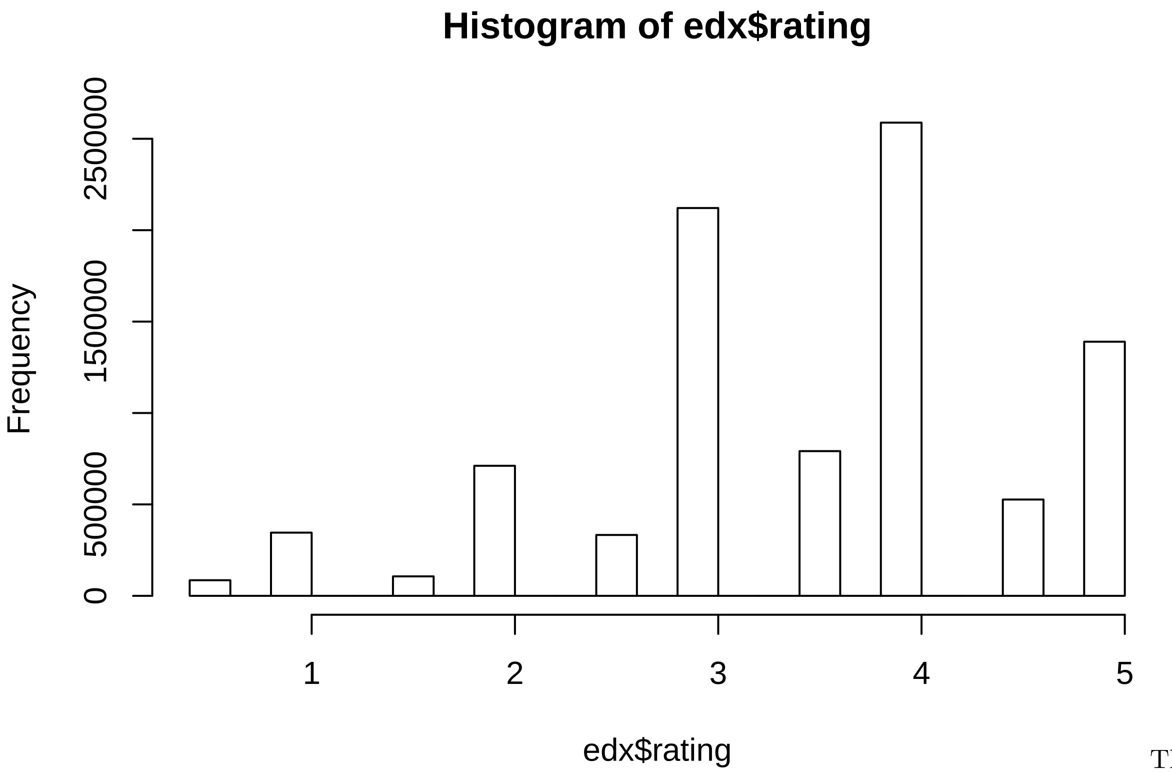
The distribution of rating values in the edx dataset:

```r
table(edx$rating)
```

```
##
##     0.5       1     1.5       2     2.5       3     3.5       4     4.5
##   85374  345679  106426  711422  333010 2121240  791624 2588430  526736
##       5
## 1390114
```

From this distribution we can see that a rating of four is more often given than other ratings. Half point ratings are given less often than full point ratings.

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



The edx dataset has the following data fields:

```r
names(edx)
```

```
## [1] "userId"    "movieId"   "rating"    "timestamp" "title"     "genres"
```

This project will attempt to build a model to predict the rating values in the validation set. Possible effects that can create bias are: * user effect - some users score movies lower than others * movie effect - some movies are scored higher or lower than average * genre effect - some genres are scored higher than others * time effect - there could be an effect based on when the movie was released.

Number of movies in the edx dataset:

```r
n_distinct(edx$movieId)
```

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

Number of users in the edx dataset:

```r
n_distinct(edx$userId)
```

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

Number of movies * number of users > Number of records in edx, which means that not every user rated every movie.

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

```
##   n_users n_movies   n_total
## 1   69878    10677 746087406
```

Genres represented in the edx dataset are represented in a pipe delimited format because some mavies are classified into more than one genre:

```
edx$genres[1:10]
```

```
##  [1] "Comedy|Romance"
##  [2] "Action|Crime|Thriller"
##  [3] "Action|Drama|Sci-Fi|Thriller"
##  [4] "Action|Adventure|Sci-Fi"
##  [5] "Action|Adventure|Drama|Sci-Fi"
##  [6] "Children|Comedy|Fantasy"
##  [7] "Comedy|Drama|Romance|War"
##  [8] "Adventure|Children|Romance"
##  [9] "Adventure|Animation|Children|Drama|Musical"
## [10] "Action|Comedy"
```

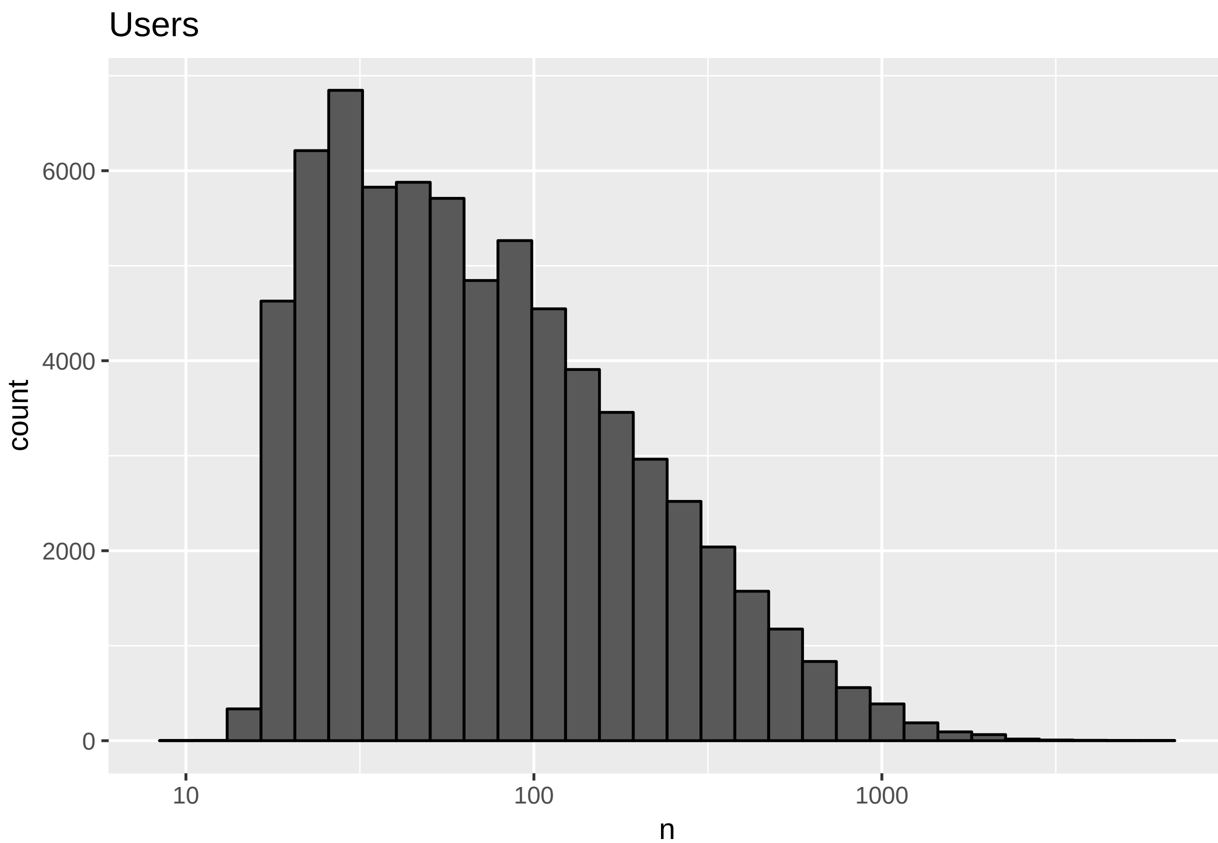
To separate the genres and count the number of ratings per genre the following code can be used

```
edx %>% separate_rows(genres, sep = "\\|") %>%
    group_by(genres) %>%
    summarize(count = n()) %>%
    arrange(desc(count))
```

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

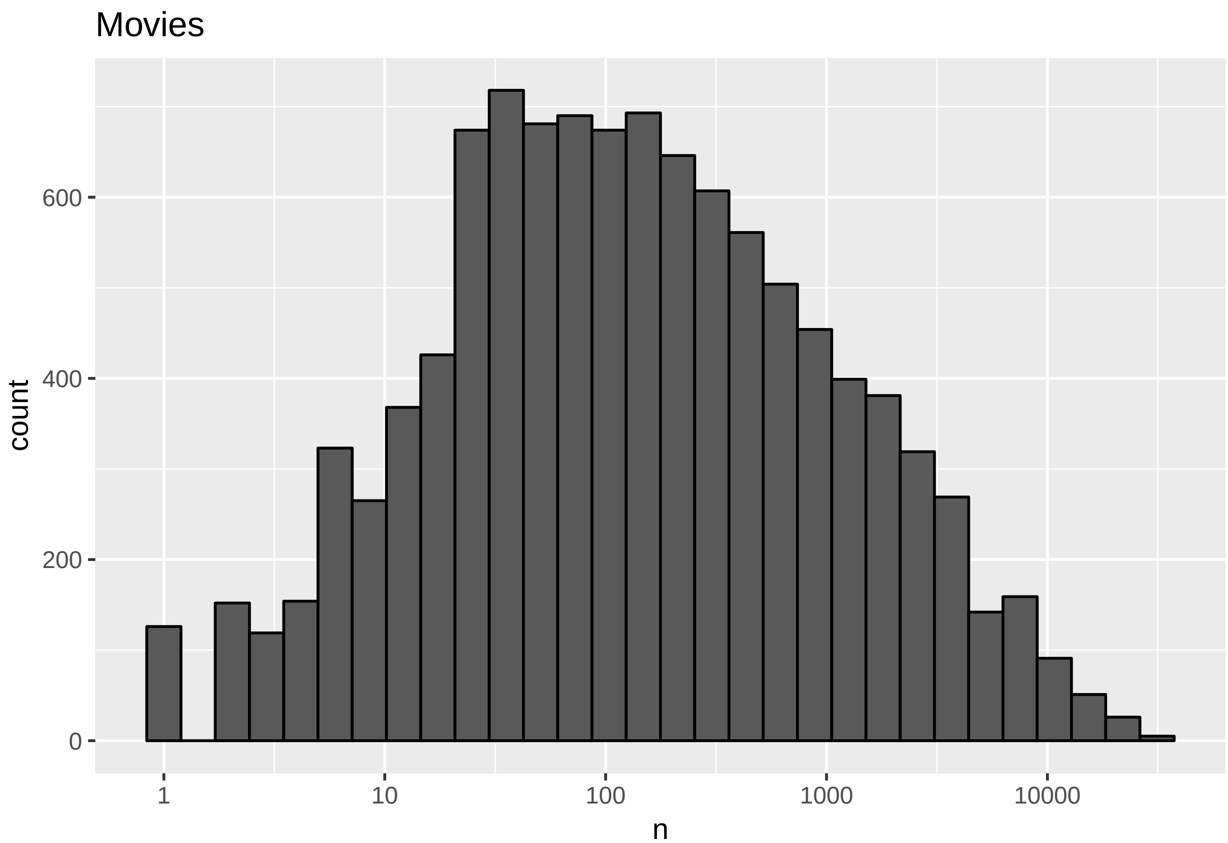
Some users are more active than others:

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



Some movies are rated more often than others:

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



Movie titles showing how the release date is embedded in the title:

```
edx$title[1:10]
```

```
##  [1] "Boomerang (1992)"
##  [2] "Net, The (1995)"
##  [3] "Outbreak (1995)"
##  [4] "Stargate (1994)"
##  [5] "Star Trek: Generations (1994)"
##  [6] "Flintstones, The (1994)"
##  [7] "Forrest Gump (1994)"
##  [8] "Jungle Book, The (1994)"
##  [9] "Lion King, The (1994)"
## [10] "Naked Gun 33 1/3: The Final Insult (1994)"
```

2. Data Cleaning

The release year of the movie is embedded in the title of the movie. This can be extracted and a new column added to both edx and validation for the year.

```
#Date -> Movie year from name
edx <- edx %>%
  mutate(title = str_trim(title)) %>%
  #extract year out of title
  extract(title, c("title_tmp","year"), regex = "^(.*) \\(([0-9 \\-]*)\\)$", remove = F) %>%
  mutate(year = if_else(str_length(year) > 4, as.integer(str_split(year, "-",simplify = T)[1]), as.inte
  #replace title na with orig title
  mutate(title = if_else(is.na(title_tmp), title, title_tmp)) %>%
```

```
  #drop title_tmp
  select(-title_tmp)

validation <- validation %>%
  mutate(title = str_trim(title)) %>%
  #extract year out of title
  extract(title, c("title_tmp","year"), regex = "^(.*) \\(([0-9 \\-]*)\\)$", remove = F) %>%
  mutate(year = if_else(str_length(year) > 4, as.integer(str_split(year, "-",simplify = T)[1]), as.integ
  #replace title na with orig title
  mutate(title = if_else(is.na(title_tmp), title, title_tmp)) %>%
  #drop title_tmp
  select(-title_tmp)
```

For every movie add a column for the genre and use a 1 or 0 to indicate if that movie falls into that genre.

```
names(edx)
```

```
## [1] "userId"    "movieId"   "rating"    "timestamp" "title"     "year"
## [7] "genres"
```

```
edx <- edx %>% separate_rows(genres, sep = "\\|") %>%
  spread(genres,genres) %>%
  #remove (no genre listed)
  select(-`(no genres listed)`)

#fix genre names with hyphens
colnames(edx)[c(16,22)] <- c("FilmNoir","SciFi")

#replace genre na values with 0
edx <- edx %>%  replace_na(list(NoGenre=0,Action=0,Adventure=0,Animation=0,
                  Children=0,Comedy=0,Crime=0,Documentary=0,Drama=0,
                  Fantasy=0,FilmNoir=0,Horror=0,IMAX=0,Musical=0,Mystery=0,
                  Romance=0,SciFi=0,Thriller=0,War=0,Western=0))

#replace genre names in genre columns with 1
edx[edx=="(no genres listed)"|edx=="Action"|edx=="Adventure"|edx=="Animation"|
               edx=="Children"|edx=="Comedy"|edx=="Crime"|edx=="Documentary"|
               edx=="Drama"|edx=="Fantasy"|edx=="Film-Noir"|
               edx=="Horror"|edx=="IMAX"|edx=="Musical"|
               edx=="Mystery"|edx=="Romance"|edx=="Sci-Fi"|
               edx=="Thriller"|edx=="War"|edx=="Western"]<-1

#set up genre columns in validation set
validation <- validation %>% separate_rows(genres, sep = "\\|") %>%
  spread(genres,genres)
#fix genre names with hyphens or (no genre listed)
colnames(validation)[c(16,22)] <- c("FilmNoir","SciFi")


#replace genre na values with 0
validation <- validation %>%  replace_na(list(NoGenre=0,Action=0,Adventure=0,Animation=0,
                                   Children=0,Comedy=0,Crime=0,Documentary=0,Drama=0,
                                   Fantasy=0,FilmNoir=0,Horror=0,IMAX=0,Musical=0,Mystery=0,
                                   Romance=0,SciFi=0,Thriller=0,War=0,Western=0))
```

```
#replace genre names in genre columns with 1
validation[validation=="(no genres listed)"|validation=="Action"|validation=="Adventure"|validation=="An
        validation=="Children"|validation=="Comedy"|validation=="Crime"|validation=="Documentary"|
        validation=="Drama"|validation=="Fantasy"|validation=="Film-Noir"|
        validation=="Horror"|validation=="IMAX"|validation=="Musical"|
        validation=="Mystery"|validation=="Romance"|validation=="Sci-Fi"|
        validation=="Thriller"|validation=="War"|validation=="Western"]<-1
```

3. Build and test models

Here is the loss function that will be used for evaluating the RMSE:

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

Model 0 - Mean only

The first model is to predict that the rating in the validation set is equal to the average of the ratigs in edx.

```
#First Model - all ratings = mean rating
# Y(u,i) = mu + eps(u,i)
mu_hat <- mean(edx$rating)
mu_hat
```

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

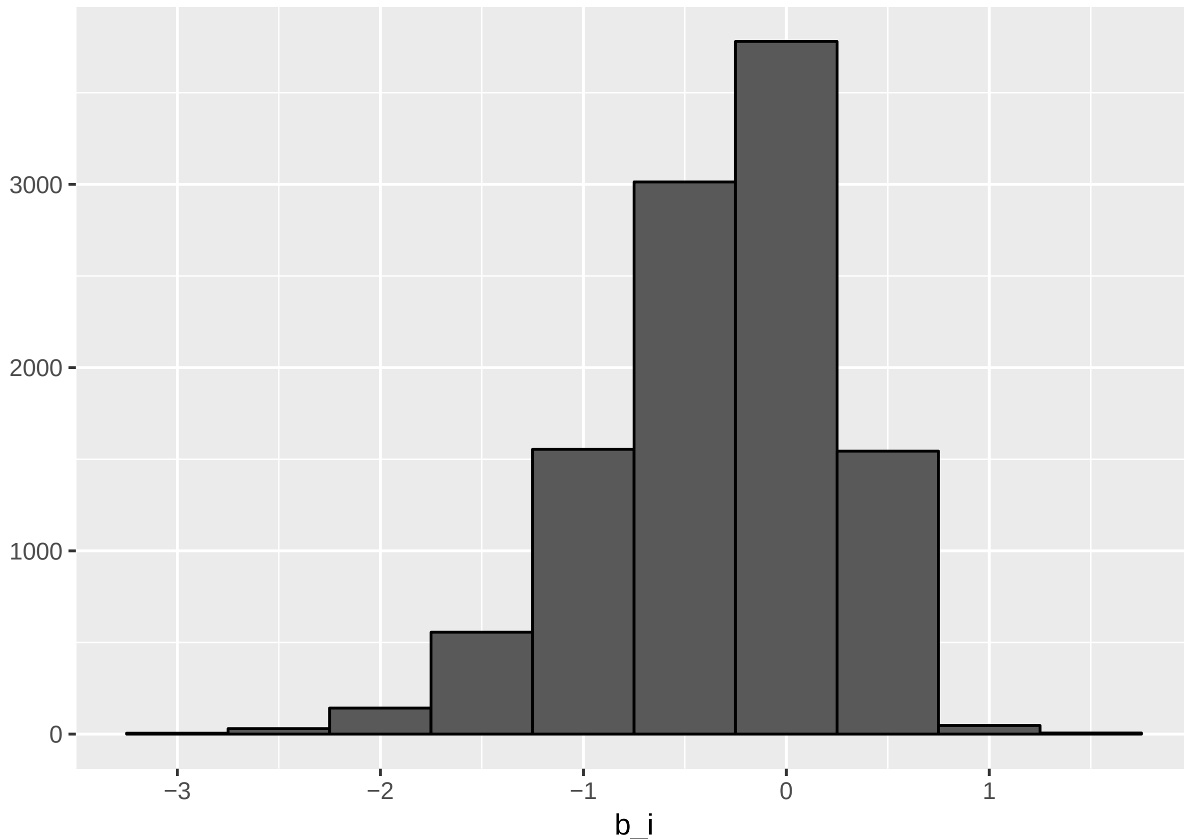
```
#RMSE with all ratings = mu
naive_rmse <-  RMSE(validation$rating, mu_hat)
naive_rmse
```

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

```
#RMSE of 1 is definitely not good enough, however log this result in results df
rmse_results <- data_frame(method = "Mean only", RMSE = naive_rmse)
```

Model 1 - Movie effect

```
#Add movie effect - some movies are rated higher
# Y(u,i) = mu + b(i) + eps(u,i)
#fit <- lm(rating ~ as.factor(userId), data = edx)  -creates memory issue
# get around this by using average of Y(u,i)-mu_hat for each movie i.
mu <- mean(edx$rating)
movie_avgs <- edx %>%
  group_by(movieId) %>%
  summarize(b_i = mean(rating - mu))
#look at variation in these estimates
movie_avgs %>% qplot(b_i, geom = "histogram", bins = 10, data = ., color = I("black"))
```



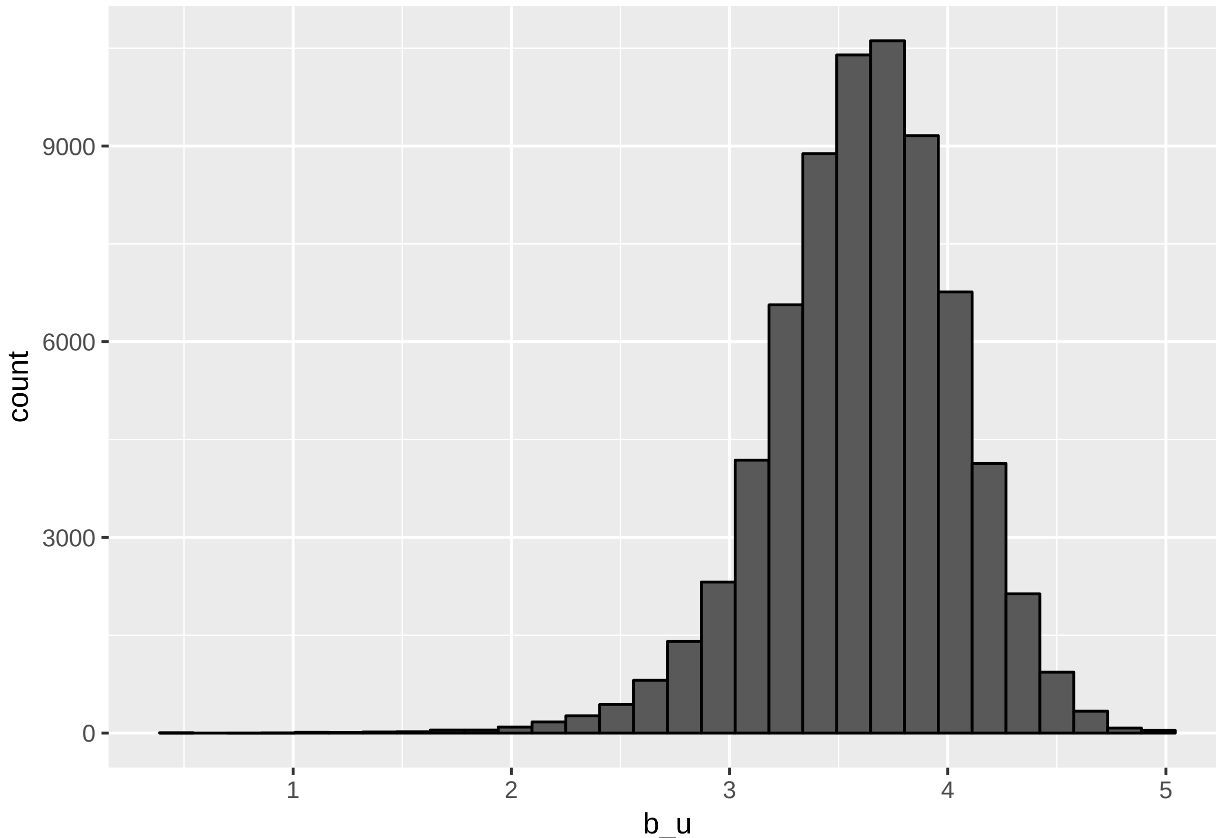

```
#A movie effect of 1.5 implies a rating of 5
#mu=3.5 and b_i=1.5 => rating = 3.5 + 1.5 = 5
#see if prediction improves when using y(u,i) = mu + b_i
predicted_ratings <- mu +  validation %>%
  left_join(movie_avgs, by='movieId') %>%
  .$b_i

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

method	RMSE
Mean only	1.0612018
Movie Effect Model	0.9439087

Model 2 - Movie effect + User effect

```
#user effect
#Y(u,i) = mu + b(i) + b(u) + eps(u,i)
#look at variability of average ratings from users with more than 100 ratings
edx %>%
  group_by(userId) %>%
  summarize(b_u = mean(rating)) %>%
  filter(n()>=100) %>%
  ggplot(aes(b_u)) +
```

```r
  geom_histogram(bins = 30, color = "black")
```



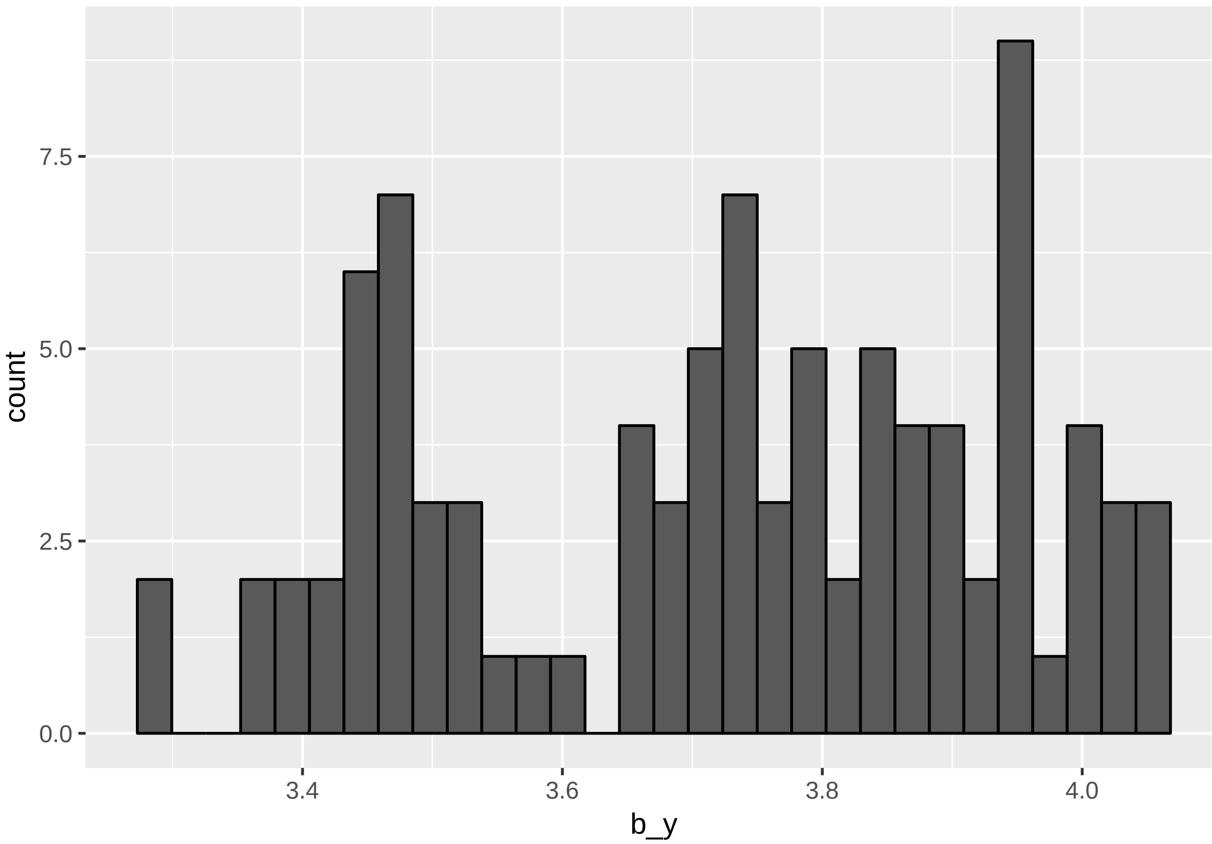

```r
#fit <- lm(rating ~ as.factor(movieId)+as.factor(userId), data = edx)  -creates memory issue
# get around this by using average of Y(u,i)-mu_hat-b(i) for each user u.
user_avgs <- edx %>%
  left_join(movie_avgs, by='movieId') %>%
  group_by(userId) %>%
  summarize(b_u = mean(rating - mu - b_i))
#how much did it improve now
predicted_ratings <- validation %>%
  left_join(movie_avgs, by='movieId') %>%
  left_join(user_avgs, 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="Movie + User Effects Model",
                                     RMSE = model_2_rmse))
rmse_results %>% knitr::kable()
```

method	RMSE
Mean only	1.0612018
Movie Effect Model	0.9439087
Movie + User Effects Model	0.8653488

Model 3 - Release Date Effect

During data exploration it was noticed that the release date is embedded in the title of the movie. The year can be extracted into a separate field as described in the Data Cleaning section of this report. The year will be used in this model to examine if there is an effect due to release year of the movie.

```
#year effect
#Y(y,u,i) = mu + b(i) + b(u) + b(y) + eps(u,i)
#look at variability of average ratings per year
edx %>%
  group_by(year) %>%
  summarize(b_y = mean(rating)) %>%
  ggplot(aes(b_y)) +
  geom_histogram(bins = 30, color = "black")
```




```
#Y(y,u,i)-mu-b(i)-b(u)
year_avgs <- edx %>%
  left_join(user_avgs, by='userId') %>%
  left_join(movie_avgs, by='movieId') %>%
  group_by(year) %>%
  summarize(b_y = mean(rating - mu - b_i - b_u))
#how much did it improve now
predicted_ratings <- validation %>%
  left_join(movie_avgs, by='movieId') %>%
  left_join(user_avgs, by='userId') %>%
  left_join(year_avgs, by='year') %>%
  mutate(pred = mu + b_i + b_u + b_y) %>%
  .$pred
```

```r
model_3_rmse <- RMSE(predicted_ratings, validation$rating)
rmse_results <- bind_rows(rmse_results,
                          data_frame(method="Movie + User + Year Effects Model",
                                     RMSE = model_3_rmse))
rmse_results %>% knitr::kable()
```

method	RMSE
Mean only	1.0612018
Movie Effect Model	0.9439087
Movie + User Effects Model	0.8653488
Movie + User + Year Effects Model	0.8650043

Model 4 - Genre Effect

```r
#Genre effect
#Y(y,u,i) = mu + b(i) + b(u) + b(y) + b(g) + eps(u,i)
#look at variability of average ratings per genre
genres <- c("Action","Adventure","Animation","Children",
            "Comedy","Crime","Documentary","Drama",
            "Fantasy","FilmNoir","Horror","IMAX","Musical",
            "Mystery","Romance","SciFi","Thriller","War","Western")

#Calculate the average rating per genre. Apply the average for each genre that a movie belongs to.
genre_avgs <- sapply(genres, function(gnre){

  edx[edx[[gnre]]==1,] %>%
    left_join(user_avgs, by='userId') %>%
    left_join(movie_avgs, by='movieId') %>%
    left_join(year_avgs, by='year') %>%
    group_by(by=gnre) %>%
    summarize(b_g = mean(rating - mu - b_i - b_u - b_y))

})

genre_avgs
```

```
##     Action      Adventure   Animation   Children    Comedy
## by  "Action"    "Adventure" "Animation" "Children"  "Comedy"
## b_g -0.01209246 -0.0157208  -0.01731843 -0.02412881 -0.0009876096
##     Crime       Documentary   Drama      Fantasy      FilmNoir
## by  "Crime"     "Documentary" "Drama"    "Fantasy"    "FilmNoir"
## b_g 0.008350267 0.05503319    0.01064426 -0.008623106 0.01532597
##     Horror      IMAX        Musical    Mystery    Romance      SciFi
## by  "Horror"    "IMAX"      "Musical"  "Mystery"  "Romance"    "SciFi"
## b_g 0.002223779 -0.01471761 -0.0155032 0.01022479 -0.001925849 -0.01423865
##     Thriller     War         Western
## by  "Thriller"   "War"       "Western"
## b_g -0.003128861 0.002100078 -0.009352553
```

```r
#Create genre_temp to with column g to save total of each genre effect for every movie
```

```r
genre_temp <- edx %>% mutate(movieId, g=0)
genre_temp$g <- genre_temp$g + as.numeric(genre_temp[["Action"]])*genre_avgs[,"Action"]$b_g
genre_temp$g <- genre_temp$g + as.numeric(genre_temp[["Adventure"]])*genre_avgs[,"Adventure"]$b_g
genre_temp$g <- genre_temp$g + as.numeric(genre_temp[["Animation"]])*genre_avgs[,"Animation"]$b_g
genre_temp$g <- genre_temp$g + as.numeric(genre_temp[["Children"]])*genre_avgs[,"Children"]$b_g
genre_temp$g <- genre_temp$g + as.numeric(genre_temp[["Comedy"]])*genre_avgs[,"Comedy"]$b_g
genre_temp$g <- genre_temp$g + as.numeric(genre_temp[["Crime"]])*genre_avgs[,"Crime"]$b_g
genre_temp$g <- genre_temp$g + as.numeric(genre_temp[["Documentary"]])*genre_avgs[,"Documentary"]$b_g
genre_temp$g <- genre_temp$g + as.numeric(genre_temp[["Drama"]])*genre_avgs[,"Drama"]$b_g
genre_temp$g <- genre_temp$g + as.numeric(genre_temp[["Fantasy"]])*genre_avgs[,"Fantasy"]$b_g
genre_temp$g <- genre_temp$g + as.numeric(genre_temp[["FilmNoir"]])*genre_avgs[,"FilmNoir"]$b_g
genre_temp$g <- genre_temp$g + as.numeric(genre_temp[["Horror"]])*genre_avgs[,"Horror"]$b_g
genre_temp$g <- genre_temp$g + as.numeric(genre_temp[["IMAX"]])*genre_avgs[,"IMAX"]$b_g
genre_temp$g <- genre_temp$g + as.numeric(genre_temp[["Musical"]])*genre_avgs[,"Musical"]$b_g
genre_temp$g <- genre_temp$g + as.numeric(genre_temp[["Mystery"]])*genre_avgs[,"Mystery"]$b_g
genre_temp$g <- genre_temp$g + as.numeric(genre_temp[["Romance"]])*genre_avgs[,"Romance"]$b_g
genre_temp$g <- genre_temp$g + as.numeric(genre_temp[["SciFi"]])*genre_avgs[,"SciFi"]$b_g
genre_temp$g <- genre_temp$g + as.numeric(genre_temp[["Thriller"]])*genre_avgs[,"Thriller"]$b_g
genre_temp$g <- genre_temp$g + as.numeric(genre_temp[["War"]])*genre_avgs[,"War"]$b_g
genre_temp$g <- genre_temp$g + as.numeric(genre_temp[["Western"]])*genre_avgs[,"Western"]$b_g

#get average genre effect per movie id
ga_avgs <- genre_temp %>%
  group_by(movieId) %>%
  select(movieId,g) %>%
  summarize(b_g = mean(g))

#remove genre_temp to free up memory
rm(genre_temp)

#how much did it improve now with genre effects
predicted_ratings <- validation %>%
  left_join(movie_avgs, by='movieId') %>%
  left_join(user_avgs, by='userId') %>%
  left_join(year_avgs, by='year') %>%
  left_join(ga_avgs, by='movieId') %>%
  mutate(pred = mu + b_i + b_u + b_y + b_g) %>%
  .$pred

model_5_rmse <- RMSE(predicted_ratings, validation$rating)
rmse_results <- bind_rows(rmse_results,
                          data_frame(method="Movie + User + Year + Genre",
                                     RMSE = model_5_rmse))
rmse_results %>% knitr::kable()
```

method	RMSE
Mean only	1.0612018
Movie Effect Model	0.9439087
Movie + User Effects Model	0.8653488
Movie + User + Year Effects Model	0.8650043
Movie + User + Year + Genre	0.8649543

Results

The following RMSE results were obtained:

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

method	RMSE
Mean only	1.0612018
Movie Effect Model	0.9439087
Movie + User Effects Model	0.8653488
Movie + User + Year Effects Model	0.8650043
Movie + User + Year + Genre	0.8649543

Conclusion

Fairly good RMSE values are possble with linear regression methods. This will allow for a decent recommendation system to be built based on these effects.