MovieLens Report

Masayoshi Sato

2021/5/3

Introduction

Nowadays, recommendation systems have become common to us. Many products and services are rated by consumers using stars and points, and these ratings are utilized by companies and organizations. They are precious; they are not only helpful for customers to choose goods or services, but are also valuable for makers and service providers in terms of predicting consumer behaviors. If you explore user ratings in detail, you will find their customers preference, and be able to recommend things which may be most likely to be bought or used.

One of the field in which this system is used is movie industries. In this paper, I will take a movie recommendation dataset, and build a model which will predict rating using factors as accurately as possible.

Dataset

The data used in this report, MovieLens 10M Dataset, is available at the Grouplens website. According to the site, this data set has 10000054 ratings and 95580 tags applied to 10681 movies by 71567 users of the online movie recommender service MovieLens.

MovieLens 10M dataset https://grouplens.org/datasets/movielens/10m/

Goal

The goal is to train a machine learning algorithm to predict movie ratings based on factors in the provided dataset.

As GroupLens web site says, it has several factors, such as user ID, movie title, genres, rating as well. The assumption in this paper is that they are inter-correlated to some extent and helpful to forecast ratings. The challenge of a recommendation system, however, is that the data is sparse and each factor in the dataset might affect others. Namely, some users rate movies more than others, and some movies are rated more. Genres, released year may also affect ratings.

Therefore, a linear regression will be used to find weights that minimize the residual mean squared error, the RMSE. Predictors are user ID, movie ID, genres. and released year. In addition to this, regulation is used to penalize large estimates that are produced by using small samples.

Exploratory Analysis

The downloaded data is split into two, edx dataset (90%) and validation dataset (10%).

The validation dataset has 999999 rows and 6 columns. And the edx dataset has 9000055 rows and 6 columns.The validation dataset is left to final evaluation of a model explored in this paper.

The first six rows of the edx dataset is as follows;

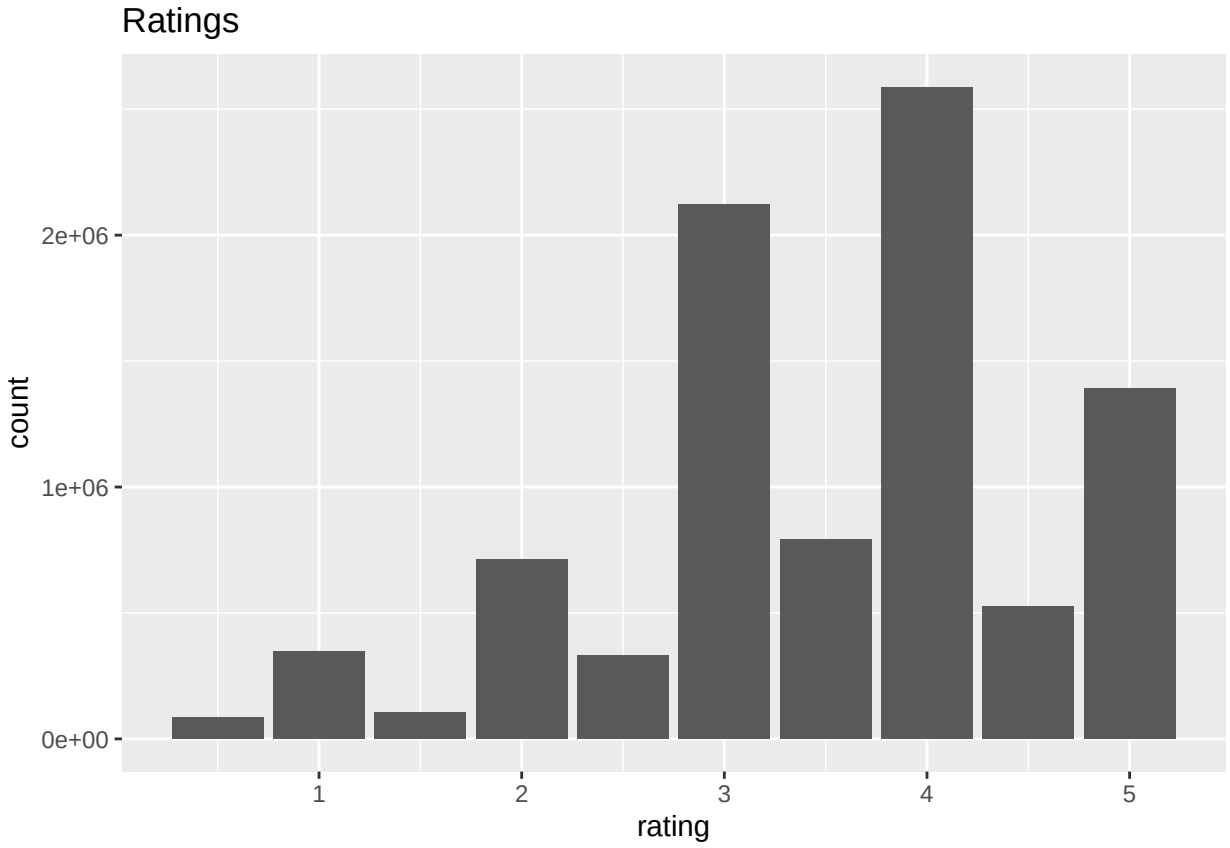
```
##    userId movieId rating timestamp                         title
## 1:      1     122      5 838985046               Boomerang (1992)
## 2:      1     185      5 838983525                Net, The (1995)
## 3:      1     292      5 838983421                Outbreak (1995)
## 4:      1     316      5 838983392                Stargate (1994)
## 5:      1     329      5 838983392 Star Trek: Generations (1994)
## 6:      1     355      5 838984474       Flintstones, The (1994)
##                           genres
## 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
```

UserId indicates a person who rate movies (rating). Users rate multiple movies. Each movie corresponds to movieId and title. Title has a movie's release year in a parenthesis. Movies are categorized by genres, which consists of several words, such as "Comedy", "Crime", or "Sci-Fi". They are separated by "|".

We will investigate columns to find out their features.

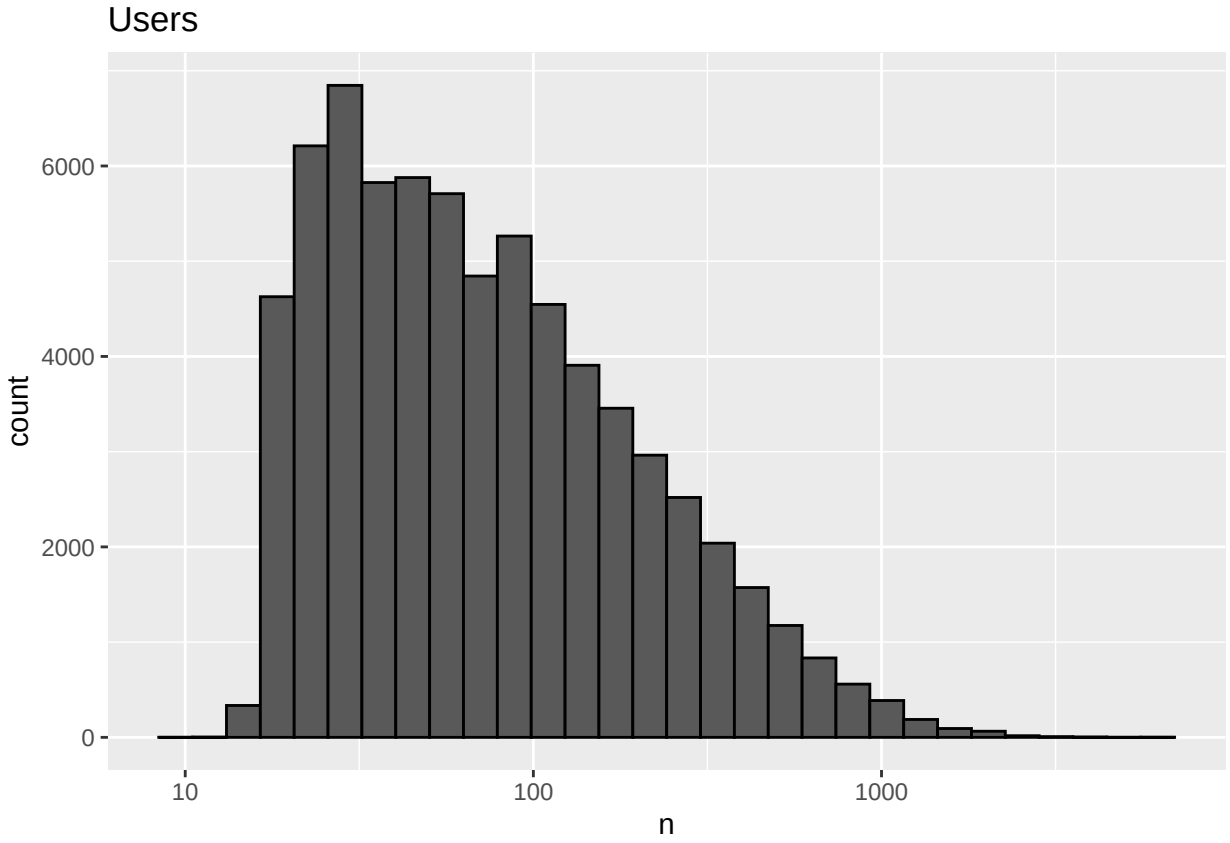
Rating

Ratings ranges from 0.5 to 5.0 by 0.5. 5.0 is the highest. The distribution is right-skewed. The mean of the rating is 3.51. The median is 4.



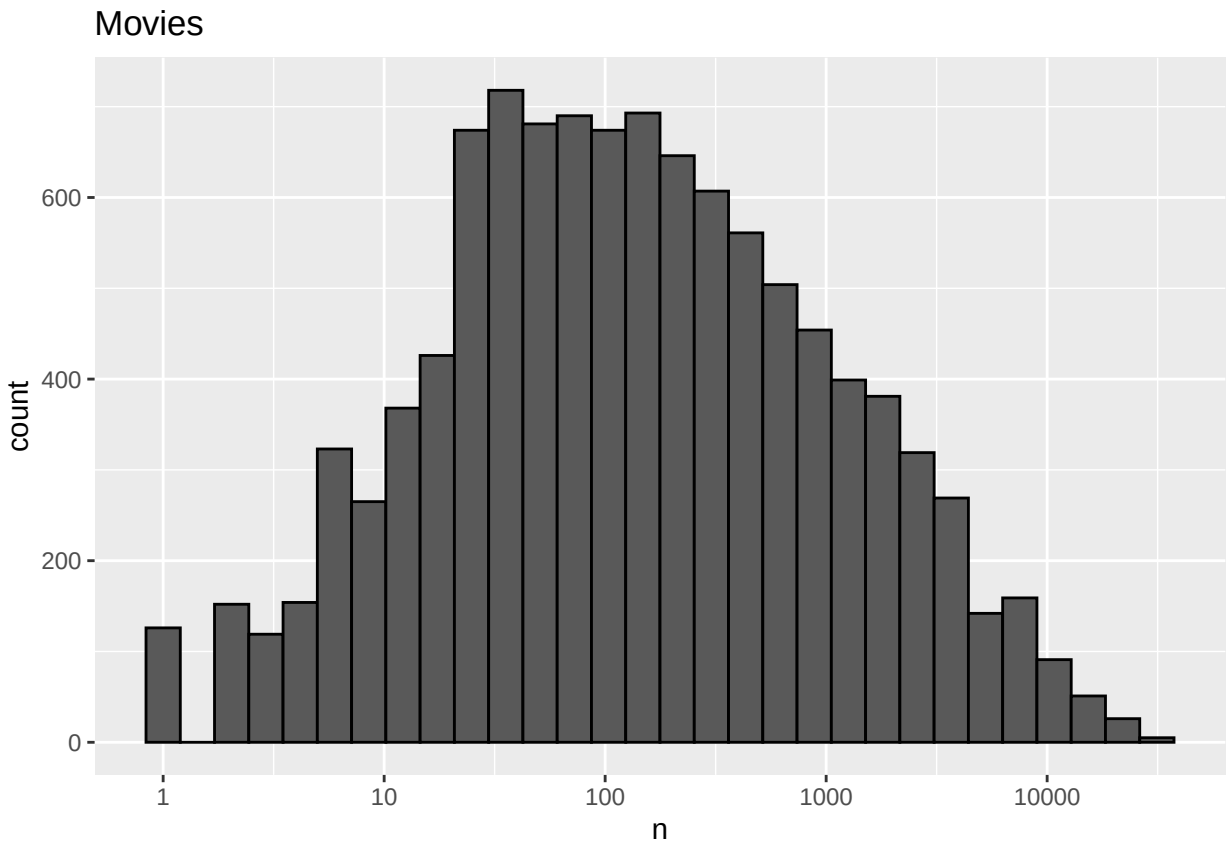
Users

The number of user (unique user) is 69878. As is seen in the first six rows of the edx, users rate multiple movies. It's distribution is left-skewed. Some users are very active in rating movies. We clearly see that there is a user effect which affect ratings.



Movies

10677 movies (the number of unique movies) are rated. They are rated by multiple users. The histogram tells us some movies get very small amount of ratings and some get huge amount of ratings. It surely leads to producing effects.



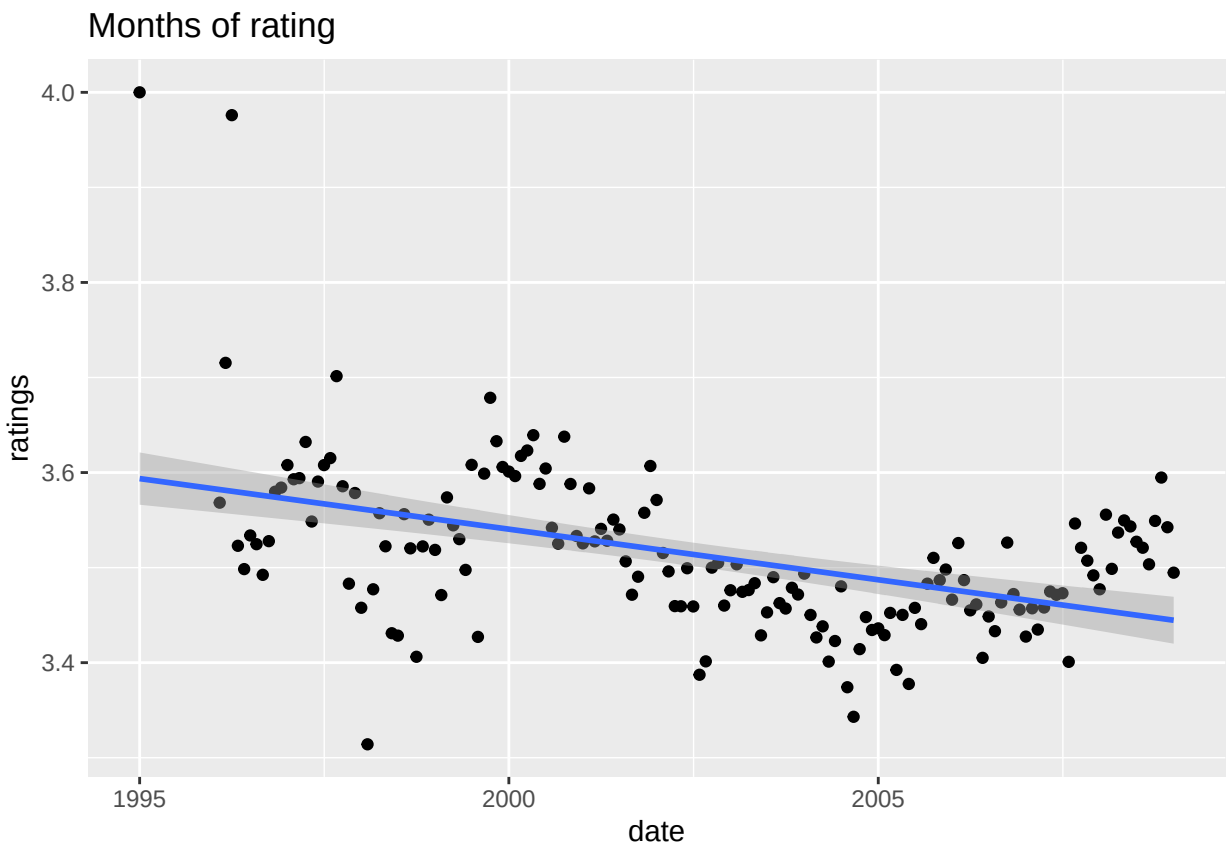
Timestamp

The class of "timestamp" is an integer. They are the exact date of rating, produced by counting days from the day, "1970.01.01". To deal with them, these figures need to covert to POSIXct data.

To find out the correlation between the date of rate and rating, we create a new column "date". The column have a month and year of the date when being rated.

We plot rating and date showing a regression line. Points are mainly gathered around 3.5 rating, and the regression line is relatively flat. We leave out the rate dates since they seem not to have much significance.

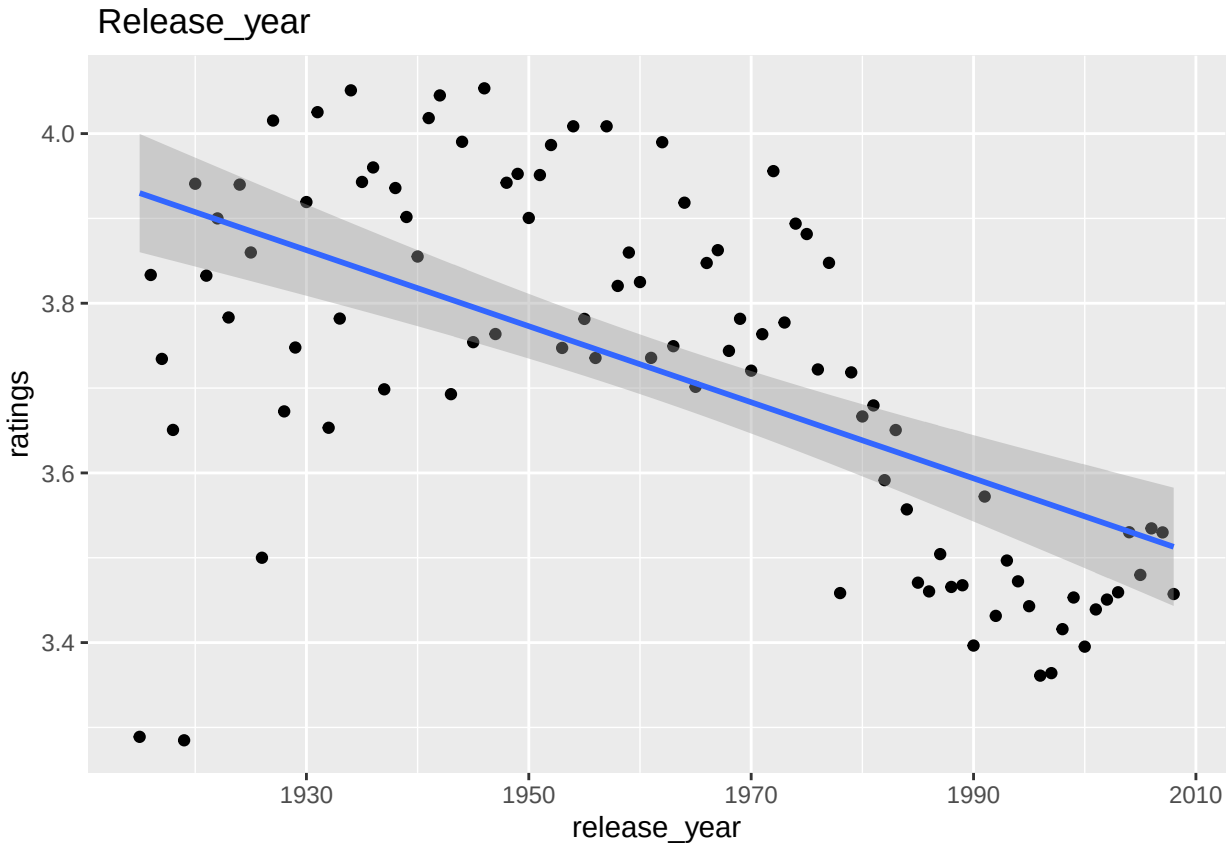
```
## `geom_smooth()` using formula 'y ~ x'
```



Title

Titles are difficult to deal with as they are. But they have release years in parentheses. We extract years from titles. Then, we mutate new columns,“release year”. We plot rating and release year along showing a regression line. There seems to be a tendency that recent movies are rated low and old movies are rated high. The release year works as effects to predict rating.

```
## `geom_smooth()` using formula 'y ~ x'
```



Genres

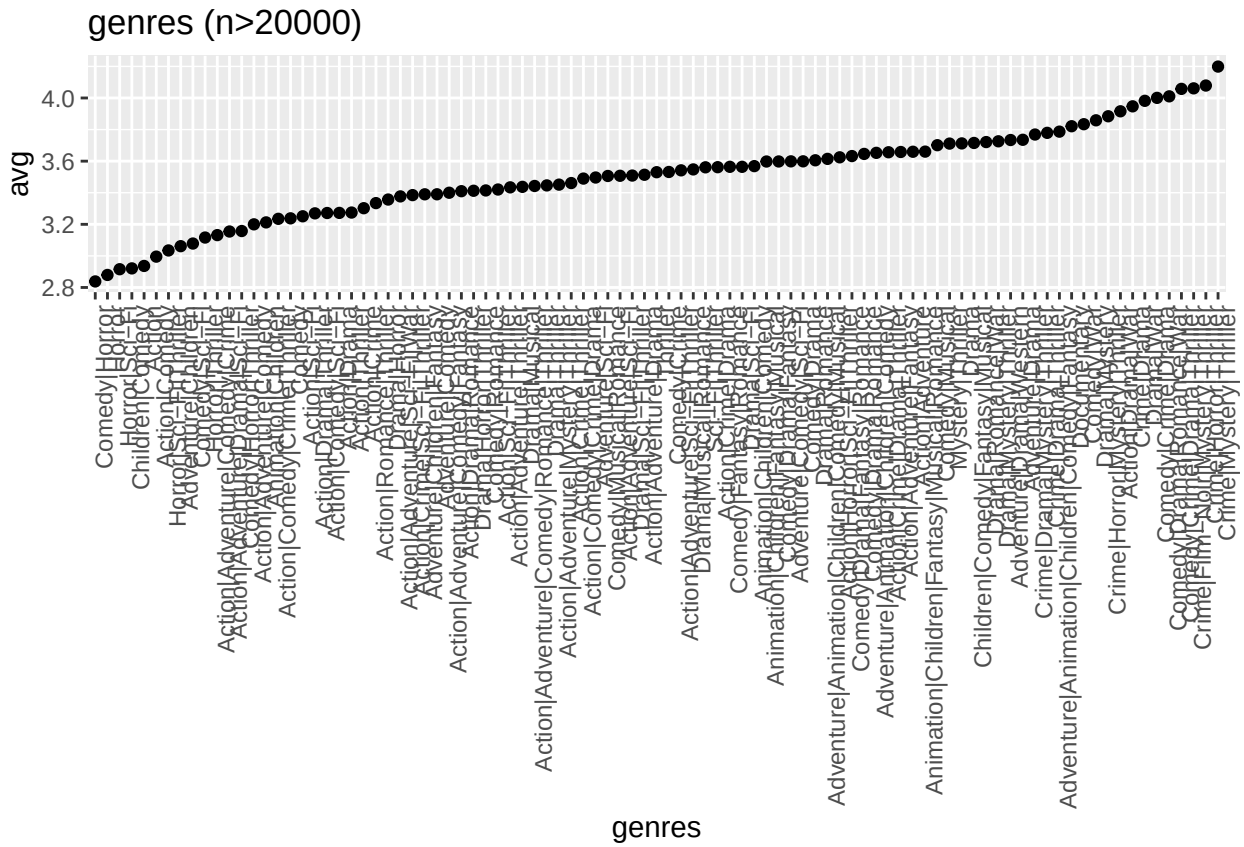
Some like comedy, but do not like thriller. Some like Sci-fi but do not like animation. Such a preference naturally affects movies' rating.

The edx has unique 797 genres. Here are top 10 genres in terms of high rating.

```
## # A tibble: 10 x 2
##    genres                                ratings
##    <chr>                                   <dbl>
##  1 Animation|IMAX|Sci-Fi                    4.71
##  2 Drama|Film-Noir|Romance                  4.30
##  3 Action|Crime|Drama|IMAX                  4.30
##  4 Animation|Children|Comedy|Crime          4.28
##  5 Film-Noir|Mystery                        4.24
##  6 Crime|Film-Noir|Mystery                  4.22
##  7 Film-Noir|Romance|Thriller               4.22
##  8 Crime|Film-Noir|Thriller                 4.21
##  9 Crime|Mystery|Thriller                   4.20
## 10 Action|Adventure|Comedy|Fantasy|Romance  4.20
```

Each genre consists of several words, such as "Action", "Drama", or"Crime". Instead of splitting them, we try to deal with them as one combination, and find correlation between rating and genre. To make it simple, genres which have more than 20000 ratings are plotted in the graph.

As in seen in the graph, some genres are rated very low (minimum approx. 2.8) and some are rated very high (maximum approx. 4.2). From this, we understand that genres (even though they are combined) affect ratings considerably.



Exploration Summary

From these studies, we found that "movieId", "userId", "genres", and "release year (extracted from"title") are important factors deciding rating.

In order to make models, we split the edx dataset into training set and test set.

```
set.seed(1, sample.kind = "Rounding")
```

```
## Warning in set.seed(1, sample.kind = "Rounding"): non-uniform 'Rounding' sampler
## used
```

```
test_index <- createDataPartition(y = edx$rating, times = 1, p = 0.1, 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")
```

To assess models, we use the RMSE. It is defined as :

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

Here, $y_{u,i}$ is rating for movie i by user i. $\hat{y}_{u,i}$ is prediction, and $N$is the number of user/ movie combinations. Models, which will be described in the next chapter, are evaluated by the RMSE. We create R function for this purpose.

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

The model which minimize the RMSE in the test set will be applied to the validation dataset, and extract its RMSE.

Model Building

1 Baseline model, assuming the same rating for all movies

First, we make a model assuming ratings of all movies are the same. We use the average of all movies ratings. Here is a code to produce its RMSE.

```
mu <- mean(train_set$rating)
naive_rmse <- RMSE(test_set$rating, mu)
naive_rmse
```

```
## [1] 1.060054
```

This is a baseline of our further modeling.

```
## # A tibble: 1 x 2
##   method             RMSE
##   <chr>             <dbl>
## 1 1. Average rating  1.06
```

2 Movie effects

Next, we create a linear regression model in which movie effects are taken into account. Here is an equation. Yu, i is outcome. μ is the average of ratings. b_i is movie effect. $\epsilon_{u,i}$ is independent error.

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

This is a code to calculate the RMSE.

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

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

movie_effect_pred <- mu + test_set %>%
  left_join(movie_avg, by="movieId") %>%
  pull(b_i)

movie_effect_rmse <- RMSE(test_set$rating, movie_effect_pred)
movie_effect_rmse
```

```
## [1] 0.9429615
```

The RMSE is improved.

```
## # A tibble: 2 x 2
##   method                 RMSE
##   <chr>                 <dbl>
## 1 1. Average rating     1.06
## 2 2 Movie Effects Model 0.943
```

3 Movie and user effects

As well as movie effect, user effect can affect the prediction. In this model, both movie and user effect are considered. This model can be represented in this equation. b_u is user effect. ;

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

Here is a code to produce the RMSE.

```
user_avg <- train_set %>%
  left_join(movie_avg, by= "movieId") %>%
  group_by(userId) %>%
  summarize(b_u = mean(rating- mu -b_i))

movie_user_effect_pred <- test_set %>%
  left_join(movie_avg, by="movieId") %>%
  left_join(user_avg, by="userId") %>%
  mutate(pred = mu + b_i + b_u)%>%
  pull(pred)

movie_user_effect_rmse <- RMSE(test_set$rating, movie_user_effect_pred)
movie_user_effect_rmse
```

```
## [1] 0.8646844
```

```
## # A tibble: 3 x 2
##   method                       RMSE
```

```
##   <chr>                      <dbl>
## 1 1. Average rating          1.06
## 2 2 Movie Effects Model      0.943
## 3 3 Movie User Effects Model 0.865
```

4 Genres and release year effects

In our exploration, we found that genres and release year also seem to affect the prediction. Does considering these two factors improve the RMSE? We assume genre effect and release year effect. This can be represented as a following linear regression model. $g_{u,i}$ is genre effect, and $yr_{u,i}$ is year effect.

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

The RMSE is produced by this code.

```
g_avg <- train_set %>%
  left_join(movie_avg, by= "movieId")%>%
  left_join(user_avg, by="userId") %>%
  group_by(genres) %>%
  summarize(g = mean(rating- mu -b_i -b_u))

y_avg <- train_set %>%
  left_join(movie_avg, by= "movieId")%>%
  left_join(user_avg, by="userId") %>%
  left_join(g_avg, by="genres") %>%
  group_by(release_year) %>%
  summarize(y = mean(rating- mu -b_i -b_u -g))

movie_user_genre_year_effect_pred <- test_set %>%
  left_join(movie_avg, by="movieId") %>%
  left_join(user_avg, by="userId") %>%
  left_join(g_avg, by="genres") %>%
  left_join(y_avg, by="release_year")%>%
  mutate(pred = mu + b_i + b_u + g + y)%>%
  pull(pred)

movie_user_genre_year_effect_rmse <- RMSE(test_set$rating,
                                          movie_user_genre_year_effect_pred)
movie_user_genre_year_effect_rmse
```

```
## [1] 0.8641262
```

It has been much improved.

```
## # A tibble: 4 x 2
##   method                                      RMSE
##   <chr>                                      <dbl>
## 1 1. Average rating                          1.06
## 2 2 Movie Effects Model                      0.943
## 3 3 Movie User Effects Model                 0.865
## 4 4 Movie User Genre Release Year Effects Model 0.864
```

5 Regularization (movie, user)

As we mentioned in the introduction, the challenge of a recommendation system is that the dataset is sparse and uneven. Namely, some users rate movies more than others, some movies are rated more. And some movies are rated high by very few users. That produces large errors, in consequence increases the RMSE. To improve this, regularization should be introduced. It penalizes large estimates which come from small sample sizes, and constrains the total variability.

We introduce a tuning parameter λ.

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

The first term is a mean squared error. The second term is a penalty term. If sum of b_i becomes larger, the penalty term increases the equation's value.

The value of b that minimizes the equation is

$$\hat{b}_i(\lambda)=\frac{1}{\lambda+n_i}\sum_{u=1}^{n_i}(Y_{u,i}-\hat{\mu}),$$

n is a number of ratings b for movie i. b can be expressed as a function of λ. Therefore, we can create a linear regression model using a parameter λ.

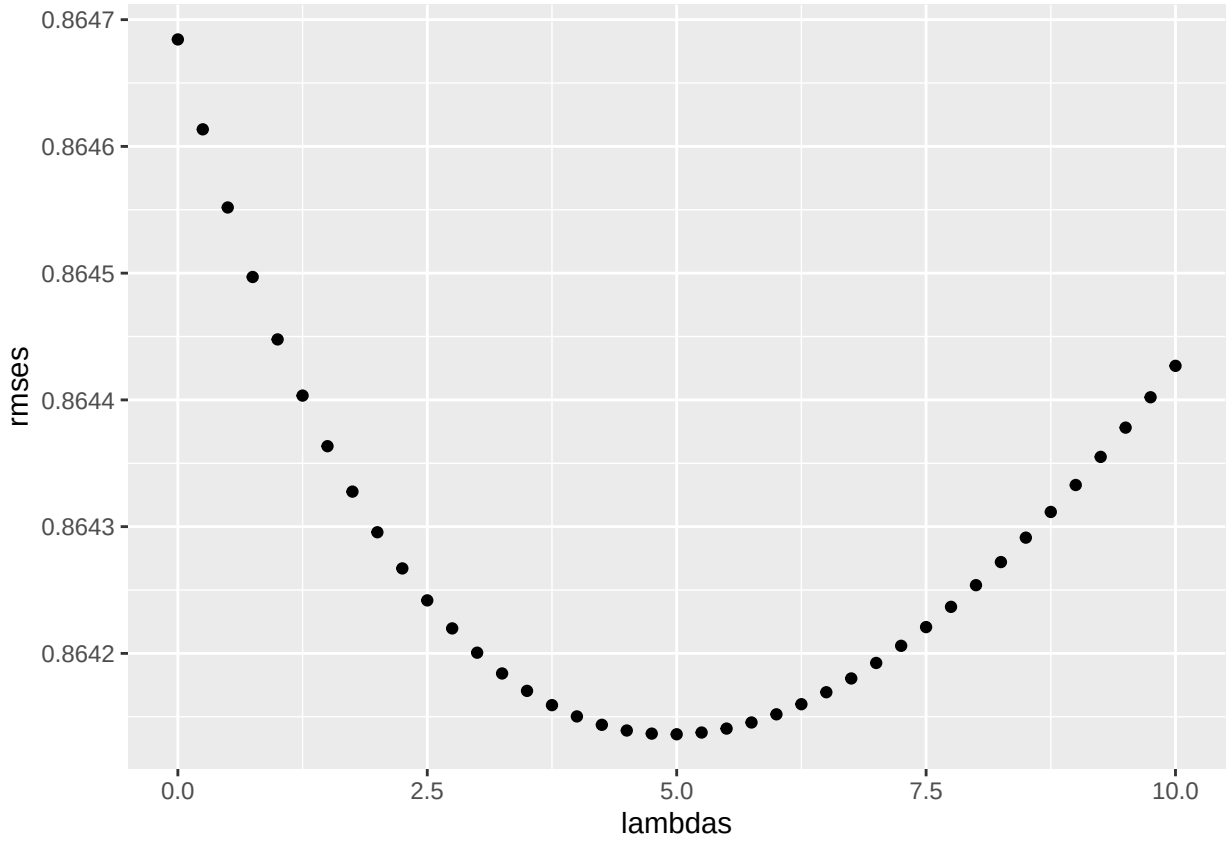
Based on this idea, we can incorporate the movie effect as follows.

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

First, we look for he value of λ that minimize the RMSE. using a following code.

```
lambdas <- seq(0, 10, 0.25)

rmses <- sapply(lambdas, function(l){

  mu <- mean(train_set$rating)

  reg_movie_avg <- train_set %>%
    group_by(movieId) %>%
    summarize(b_i = sum(rating - mu)/(n()+l))

  reg_user_avg <- train_set %>%
    left_join(reg_movie_avg, by="movieId") %>%
    group_by(userId) %>%
    summarize(b_u = sum(rating - b_i - mu)/(n()+l))

  predicted_ratings <- test_set %>%
    left_join(reg_movie_avg, by = "movieId") %>%
    left_join(reg_user_avg ,by = "userId") %>%
    mutate(pred = mu + b_i + b_u) %>%
    pull(pred)

  return(RMSE(predicted_ratings, test_set$rating))

})
```

Then we plot λ and rmses.



The value of λ which minimize rmses is

```
## [1] 5
```

The minimized value of rmses is

```
## [1] 0.8644477
```

```
## # A tibble: 5 x 2
##   method                                     RMSE
##   <chr>                                     <dbl>
## 1 1. Average rating                         1.06
## 2 2 Movie Effects Model                     0.943
## 3 3 Movie User Effects Model                0.865
## 4 4 Movie User Genre Release Year Effects Model 0.864
## 5 5 Reg Movie User Effects Model            0.864
```

Compared to the previous “Movie User Effect Model” ,the RMSE is improved, but it is worse than " Movie User Genre Release Year Effect Model". Thus we try to create another model.

6 Regularization (movie, user, genres, release_year)

In this model, we will take into account both genres and release year in the regularization. To find the value of λ that minimize the RMSE, we use a following code.

```
lambdas <- seq(0, 10, 0.25)

rmses <- sapply(lambdas, function(l){

  mu <- mean(train_set$rating)

  reg_movie_avg <- train_set %>%
    group_by(movieId) %>%
    summarize(b_i = sum(rating - mu)/(n()+l))

  reg_user_avg <- train_set %>%
    left_join(reg_movie_avg, by="movieId") %>%
    group_by(userId) %>%
    summarize(b_u = sum(rating - b_i - mu)/(n()+l))

  g_avg <- train_set %>%
    left_join(reg_movie_avg, by="movieId") %>%
    left_join(reg_user_avg, by="userId") %>%
    group_by(genres) %>%
    summarize(g = sum(rating - b_i -b_u - mu)/(n()+l))

    y_avg <- train_set %>%
    left_join(reg_movie_avg, by="movieId") %>%
    left_join(reg_user_avg, by="userId") %>%
    left_join(g_avg, by="genres") %>%
    group_by(release_year) %>%
    summarize(y = sum(rating - b_i -b_u -g - mu)/(n()+l))

  predicted_ratings <-test_set %>%
    left_join(reg_movie_avg, by = "movieId") %>%
    left_join(reg_user_avg, by = "userId") %>%
    left_join(g_avg, by="genres") %>%
    left_join(y_avg, by="release_year")%>%
    mutate(pred = mu + b_i + b_u + g + y) %>%
    pull(pred)

  return(RMSE(predicted_ratings, test_set$rating))
})
```

The value of λ which minimize the RMSE is ;

```
## [1] 4.5
```

The RMSE is;

```
## [1] 0.8636478
```

```
## # A tibble: 6 x 2
##   method                                    RMSE
##   <chr>                                    <dbl>
## 1 1. Average rating                        1.06
## 2 2 Movie Effects Model                    0.943
```

```
## 3 3 Movie User Effects Model                          0.865
## 4 4 Movie User Genre Release Year Effects Model     0.864
## 5 5 Reg Movie User Effects Model                      0.864
## 6 6 Reg Movie User Genre Release Year Effects Model 0.864
```

Final model "Reg Movie User Genre Release Year Effects Model" is proved to extract the least RMSE Value in the test set.

Evaluation

Before the final evaluation, the validation set needs to be arranged to fit the train set. The Release Year column should be added from title column.

Find the value of λ that minimize the RMSE. During the process, will will replace NAs in the validation with mu (the average rating in the training).

Then we find lambda which minimize the RMSE.

```
## [1] 5
```

The final RMSE is;

```
## [1] 0.8646978
```

We found out that the last model in which a regularized linear regression with predictors (userId, movieId, genres, and release year) produces the least value of the RMSE. **The value is 0.8646978**

Conclusion

In this report, we made 6 models. The basic method is a linear regression. In the test set, the simple regression model "Movie User Genre Release Year Effects Model" showed improvement by 18.483% compared to the first baseline model (only the average rating is considered).

Regularization has prove to be effective. However, if you take only two factors, our exploration showed that it was less effective than the 4th model. But the regularized model was improved once it took factors, genres and release year into the regularized model. The "Reg Movie User Genre Release Year Effects Model" proved to the best in the test set. Compared to the first model, it illustrated 18.499% improvement. Our final RMSE result from the validation set is 0.86469.

Having said that, this exploration revealed its limitation. As we increase predictors, the improvement becomes more and more small. The difference between 4th model and 6th model in terms of the RMSE is approximately 0.00017. To improve the result, it is necessary to consider other factors like correlations between genres and introduce other method such as matrix factorization.