HarvardX PH125.9x Data Science Capstone: MovieLens Project

Daoxia Ding

7/1/2022

Contents

1 Introduction

The MovieLens data is provided as part of HarvardX PH125.9x Data Science Capstone course. The reduced size version was also used throughout the textbook by Rafael Irizarry. It includes more than 10 million ratings user submitted for movies.
The data set can be found through the link below.
[linked phrase]https://files.grouplens.org/datasets/movielens/ml-10m.zip
The target of this project is to develop and train a recommendation system model based on the MovieLens 10M dataset. The Residual Mean Square Error (RMSE) is used to evaluate the loss of the algorithm. The ultimate target of RMSE is to reach below 0.86490.
Due to the large size of the data, existing R lm() model is not used due to computational limits on the laptop. Instead, we are computing it without using lm() in R. We will also use regularization in the model to penalize large estimates that are formed using small sample sizes.
This report, following the course requirements, will explain the process to explore the data, clean the data, split the data for training and testing, identify four effects to be included in the linear model, implement regularization, and eventually apply the model on the validation set and conclude with the RMSE result.

2 Data Exploration

2.1 General Overview

We review the dimension of the dataset "edx",

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

as well as the dimension of the dataset "validation":

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

A look at the head and summary of the dataset "edx" reveals that there are 6 columns. timestamp column would need to be converted to date format. We may need to split the genre column to individual categories instead of a single string.

userId	movieId	rating	timestamp	title	genres
1	122	5	838985046	Boomerang (1992)	Comedy|Romance
1	185	5	838983525	Net, The (1995)	Action|Crime|Thriller
1	292	5	838983421	Outbreak (1995)	Action|Drama|Sci-Fi|Thriller
1	316	5	838983392	Stargate (1994)	Action|Adventure|Sci-Fi
1	329	5	838983392	Star Trek: Generations (1994)	Action|Adventure|Drama|Sci-Fi
1	355	5	838984474	Flintstones, The (1994)	Children|Comedy|Fantasy

```
##      userId         movieId          rating        timestamp
##  Min.   :    1   Min.   :    1   Min.   :0.500   Min.   :7.897e+08
##  1st Qu.:18124   1st Qu.:  648   1st Qu.:3.000   1st Qu.:9.468e+08
##  Median :35738   Median : 1834   Median :4.000   Median :1.035e+09
##  Mean   :35870   Mean   : 4122   Mean   :3.512   Mean   :1.033e+09
##  3rd Qu.:53607   3rd Qu.: 3626   3rd Qu.:4.000   3rd Qu.:1.127e+09
##  Max.   :71567   Max.   :65133   Max.   :5.000   Max.   :1.231e+09
##     title              genres
##  Length:9000055     Length:9000055
##  Class :character   Class :character
##  Mode  :character   Mode  :character
##
##
##
```

Below is the head and summary of the dataset "validation". It's very similar to "edx".

userId	movieId	rating	timestamp	title	genres
1	231	5	838983392	Dumb & Dumber (1994)	Comedy
1	480	5	838983653	Jurassic Park (1993)	Action|Adventure|Sci-Fi|Thriller
1	586	5	838984068	Home Alone (1990)	Children|Comedy
2	151	3	868246450	Rob Roy (1995)	Action|Drama|Romance|War
2	858	2	868245645	Godfather, The (1972)	Crime|Drama
2	1544	3	868245920	Lost World: Jurassic Park, The (Jurassic Park 2) (1997)	Action|Adventure|Horror|Sci-Fi|Thriller

```
##      userId         movieId          rating        timestamp
##  Min.   :    1   Min.   :    1   Min.   :0.500   Min.   :7.897e+08
##  1st Qu.:18096   1st Qu.:  648   1st Qu.:3.000   1st Qu.:9.467e+08
##  Median :35768   Median : 1827   Median :4.000   Median :1.035e+09
```

```
##  Mean   :35870   Mean   : 4108   Mean   :3.512   Mean   :1.033e+09
##  3rd Qu.:53621   3rd Qu.: 3624   3rd Qu.:4.000   3rd Qu.:1.127e+09
##  Max.   :71567   Max.   :65133   Max.   :5.000   Max.   :1.231e+09
##     title               genres
##  Length:999999      Length:999999
##  Class :character   Class :character
##  Mode  :character   Mode  :character
##
##
##
```

The "edx" dataset includes close to 70,000 unique users and over 10,000 unique movies:

n_users	n_movies
69878	10677

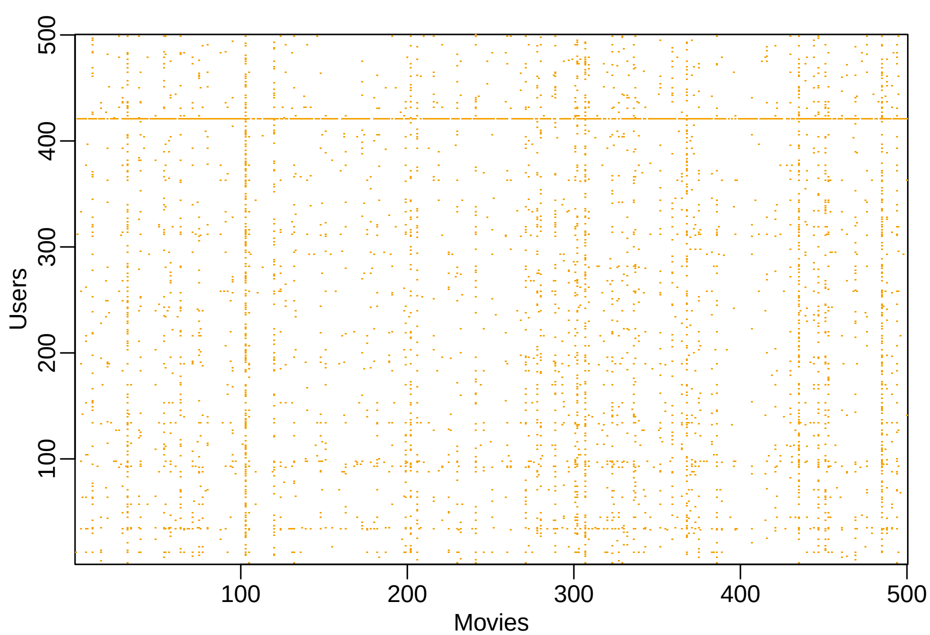
The data sets are pretty clean since analysis shows no missing value in "edx",

```
##    userId   movieId    rating timestamp     title    genres
##         0         0         0         0         0         0
```

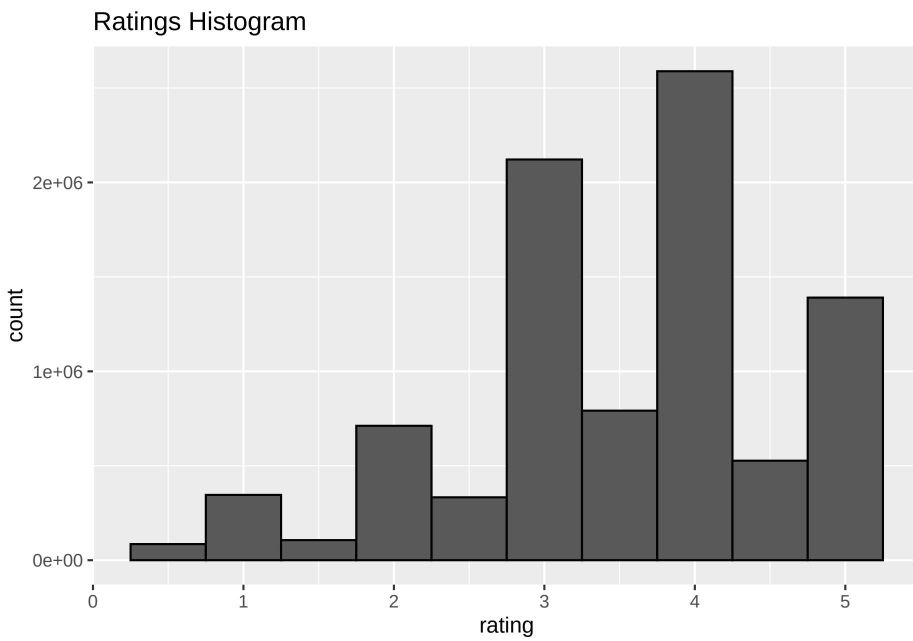
And no missing value in "validation" as well. Therefore no action is needed on treating missing values.

```
##    userId   movieId    rating timestamp     title    genres
##         0         0         0         0         0         0
```

Imagine if we set row as each unique users, column as each unique movie. Then the whole question/target becomes to fill in the blanks in this matrix. Below visualization shows how sparse this matrix is with 500 users and 500 movies.



The movie ratings are between 0.5 and 5.0 with 0.5 increments. As you can see below, in general, half star ratings are less common than whole star ratings.

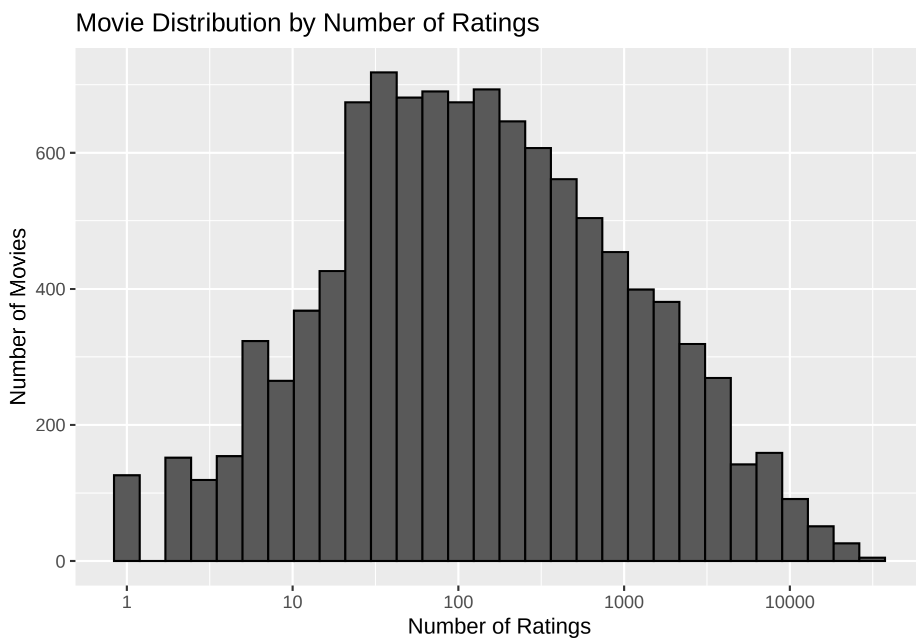


Here is the same information but in table view.

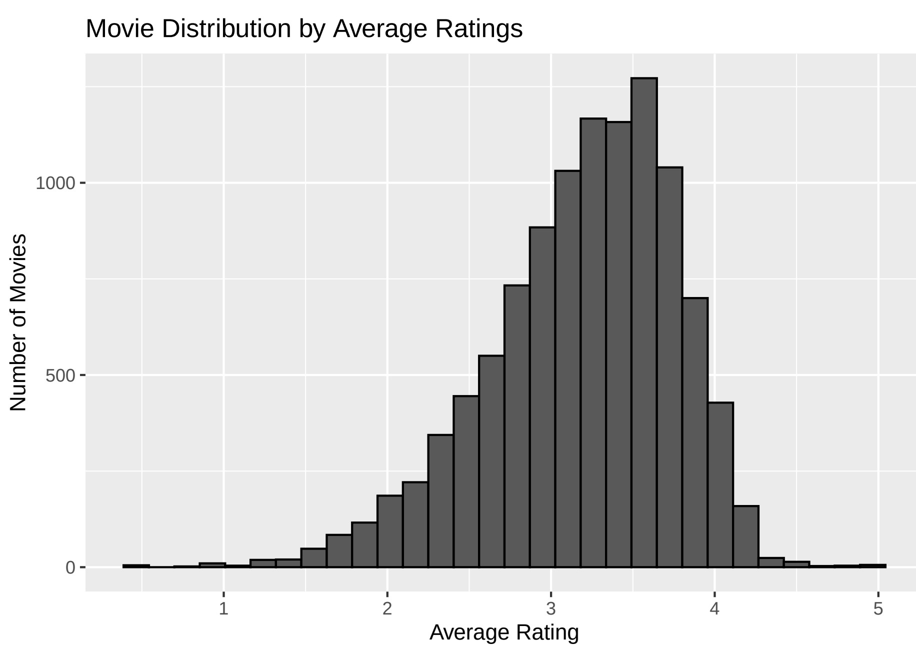
rating	n
4.0	2588430
3.0	2121240
5.0	1390114
3.5	791624
2.0	711422
4.5	526736
1.0	345679
2.5	333010
1.5	106426
0.5	85374

2.2 Movie Effects

Let's take a look at some distributions relevant to the movies. Below shows Movie Distribution by Number of Ratings. We can see some movies get rated more than others.



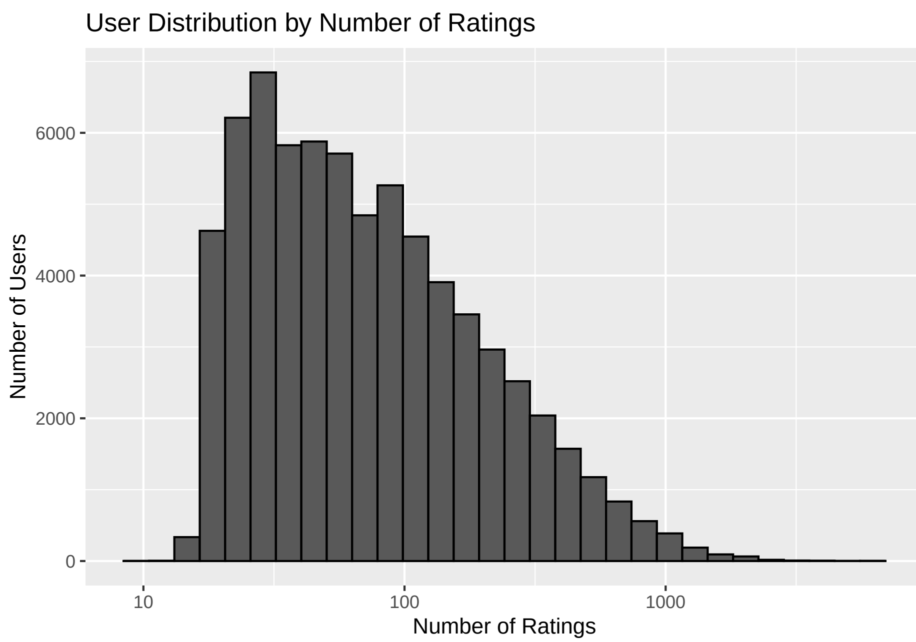
Below shows Movie Distribution by Average Ratings. We can see some movies get rated higher than others.



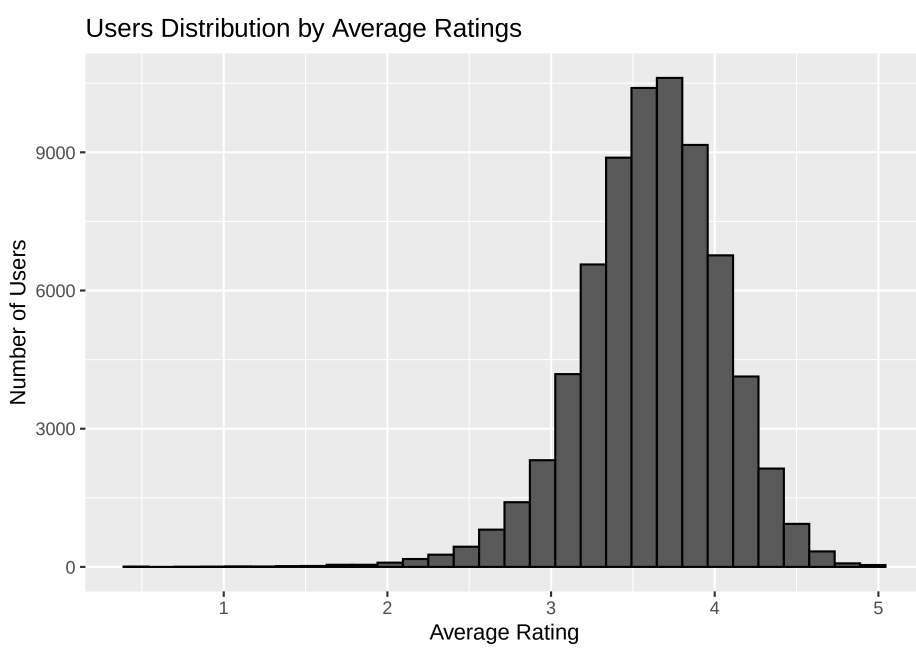
In summary, above indicates in a linear model, we should consider movie effects.

2.3 User Effects

Let's now take a look at distributions relevant to the users. Below shows User Distribution by Number of Ratings. It clearly shows some users are more active than others at rating movies.



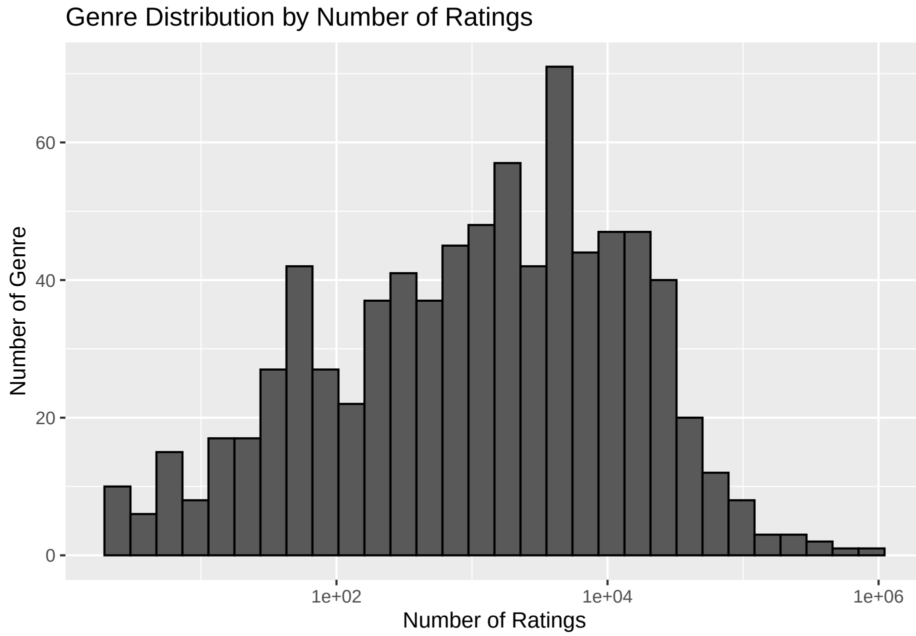
Below shows Users Distribution by Average Ratings. It clearly show some users give higher ratings than other users.



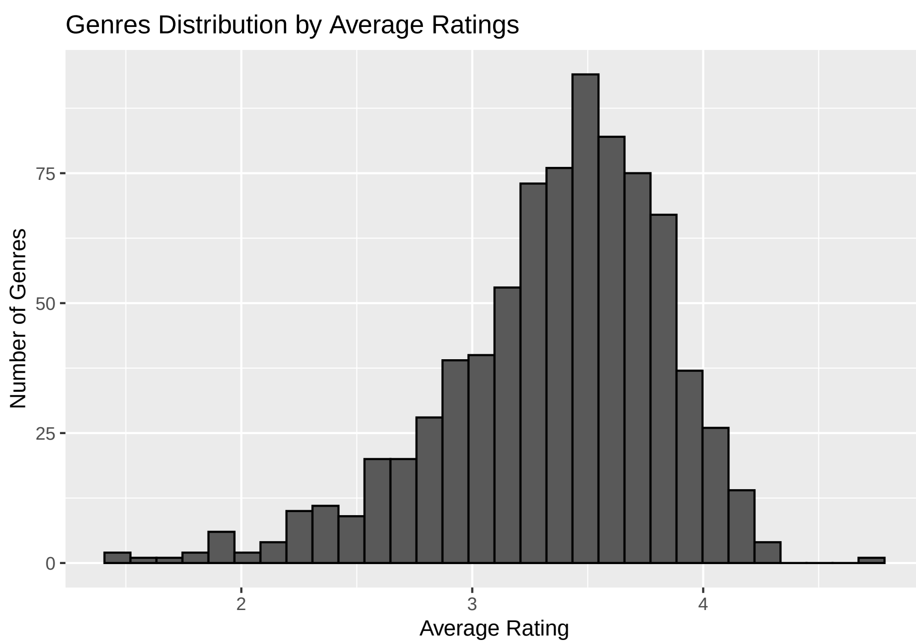
In summary, above indicates in a linear model, we should consider user effects as well.

2.4 Genre Effects

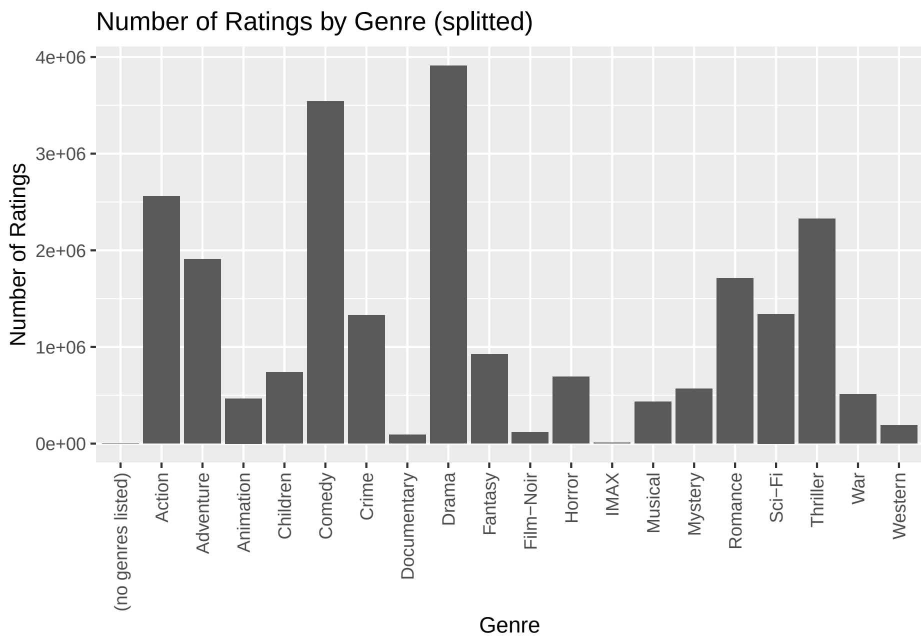
Now let's do similar visualizations on the distributions relevant to the genres. Below shows Genre Distribution by Number of Ratings. Some genres receive more ratings than others.



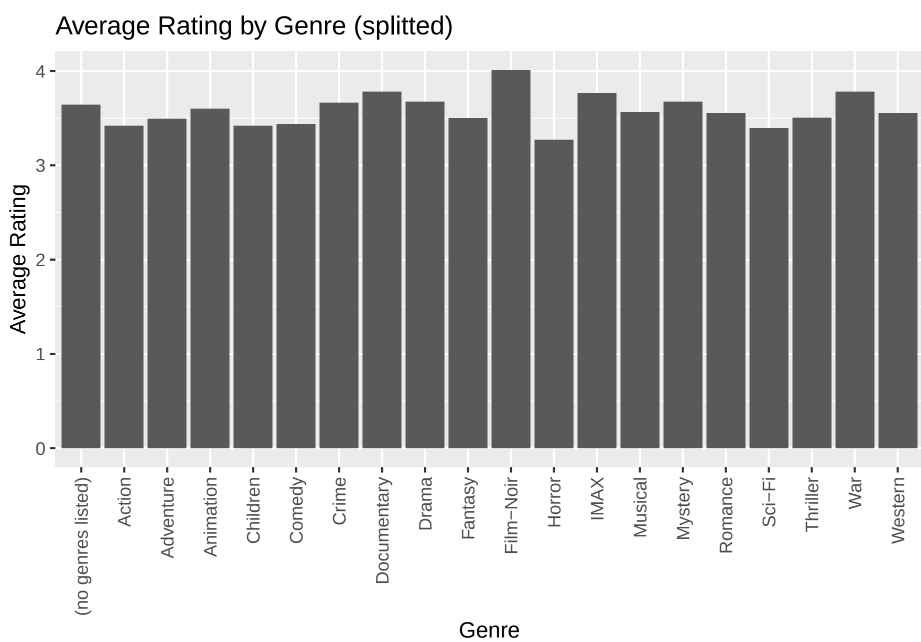
Below shows Genre Distribution by Average Ratings. Some genres receive higher ratings than others.



In summary, above indicates in a linear model, we should consider genre effects as well.
But do we need to split the genre column into individual genres? Let's split the genre column and look at same distributions analysis.
Below shows Number of Ratings by Genre (splitted).



Below shows Average Rating by Genre (splitted) .



Above reinforced the idea that in a linear model, we should definitely consider genre effects.
Also, due to less variability on Average Rating by the individual genres, I decided to use the original combined genre column without splitting it.

2.5 Data Cleaning

So far our data exploration covered user, movie, and genres. In order to further explore the effects of rating timestamps, we need to clean the data.
Below code is to convert timestamp to Date type, extract movie release year from movie title, and calculate the year between movie release and rating time stamp.

```
edx <- edx %>%
    mutate(title_temp = str_trim(title), date = as.Date(as_datetime(timestamp))) %>%
    # year_after_release: the year between movie
    # release and rating time stamp
extract(title_temp, c("title_temp", "releaseyear"), regex = "^(.*) \\(([0-9 \\-]*)\\)$",
    remove = FALSE) %>%
    mutate(releaseyear = if_else(str_length(releaseyear) >
        4, as.integer(str_split(releaseyear, "-", simplify = TRUE)[1]),
        as.integer(releaseyear))) %>%
    mutate(year_after_release = isoyear(date) - releaseyear) %>%
    select(-title_temp, -timestamp)
head(edx) %>%
    kable() %>%
    kable_styling(position = "center", latex_options = c("hold_position",
        "scale_down"))
```

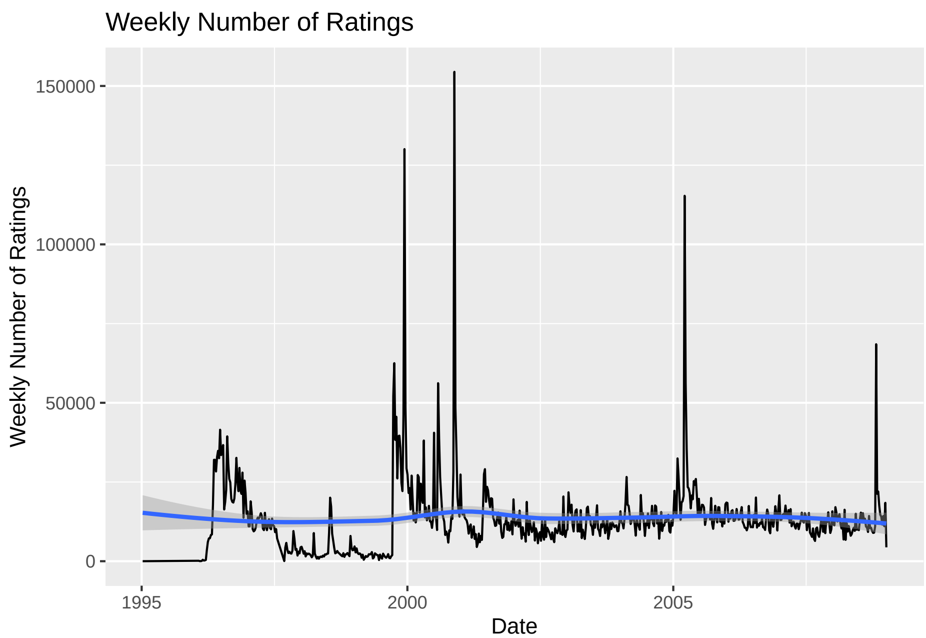
userId	movieId	rating	title	genres	date	releaseyear	year_after_release
1	122	5	Boomerang (1992)	Comedy|Romance	1996-08-02	1992	4
1	185	5	Net, The (1995)	Action|Crime|Thriller	1996-08-02	1995	1
1	292	5	Outbreak (1995)	Action|Drama|Sci-Fi|Thriller	1996-08-02	1995	1
1	316	5	Stargate (1994)	Action|Adventure|Sci-Fi	1996-08-02	1994	2
1	329	5	Star Trek: Generations (1994)	Action|Adventure|Drama|Sci-Fi	1996-08-02	1994	2
1	355	5	Flintstones, The (1994)	Children|Comedy|Fantasy	1996-08-02	1994	2

```
# do the same to validation dataset (I'm not
# adding/deleting any rows.)
validation <- validation %>%
    mutate(title_temp = str_trim(title), date = as.Date(as_datetime(timestamp))) %>%
    extract(title_temp, c("title_temp", "releaseyear"),
        regex = "^(.*) \\(([0-9 \\-]*)\\)$", remove = FALSE) %>%
    mutate(releaseyear = if_else(str_length(releaseyear) >
        4, as.integer(str_split(releaseyear, "-", simplify = TRUE)[1]),
        as.integer(releaseyear))) %>%
    mutate(year_after_release = isoyear(date) - releaseyear) %>%
    select(-title_temp, -timestamp)
head(validation) %>%
    kable() %>%
    kable_styling(position = "center", latex_options = c("hold_position",
        "scale_down"))
```

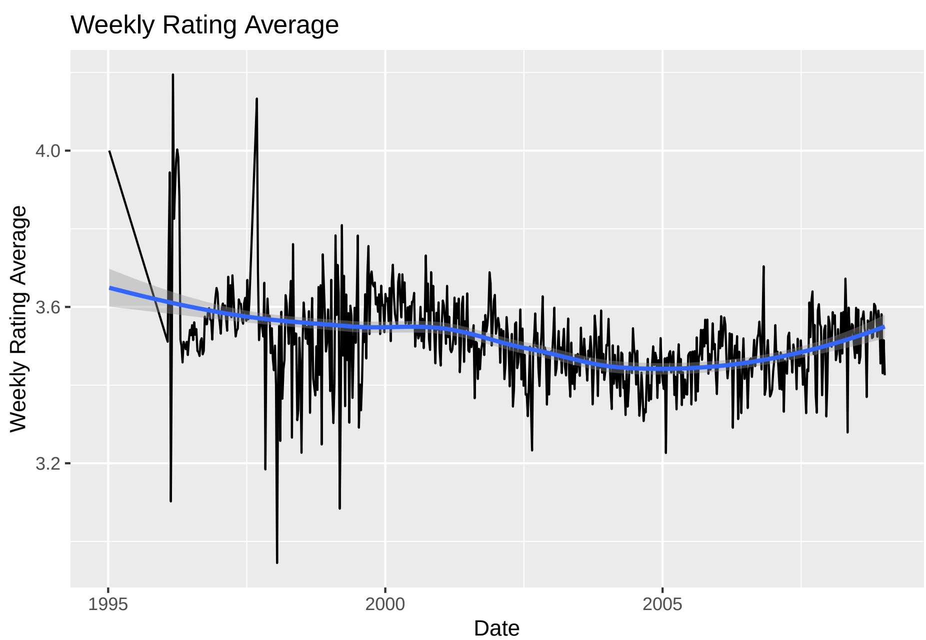
userId	movieId	rating	title	genres	date	releaseyear	year_after_release
1	231	5	Dumb & Dumber (1994)	Comedy	1996-08-02	1994	2
1	480	5	Jurassic Park (1993)	Action|Adventure|Sci-Fi|Thriller	1996-08-02	1993	3
1	586	5	Home Alone (1990)	Children|Comedy	1996-08-02	1990	6
2	151	3	Rob Roy (1995)	Action|Drama|Romance|War	1997-07-07	1995	2
2	858	2	Godfather, The (1972)	Crime|Drama	1997-07-07	1972	25
2	1544	3	Lost World: Jurassic Park, The (Jurassic Park 2) (1997)	Action|Adventure|Horror|Sci-Fi|Thriller	1997-07-07	1997	0

2.6 Rate Time Effects

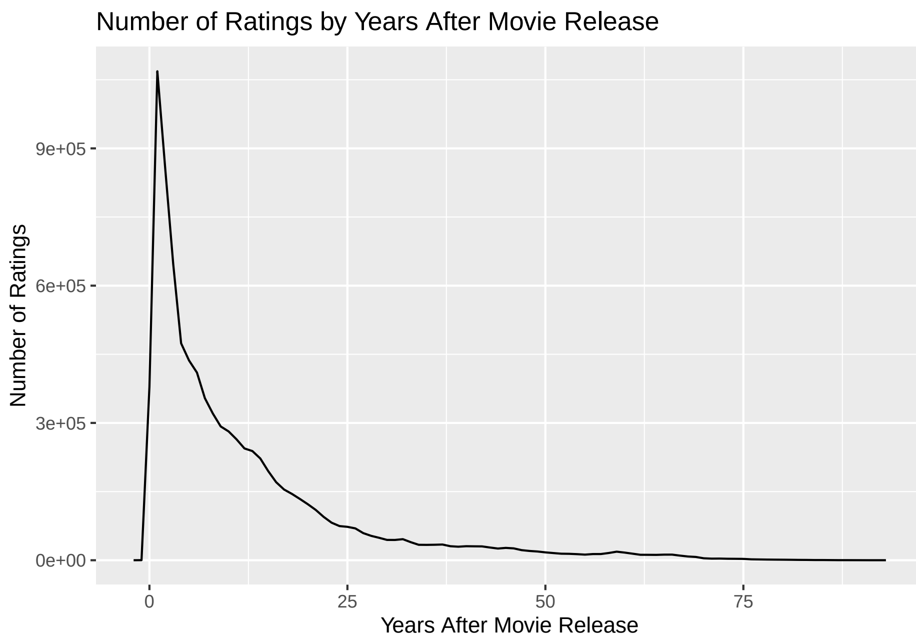
Now we can take a look at possible Rate Time Effects.
Let's first look at Weekly Number of Ratings. It turns out there is not much insight.



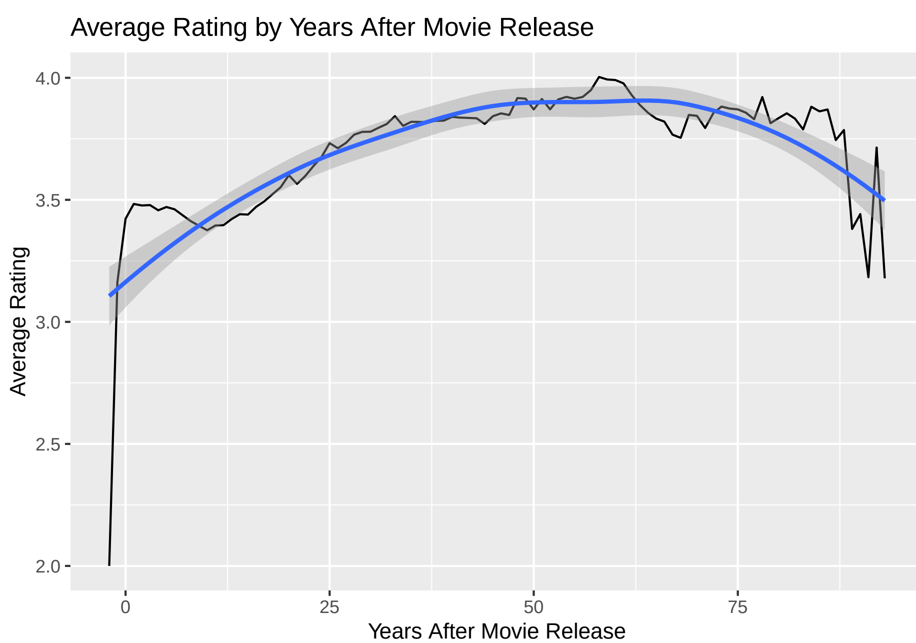
We then look at Weekly Rating Average. Again, not much insight.



Instead of using rating time, let's look at the rating time in comparison to the movie release year - "Years After Movie Release".
Below we look at Number of Ratings by Years After Movie Release. As you can see below, the number of ratings for a new released movie topped in the first 5 years, and then reduce over the time.



Let's also take a look at Average Rating by Years After Movie Release.



Since the online movie rating website came with Web 2.0 and is something less than 30 years old, I'm focusing on the trend within 30 years after movie release. During that period of time, the average rating tends to go up as time passes. Therefore we should consider rating time effects.

3 Model Building

I'm using the code below to split data into training and testing sets in order to prepare for model building.

```
# Split data sets and prepare for model building
# set.seed(755) # if using R 3.5 or earlier
set.seed(755, sample.kind = "Rounding")  # if using R 3.6 or later
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 movies, users, genres,
# year_after_release 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") %>%
    semi_join(train_set, by = "genres") %>%
    semi_join(train_set, by = "year_after_release")
# Add rows removed from test_set back into train_set
test_set_removed <- anti_join(edx[test_index, ], test_set)
train_set <- rbind(train_set, test_set_removed)
```

We will use RMSE as the loss function, here we write a custom function for that:

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

3.1 Model#1: Just the average

This is the simplest model we start with. In this model, we predict all movies with just the mean. The formula is

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

where μ is one "true" rating for all movies. ϵ is independent errors sampled from the same distribution centered at zero, i is movie, u is user.

```
mu_hat <- mean(train_set$rating)
model_1_rmse <- RMSE(test_set$rating, mu_hat)
# create a results table.
rmse_results <- tibble(method = "Just the average", RMSE = model_1_rmse)
rmse_results %>% knitr::kable()
```

method	RMSE
Just the average	1.060561

The RMSE result shows our typical error is larger than one star, which is not good prediction.

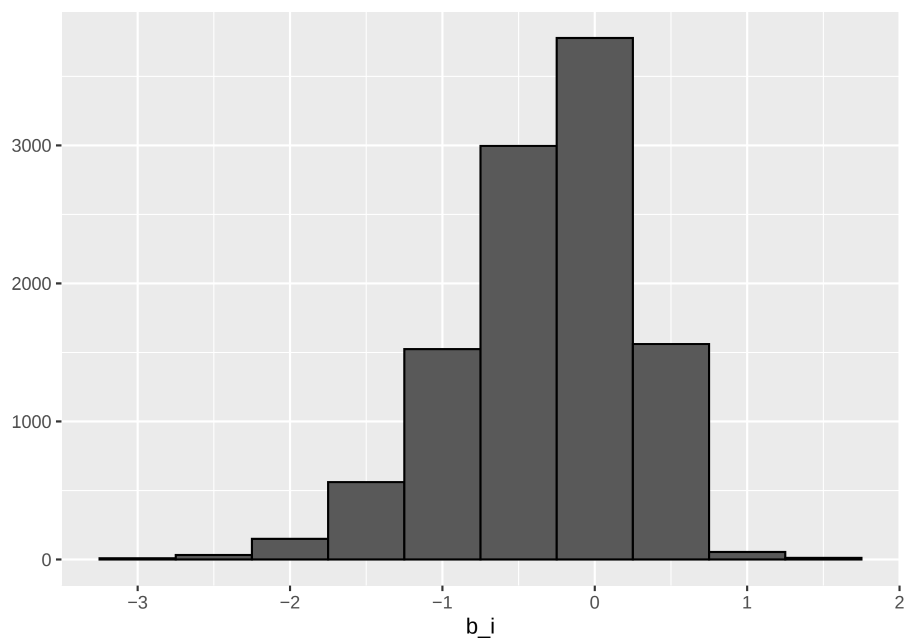
3.2 Model#2.1: Movie Effect Model (b_i)

We then take into account the Movie Effects. The formula is

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

where b_i is average rating effect of the movie i.
By plotting a chart, we see it proves each movie's b_i (bias) varies substantially.

```r
# Since lm() is too slow and probably crash your
# laptop as complexity grows, we will calculate using
# code below.
mu <- mean(train_set$rating)
movie_avgs <- train_set %>%
    group_by(movieId) %>%
    summarize(b_i = mean(rating - mu))
# by plotting a chart, we see it proves each movie's
# b_i (bias) varies substantially
movie_avgs %>%
    qplot(b_i, geom = "histogram", bins = 10, data = .,
        color = I("black"))
```




```r
# Add the result to the results table
predicted_ratings <- mu + test_set %>%
    left_join(movie_avgs, by = "movieId") %>%
    pull(b_i)
model_2_1_rmse <- RMSE(predicted_ratings, test_set$rating)
rmse_results <- bind_rows(rmse_results, tibble(method = "Movie Effect Model",
    RMSE = model_2_1_rmse))
rmse_results %>%
    knitr::kable()
```

method	RMSE
Just the average	1.0605613
Movie Effect Model	0.9439868

The RMSE result shows improvements compared to "Just the average" model.

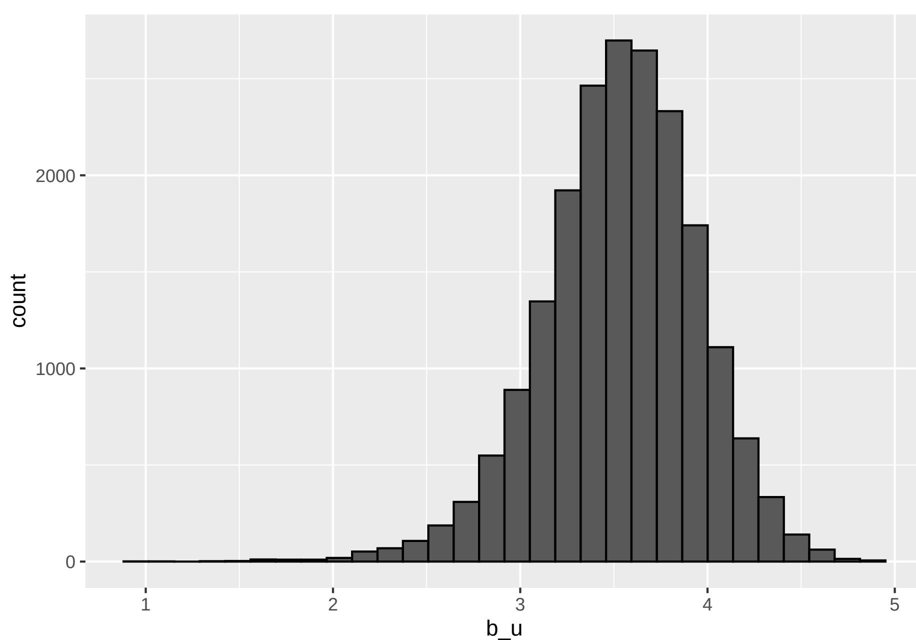
3.3 Model#2.2: Movie + User Effects Model (b_i + b_u)

Let's add User Effects as well. The formula is

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

where b_u is average rating effect of the user u.
By plotting a chart of the average rating for user u for those that have rated 100 or more movies. It proves that there is substantial variability across users as well.

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




```
user_avgs <- train_set %>%
    left_join(movie_avgs, by = "movieId") %>%
    group_by(userId) %>%
    summarize(b_u = mean(rating - mu - b_i))
# Add the result to the results table
predicted_ratings <- test_set %>%
    left_join(movie_avgs, by = "movieId") %>%
    left_join(user_avgs, by = "userId") %>%
    mutate(pred = mu + b_i + b_u) %>%
    pull(pred)
```

```r
model_2_2_rmse <- RMSE(predicted_ratings, test_set$rating)
rmse_results <- bind_rows(rmse_results, data_frame(method = "Movie + User Effects Model",
    RMSE = model_2_2_rmse))
rmse_results %>%
    knitr::kable()
```

method	RMSE
Just the average	1.0605613
Movie Effect Model	0.9439868
Movie + User Effects Model	0.8666408

The RMSE result shows improvements compared to "Movie Effect Model" model.

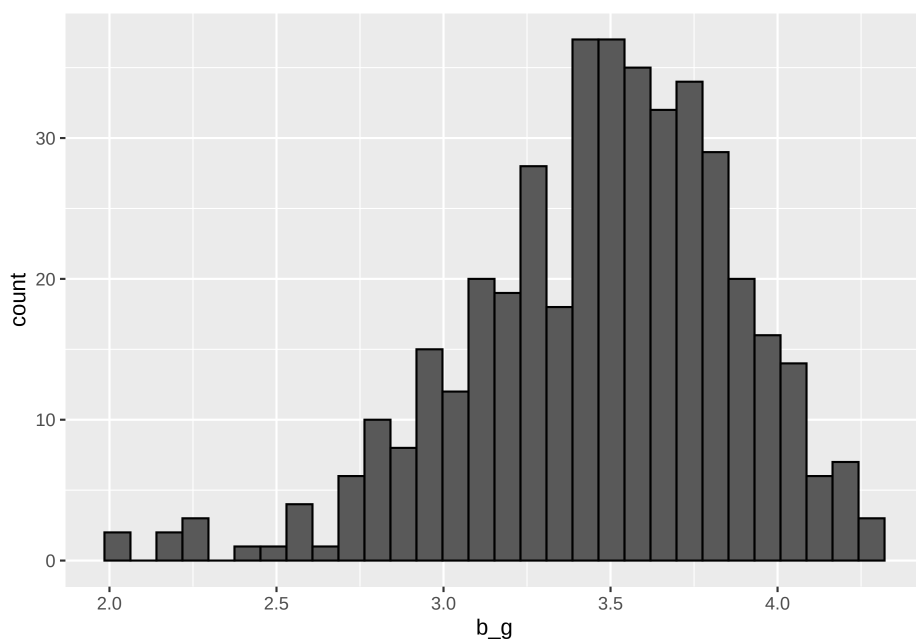
3.4 Model#2.3: Movie + User + Genre Effects Model (b_i + b_u + b_g)

Next, we are adding Genre Effects. The formula is

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

where b_g is average rating effect of the genre g.
By plotting a chart of the average rating for genre g for those that have 1000 or more movies. It proves that there is substantial variability across genres as well.

```r
train_set %>%
    group_by(genres) %>%
    filter(n() >= 1000) %>%
    summarize(b_g = mean(rating)) %>%
    ggplot(aes(b_g)) + geom_histogram(bins = 30, color = "black")
```



```r
genre_avgs <- train_set %>%
    left_join(movie_avgs, by = "movieId") %>%
    left_join(user_avgs, by = "userId") %>%
    group_by(genres) %>%
    summarize(b_g = mean(rating - mu - b_i - b_u))
# Add the result to the results table
predicted_ratings <- test_set %>%
    left_join(movie_avgs, by = "movieId") %>%
    left_join(user_avgs, by = "userId") %>%
    left_join(genre_avgs, by = "genres") %>%
    mutate(pred = mu + b_i + b_u + b_g) %>%
    pull(pred)
model_2_3_rmse <- RMSE(predicted_ratings, test_set$rating)
rmse_results <- bind_rows(rmse_results, data_frame(method = "Movie + User + Genre Effects Model",
    RMSE = model_2_3_rmse))
rmse_results %>%
    knitr::kable()
```

method	RMSE
Just the average	1.0605613
Movie Effect Model	0.9439868
Movie + User Effects Model	0.8666408
Movie + User + Genre Effects Model	0.8662908

The RMSE result shows improvements compared to "Movie + User Effects Model" model.

3.5 Model#2.4: Movie + User + Genre + Rate Time Effects Model (b_i + b_u + b_g + b_t)

Let's add the last effects to our model - Rate Time Effects. The formula is

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

where b_t is average rating effect of the rate time t.

```r
rate_time_avgs <- train_set %>%
    left_join(movie_avgs, by = "movieId") %>%
    left_join(user_avgs, by = "userId") %>%
    left_join(genre_avgs, by = "genres") %>%
    group_by(year_after_release) %>%
    summarize(b_t = mean(rating - mu - b_i - b_u - b_g))
# Add the result to the results table
predicted_ratings <- test_set %>%
    left_join(movie_avgs, by = "movieId") %>%
    left_join(user_avgs, by = "userId") %>%
    left_join(genre_avgs, by = "genres") %>%
    left_join(rate_time_avgs, by = "year_after_release") %>%
    mutate(pred = mu + b_i + b_u + b_g + b_t) %>%
    pull(pred)
model_2_4_rmse <- RMSE(predicted_ratings, test_set$rating)
rmse_results <- bind_rows(rmse_results, data_frame(method = "Movie + User + Genre + Rate Time Effects Mo
    RMSE = model_2_4_rmse))
rmse_results %>%
    knitr::kable()
```

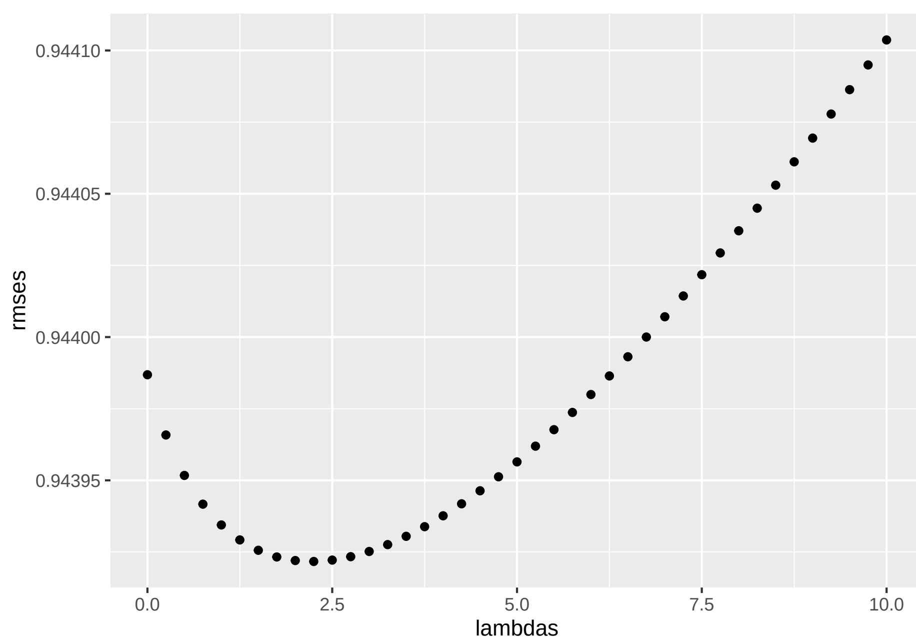
method	RMSE
Just the average	1.0605613
Movie Effect Model	0.9439868
Movie + User Effects Model	0.8666408
Movie + User + Genre Effects Model	0.8662908
Movie + User + Genre + Rate Time Effects Model	0.8658614

The RMSE result shows best performance so far compared to all previous models.

3.6 Model#3.1: Regularized Movie Effect Model

Regularization permits us to penalize large estimates that are formed using small sample sizes. This should help us further improve the model.
Let's try it on "Movie Effect Model".

```
lambdas <- seq(0, 10, 0.25)
just_the_sum <- train_set %>%
    group_by(movieId) %>%
    summarize(s = sum(rating - mu), n_i = n())
rmses <- sapply(lambdas, function(l) {
    predicted_ratings <- test_set %>%
        left_join(just_the_sum, by = "movieId") %>%
        mutate(b_i = s/(n_i + l)) %>%
        mutate(pred = mu + b_i) %>%
        pull(pred)
    return(RMSE(predicted_ratings, test_set$rating))
})
# lambda_b_i=2.25
qplot(lambdas, rmses)
```



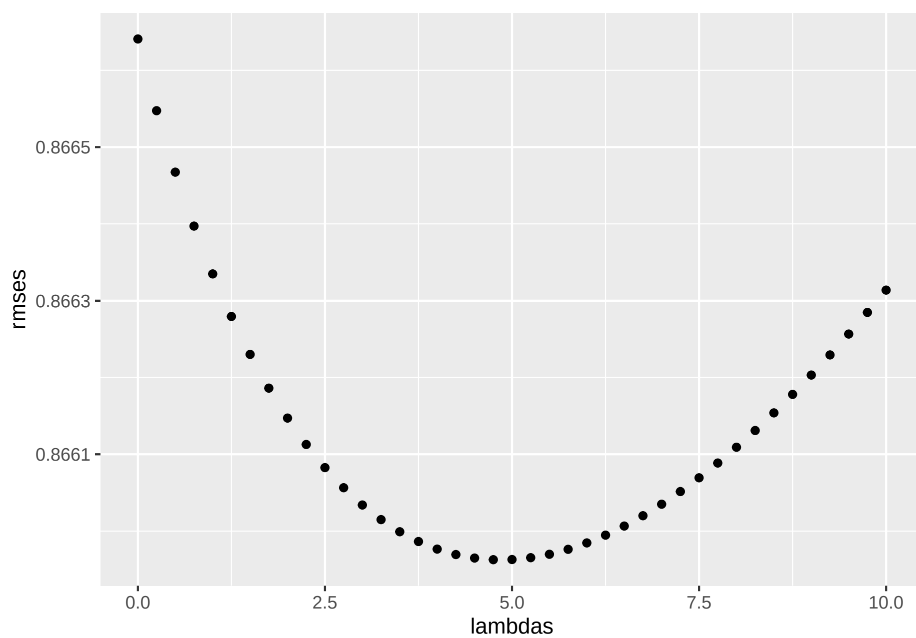

```
lambda_b_i <- lambdas[which.min(rmses)]
model_3_1_rmse <- min(rmses)
rmse_results <- bind_rows(rmse_results, data_frame(method = "Regularized Movie Effect Model",
    RMSE = model_3_1_rmse))
rmse_results %>%
    knitr::kable()
```

method	RMSE
Just the average	1.0605613
Movie Effect Model	0.9439868
Movie + User Effects Model	0.8666408
Movie + User + Genre Effects Model	0.8662908
Movie + User + Genre + Rate Time Effects Model	0.8658614
Regularized Movie Effect Model	0.9439217

The RMSE result did show improvements compared to the original “Movie Effects Model” model.

3.7 Model#3.2: Regularized Movie + User Effect Model

```
lambdas <- seq(0, 10, 0.25)
rmses <- sapply(lambdas, function(l) {
    mu <- mean(train_set$rating)
    b_i <- train_set %>%
        group_by(movieId) %>%
        summarize(b_i = sum(rating - mu)/(n() + l))
    b_u <- train_set %>%
        left_join(b_i, by = "movieId") %>%
        group_by(userId) %>%
        summarize(b_u = sum(rating - mu - b_i)/(n() + l))
    predicted_ratings <- test_set %>%
        left_join(b_i, by = "movieId") %>%
        left_join(b_u, by = "userId") %>%
        mutate(pred = mu + b_i + b_u) %>%
        pull(pred)
    return(RMSE(predicted_ratings, test_set$rating))
})
# lambda_b_u=4.75
qplot(lambdas, rmses)
```



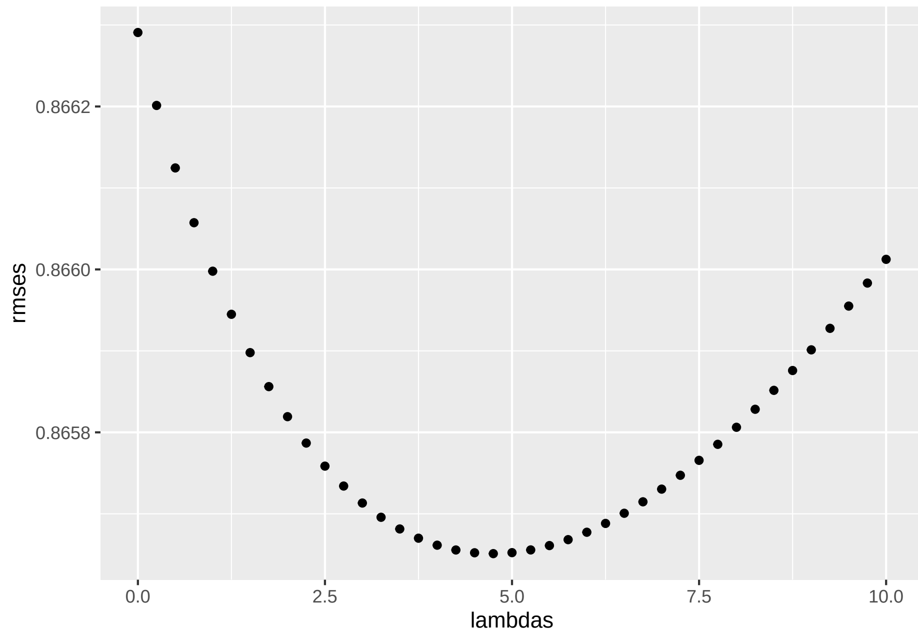
```r
lambda_b_u <- lambdas[which.min(rmses)]
model_3_2_rmse <- min(rmses)
rmse_results <- bind_rows(rmse_results, data_frame(method = "Regularized Movie + User Effect Model",
    RMSE = model_3_2_rmse))
rmse_results %>%
    knitr::kable()
```

method	RMSE
Just the average	1.0605613
Movie Effect Model	0.9439868
Movie + User Effects Model	0.8666408
Movie + User + Genre Effects Model	0.8662908
Movie + User + Genre + Rate Time Effects Model	0.8658614
Regularized Movie Effect Model	0.9439217
Regularized Movie + User Effect Model	0.8659628

The RMSE result did show improvements compared to the original "Movie + User Effects Model" model.

3.8 Model#3.3: Regularized Movie + User + Genre Effect Model

```r
lambdas <- seq(0, 10, 0.25)
rmses <- sapply(lambdas, function(l) {
    mu <- mean(train_set$rating)
    b_i <- train_set %>%
        group_by(movieId) %>%
        summarize(b_i = sum(rating - mu)/(n() + l))
    b_u <- train_set %>%
        left_join(b_i, by = "movieId") %>%
        group_by(userId) %>%
        summarize(b_u = sum(rating - mu - b_i)/(n() + l))
    b_g <- train_set %>%
        left_join(b_i, by = "movieId") %>%
        left_join(b_u, by = "userId") %>%
        group_by(genres) %>%
        summarize(b_g = sum(rating - mu - b_i - b_u)/(n() +
            l))
    predicted_ratings <- test_set %>%
        left_join(b_i, by = "movieId") %>%
        left_join(b_u, by = "userId") %>%
        left_join(b_g, by = "genres") %>%
        mutate(pred = mu + b_i + b_u + b_g) %>%
        pull(pred)
    return(RMSE(predicted_ratings, test_set$rating))
})
# lambda_b_g=4.75
qplot(lambdas, rmses)
```




```
lambda_b_g <- lambdas[which.min(rmses)]
model_3_3_rmse <- min(rmses)
rmse_results <- bind_rows(rmse_results, data_frame(method = "Regularized Movie + User + Genre Effect Mo
    RMSE = model_3_3_rmse))
rmse_results %>%
    knitr::kable()
```

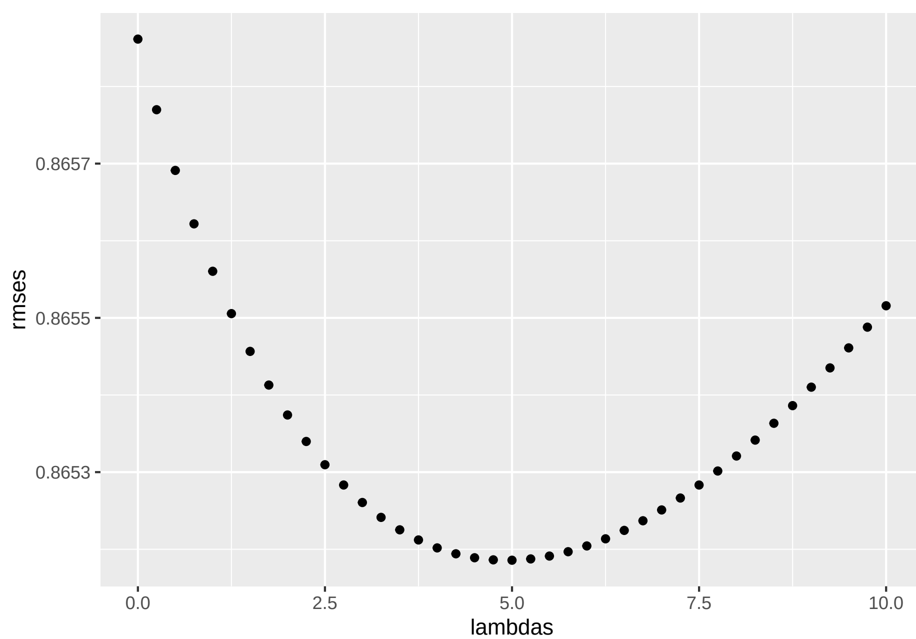
method	RMSE
Just the average	1.0605613
Movie Effect Model	0.9439868
Movie + User Effects Model	0.8666408
Movie + User + Genre Effects Model	0.8662908
Movie + User + Genre + Rate Time Effects Model	0.8658614
Regularized Movie Effect Model	0.9439217
Regularized Movie + User Effect Model	0.8659628
Regularized Movie + User + Genre Effect Model	0.8656511

The RMSE result did show improvements compared to the original "Movie + User + Genre Effects Model" model.

3.9 Model#3.4: Regularized Movie + User + Genre + Rate Time Effect Model

```
lambdas <- seq(0, 10, 0.25)
rmses <- sapply(lambdas, function(l) {
    mu <- mean(train_set$rating)
    b_i <- train_set %>%
        group_by(movieId) %>%
        summarize(b_i = sum(rating - mu)/(n() + l))
    b_u <- train_set %>%
        left_join(b_i, by = "movieId") %>%
```

```
        group_by(userId) %>%
        summarize(b_u = sum(rating - mu - b_i)/(n() + l))
    b_g <- train_set %>%
        left_join(b_i, by = "movieId") %>%
        left_join(b_u, by = "userId") %>%
        group_by(genres) %>%
        summarize(b_g = sum(rating - mu - b_i - b_u)/(n() +
            l))
    b_t <- train_set %>%
        left_join(b_i, by = "movieId") %>%
        left_join(b_u, by = "userId") %>%
        left_join(b_g, by = "genres") %>%
        group_by(year_after_release) %>%
        summarize(b_t = sum(rating - mu - b_i - b_u - b_g)/(n() +
            l))
    predicted_ratings <- test_set %>%
        left_join(b_i, by = "movieId") %>%
        left_join(b_u, by = "userId") %>%
        left_join(b_g, by = "genres") %>%
        left_join(b_t, by = "year_after_release") %>%
        mutate(pred = mu + b_i + b_u + b_g + b_t) %>%
        pull(pred)
    return(RMSE(predicted_ratings, test_set$rating))
})
# lambda_b_t=5
qplot(lambdas, rmses)
```




```
lambda_b_t <- lambdas[which.min(rmses)]
model_3_4_rmse <- min(rmses)
rmse_results <- bind_rows(rmse_results, data_frame(method = "Regularized Movie + User + Genre + Rate Ti
    RMSE = model_3_4_rmse))
```

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

method	RMSE
Just the average	1.0605613
Movie Effect Model	0.9439868
Movie + User Effects Model	0.8666408
Movie + User + Genre Effects Model	0.8662908
Movie + User + Genre + Rate Time Effects Model	0.8658614
Regularized Movie Effect Model	0.9439217
Regularized Movie + User Effect Model	0.8659628
Regularized Movie + User + Genre Effect Model	0.8656511
Regularized Movie + User + Genre + Rate Time Effect Model	0.8651858

The RMSE result is so far the best, even better than the original "Movie + User + Genre + Rate Time Effects Model" model.

3.10 Final Model

Based on the lowest RMSE, we are choosing "Regularized Movie + User + Genre + Rate Time Effect Model" as our final model.

4 Results

Let's take a quick recap on all the lambda parameters we picked:

lambda_b_i	lambda_b_u	lambda_b_g	lambda_b_t
2.25	4.75	4.75	5

The final model is developed based from the "edx" data set. The validation set is not used at all.

```r
# use the final model developed based on edx data set
mu <- mean(train_set$rating)
movie_reg_avgs <- train_set %>%
    group_by(movieId) %>%
    summarize(b_i = sum(rating - mu)/(n() + lambda_b_i))
user_reg_avgs <- train_set %>%
    left_join(movie_reg_avgs, by = "movieId") %>%
    group_by(userId) %>%
    summarize(b_u = sum(rating - b_i - mu)/(n() + lambda_b_u))
genre_reg_avgs <- train_set %>%
    left_join(movie_reg_avgs, by = "movieId") %>%
    left_join(user_reg_avgs, by = "userId") %>%
    group_by(genres) %>%
    summarize(b_g = sum(rating - b_i - mu - b_u)/(n() +
        lambda_b_g))
ratetime_reg_avgs <- train_set %>%
    left_join(movie_reg_avgs, by = "movieId") %>%
    left_join(user_reg_avgs, by = "userId") %>%
    left_join(genre_reg_avgs, by = "genres") %>%
    group_by(year_after_release) %>%
    summarize(b_t = sum(rating - b_i - mu - b_u - b_g)/(n() +
        lambda_b_t))
# implement the model on validation set, and see
# result RMSE=0.8648385, which has achieved ultimate
# target: RMSE < 0.86490
predicted_ratings <- validation %>%
    left_join(movie_reg_avgs, by = "movieId") %>%
    left_join(user_reg_avgs, by = "userId") %>%
    left_join(genre_reg_avgs, by = "genres") %>%
    left_join(ratetime_reg_avgs, by = "year_after_release") %>%
    mutate(pred = mu + b_i + b_u + b_g + b_t) %>%
    pull(pred)
model_final_rmse <- RMSE(predicted_ratings, validation$rating)
RMSE(predicted_ratings, validation$rating)
```

```
## [1] 0.8648385
```

The final RMSE we get by applying our final model on the "validation" dateset is 0.8648385. This proves that we have achieved ultimate target: RMSE < 0.86490.

5 Conclusion

After using only "edx" dataset to test different models, we ended up constructed the "Regularized Movie + User + Genre + Rate Time Effect Model". The final model takes into consideration the effects from movie, user, genre and rate time. Due to sparsity of the data, we added regularization to further improve the model performance.
After applying our final model on the "validation" dateset (previously unused), We have successfully achieved RMSE of 0.8648385. This is beyond the ultimate target of RMSE below 0.86490.

Even though we have reached the target set by this course, we need to realize there are limitations on this final model. One example is that it assumes movie, user, genre and rate time are all independent, which is most likely not the case in real world.

Looking forward, I would be interested to see more information regarding the users (such as location, age, gender, etc.) and the movies (such as box office sales, investment, director, actors, country, etc.) in order to improve further the model performance. Matrix factorization would be another method to consider as well.