Movie Recommendation Project

Andy Zhao

2023-03-20

Contents

1 Overview

The objective of the project is to compare rmse values and find a better model to recommend movies.

Rmse-root mean squared error formula is given: sq (sum (error)^2 / n), where error is the difference between predicted rating and true rating and n the number of datapoints.

The dataset contains around 10 Millions of datapoints partitioned into 9 Millions for training(edx) and 1 Million for testing(final_holdout_test).

Only use the average of rating(naïve mean baseline) to have a rmse value of 1.061202. When movie(movieId) and user(userId) effects are taken into account, the rmse value could be lowered to a good value 0.8653488. After regularization is applied, the value gets lowered further a tiny bit to 0.8648170. Which fulfills the project target RMSE < 0.86490.

2 Exploratory data analysis

2.1 edx dataset

First 5 rows of edx dataset

```
#load the kable function package
if(!require(kableExtra)) install.packages("kableExtra")
```

```
## Loading required package: kableExtra
```

```
##
## Attaching package: 'kableExtra'
```

```
## The following object is masked from 'package:dplyr':
##
##     group_rows
```

```
library(kableExtra)

#show the first 5 rows of the edx dataset
edx %>% head(5) %>%
  kable() %>%
  kable_styling(latex_options = "HOLD_position",bootstrap_options = c("striped", "hover", "condensed",
                position = "center",
                full_width = FALSE) %>% row_spec(c(1,2,3,4,5),color = "white" , background ="blue")
```

	userId	movieId	rating	timestamp	title	genres
1	1	122	5	838985046	Boomerang (1992)	Comedy|Romance
2	1	185	5	838983525	Net, The (1995)	Action|Crime|Thriller
4	1	292	5	838983421	Outbreak (1995)	Action|Drama|Sci-Fi|Thriller
5	1	316	5	838983392	Stargate (1994)	Action|Adventure|Sci-Fi
6	1	329	5	838983392	Star Trek: Generations (1994)	Action|Adventure|Drama|Sci-Fi

Missing values

```
#check the missing values in edx
sapply(edx, function(n) sum(is.na(n))) %>%
  kable(col.names = c("Missing values per variable")) %>%
  kable_styling(latex_options = "HOLD_position",bootstrap_options = c("striped", "hover", "condensed",
                position = "center",
                full_width = FALSE) %>% column_spec(1 ,color = "white" , background ="blue")
```

	Missing values per variable
userId	0
movieId	0
rating	0
timestamp	0
title	0
genres	0

2.2 Rating analysis

The summary of the rating column in edx shows: the mean is 3.512 and median is 4.

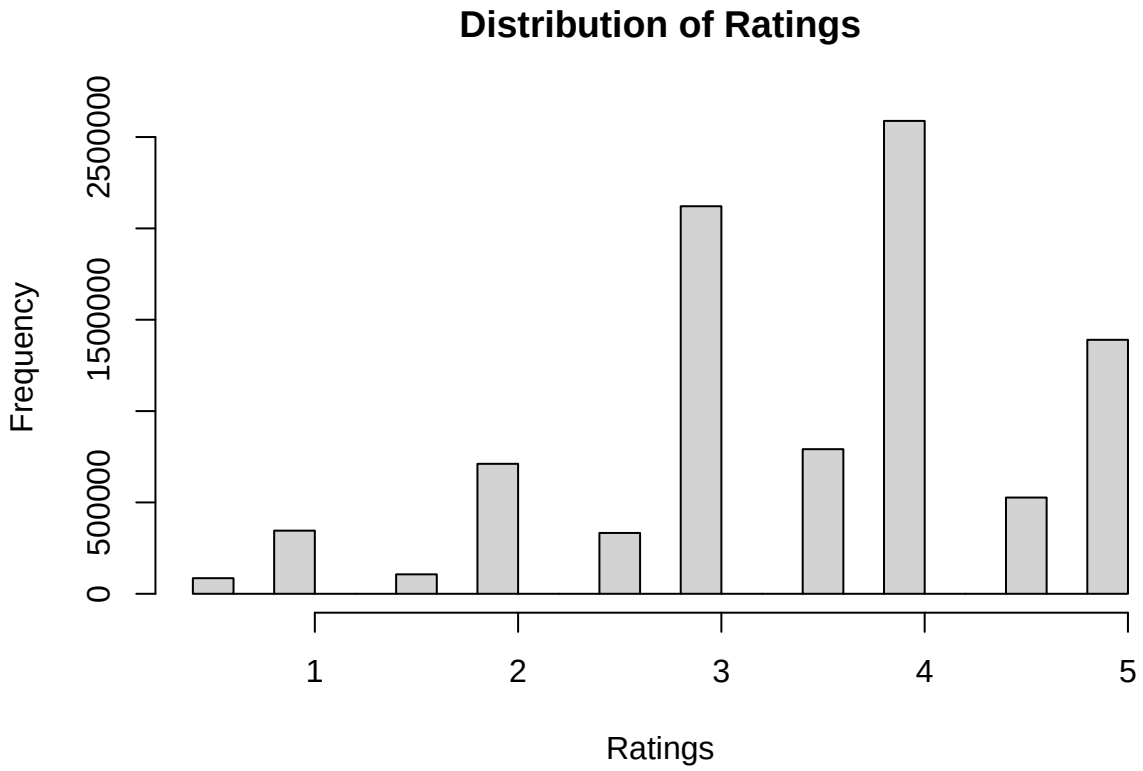
```
#summary of rating
summary(edx$rating)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   0.500   3.000   4.000   3.512   4.000   5.000
```

The histogram shows: there are more full stars than half-stars.

Overall frequency of ratings

```
#plot histogram of Distribution of Ratings
hist(edx$rating, main="Distribution of Ratings", xlab="Ratings")
```



2.3 MoiveId analysis

After visualizations, we notice some movies are rated tens of thousands of times, but some only rated once. The distributions show there might be a strong movie effect on decreasing the value of RSME.

Top 10 most rated movies

```
#top 10 most rated movies
edx %>% group_by(title) %>%
  summarize(count = n()) %>%
  arrange(desc(count)) %>% top_n(10,count) %>%
  kable() %>%
  kable_styling(latex_options = "HOLD_position",bootstrap_options = c("striped", "hover", "condensed",
                position = "center",
                full_width = FALSE) %>% row_spec(c(1,3,5,7,9),color = "white" , background ="blue")
```

title	count
Pulp Fiction (1994)	31362
Forrest Gump (1994)	31079
Silence of the Lambs, The (1991)	30382
Jurassic Park (1993)	29360
Shawshank Redemption, The (1994)	28015
Braveheart (1995)	26212
Fugitive, The (1993)	25998
Terminator 2: Judgment Day (1991)	25984
Star Wars: Episode IV - A New Hope (a.k.a. Star Wars) (1977)	25672
Apollo 13 (1995)	24284

Top 10 least rated movies

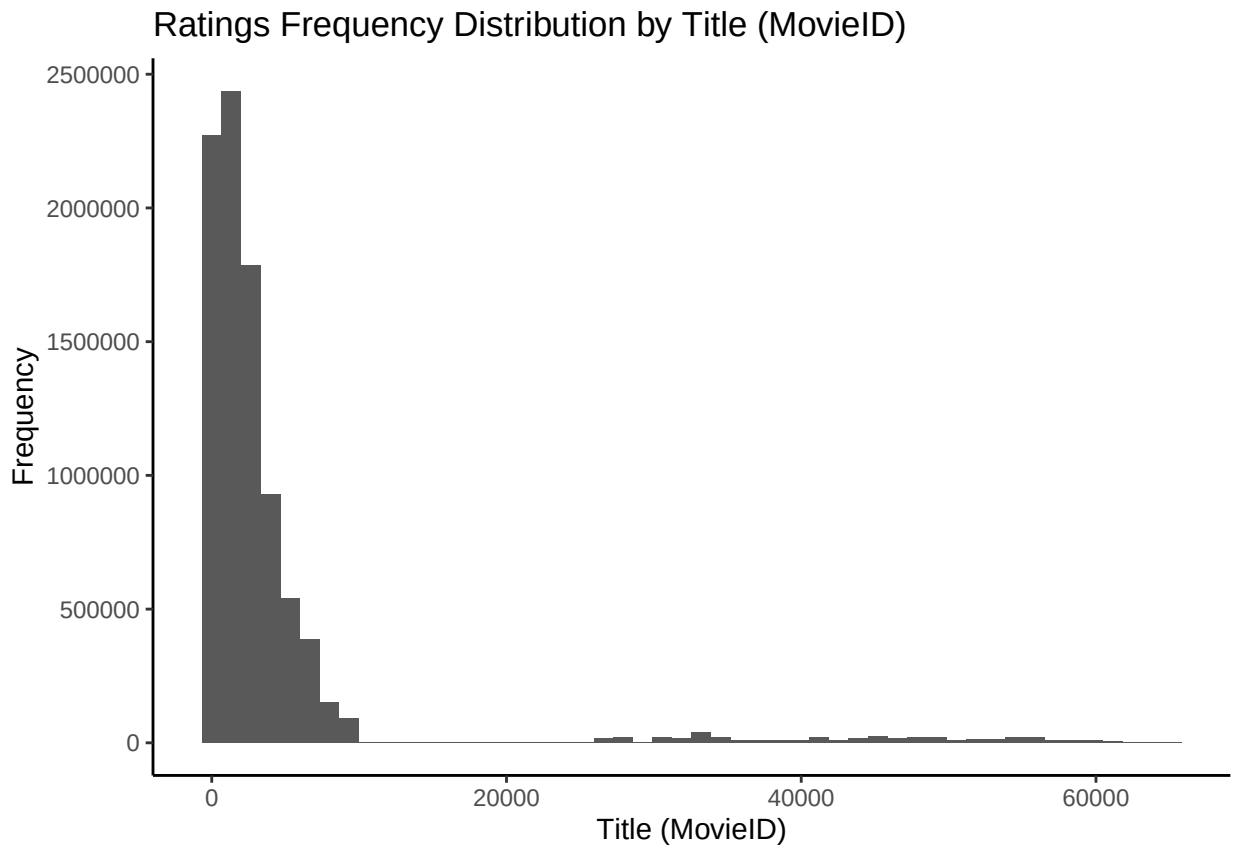
```
#top 10 least rated movies
edx %>% group_by(title) %>%
  summarize(count = n()) %>%
  arrange(count) %>% head(10) %>%
  kable() %>%
  kable_styling(latex_options = "HOLD_position",bootstrap_options = c("striped", "hover", "condensed",
                position = "center",
                full_width = FALSE) %>% row_spec(c(1,3,5,7,9),color = "white" , background ="red")
```

title	count
1, 2, 3, Sun (Un, deuz, trois, soleil) (1993)	1
100 Feet (2008)	1
4 (2005)	1
Accused (Anklaget) (2005)	1
Ace of Hearts (2008)	1
Ace of Hearts, The (1921)	1
Adios, Sabata (Indio Black, sai che ti dico: Sei un gran figlio di...) (1971)	1
Africa addio (1966)	1
Aleksandra (2007)	1
Bad Blood (Mauvais sang) (1986)	1

Frequency of ratings by movieId

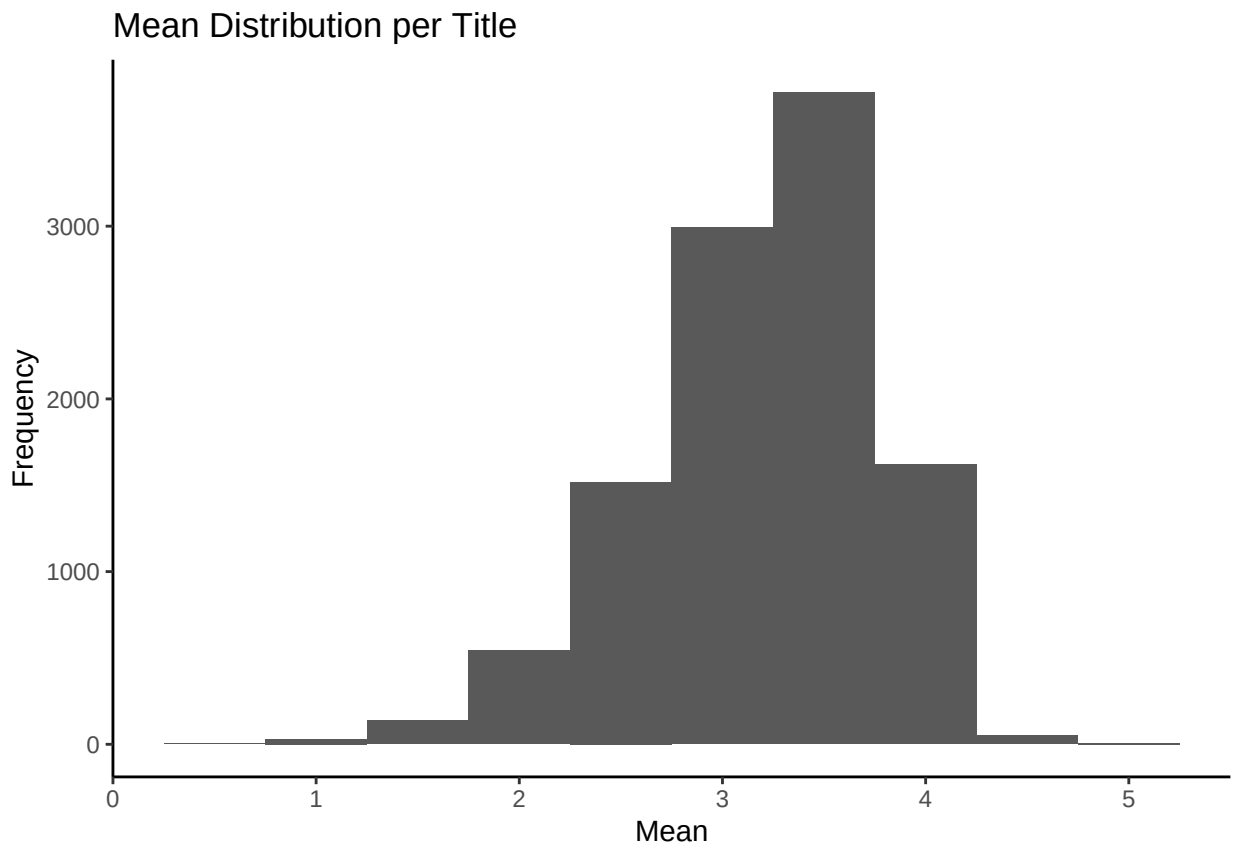
```
#plot histogram of ratings frequency distribution by movieid
ggplot(edx, aes(movieId)) +
  theme_classic()  +
  geom_histogram(bins=50) +
```

```
  labs(title = "Ratings Frequency Distribution by Title (MovieID)",
       x = "Title (MovieID)",
       y = "Frequency")
```



Mean distribution of ratings by movieId

```
#plot histogram of ratings mean distribution by movieid
edx %>%
  group_by(movieId) %>%
  summarise(mean = mean(rating)) %>%
  ggplot(aes(mean)) +
  theme_classic()  +
  geom_histogram(bins=10) +
  labs(title = "Mean Distribution per Title",
       x = "Mean",
       y = "Frequency")
```



2.4 UserId analysis

Notice users' rating times vary a lot with a maximum of 6616 and a minimum 10 in the edx dateset. The rating frequency distribution shows user effect should be considered when decreasing the value of RMSE.

Top rating users

```
#top rating users
edx %>% group_by(userId) %>%
  summarize(count = n()) %>%
  arrange(desc(count)) %>% head(10) %>%
  kable() %>%
  kable_styling(latex_options = "HOLD_position",bootstrap_options = c("striped", "hover", "condensed",
                position = "center",
                full_width = FALSE)
```

userId	count
59269	6616
67385	6360
14463	4648
68259	4036
27468	4023
19635	3771
3817	3733
63134	3371
58357	3361
27584	3142

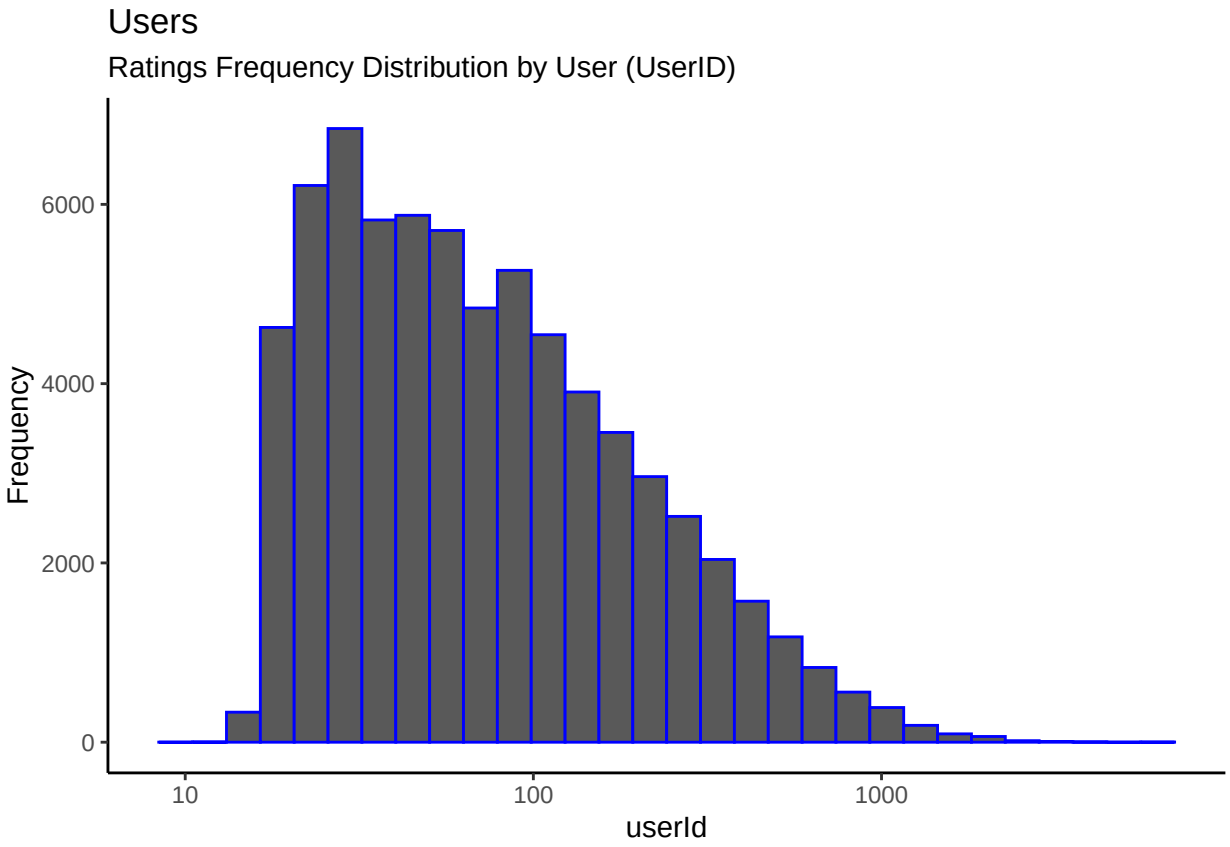
Bottom rating users

```
#bottom rating users
edx %>% group_by(userId) %>%
  summarize(count = n()) %>%
  arrange(desc(count)) %>% tail(10) %>%
  kable() %>%
  kable_styling(latex_options = "HOLD_position",bootstrap_options = c("striped", "hover", "condensed",
                position = "center",
                full_width = FALSE)
```

userId	count
57894	14
62317	14
63143	14
68161	14
68293	14
71344	14
15719	13
50608	13
22170	12
62516	10

Frequency distribution of ratings by userId

```
#plot number of ratings against userid
edx %>%
  count(userId) %>%
  ggplot(aes(n)) +
  geom_histogram( bins=30, color = "blue") + scale_x_log10() +
  ggtitle("Users") +
  labs(subtitle ="Ratings Frequency Distribution by User (UserID)",
       x="userId" ,y="Frequency") +
  theme_classic()
```



2.5 Genres analysis

Because each movie could be categorized in multiple genres, so the unique genres in edx is 797. Simply visualizing the frequency and mean of ratings by genres doesn't fit the real life situation. Say even a fanatic movie hobbyist probably dislike picking movies to watch based on random 797 genre labels. So we try to separate each row of the genres to have distinct genres, one for each row.

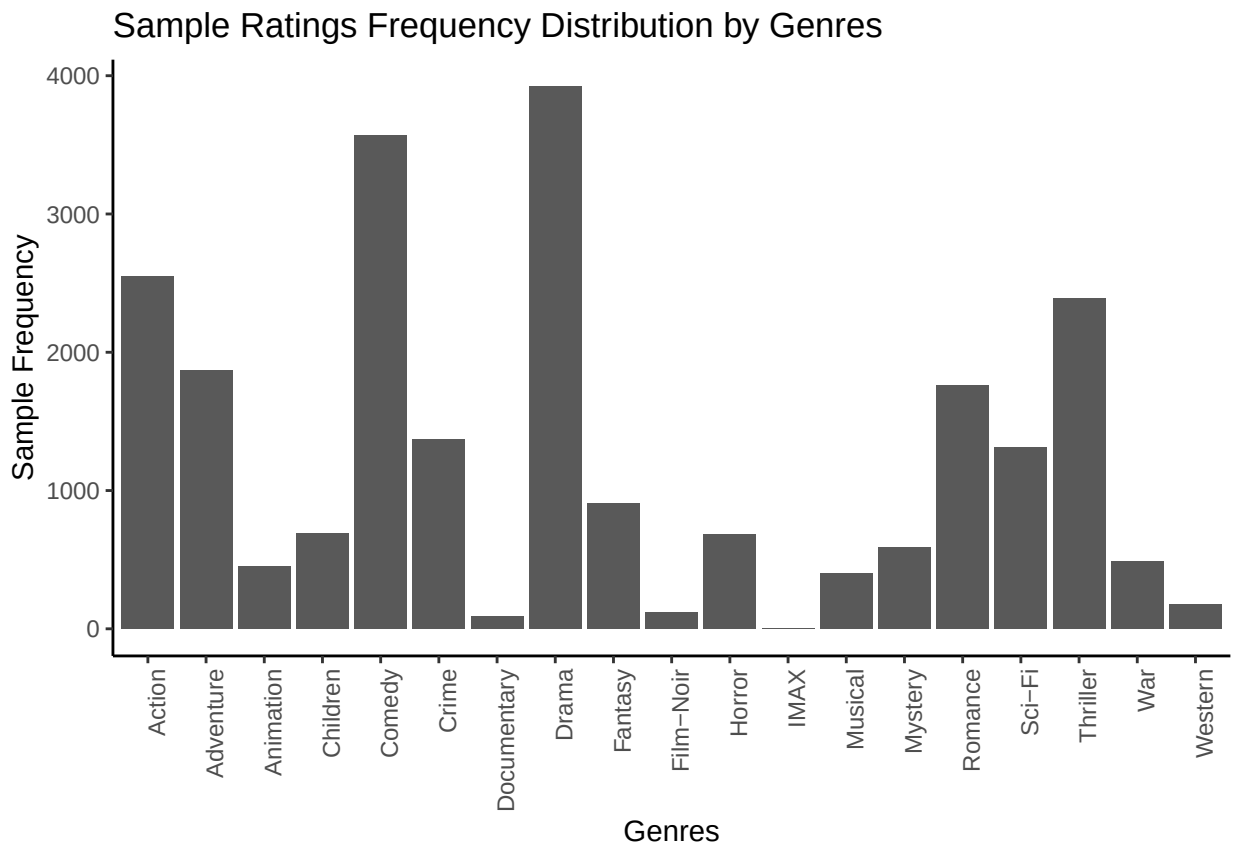
Since the edx dataset is quite ginormous, we can have a sample and check its frequency and mean of ratings. The downside of this implementation is that 1 or 2 genre types might be missed, and some movies with the majority of ratings can increase the value their respective genres exponentially, and then the movies with less ratings generate far small number of genres comparatively.

Overall, the genres might have trivial effects in lowering the RSME value in comparison with movieID and userID.

Sample frequency of ratings by genres

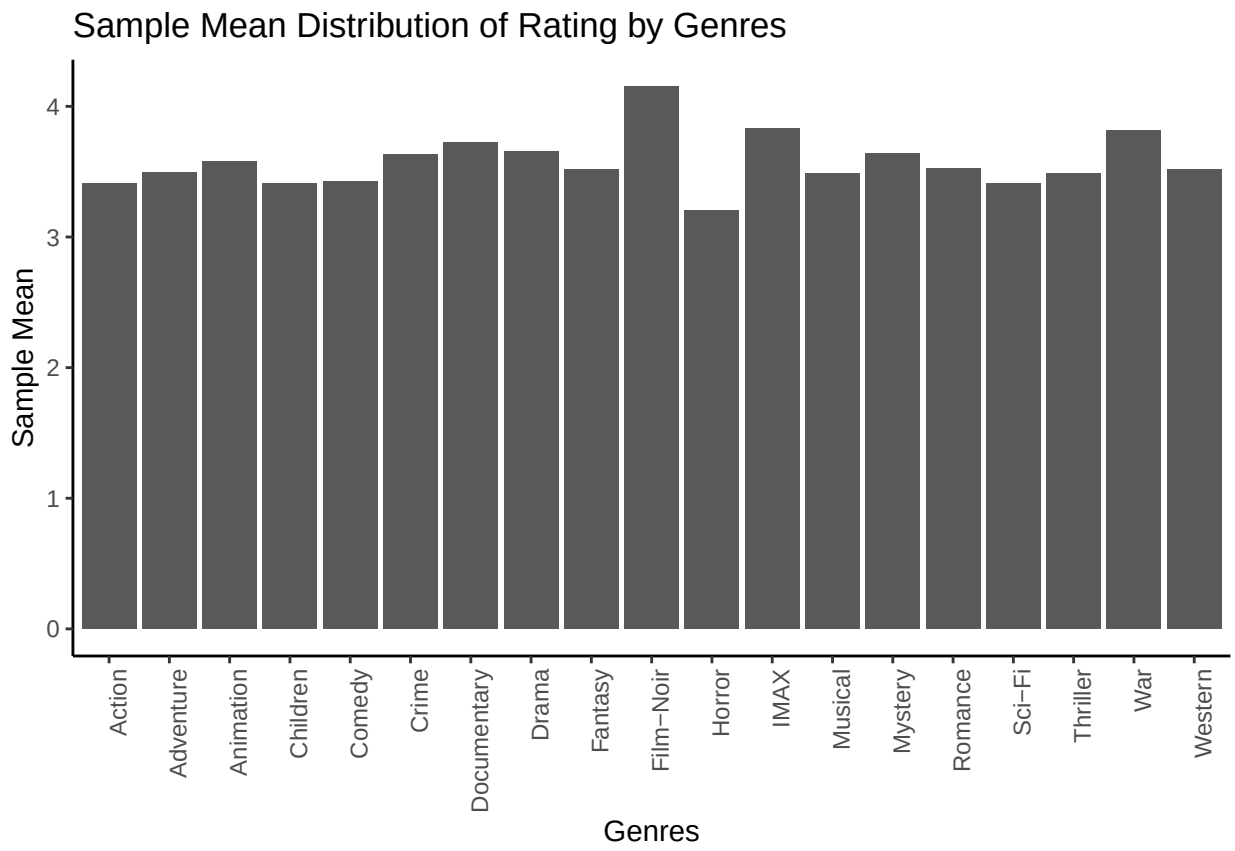
```
#check the unique genres
unique <- edx %>% summarize(genres=n_distinct(genres))

#sample a dataset from edx
t1000<-sample(1:nrow(edx),as.integer(0.001*nrow(edx)))
dat1000<-edx[t1000,]

#plot Sample Ratings Frequency Distribution by Genres
dat1000 %>% separate_rows(genres, sep = "\\|") %>% group_by(genres) %>%
```

```
  summarise(count=n()) %>%
  ggplot(aes(genres, count)) +
  theme_classic()  +
  geom_col() +
  theme(axis.text.x = element_text(angle = 90, hjust = 1)) +
  labs(title = "Sample Ratings Frequency Distribution by Genres",
       x = "Genres",
       y = "Sample Frequency")
```



Sample mean distribution of ratings by genres

```
#plot the Sample Mean Distribution of Rating by Genres

dat1000 %>% separate_rows(genres, sep = "\\|") %>%
  group_by(genres) %>%
  summarise(mean = mean(rating)) %>%
  ggplot(aes(genres, mean)) +
  theme_classic()  +
  geom_col() +
  theme(axis.text.x = element_text(angle = 90, hjust = 1)) +
  labs(title = "Sample Mean Distribution of Rating by Genres",
       x = "Genres",
       y = "Sample Mean")
```



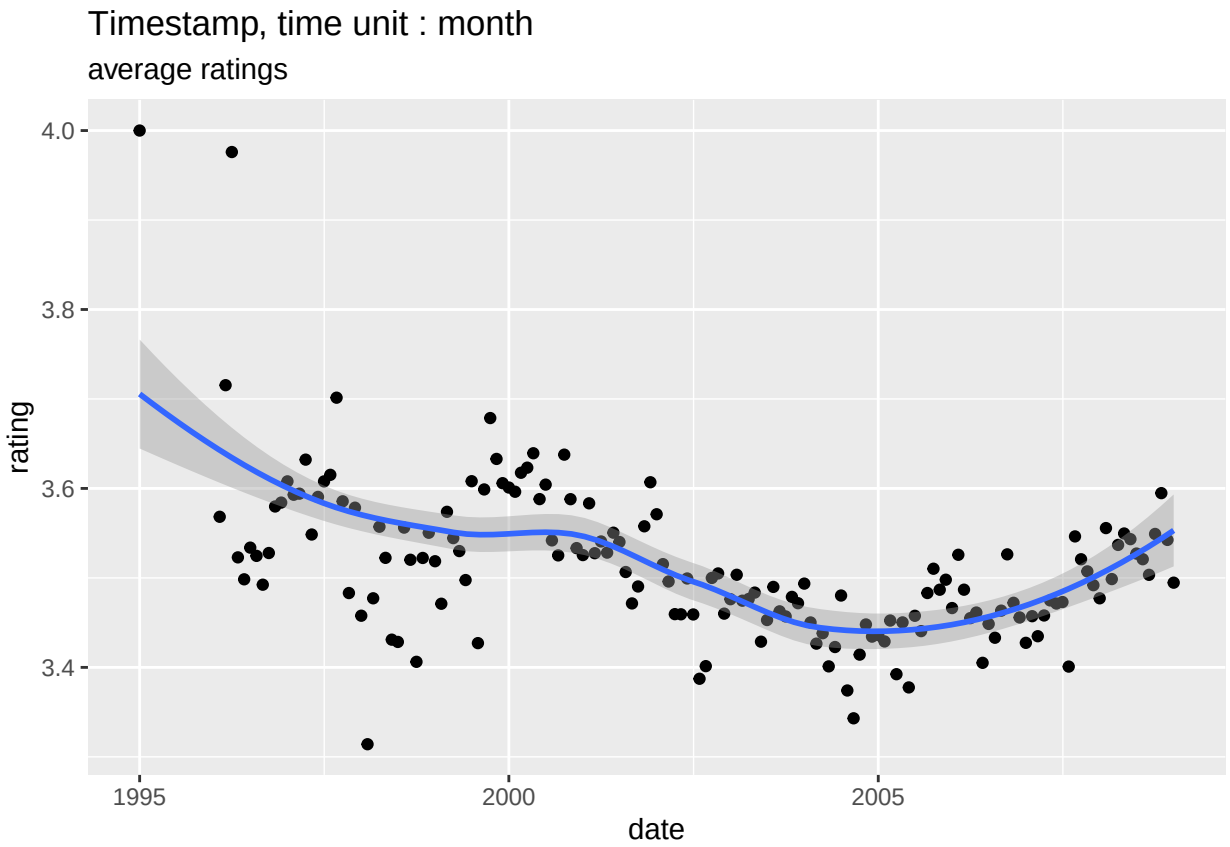
2.6 Timestamp analysis

After the timestamp is changed into normal date, the mean distribution of ratings is visualized. The distributions by month and by week don't show specific patterns. Thus the timestamp might also have trivial effects in lowering the RSME value in comparison with movieID and userID.

Mean distribution of ratings by by month

```
#plot mean distribution of ratings by month
edx %>%
  mutate(date = round_date(as_datetime(timestamp), unit = "month")) %>% group_by(date) %>%
  summarize(rating = mean(rating)) %>%
  ggplot(aes(date, rating)) +
  geom_point() +
  geom_smooth() +
  ggtitle("Timestamp, time unit : month")+
  labs(subtitle = "average ratings")
```

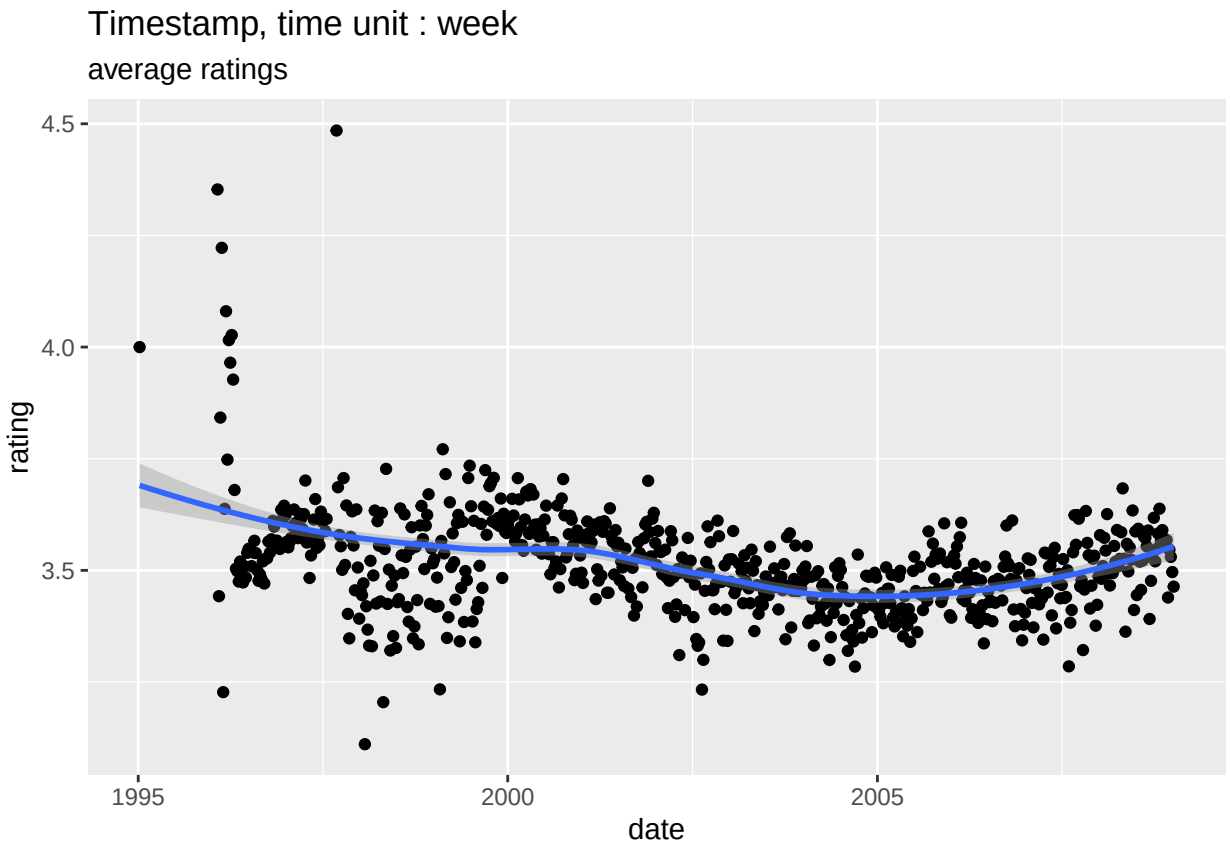
```
## `geom_smooth()` using method = 'loess' and formula = 'y ~ x'
```



Mean distribution of ratings by by week

```
#plot mean distribution of ratings by week
edx %>%
  mutate(date = round_date(as_datetime(timestamp), unit = "week")) %>% group_by(date) %>%
  summarize(rating = mean(rating)) %>%
  ggplot(aes(date, rating)) +
  geom_point() +
  geom_smooth() +
  ggtitle("Timestamp, time unit : week")+
  labs(subtitle = "average ratings")
```

```
## `geom_smooth()` using method = 'loess' and formula = 'y ~ x'
```



3 Methods

When only predict the mean, we can have the simplest model, naïve mean-baseline model, nevertheless, it'll be the least one to consider. After we add the movie effect(movieId), we have the content-based model. Since user effect(userId) shouldn't be ignored, we then have the user-based model. If we also consider genre effect, we can add that to have another model. The respective model formula are:

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

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

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

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

Mu is the rating mean, error_u_i is the errors, b_i is the measure of movie effect, b_u is the measure of user effect, and b_u_g is the measure of genre effect.

The RMSE values for movie effect, movie+user effects, and movie+user+genre effects are: 0.9439087, 0.8653488, 0.8649469 respectively. Notice the huge improvement from movie effect model to movie+user effects, but tiny increase in model quality after genre is considered. The results fit the assumptions from the EDA section. After regularization, we are able to achieve a target of RMSE < 0.86490 based on the grading rubric.

3.1 movie effect model

```
##-------------Methods--------------------

#define RMSE function
RMSE <- function(true_ratings = NULL, predicted_ratings = NULL) {
  sqrt(mean((true_ratings - predicted_ratings)^2))
}

#change name of the testing dataset
test<-final_holdout_test

#calculate the average of all ratings of the edx dataset
mu <- mean(edx$rating)

#naive mean-baseline model
rmse_model_0 <- RMSE(test$rating,mu)

#calculate b_i on the edx dataset
movie_m <- edx %>%
  group_by(movieId) %>% summarize(b_i = mean(rating - mu))

# predicted ratings
predicted_ratings_bi <- mu + test %>% left_join(movie_m, by='movieId') %>% .$b_i

#calculate the RMSE for movies
rmse_model_1 <- RMSE(test$rating,predicted_ratings_bi)
```

3.2 movie + user effects model

```
#calculate b_u using the edx set
user_m <- edx %>%
  left_join(movie_m, by='movieId') %>% group_by(userId) %>%
  summarize(b_u = mean(rating - mu - b_i))

#predicted ratings
predicted_ratings_bu <- test %>% left_join(movie_m, by='movieId') %>%
    left_join(user_m, by='userId') %>%
    mutate(pred = mu + b_i + b_u) %>% .$pred

#calculate the RMSE for movies + users
rmse_model_2 <- RMSE(test$rating,predicted_ratings_bu)
```

3.3 movie + user + genre effects model

```
#calculate b_u_g using the edx set
genre_m <- edx %>%
  left_join(movie_m, by='movieId') %>%
  left_join(user_m, by='userId') %>%
```

```
  group_by(genres) %>%
  summarize(b_u_g = mean(rating - mu - b_i - b_u))

#predicted ratings
predicted_ratings_bug <- test %>%
  left_join(movie_m, by='movieId') %>%
  left_join(user_m, by='userId') %>%
  left_join(genre_m, by='genres') %>%
  mutate(pred = mu + b_i + b_u + b_u_g) %>%
  pull(pred)

#calculate the RMSE for movies + users + genres effects
rmse_model_3 <- RMSE(test$rating,predicted_ratings_bug)
```

3.4 regularized models

The regularization formula is given:

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

. A penalty value lambda is introduced to minimize b_i in this equation:

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

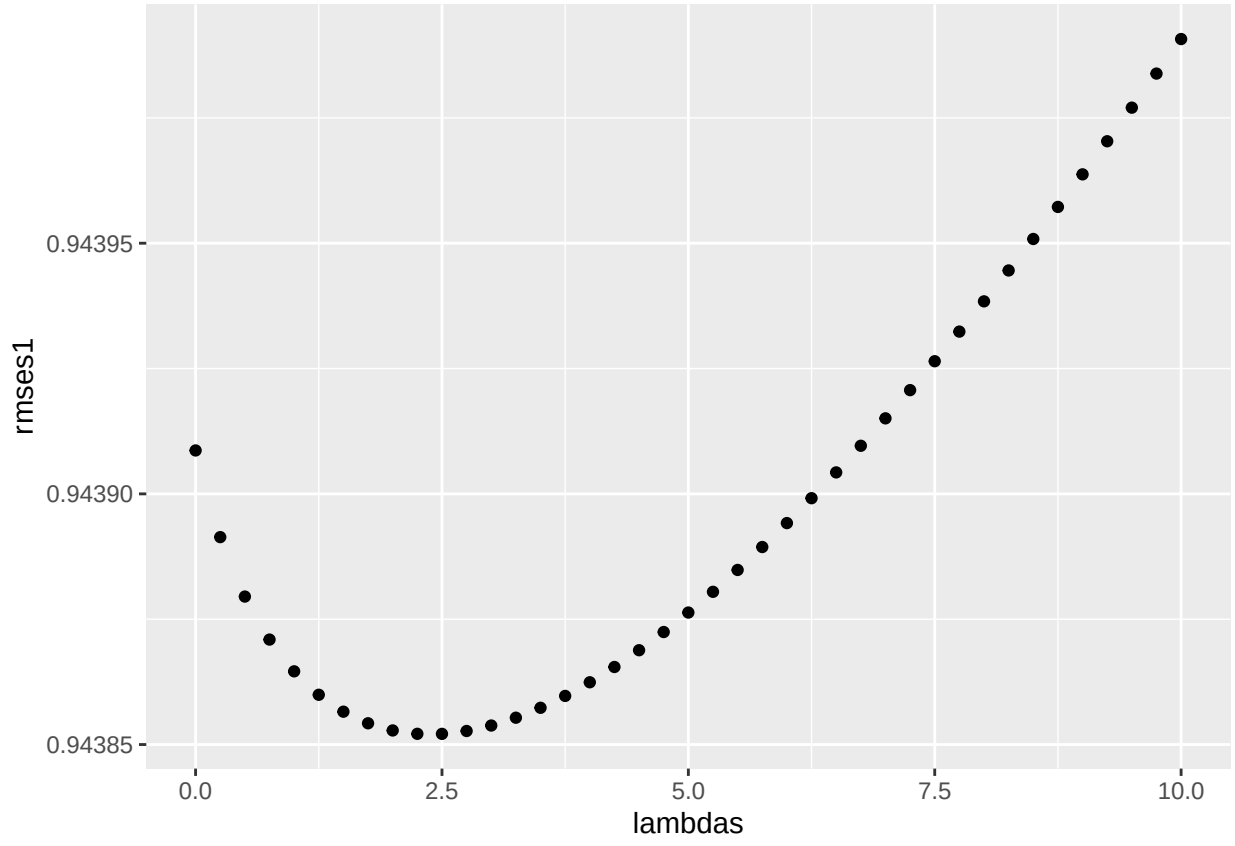
The respective minimum rmse values both decrease by about 0.0001for movie effect and for movie+user effects models, and it's a tiny improvement comparatively. Nevertheless, when lambda is 5.25, we have the rmse min value 0.8648170. Thus the target RMSE $<$ 0.86490 is reached.

The plot of rmse values against lambdas for movie effect model

```
#set up the values for the regularized tuning parameter lambda
lambdas <- seq(0, 10, 0.25)

#apply all the values of lambda on regularized models
rmses1 <- sapply(lambdas, function(x){
  mu_reg <- mean(edx$rating)

  b_i_reg <- edx %>%
    group_by(movieId) %>%
    summarize(b_i_reg = sum(rating - mu_reg)/(n()+x))

  predicted_ratings_b_i <-
    test %>%
    left_join(b_i_reg, by = "movieId") %>% mutate(pred = mu_reg + b_i_reg) %>% .$pred
  return(RMSE(test$rating,predicted_ratings_b_i)) })

rmses2 <- sapply(lambdas, function(x){
  mu_reg <- mean(edx$rating)

  b_i_reg <- edx %>%
    group_by(movieId) %>%
```

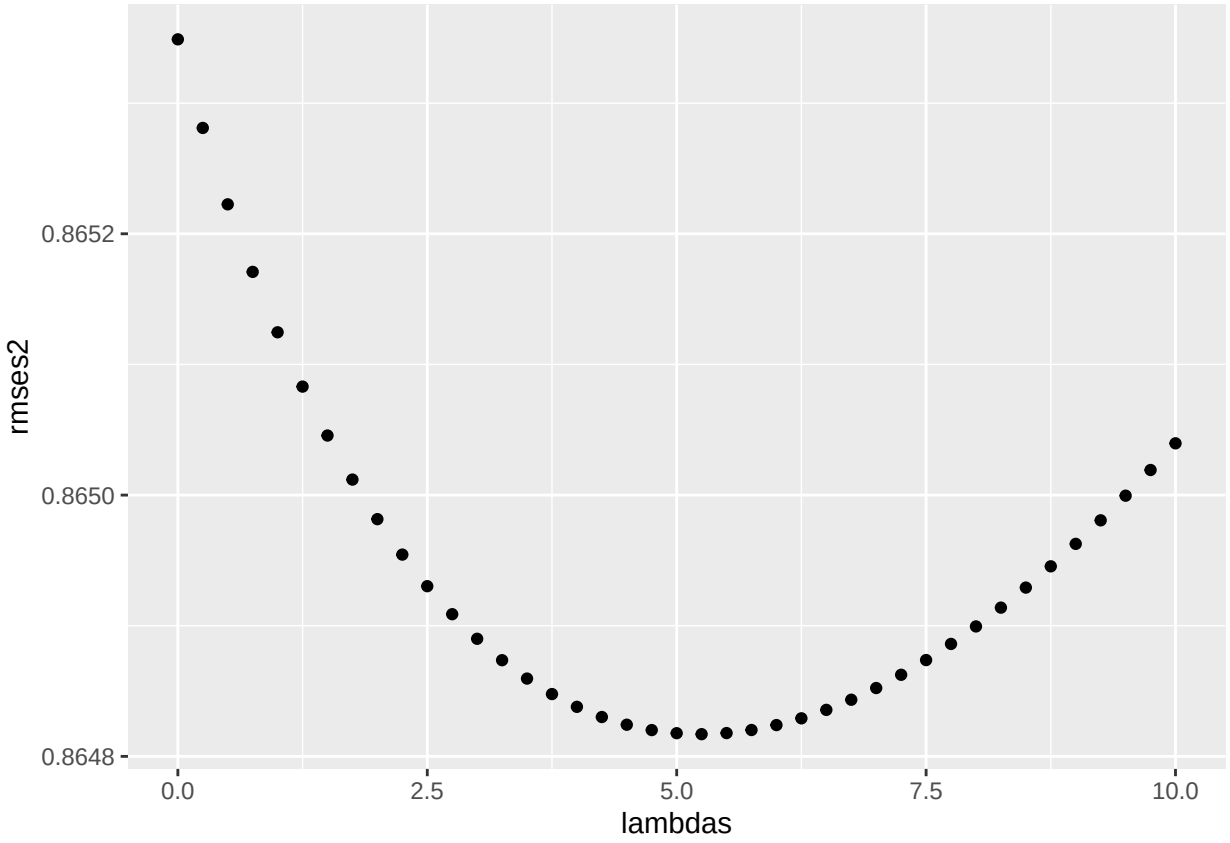
```
    summarize(b_i_reg = sum(rating - mu_reg)/(n()+x))

  b_u_reg <- edx %>%
    left_join(b_i_reg, by="movieId") %>%
    group_by(userId) %>%
    summarize(b_u_reg = sum(rating - b_i_reg - mu_reg)/(n()+x))

  predicted_ratings_b_i_u <-
    test %>%
    left_join(b_i_reg, by = "movieId") %>% left_join(b_u_reg, by = "userId") %>%
    mutate(pred = mu_reg + b_i_reg + b_u_reg) %>% .$pred
  return(RMSE(test$rating,predicted_ratings_b_i_u)) })

#plot rmse values against lambdas for movie effect
qplot(lambdas, rmses1)
```

```
## Warning: `qplot()` was deprecated in ggplot2 3.4.0.
```



The plot of rmse values against lambdas for movie+user effects model

```
#plot rmse values against lambdas for movie+user effects
qplot(lambdas, rmses2)
```




```
#find the optimal lambda values respectively
lambda1 <- lambdas[which.min(rmses1)]
lambda2 <- lambdas[which.min(rmses2)]

#get the minimum rmse values
rmse_model_1_reg <- min(rmses1)
rmse_model_2_reg <- min(rmses2)
```

4 Results

This the summary of all the models built, trained on the edx dataset, and tested on the final_hold_out dataset.

```
##-------------Results--------------------

#summarize all the rmse on final_holdout_test data set for models
rmse_results <- data.frame(methods=c("movie effect","movie + user effects",
                                     "movie + user + genre effects", "regularized movie effect",
                                     "regularized movie + user effects"),
                           rmse=c(rmse_model_1,rmse_model_2,rmse_model_3,rmse_model_1_reg,
                                  rmse_model_2_reg))

kable(rmse_results) %>% kable_styling(latex_options = "HOLD_position",bootstrap_options = c("striped",
```

```
                          position = "center",
                          full_width = FALSE) %>% row_spec(c(4,5),color = "white" , background ="blue")
```

methods	rmse
movie effect	0.9439087
movie + user effects	0.8653488
movie + user + genre effects	0.8649469
regularized movie effect	0.9438521
regularized movie + user effects	0.8648170

5 Conclusion

Apparently, all 5 rmse values from the result summary are less than the value 1.061202 from naïve baseline model. The overall best model is regularized movie + user effects one due to their overwhelming enhancement in decreasing rmse over genre predictor. Though, the regularization process just slightly improved the model quality, it helps to achieve the objective.

Similarity measures like euclidean distance, pearson correlation, cosine distance and dimensionality reduction like PCA/SVD can be applied to deal with the problem of sparsity of a more effective matrix for future work. Collaborative filtering recommender system and even ensemble algorithms could be implemented with supercomputing devices.