MovieLens Project - RMSE-Analysis

Mruthyum J Thatipamala

2020-10-21

----------Begin Report----------

1. Introduction/Overview/Executive summary: The goal of movielens project is to simulate and develop a movie recommendation system to give ratings to movies on test data (as if prior ratings are not available) based on ratings available in train data. It is a sort of machine learning challenge project.

GroupLens research labs generated data with over 20 million ratings for over 27,000 movies by more than 138,000 users. A section of this data, about 9 million records, is provided by edx team in the form of training R data set and validation R data set (.rds). These datasets are downloaded and stored into temporary files. From saved files, corresponding dataframes are created using readRDS function.

An algorithm is developed using the train set to calculate the Residual Mean Square Error (RMSE), similar to standard deviation, is calculated. As a final test to the final algorithm, predict movie ratings in the set (the final hold-out test set) as if they were unknown. RMSE will be used to evaluate how close your predictions are to the true values in the validation set (the final hold-out test set).

----------****----------

2. Methods/Analysis: Data cleaning: Movie title column is not used anywhere at any level of the algorithm. Hence, it is deleted from both train and test sets. Several partitions of train data are created using caret package functions to test the biases within the training phase. It is observed that some of the ratings are blank, those are filled with average value rather than deleting the rows.

Modelling: First model, the simplest model, is developed based on the assumption that same rating is given by all users to all movies, which is the average of the ratings (mu). Then 'naive rmse' is calculated using the mu and the actual rating given for each movie.

It is an interesting observation that users give ratings differently to different movies. Hence, the basic model is augmented by adding movie bias (b_i). The rmse calculated at this level is less than naive rmse.

From the train data, we can make an observation that ratings given to a movie by different users vary substantially. Hence, a new bais, b_u, is added to the model.

Also, movies belonging to a particular set of genres are given higher ratings in general compared to other genres that attract lower ratings. Also, rating will also vary (though slightly) on the day of the week. Hence, the RMSE model is augmented by adding genre bias (b_g) and time bias (b_t).

It can be easily observed in the data that some movies belonging to a particular get 100+ ratings whereas there are certain movies which get only one rating. Hence, there will be a huge variation in calculating RMSE. The movie that gets more ratings is estimated correctly. To constrain the effect of sizes (the general idea behind regularization), a range of (3.5, 5.5, 0.25) lambda values are introduced as a part of last modelling.

Visualization: To support key concepts and data observations used in analysis, bar plots, scatter plots, and box plots are used as part of data visualization.

Insights: As the model is improved by incorporating more biases, computed RMSE value has reduced considerably. Also, the bigger the size of train and validation data, the more accurate the rating predictions are. Let us look at the code of Model and RSME Algorithm.

_____****_____

Step1: Clean up heap memory and plots - Optimizing memory of environment. This happens at several places in the code

```
# Clear environment
rm(list = ls())
# Clear console
cat("\014")
```

```
# Clear plots
if(!is.null(dev.list())) dev.off()
```

```
## null device
##           1
```

Step2: Installing packages and loading libraries

```
if(!require(tidyverse)) install.packages("tidyverse", repos = "http://cran.us.r-project.org")
if(!require(caret)) install.packages("caret", repos = "http://cran.us.r-project.org")
if(!require(lubridate)) install.packages("lubridate", repos = "http://cran.us.r-project.org")
if(!require(ggplot2)) install.packages("ggplot2", repos = "http://cran.us.r-project.org")
if(!require(dplyr)) install.packages("dplyr", repos = "http://cran.us.r-project.org")


library(caret)
library(tidyverse)
library(dplyr)
library(ggplot2)
library(lubridate)
library(rmarkdown)
```

Step3: Reading the smaller set of edx and validation data from CSV files and saving them to dataframes. Merging them into one big dataframe each using 'rbind' function

```
newtrialset1 <- read.csv(file = "newtrialset1.csv", head = TRUE, sep="\t")
newtrialset2 <- read.csv(file = "newtrialset2.csv", head = TRUE, sep="\t")
newtrialset3 <- read.csv(file = "newtrialset3.csv", head = TRUE, sep="\t")
newtrialset4 <- read.csv(file = "newtrialset4.csv", head = TRUE, sep="\t")
newtrialset5 <- read.csv(file = "newtrialset5.csv", head = TRUE, sep="\t")

newvalidation1 <- read.csv(file = "newvalidation1.csv", head = TRUE, sep="\t")
newvalidation2 <- read.csv(file = "newvalidation2.csv", head = TRUE, sep="\t")
newvalidation3 <- read.csv(file = "newvalidation3.csv", head = TRUE, sep="\t")

trialset <- rbind(newtrialset1, newtrialset2, newtrialset3, newtrialset4, newtrialset5)
rm(list = c("newtrialset1", "newtrialset2", "newtrialset3", "newtrialset4", "newtrialset5"))

validation <- rbind(newvalidation1, newvalidation2, newvalidation3)
rm(list = c("newvalidation1", "newvalidation2", "newvalidation3"))
```

Step4: It is observed from the data that movies belonging some genres are watched more and ratings are also higher. On contrary, some genres are watched less and rating are also low.

```
high_boxplot_genres_rating <- trialset %>% filter(genres %in% c("Drama", "Comedy", "Comedy|Romance", "C
str(high_boxplot_genres_rating)
```

```
## 'data.frame':    794025 obs. of  2 variables:
##  $ genres: chr  "Comedy|Romance" "Drama" "Drama" "Comedy|Drama|Romance" ...
##  $ rating: num  5 5 4 4.5 3 2 3 3 3 3 ...
```

```
head(high_boxplot_genres_rating)
```

```
##                 genres rating
## 1       Comedy|Romance    5.0
## 2                Drama    5.0
## 3                Drama    4.0
```

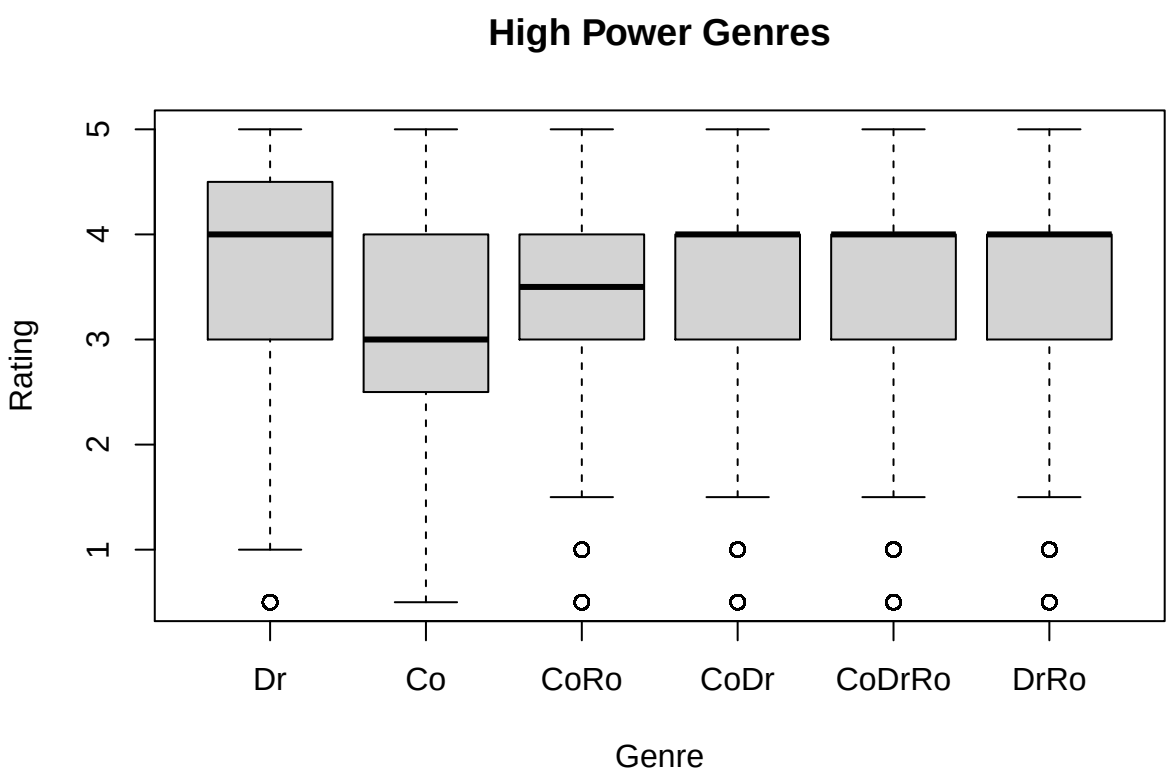
```
## 4 Comedy|Drama|Romance    4.5
## 5       Comedy|Romance    3.0
## 6         Comedy|Drama    2.0
```

```
mean(high_boxplot_genres_rating$rating)
```

```
## [1] 3.513566
```

```
high_boxplot_genres_rating$genres <- factor(high_boxplot_genres_rating$genres, levels = c("Drama", "Como

boxplot(rating ~ genres, data = high_boxplot_genres_rating, xlab = "Genre", ylab = "Rating", main = "Hig
```




```
# Clear plot
if(!is.null(dev.list())) dev.off()
```

```
## null device
##           1
```

```
low_boxplot_genres_rating <- trialset %>% filter(genres %in% c("Action|Drama|Horror|Sci-Fi", "Action|Ro
str(low_boxplot_genres_rating)
```

```
## 'data.frame':    11 obs. of  2 variables:
##  $ genres: chr  "Adventure|Comedy|Drama|Fantasy|Mystery|Sci-Fi" "Adventure|Horror|Romance|Sci-Fi" "A
##  $ rating: num  4.5 5 3 4.5 5 3 2 4 0.5 4 ...
```

```
head(low_boxplot_genres_rating)
```

```
##                                       genres rating
## 1 Adventure|Comedy|Drama|Fantasy|Mystery|Sci-Fi    4.5
## 2               Adventure|Horror|Romance|Sci-Fi    5.0
## 3               Adventure|Crime|Horror|Thriller    3.0
## 4 Adventure|Comedy|Drama|Fantasy|Mystery|Sci-Fi    4.5
## 5 Adventure|Comedy|Drama|Fantasy|Mystery|Sci-Fi    5.0
```

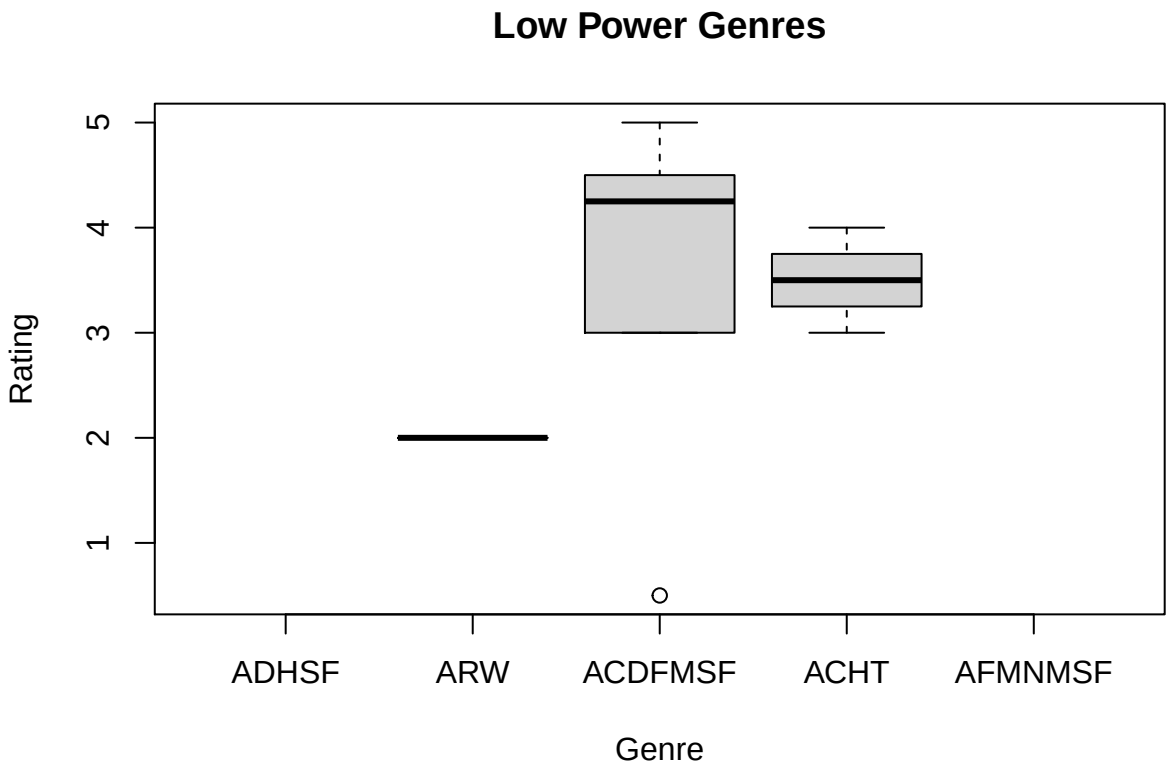
```
## 6 Adventure|Comedy|Drama|Fantasy|Mystery|Sci-Fi    3.0
```

```
mean(low_boxplot_genres_rating$rating)
```

```
## [1] 3.545455
```

```
low_boxplot_genres_rating$genres <- factor(low_boxplot_genres_rating$genres, levels = c("Action|Drama|H

boxplot(rating ~ genres, data = low_boxplot_genres_rating, xlab = "Genre", ylab = "Rating", main = "Low
```




```
# Clear plot
if(!is.null(dev.list())) dev.off()
```

```
## null device
##           1
```

Step5:Defining RMSE function, a function that computes the RMSE for vectors of ratings and their corresponding predictors

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

Step6:Instead of taking the mean of entire dataset, we take the lower end, add increments to reach upper end to calculate the lowest naive rmse and its 'mu'. This a similar to weighted/average distribution analysis

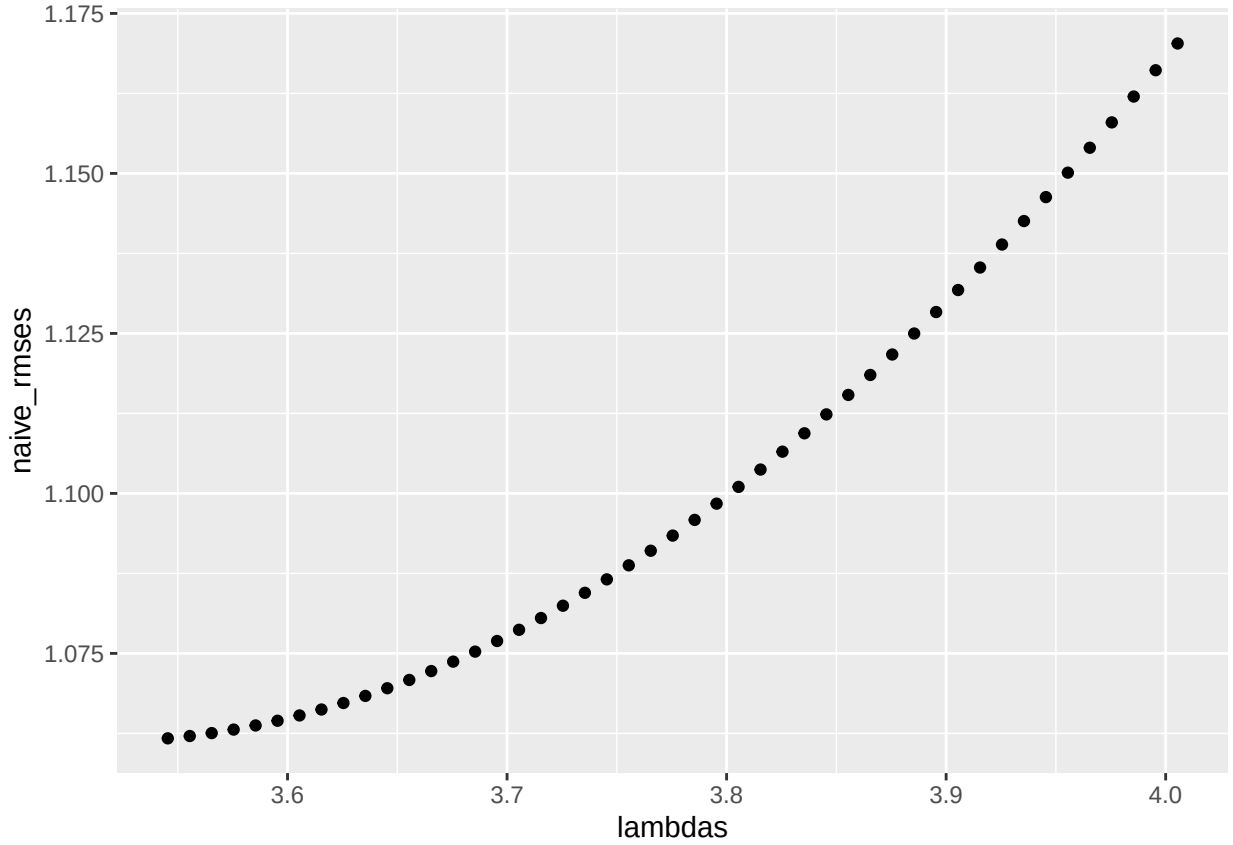
```
lambdas<-seq(mean(low_boxplot_genres_rating$rating),mean(high_boxplot_genres_rating$rating)+0.5, 0.01)

rm(list = c("high_boxplot_genres_rating", "low_boxplot_genres_rating"))

naive_rmses <- sapply(lambdas, function(l){
  return(RMSE(validation$rating,l))
```

```
})

qplot(lambdas, naive_rmses)
```




```
# Clear plots
if(!is.null(dev.list())) dev.off()
```

```
## null device
##           1
```

```
#(Weighted) average of ratings across the train set
mu <- lambdas[which.min(naive_rmses)]
mu
```

```
## [1] 3.545455
```

```
#Storing rmse results for various baises in training set
rmse_results <- tibble(method = "naive", RMSE = min(naive_rmses))
rm(list = c("lambdas", "naive_rmses"))
```

Step6: Movie bias - We observe from the data that different movies are rated differently. Modeling movie effects by a bias, b_i

```
movie_avgs <- trialset %>%
  group_by(movieId) %>%
  summarize(b_i = mean(rating - mu))
```

```
## `summarise()` ungrouping output (override with `.groups` argument)
```

```
#Data set for testng the movie bias effect and calculating RMSE
movieset <- read.csv(file = "movieset.csv", head = TRUE, sep="\t")

predicted_ratings <- mu + movieset %>% left_join(movie_avgs, by='movieId') %>% pull(b_i)

predicted_ratings <- predicted_ratings %>% replace_na(mu)

movie_rmse <- RMSE(predicted_ratings, movieset$rating)
movie_rmse
```

```
## [1] 0.940034
```

```
rmse_results <- rmse_results %>% add_row(method = "movie", RMSE = movie_rmse)
rm(list = c("predicted_ratings", "movieset"))
```

Step7: User bias - there is substantial variability in rating for a given movie. Different users give different rating for the same movie. Calculating user effects thru a bias

```
user_avgs <- trialset %>%
  left_join(movie_avgs, by='movieId') %>%
  group_by(userId) %>%
  summarize(b_u = mean(rating - mu - b_i))
```

```
## `summarise()` ungrouping output (override with `.groups` argument)
```

```
#Data set for testng the user bias effect and calculating RMSE
userset <- read.csv(file = "userset.csv", head = TRUE, sep="\t")

predicted_ratings <- userset %>%
  left_join(movie_avgs, by='movieId') %>%
  left_join(user_avgs, by='userId') %>%
  mutate(pred = mu + b_i + b_u) %>%
  pull(pred)

predicted_ratings <- predicted_ratings %>% replace_na(mu)

user_rmse <- RMSE(predicted_ratings, userset$rating)
user_rmse
```

```
## [1] 0.8469732
```

```
rmse_results <- rmse_results %>% add_row(method = "user", RMSE = user_rmse)
rm(list = c("predicted_ratings", "userset"))
```

Step7: Genres bias - As calculated and observed in Step4movies belonging some genres are watched more and ratings are also higher. On contrary, some genres are watched less and rating are also low. Inclusion of genre bias

```
genres_avgs <- trialset %>%
  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))
```

```
## `summarise()` ungrouping output (override with `.groups` argument)
```

```
#Data set for testng the genres bias effect and calculating RMSE
genresset <- read.csv(file = "genresset.csv", head = TRUE, sep="\t")
```

```
predicted_ratings <- genresset %>%
  left_join(movie_avgs, by='movieId') %>%
  left_join(user_avgs, by='userId') %>%
  left_join(genres_avgs, by='genres') %>%
  mutate(pred = mu + b_i + b_u + b_g) %>% pull(pred)

predicted_ratings <- predicted_ratings %>% replace_na(mu)

genre_rmse <- RMSE(predicted_ratings, genresset$rating)
genre_rmse
```

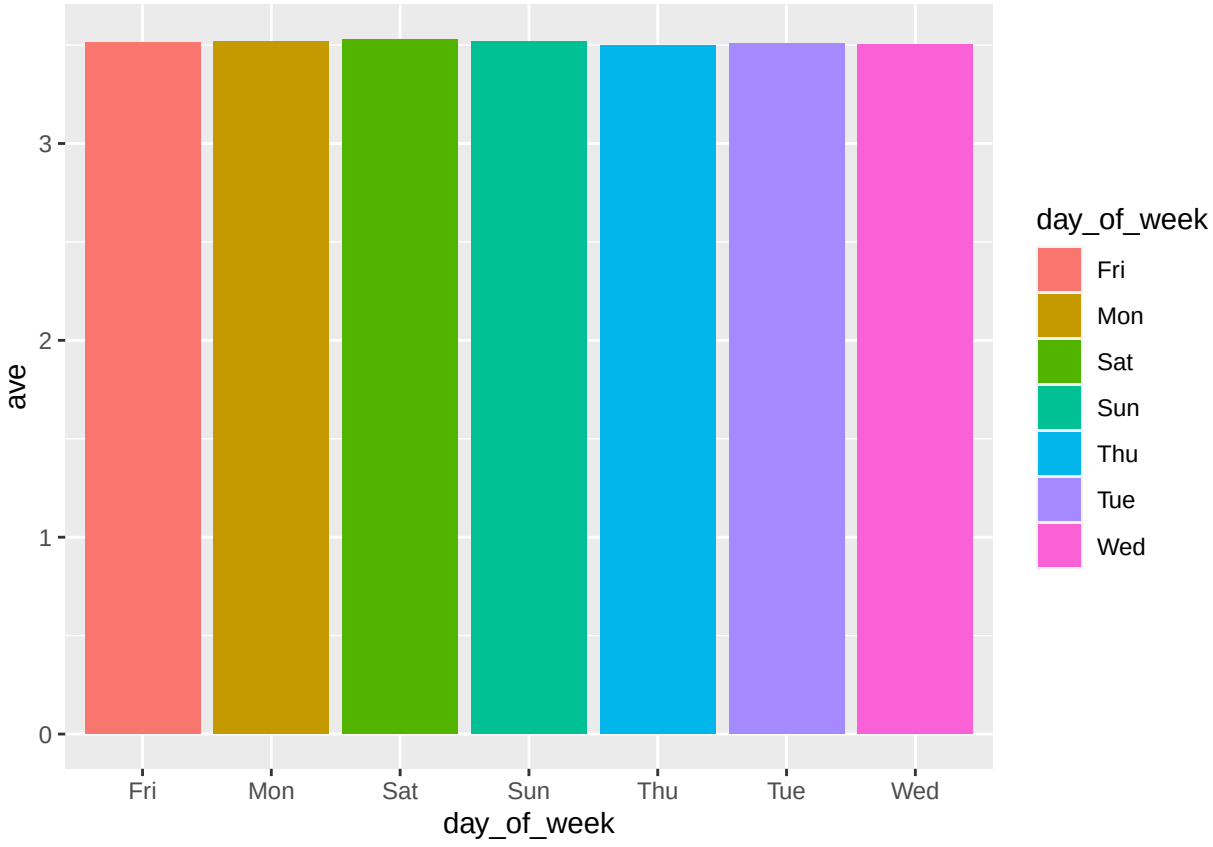
```
## [1] 0.846615
```

```
rmse_results <- rmse_results %>% add_row(method = "genre", RMSE = genre_rmse)
rm(list = c("predicted_ratings", "genresset"))
```

Step8: Time bias. How the ratings averages are varying on a day of the week can be observed from the bar charts plotted in the below given code. Though small change, the average of ratings given on Tues/Wed/Thursdays are less compared to other days. Accommodating time/day bias.

```
trialset %>% group_by(day_of_week) %>% summarize(total = n(), ave = mean(rating)) %>% ggplot(aes(x=day_
```

```
## `summarise()` ungrouping output (override with `.groups` argument)
```




```
#Clear plots
if(!is.null(dev.list())) dev.off()
```

```
## null device
```

```
##           1
```

```
time_avgs <- trialset %>%
  left_join(movie_avgs, by='movieId') %>%
  left_join(user_avgs, by='userId') %>%
  left_join(genres_avgs, by='genres') %>%
  group_by(day_of_week) %>%
  summarize(b_d = mean(rating - mu - b_i - b_u - b_g))
```

```
## `summarise()` ungrouping output (override with `.groups` argument)
```

```
#Data set for testng the time bias effect and calculating RMSE
timeset <- read.csv(file = "timeset.csv", head = TRUE, sep="\t")

predicted_ratings <- timeset %>%
  left_join(movie_avgs, by='movieId') %>%
  left_join(user_avgs, by='userId') %>%
  left_join(genres_avgs, by='genres') %>%
  left_join(time_avgs, by='day_of_week') %>%
  mutate(pred = mu + b_i + b_u + b_g + b_d) %>% pull(pred)

time_rmse <- RMSE(predicted_ratings, timeset$rating)
time_rmse
```

```
## [1] 0.8466151
```

```
rmse_results <- rmse_results %>% add_row(method = "time", RMSE = time_rmse)
rm(list = c("predicted_ratings", "timeset"))
```

Final Step: Putting together all biases together and calculating the RMSE against validation set. This analysis also includes a tuning parameter lambda and use it for cross-validation to accommodate regularization for total variability of effect of sizes.

```
lambdas <- seq(3.5, 5.5, 0.25)
rmses <- sapply(lambdas, function(l){

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

b_u <- trialset %>% left_join(b_i, by="movieId") %>% group_by(userId) %>% group_by(userId) %>% summariz

b_g <- trialset %>% left_join(movie_avgs, by='movieId') %>% left_join(user_avgs, by='userId') %>%
  group_by(genres) %>% summarize(b_g = sum(rating - mu - b_i - b_u)/(n()+l))

b_d <- trialset %>% left_join(movie_avgs, by='movieId') %>%
  left_join(user_avgs, by='userId') %>%
  left_join(genres_avgs, by='genres') %>%
  group_by(day_of_week) %>%
  summarize(b_d = sum(rating - mu - b_i - b_u - b_g)/(n()+l))

predicted_ratings <- validation %>%
  left_join(b_i, by='movieId') %>%
  left_join(b_u, by='userId') %>%
  left_join(b_g, by='genres') %>%
  left_join(b_d, by='day_of_week') %>%
  mutate(pred = mu + b_i + b_u + b_g + b_d) %>% pull(pred)

predicted_ratings <- predicted_ratings %>% replace_na(mu)
```

```r
RMSE(predicted_ratings, validation$rating)
 return(RMSE(predicted_ratings, validation$rating))
})
```

```
## `summarise()` ungrouping output (override with `.groups` argument)
## `summarise()` ungrouping output (override with `.groups` argument)
## `summarise()` ungrouping output (override with `.groups` argument)
## `summarise()` ungrouping output (override with `.groups` argument)
## `summarise()` ungrouping output (override with `.groups` argument)
## `summarise()` ungrouping output (override with `.groups` argument)
## `summarise()` ungrouping output (override with `.groups` argument)
## `summarise()` ungrouping output (override with `.groups` argument)
## `summarise()` ungrouping output (override with `.groups` argument)
## `summarise()` ungrouping output (override with `.groups` argument)
## `summarise()` ungrouping output (override with `.groups` argument)
## `summarise()` ungrouping output (override with `.groups` argument)
## `summarise()` ungrouping output (override with `.groups` argument)
## `summarise()` ungrouping output (override with `.groups` argument)
## `summarise()` ungrouping output (override with `.groups` argument)
## `summarise()` ungrouping output (override with `.groups` argument)
## `summarise()` ungrouping output (override with `.groups` argument)
## `summarise()` ungrouping output (override with `.groups` argument)
## `summarise()` ungrouping output (override with `.groups` argument)
## `summarise()` ungrouping output (override with `.groups` argument)
## `summarise()` ungrouping output (override with `.groups` argument)
## `summarise()` ungrouping output (override with `.groups` argument)
## `summarise()` ungrouping output (override with `.groups` argument)
## `summarise()` ungrouping output (override with `.groups` argument)
## `summarise()` ungrouping output (override with `.groups` argument)
## `summarise()` ungrouping output (override with `.groups` argument)
## `summarise()` ungrouping output (override with `.groups` argument)
## `summarise()` ungrouping output (override with `.groups` argument)
## `summarise()` ungrouping output (override with `.groups` argument)
## `summarise()` ungrouping output (override with `.groups` argument)
## `summarise()` ungrouping output (override with `.groups` argument)
## `summarise()` ungrouping output (override with `.groups` argument)
## `summarise()` ungrouping output (override with `.groups` argument)
## `summarise()` ungrouping output (override with `.groups` argument)
## `summarise()` ungrouping output (override with `.groups` argument)
## `summarise()` ungrouping output (override with `.groups` argument)
```

```r
rm(list = c("trialset", "movie_avgs", "user_avgs", "genres_avgs"))
```

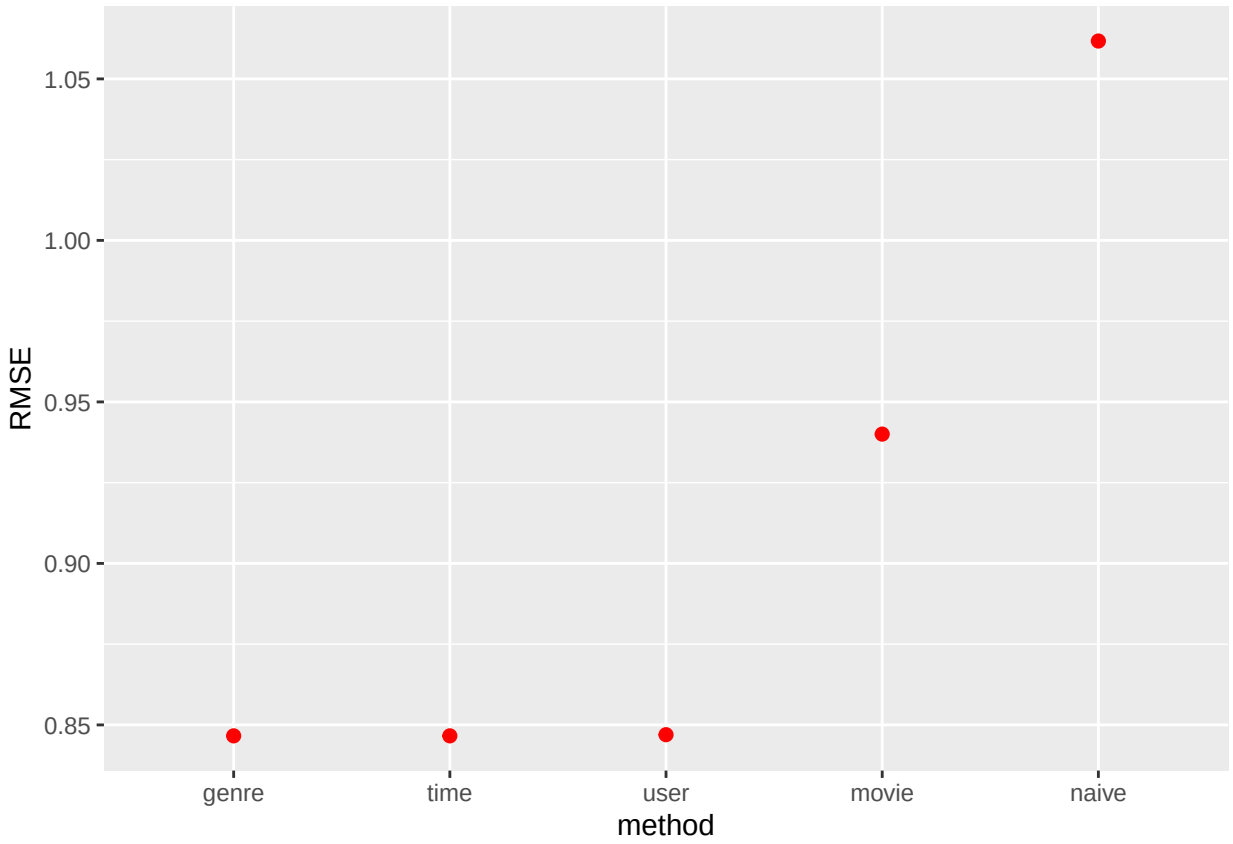
______________________________****__________________________________

3. Results:

Training Modeling results:

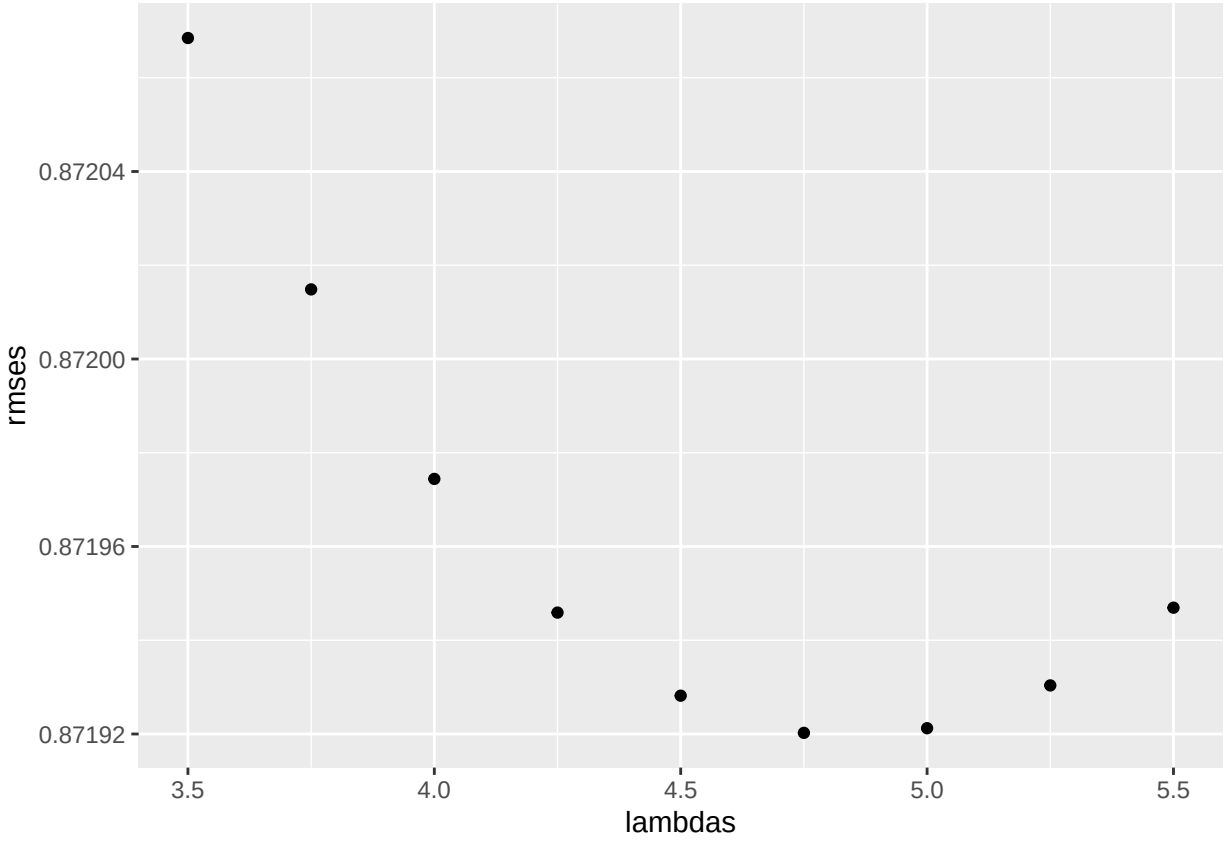
Scatter plot below shows an improvement in model performance during training phase. RMSE value is reduced from 1.055+ to 0.8466 as more biases are added to the algorithm.

```r
rmse_results$method <- factor(rmse_results$method, levels = rmse_results$method[order(rmse_results$RMSE
rmse_results %>% ggplot(aes(x=method, y=RMSE)) + geom_point(colour = 'red', size = 2)
```



Model performance:

```
qplot(lambdas, rmses)
```




```
final_rmse <- min(rmses)
final_rmse
```

```
## [1] 0.8719202
```

The lowest value of RMSE on validation set is 0.87192

Conclusion:

Summary:

Performance of the model improves as more biases are included in the algorithm. It can be improved further by applying advanced techniques as explained in the following section.

Limitations and Future work:

I got stuck in India due to lockdown during COVID-19. There is a big limitation on WiFi speed, and the very low end laptop I had. Hence, I needed to contain very limited data for training the model (2.7 million records as against 9 million records provided). Also, I wish there are more records in the validation set as against 1 million records.

Model can be further improved by applying Matrix factorization, singular value decomposition (SVD), and principal component analysis (PCA).
