MovieLens Project

Main objective of this model is to predict the rating value for a user-item combination.

```
#Loading librarries and data

library(tidyverse)
```

```
## Warning: package 'tidyverse' was built under R version 3.4.4
```

```
## -- Attaching packages --------------------------------------------------------------- tidyverse 1.2.1 -
-
```

```
## v ggplot2 3.2.0       v purrr   0.3.2
## v tibble  2.1.1       v dplyr   0.8.0.1
## v tidyr   0.8.3       v stringr 1.4.0
## v readr   1.3.1       v forcats 0.4.0
```

```
## Warning: package 'tibble' was built under R version 3.4.4
```

```
## Warning: package 'tidyr' was built under R version 3.4.4
```

```
## Warning: package 'readr' was built under R version 3.4.4
```

```
## Warning: package 'purrr' was built under R version 3.4.4
```

```
## Warning: package 'dplyr' was built under R version 3.4.4
```

```
## Warning: package 'stringr' was built under R version 3.4.4
```

```
## Warning: package 'forcats' was built under R version 3.4.4
```

```
## -- Conflicts ------------------------------------------------------------------ tidyverse_conflicts() -
-
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```
library(caret)
```

```
## Warning: package 'caret' was built under R version 3.4.4
```

```
## Loading required package: lattice
```

```
##
## Attaching package: 'caret'
```

```
## The following object is masked from 'package:purrr':
##
##     lift
```

```
library(data.table)
```

```
## Warning: package 'data.table' was built under R version 3.4.4
```

```
##
## Attaching package: 'data.table'
```

```
## The following objects are masked from 'package:dplyr':
##
##     between, first, last
```

```
## The following object is masked from 'package:purrr':
##
##     transpose
```

```
library(kableExtra)
```

```
## Warning: package 'kableExtra' was built under R version 3.4.4
```

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

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

```
library(lubridate)
```

```
## Warning: package 'lubridate' was built under R version 3.4.4
```

```
##
## Attaching package: 'lubridate'
```

```
## The following objects are masked from 'package:data.table':
##
##     hour, isoweek, mday, minute, month, quarter, second, wday,
##     week, yday, year
```

```
## The following object is masked from 'package:base':
##
##     date
```

```
library(DT)
library(RColorBrewer)
```

```
## Warning: package 'RColorBrewer' was built under R version 3.4.4
```

```
library(ggthemes)


setwd("C:/Users/arpitagupta/Documents/ml-10M100K")
edx <- readRDS("edx.rds")
validation <- readRDS("validation.rds")
```

Dataset Summary and Exploration

```
str(edx)
```

```
## 'data.frame':    9000055 obs. of  6 variables:
##  $ userId   : int  1 1 1 1 1 1 1 1 1 1 ...
##  $ movieId  : num  122 185 292 316 329 355 356 362 364 370 ...
##  $ rating   : num  5 5 5 5 5 5 5 5 5 5 ...
##  $ timestamp: int  838985046 838983525 838983421 838983392 838983392 838984474 838983653 838984885 838983
707 838984596 ...
##  $ title    : chr  "Boomerang (1992)" "Net, The (1995)" "Outbreak (1995)" "Stargate (1994)" ...
##  $ genres   : chr  "Comedy|Romance" "Action|Crime|Thriller" "Action|Drama|Sci-Fi|Thriller" "Action|Advent
ure|Sci-Fi" ...
```

edX dataset is made of 6 features for a total of about 9,000,055 observations. Below are the features and their characteristics:

quantitative features: userId, MovieID, timestamp

qualitative features: title, genres

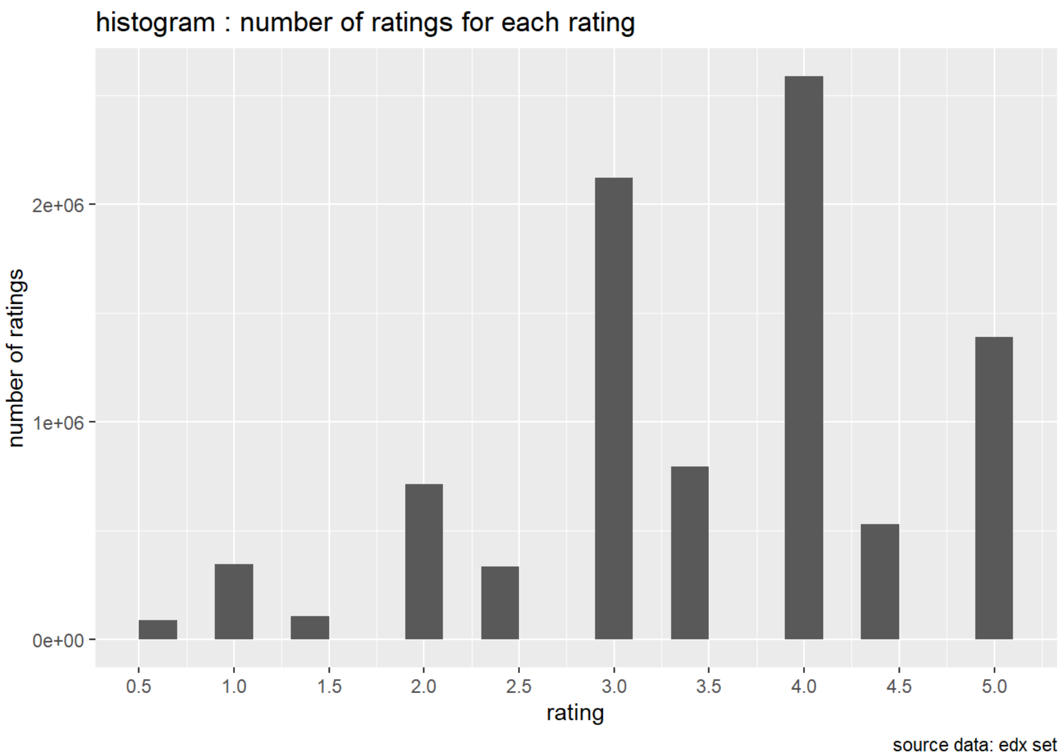
outcome,y: rating

```
summary(edx)
```

```
##      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
##
##
##
```

Lets see the proportion of ratings. Which rating is used most by the users

```
# histogram of ratings
ggplot(edx, aes(x= rating)) +
  geom_histogram( binwidth = 0.2) +
  scale_x_continuous(breaks=seq(0, 5, by= 0.5)) +
  labs(x="rating", y="number of ratings", caption = "source data: edx set") +
  ggtitle("histogram : number of ratings for each rating")
```



From the above chart we can see that no user gave 0 rating, 4 rating is most used.

Now lets see which genre and title is at top according to the rating

```
#Create the plot to identify which single genre is top rated.
top_genr <- edx %>% separate_rows(genres, sep = "\\|") %>%
  group_by(genres) %>%
  summarize(count = n()) %>%
  arrange(desc(count))

print(head(top_genr,5))
```

```
## # A tibble: 5 x 2
##   genres      count
##   <chr>       <int>
## 1 Drama     3910127
## 2 Comedy    3540930
## 3 Action    2560545
## 4 Thriller  2325899
## 5 Adventure 1908892
```

we can see that the âDramaâ genre has the top number of movies ratings, followed by the âComedyâ and the âActionâ genres.

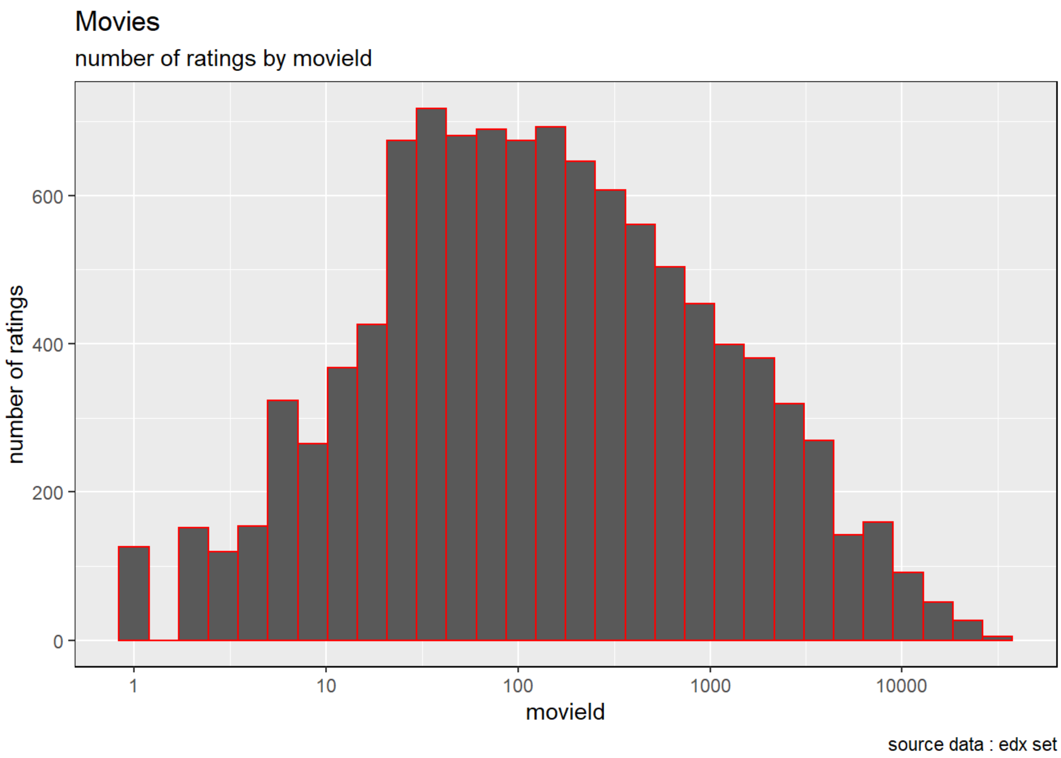
```
# Create dataframe to see which company is top rated

top_title <- edx %>%
  group_by(title,genres) %>%
  summarize(count=n()) %>%
  top_n(20,count) %>%
  arrange(desc(count))

print(head(top_title,10))
```

```
## # A tibble: 10 x 3
## # Groups:   title [10]
##    title                                       genres                    count
##    <chr>                                       <chr>                     <int>
##  1 Pulp Fiction (1994)                         Comedy|Crime|Drama        31362
##  2 Forrest Gump (1994)                         Comedy|Drama|Romance|W~ 31079
##  3 Silence of the Lambs, The (1991)            Crime|Horror|Thriller     30382
##  4 Jurassic Park (1993)                        Action|Adventure|Sci-F~ 29360
##  5 Shawshank Redemption, The (1994)            Drama                     28015
##  6 Braveheart (1995)                           Action|Drama|War          26212
##  7 Fugitive, The (1993)                        Thriller                  25998
##  8 Terminator 2: Judgment Day (1991)           Action|Sci-Fi             25984
##  9 Star Wars: Episode IV - A New Hope (a.k.a~ Action|Adventure|Sci-Fi 25672
## 10 Apollo 13 (1995)                            Adventure|Drama           24284
```

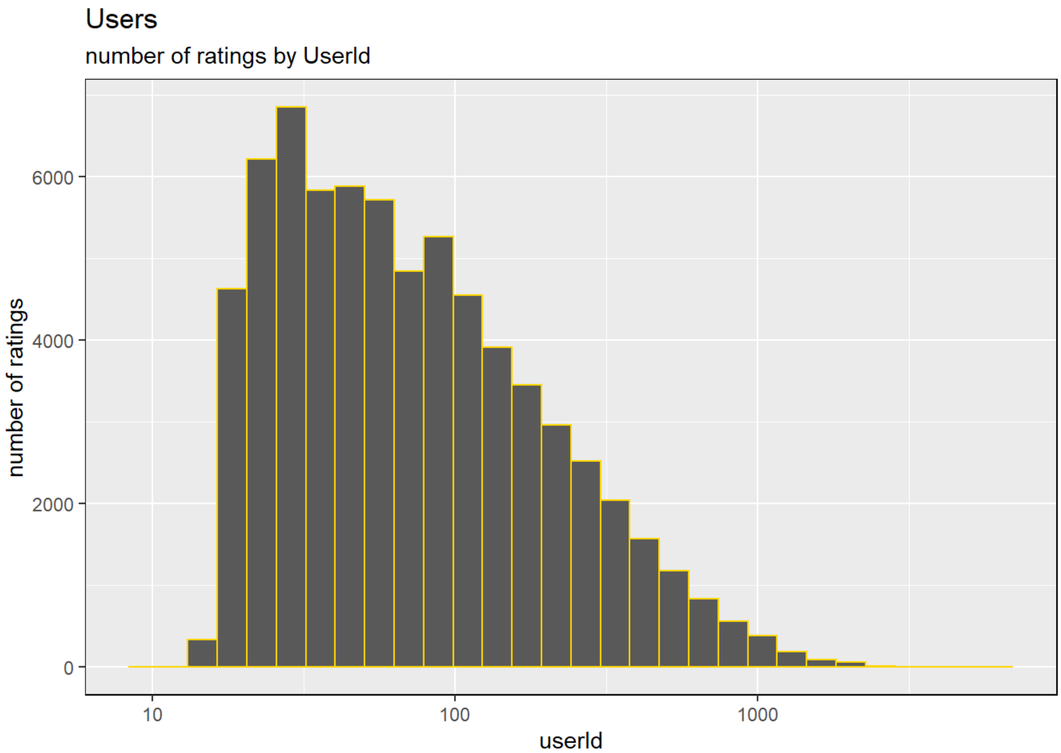
The movies which have the highest number of ratings are in the top genres categories : movies like Pulp fiction (1994), Forrest Gump(1994) or Jurrasic Park(1993) which are in the top 5 of movieâs ratings number , are part of the Drama, Comedy or Action genres.

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

```
# histogram of number of ratings by movieId

edx %>%
  count(movieId) %>%
  ggplot(aes(n)) +
  geom_histogram( bins=30, color = "red") +
  scale_x_log10() +
  ggtitle("Movies") +
  labs(subtitle  ="number of ratings by movieId",
       x="movieId" ,
       y="number of ratings",
       caption ="source data : edx set") +
  theme(panel.border = element_rect(colour="black", fill=NA))
```



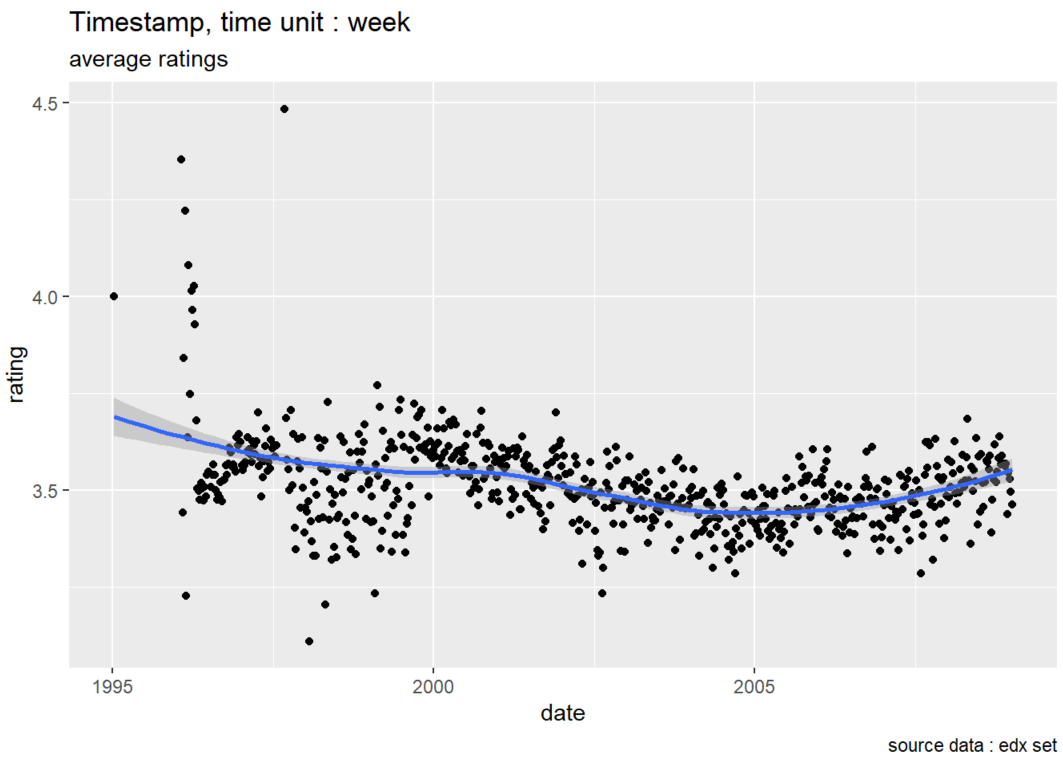

```
# histogram of number of ratings by userId


edx %>%
  count(userId) %>%
  ggplot(aes(n)) +
  geom_histogram( bins=30, color = "gold") +
  scale_x_log10() +
  ggtitle("Users") +
  labs(subtitle ="number of ratings by UserId",
       x="userId" ,
       y="number of ratings") +
  theme(panel.border = element_rect(colour="black", fill=NA))
```



From the above two graph we can see that some movies get rated more than others, and some users are more active than others at rating movies. This should presumably explain the presence of movies effects and users effects.

```
#we know that the edx set contains the timestamp variable which represents the time and data in which the ra
ting was provided. The units are seconds since January 1, 1970. with the as_datetime function in the lubrida
te package , we can have each timestamp in the right format . I then use the point geom to create scatterplo
t of y = average ratings vs x  = date ,  and smooth geom to aid the eyes in seeing patterns in the presence
of overplotting.


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",
       caption = "source data : edx set")
```

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



Analyzing the trend of the average ratings versus the date , we can notice that there is some evidence of a time effect in the plot, but there is not a strong effect of time.

Identify the Optimal Model to predict ratings

1)Regression Model

```r
#i define the RMSE function as:
RMSE <- function(true_ratings, predicted_ratings){
    sqrt(mean((true_ratings - predicted_ratings)^2))
  }


#a.movie effect

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

# i calculate b_i on the training set
movie_avgs <- edx %>%
  group_by(movieId) %>%
  summarize(b_i = mean(rating - mu))

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

rmse_model1 <- RMSE(validation$rating,predicted_ratings_bi)
rmse_model1
```

```
## [1] 0.9439087
```

```r
#b.movie + user effect

#i calculate b_u using the training set
user_avgs <- edx %>%
  left_join(movie_avgs, by='movieId') %>%
  group_by(userId) %>%
  summarize(b_u = mean(rating - mu - b_i))

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

rmse_model2 <- RMSE(validation$rating,predicted_ratings_bu)
rmse_model2
```

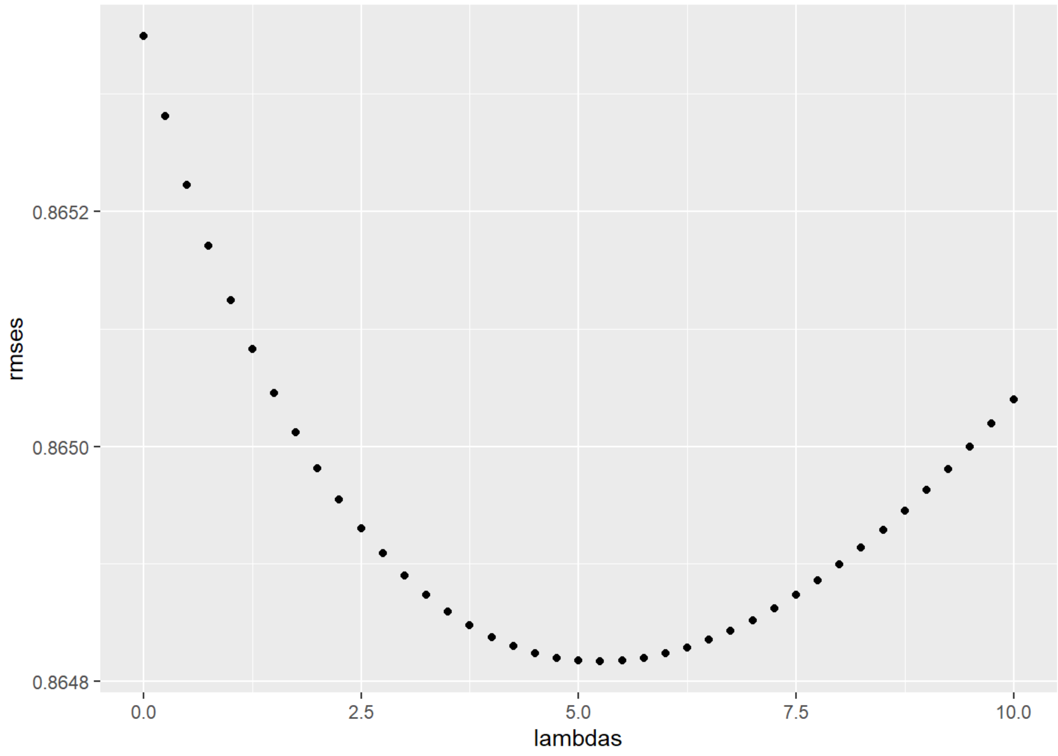
```
## [1] 0.8653488
```

```r
#c.movie + user + time effect

#i create a copy of validation set , valid, and create the date feature which is the timestamp converted to a datetime object  and  rounded by week.

valid <- validation
valid <- valid %>%
  mutate(date = round_date(as_datetime(timestamp), unit = "week"))

# i calculate time effects ( b_t) using the training set
temp_avgs <- edx %>%
  left_join(movie_avgs, by='movieId') %>%
  left_join(user_avgs, by='userId') %>%
  mutate(date = round_date(as_datetime(timestamp), unit = "week")) %>%
  group_by(date) %>%
  summarize(b_t = mean(rating - mu - b_i - b_u))

# predicted ratings
  predicted_ratings_bt <- valid %>%
  left_join(movie_avgs, by='movieId') %>%
  left_join(user_avgs, by='userId') %>%
  left_join(temp_avgs, by='date') %>%
  mutate(pred = mu + b_i + b_u + b_t) %>%
  .$pred

rmse_model3 <- RMSE(valid$rating,predicted_ratings_bt)
rmse_model3
```

```
## [1] 0.8652511
```

As we can see that there is not much improvement in RMSE when we added time therefore now we perform regularization using only the movie and user effects .

```r
#e. regularization

# lambda is a tuning parameter. We can use cross-validation to choose it

lambdas <- seq(0, 10, 0.25)

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

    mu_reg <- mean(edx$rating)

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

    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()+l))

    predicted_ratings_b_i_u <-
      validation %>%
      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(validation$rating,predicted_ratings_b_i_u))
  })


  qplot(lambdas, rmses)
```




```
#For the full model, the optimal  Î» is:

lambda <- lambdas[which.min(rmses)]
lambda
```

```
## [1] 5.25
```

```
rmse_model4 <- min(rmses)
rmse_model4
```

```
## [1] 0.864817
```

```
#summarize all the rmse on validation set for Linear regression models

rmse_results <- data.frame(methods=c("movie effect","movie + user effects","movie + user + time effects", "R
egularized Movie + User Effect Model"),rmse = c(rmse_model1, rmse_model2,rmse_model3, rmse_model4))

print(rmse_results)
```

```
##                                methods      rmse
## 1                         movie effect 0.9439087
## 2                 movie + user effects 0.8653488
## 3          movie + user + time effects 0.8652511
## 4 Regularized Movie + User Effect Model 0.8648170
```

Conclusion:

The main objective of this project was to build the algorithm to predict movie ratings for the 10M version of the Movielens data. Using the provided training set (edx) and validation set, we successively trained different linear regression models. The model evaluation performance through the RMSE (root mean squared error) showed that the Linear regression model with regularized effects on users and movies is the appropriate recommender systems to predict ratings on the validation set.