# HarvardX Movielens project

Luca Aceti

22/8/2022

## Introduction

The project is an application of machine learning techniques in the context of a recommendation system problem. Recommendation systems use ratings that users have given products to make specific recommendations. Netflix uses this system to predict the evaluation of movies by users. For the movielens project, we use the 10M version of the MovieLens dataset generated by the GroupLens research lab. After the data preparation and a preliminary exploratory analysis, different models are proposed, using step by step all the available covariates. We train the various models using edx input dataset and predict movie ratings using the validation set. RMSE will be used on the validation set to evaluate the accuracy of different models in order to choose the best one. Subsequently we try to test more sophisticated algoritms, in order to improve RMSE results. The goal of the project is to find a model that predict the rating of movies with a RMSE $<$ 0.86490.

## Data preparation

The first steps are related to the upload of the libraries, the download of the database and the partition of the dataset between training (edx) and validation test.

```
library(tidyverse)
```

```
## Warning: package 'tidyverse' was built under R version 4.0.2
```

```
## -- Attaching packages --------------------------------------- tidyverse 1.3.0 --
```

```
## v ggplot2 3.3.5     v purrr   0.3.4
## v tibble  3.1.6     v dplyr   1.0.8
## v tidyr   1.2.0     v stringr 1.4.0
## v readr   1.3.1     v forcats 0.5.0
```

```
## Warning: package 'ggplot2' was built under R version 4.0.5
```

```
## Warning: package 'tibble' was built under R version 4.0.5
```

```
## Warning: package 'tidyr' was built under R version 4.0.5
```

```
## Warning: package 'readr' was built under R version 4.0.2
```

```
## Warning: package 'dplyr' was built under R version 4.0.5
```

```
## Warning: package 'forcats' was built under R version 4.0.2
```

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

```
library(caret)
```

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

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

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

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

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

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

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

```
# MovieLens 10M dataset:
# https://grouplens.org/datasets/movielens/10m/
# http://files.grouplens.org/datasets/movielens/ml-10m.zip

dl <- tempfile()
download.file("https://files.grouplens.org/datasets/movielens/ml-10m.zip", dl)

ratings <- fread(text = gsub("::", "\t", readLines(unzip(dl, "ml-10M100K/ratings.dat"))),
                 col.names = c("userId", "movieId", "rating", "timestamp"))

movies <- str_split_fixed(readLines(unzip(dl, "ml-10M100K/movies.dat")), "\\::", 3)
colnames(movies) <- c("movieId", "title", "genres")

# if using R 4.0 or later:
movies <- as.data.frame(movies) %>% mutate(movieId = as.numeric(movieId),
                                           title = as.character(title),
                                           genres = as.character(genres))


movielens <- left_join(ratings, movies, by = "movieId")

# Validation set will be 10% of MovieLens data
set.seed(1, sample.kind="Rounding") # if using R 3.5 or earlier, use `set.seed(1)`
```

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

```
test_index <- createDataPartition(y = movielens$rating, times = 1, p = 0.1, list = FALSE)
edx <- movielens[-test_index,]
temp <- movielens[test_index,]

# Make sure userId and movieId in validation set are also in edx set
validation <- temp %>%
  semi_join(edx, by = "movieId") %>%
  semi_join(edx, by = "userId")

# Add rows removed from validation set back into edx set
removed <- anti_join(temp, validation)
```

```
## Joining, by = c("userId", "movieId", "rating", "timestamp", "title", "genres")
```

```
edx <- rbind(edx, removed)

rm(dl, ratings, movies, test_index, temp, movielens, removed)
```

## Exploratory Data Analysis

First of all, a preliminary analysis of the data is necessary, in order to have a picture of structure of the database and a first evidence of the effect of each variables on the rating. The dataset is composed of 6 columns: an outcome (rating), quantitative (userId, movieId, timestamp) and qualitative (title and genres) features. Timestamps represent date and time of the rating, seconds since midnight UTC January 1, 1970. The dataset presents 69,878 unique users and 10,677 movies.

```
str(edx)
```

```
## Classes 'data.table' and '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 83
##  $ title    : chr  "Boomerang (1992)" "Net, The (1995)" "Outbreak (1995)" "Stargate (1994)" ...
##  $ genres   : chr  "Comedy|Romance" "Action|Crime|Thriller" "Action|Drama|Sci-Fi|Thriller" "Action|A
##  - attr(*, ".internal.selfref")=<externalptr>
```

```
edx %>% select(-genres) %>% summary()
```

```
##      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
```

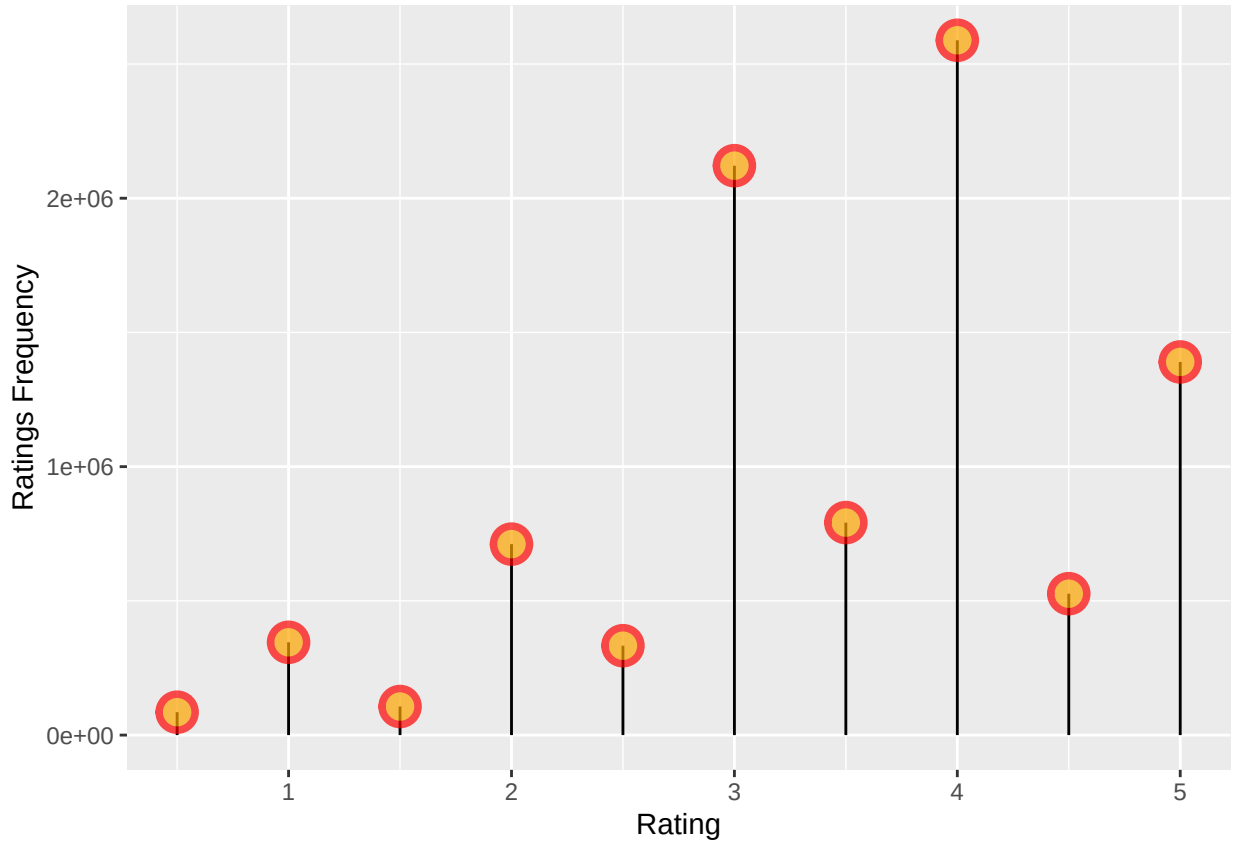
```
##  Length:9000055
##  Class :character
##  Mode  :character
##
##
##
```

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

```
##   n_users n_movies
## 1   69878    10677
```

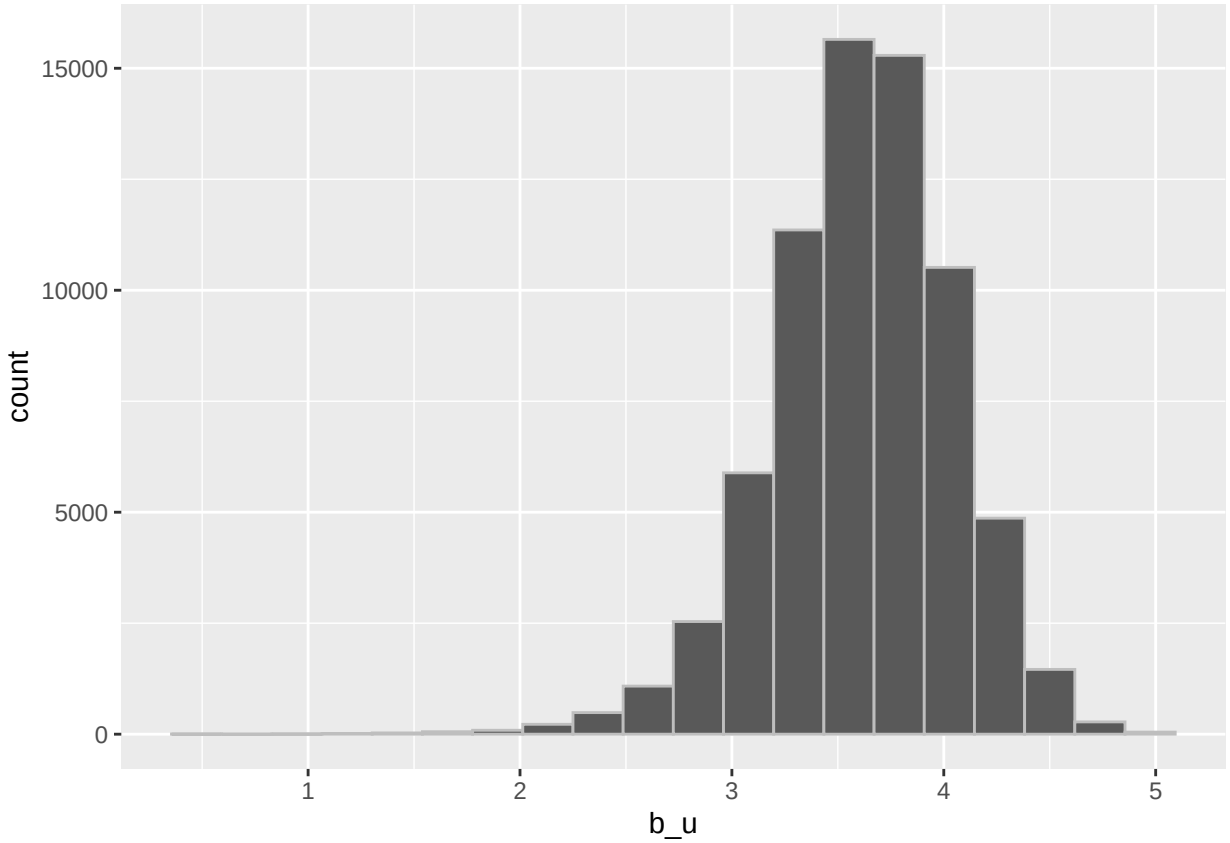
The distribution of the rating put in evidence the prevalence of intermediate valuation (3 and 4 stars are the more common valuation), with whole numbers used more often.

```
edx %>%
  group_by(rating) %>%
  summarize(count = n()) %>%
  ggplot(aes(x = rating, y = count)) +
  geom_segment( aes(x=rating, xend=rating, y=0, yend=count)) +
  geom_point( size=5, color="red", fill=alpha("orange", 0.3), alpha=0.7, shape=21, stroke=2) +
  xlab("Rating") +
  ylab("Ratings Frequency")
```



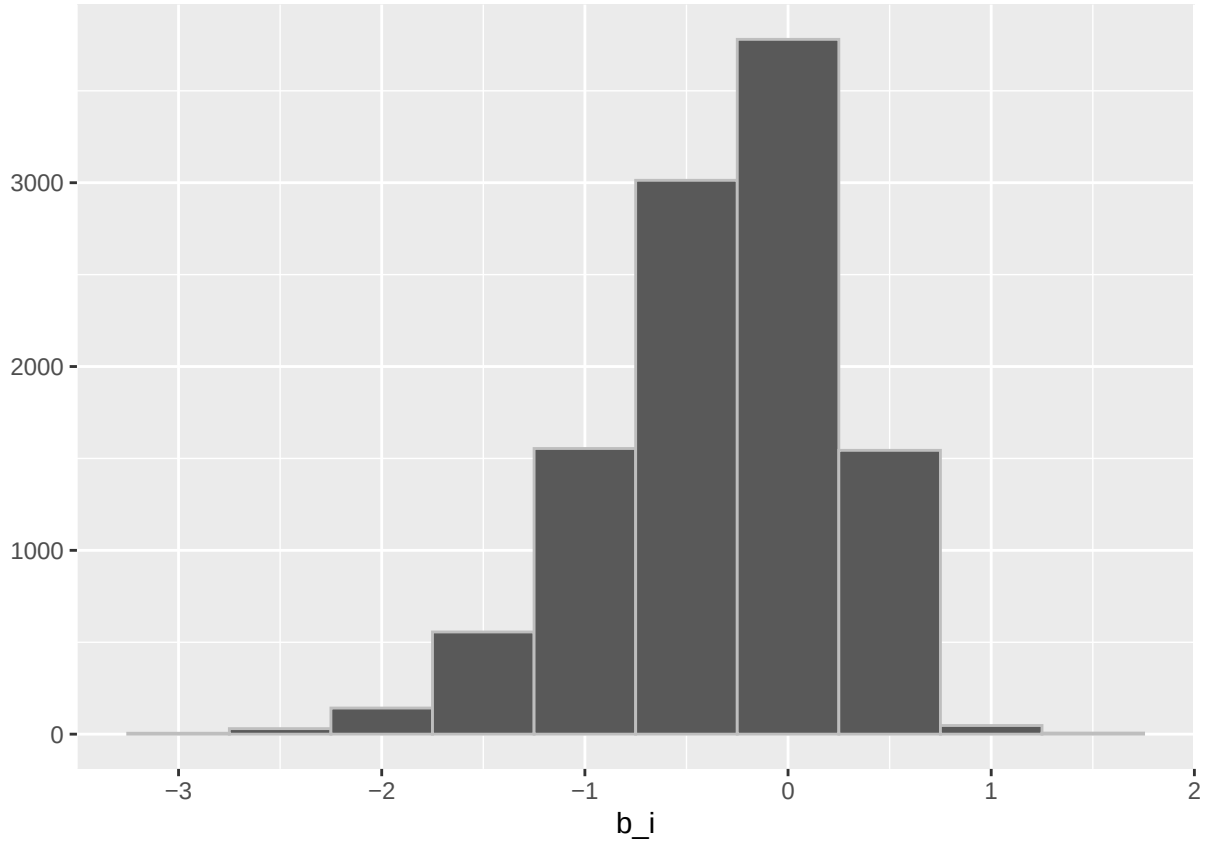

The next step is to understand the different effect of the variables on the rating.The assumption is that user and movie affect obviously the rating, but also time of the rating and movie genre could have an impact. User effect is shown with the following graph.

```
edx %>%
  group_by(userId) %>%
  summarize(b_u=mean(rating)) %>%
  ggplot(aes(b_u))+
  geom_histogram(bins=20,color="grey")
```




Likewise we do for movie effect, showing also the mean rating of the top judged movies.

```
mu<-mean(edx$rating)
movie_avg<-edx %>%
  group_by(movieId) %>%
  summarize(b_i=mean(rating-mu))
qplot(b_i,data=movie_avg,bins=10,color=I("grey"))
```



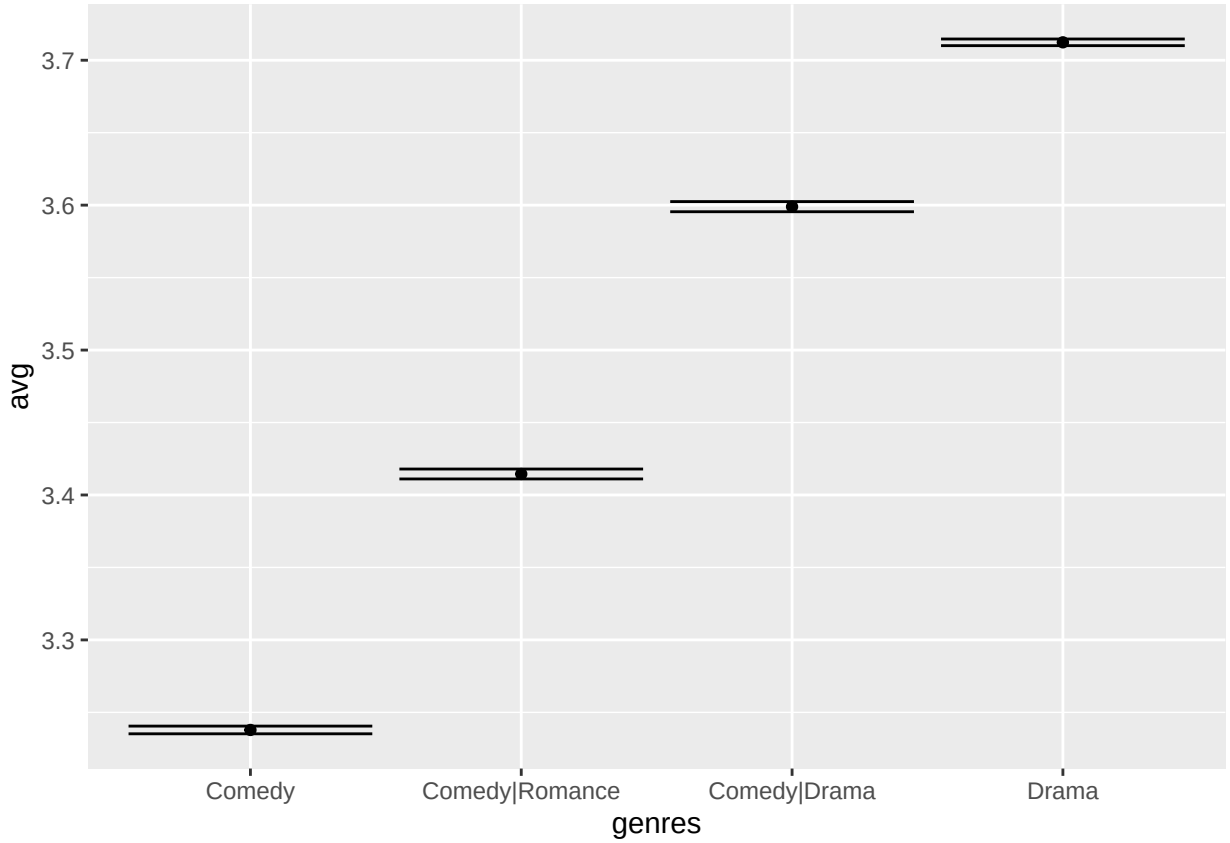

```
movie_top<-edx %>% group_by(title) %>% summarize(n=n(),avg=mean(rating)) %>% arrange(desc(n)) %>%
  top_n(10,n)
knitr::kable(movie_top)
```

| title | n | avg |
|---|---|---|
| Pulp Fiction (1994) | 31362 | 4.154789 |
| Forrest Gump (1994) | 31079 | 4.012822 |
| Silence of the Lambs, The (1991) | 30382 | 4.204101 |
| Jurassic Park (1993) | 29360 | 3.663522 |
| Shawshank Redemption, The (1994) | 28015 | 4.455131 |
| Braveheart (1995) | 26212 | 4.081852 |
| Fugitive, The (1993) | 25998 | 4.009155 |
| Terminator 2: Judgment Day (1991) | 25984 | 3.927859 |
| Star Wars: Episode IV - A New Hope (a.k.a. Star Wars) (1977) | 25672 | 4.221311 |
| Apollo 13 (1995) | 24284 | 3.885789 |

In the graph it is considered also the effect of the more frequent genres, for graphical reasons only genres with a number of ratings > 300.000 appears.The genre definition is not unique, but it can be a combination of different categories. The aim is to bring out a correlation between the genre and the rating of the movie.

```
edx %>%
  group_by(genres) %>%
  summarize(n = n(),avg = mean(rating),se=sd(rating)/sqrt(n)) %>% filter(n>300000) %>%
```

```
  mutate(genres=reorder(genres,avg)) %>%
  ggplot(aes(x=genres,y=avg,ymin=avg-2*se,ymax=avg+2*se))+geom_point()+geom_errorbar()
```




```
genre_top<- edx %>%
  group_by(genres) %>%
  summarize(n = n(),avg=mean(rating)) %>% arrange(desc(n)) %>% top_n(10,n)

knitr::kable(genre_top)
```

| genres | n | avg |
|---|---|---|
| Drama | 733296 | 3.712364 |
| Comedy | 700889 | 3.237858 |
| Comedy\|Romance | 365468 | 3.414486 |
| Comedy\|Drama | 323637 | 3.598961 |
| Comedy\|Drama\|Romance | 261425 | 3.645824 |
| Drama\|Romance | 259355 | 3.605471 |
| Action\|Adventure\|Sci-Fi | 219938 | 3.507407 |
| Action\|Adventure\|Thriller | 149091 | 3.434101 |
| Drama\|Thriller | 145373 | 3.446345 |
| Crime\|Drama | 137387 | 3.947135 |

```
genre_effect <- edx %>%
  group_by(genres) %>%
  summarise(c = mean(rating - mu))
```

Also the relation between time and rating is analyzed, even if it seems to be less significant than the other covariates.

```
library(lubridate)
```

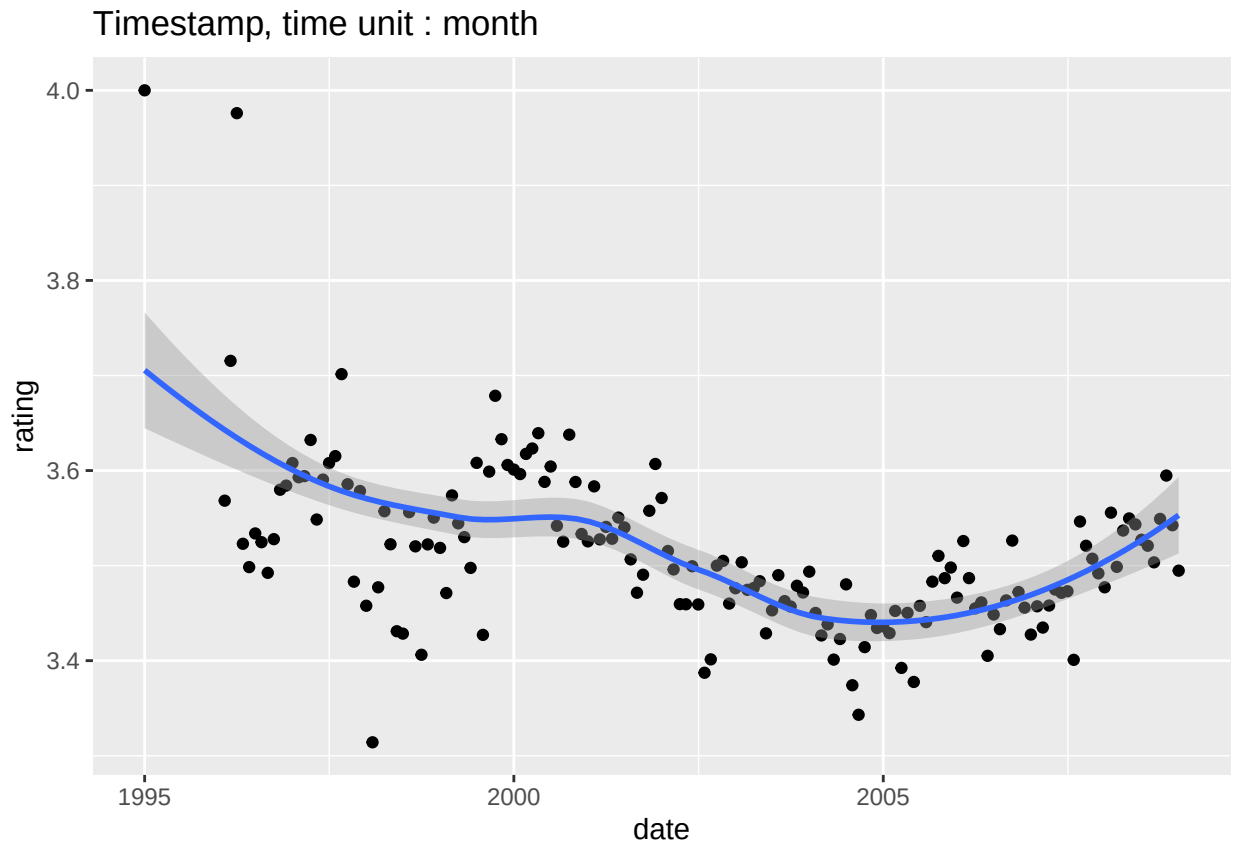
```
##
## Attaching package: 'lubridate'
```

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

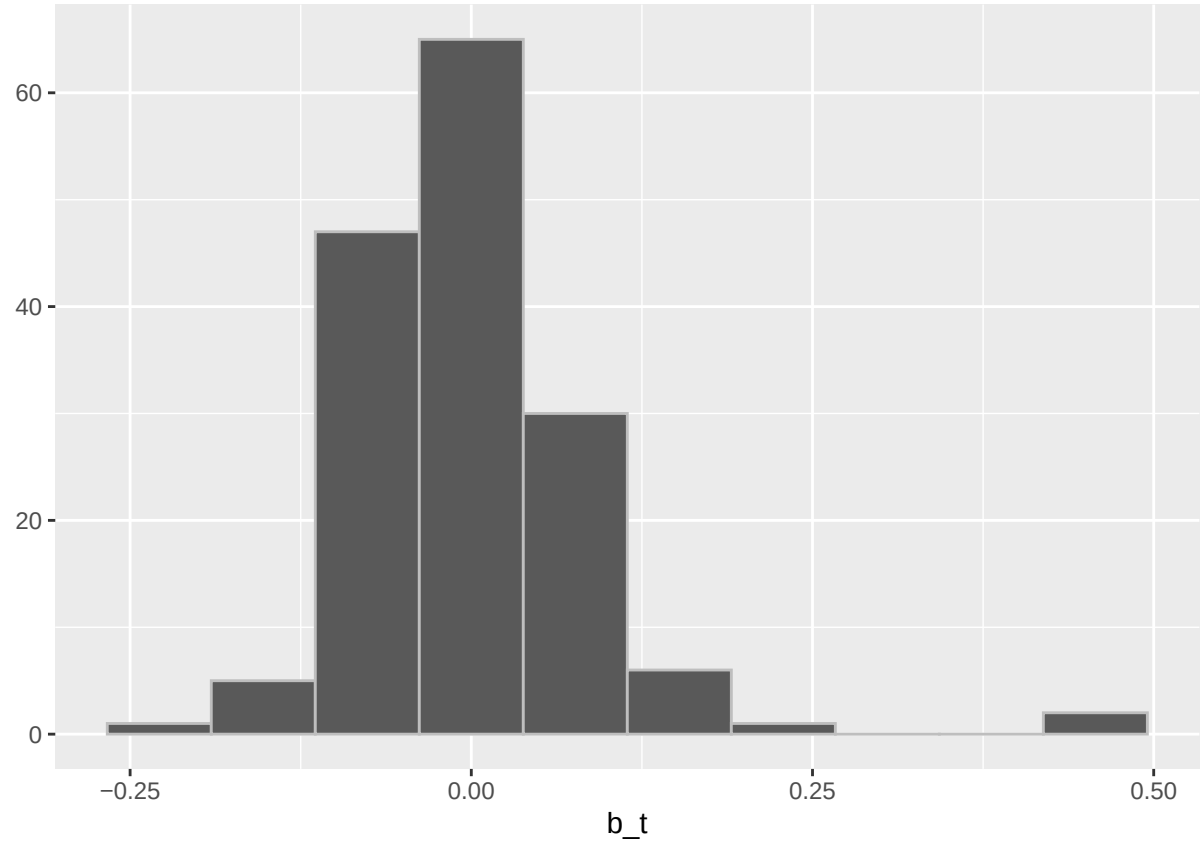
```
## The following objects are masked from 'package:base':
##
##     date, intersect, setdiff, union
```

```
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")
```

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




```
time_avg<-edx %>%
  mutate(date = round_date(as_datetime(timestamp), unit = "month")) %>%
  group_by(date) %>%
  summarize(b_t = mean(rating-mu))

qplot(b_t,data=time_avg,bins=10,color=I("grey"))
```




## Model Analysis

The aim of this section is to analyze different models, starting from a simple one and adding one predictor step by step in order to see the improvement on the RMSE result.The models are the following:

1. Movie effect

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

```
#1.movie effect

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

#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
```

2. Movie effect + User effect

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

```
#2.movie + user effect

#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
```

3. Movie effect + User effect + Time effect

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

```
#3.movie + user + time effect

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

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

#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 = "month")) %>%
  group_by(date) %>%
  summarize(b_t = mean(rating - mu - b_i - b_u))

#predicted ratings
predicted_ratings_bt <- validate %>%
  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
```

4. Movie effect + User effect + Time effect + Genre effect

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

```
#4.movie + user + time effect + genre effect

#calculate genre effect (b_g) using the training set
genre_avgs <- edx %>%
  left_join(movie_avgs, by='movieId') %>%
```

```
  left_join(user_avgs, by='userId') %>%
  mutate(date = round_date(as_datetime(timestamp), unit = "month")) %>%
  left_join(temp_avgs,by='date') %>%
  group_by(genres) %>%
  summarize(b_g = mean(rating - mu - b_i - b_u - b_t))

#predicted ratings
predicted_ratings_bg <- validate %>%
  left_join(movie_avgs, by='movieId') %>%
  left_join(user_avgs, by='userId') %>%
  left_join(temp_avgs, by='date') %>%
  left_join(genre_avgs,by='genres') %>%
  mutate(pred = mu + b_i + b_u + b_t + b_g) %>%
  .$pred
```

## RMSE results

Comparing different models we see that the best result is linked to the model that include all the predictors, but the improvement of the RMSE related to time and genre effect seems to be very poor, even if the initial goal of the project is almost reached.

```
#Root Mean Square Error Loss Function
RMSE <- function(true_ratings, predicted_ratings){
  sqrt(mean((true_ratings - predicted_ratings)^2))
}

# Model RMSE results - comparison
rmse_model1 <- round(RMSE(validation$rating,predicted_ratings_bi),6)
rmse_model2 <- round(RMSE(validation$rating,predicted_ratings_bu),6)
rmse_model3 <- round(RMSE(validation$rating,predicted_ratings_bt),6)
rmse_model4 <- round(RMSE(validation$rating,predicted_ratings_bg),6)
rmse_results<-data.frame(method=c("movie effect","movie + user effect",
                                  "movie + user + time effect",
                                  "movie + user + time + genre effect"),
                                  rmse=c(rmse_model1,rmse_model2,rmse_model3,rmse_model4))

knitr::kable(rmse_results)
```

| method | rmse |
|---|---|
| movie effect | 0.943909 |
| movie + user effect | 0.865349 |
| movie + user + time effect | 0.865315 |
| movie + user + time + genre effect | 0.864914 |

## Regularization model

The idea behind regularization is to add a penalty for large values of bias effect (bi, bu for example) to the sum of squares equation that we minimize. The reason is that large value of bias come from small sample size, due to low number of rating. This can affect negatively on the model accuracy. We introduce a penalty

term lambda to tune RMSE. The tuning parameter is chosen by using cross-validation. This is the formula to be minimized:

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

We decide to consider only movie and user effect, because models that incorporate time and genres don't improve RMSE in a meaningful way. Furthermore, in order to use cross-validation, we need an additional partition of the training set, into 2 splits: the first (train_reg) for the calculation of movie and user effect and the second (test_reg) for the selection of the optimal lambda.

```
set.seed(7, sample.kind="Rounding") # if using R 3.5 or earlier, use `set.seed(7)`
```

```
## Warning in set.seed(7, sample.kind = "Rounding"): non-uniform 'Rounding' sampler
## used
```

```
test_valid <- createDataPartition(y = edx$rating, times = 1, p = 0.1, list = FALSE)
train_reg <- edx[-test_valid,]
temp <- edx[test_valid,]

# Make sure userId and movieId in test_reg set are also in train_reg set
test_reg <- temp %>%
  semi_join(train_reg, by = "movieId") %>%
  semi_join(train_reg, by = "userId")

# Add rows removed from test_reg set back into train_reg set
removed <- anti_join(temp, test_reg)
```

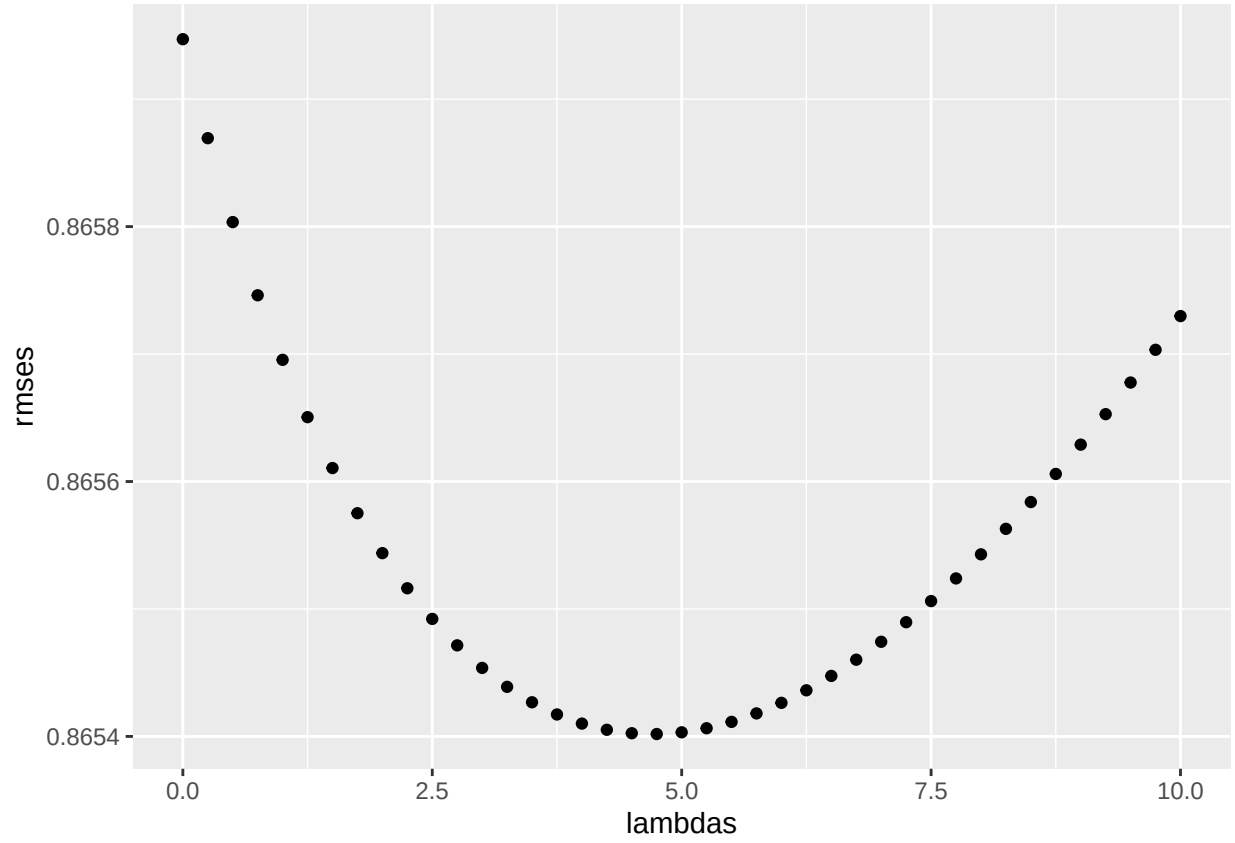
```
## Joining, by = c("userId", "movieId", "rating", "timestamp", "title", "genres")
```

```
train_reg <- rbind(train_reg, removed)
```

After this, it is possible to run the code for the calculation of the optimal lambda.

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

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

  #Calculate the mean of ratings from the training set for cross-validation
  mu_reg <- mean(train_reg$rating)

  #Adjust mean by movie effect and penalize low number of ratings
  b_i_reg <- train_reg %>%
    group_by(movieId) %>%
    summarize(b_i_reg = sum(rating - mu_reg)/(n()+l))

  #ajdust mean by user and movie effect and penalize low number of ratings
  b_u_reg <- train_reg %>%
    left_join(b_i_reg, by="movieId") %>%
    group_by(userId) %>%
    summarize(b_u_reg = sum(rating - b_i_reg - mu_reg)/(n()+l))

  #predict ratings in the test set (test_reg) to derive optimal penalty value 'lambda'
```

```
  predicted_ratings <-
    test_reg %>%
    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_reg$rating, predicted_ratings))
})

qplot(lambdas, rmses)
```




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

Finally we calculate the RMSE for the optimal lambda on the validation test and compare these model to the previous ones.

```
pred_reg <- sapply(lambda,function(l){

  #Derive the mean from the training set
  mu <- mean(edx$rating)

  #Calculate movie effect with optimal lambda
  b_i <- edx %>%
    group_by(movieId) %>%
```

```
    summarize(b_i = sum(rating - mu)/(n()+l))

  #Calculate user effect with optimal lambda
  b_u <- edx %>%
    left_join(b_i, by="movieId") %>%
    group_by(userId) %>%
    summarize(b_u = sum(rating - b_i - mu)/(n()+l))

  #Predict ratings on validation set
  predicted_ratings <-
    validation %>%
    left_join(b_i, by = "movieId") %>%
    left_join(b_u, by = "userId") %>%
    mutate(pred = mu + b_i + b_u) %>%
    .$pred #validation

  return(predicted_ratings)

})

rmse_model5 <- round(RMSE(validation$rating,pred_reg),6)
new_model<-c("Regularized Movie + User effect Model",rmse_model5)
rmse_results<-rbind(rmse_results,new_model)
knitr::kable(rmse_results)
```

| method | rmse |
|---|---|
| movie effect | 0.943909 |
| movie + user effect | 0.865349 |
| movie + user + time effect | 0.865315 |
| movie + user + time + genre effect | 0.864914 |
| Regularized Movie + User effect Model | 0.86482 |

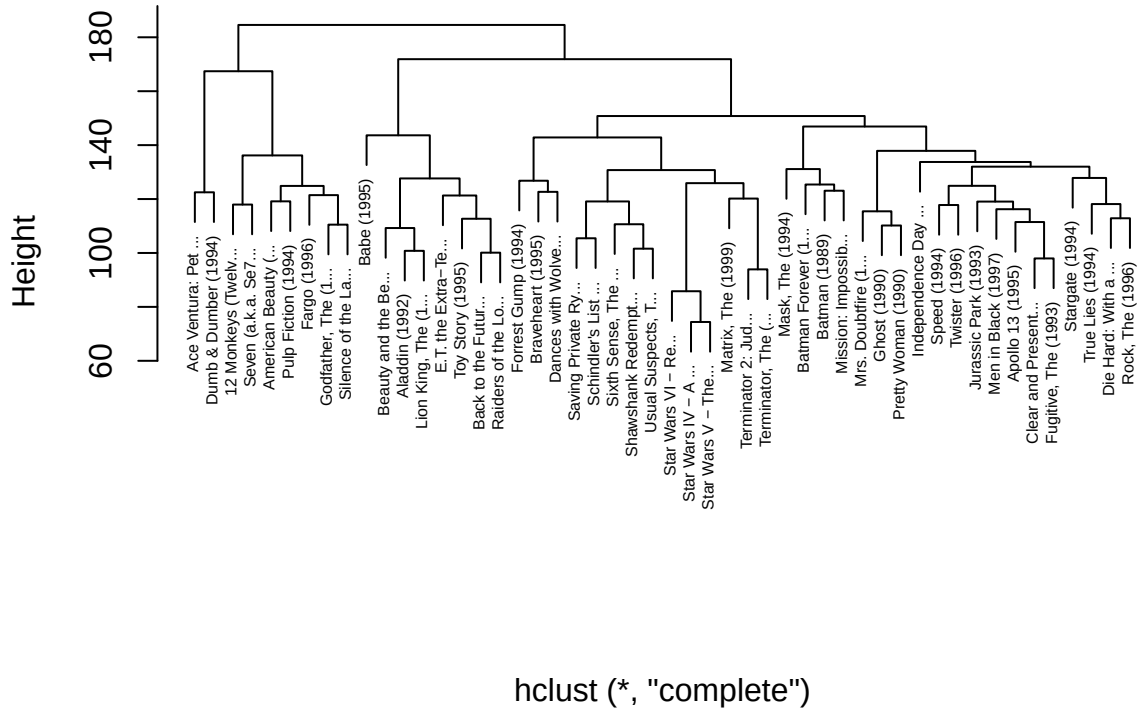
The result in RMSE show a little improvement: this indicates that the model doesn't catch completely the structure of the data.

## Matrix factorization - Recosystem

With the previous algoritms we have produced models that take into account the different effect of the covariates on the rating, but no assumption is done about the correlation between them. Using unsupervised methods we have the possibility to do a further step in order to understand better the structure of the data. In particular we decide to use hierarchical clustering, that divide iteratively movies into groups, computing the the distance between each pair of movies. The resulting groups are shown through a dendogram, then we have decided to limit to 10 groups of movies and to present some examples of them. The split intuitively seems to make sense. The correlation appears clearly using the heatmap function.

```
# unsupervised methods

top<-edx %>%
  group_by(movieId) %>%
  summarize(n=n(),title=first(title)) %>%
```

```
  top_n(50,n) %>%
  pull(movieId)

x<-edx %>%
  filter(movieId %in% top) %>%
  group_by(userId) %>%
  filter(n() >=25) %>%
  ungroup() %>%
  select(title,userId,rating) %>%
  spread(userId,rating)

row_names<-str_remove(x$title,": Episode") %>% str_trunc(20)
x<-x[,-1] %>% as.matrix()
x<-sweep(x,2,colMeans(x,na.rm=TRUE))
x<-sweep(x,1,rowMeans(x,na.rm=TRUE))
rownames(x)<-row_names

# Hierarchical cluster
d<-dist(x)
h<-hclust(d)
plot(h,cex=0.5,main="",xlab="")
```




```
# limit the groups to 10
groups<-cutree(h,k=10)
names(groups)[groups==3]
```

```
## [1] "Aladdin (1992)"       "Back to the Futur..." "Beauty and the Be..."
## [4] "E.T. the Extra-Te..." "Lion King, The (1..." "Raiders of the Lo..."
## [7] "Toy Story (1995)"
```

```
names(groups)[groups==10]
```

```
##  [1] "Matrix, The (1999)"   "Saving Private Ry..." "Schindler's List ..."
##  [4] "Shawshank Redempt..." "Sixth Sense, The ..." "Star Wars IV - A ..."
##  [7] "Star Wars V - The..." "Star Wars VI - Re..." "Terminator 2: Jud..."
## [10] "Terminator, The (..." "Usual Suspects, T..."
```

```
names(groups)[groups==5]
```

```
##  [1] "Apollo 13 (1995)"     "Clear and Present..." "Die Hard: With a ..."
##  [4] "Fugitive, The (1993)" "Independence Day ..." "Jurassic Park (1993)"
##  [7] "Men in Black (1997)"  "Rock, The (1996)"     "Speed (1994)"
## [10] "Stargate (1994)"      "True Lies (1994)"     "Twister (1996)"
```

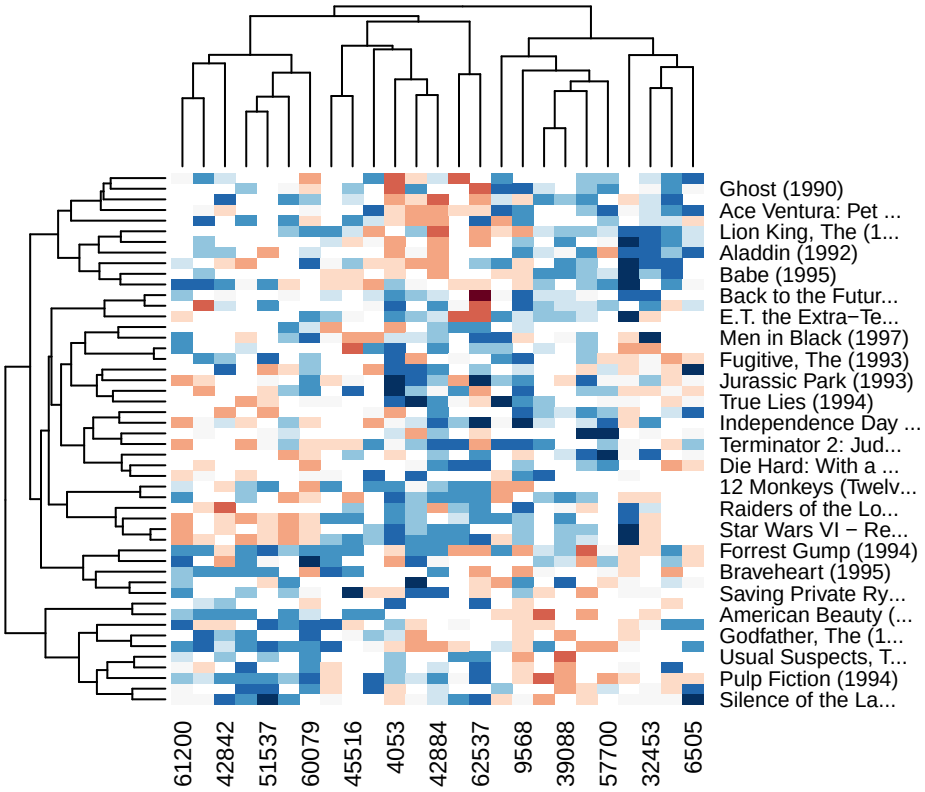
```
# heatmap representation
library(matrixStats)
```

```
## Warning: package 'matrixStats' was built under R version 4.0.5
```

```
##
## Attaching package: 'matrixStats'
```

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

```
sds<-colSds(x,na.rm=TRUE)
o<-order(sds,decreasing=TRUE)[1:25]
heatmap(x[,o],col=RColorBrewer::brewer.pal(11,"RdBu"))
```




From the previous analysis, it appears clear that a further improvement of the model is necessary in order to take into consideration the structure of correlation among groups of movies. The final model should be represented in this way:

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

where:

$$r_{u,i} = p_{u,1}q_{1,i} + p_{u,2}q_{2,i} + ... + p_{u,m}q_{m,i}$$

In order to implement this approach, I try to find an useful package in R. The recommenderlab package is appropriate for this issue, because several algoritms can be used. But I find also another package recosystem, the use of which is well explained in this Yixuan's blog post: https://statr.me/2016/07/recommender-system-using-parallel-matrix-factorization/. Therefore I try to use the available code chunk on edx data, using the suggested tuning parameters and then calculating the RMSE on the validation data set. The result improve a lot, in comparison with the previous models. The citated blog suggest also to tune the parameters, but I do not incorporated the results in the project file because the code takes a few minutes to run on my personal PC. Another comment to be made regards the reproducibility of the analysis. The blog underlines that if the parameter nthread is greater than 1 then the training result is NOT guaranteed to be reproducible, even if a random seed is set.

```
library(recosystem)
```

```
## Warning: package 'recosystem' was built under R version 4.0.5
```

```
# data preparation
train_data<-with(edx,data_memory(user_index = userId,
                                 item_index = movieId,
```

```
                                        rating=rating))
validation_data<-with(validation,data_memory(user_index=userId,
                                             item_index=movieId,
                                             rating=rating))
# training the model
r=Reco()
r$train(train_data, opts = list(dim = 20,
                                costp_l1 = 0, costp_l2 = 0.01,
                                costq_l1 = 0, costq_l2 = 0.01,
                                niter = 10,
                                nthread = 4))
```

```
## iter      tr_rmse          obj
##    0       0.9580   9.0072e+06
##    1       0.8538   7.3756e+06
##    2       0.8006   6.6297e+06
##    3       0.7716   6.2450e+06
##    4       0.7558   6.0414e+06
##    5       0.7458   5.9137e+06
##    6       0.7391   5.8284e+06
##    7       0.7340   5.7636e+06
##    8       0.7302   5.7154e+06
##    9       0.7271   5.6768e+06
```

```
# prediction on validation data set
pred = r$predict(validation_data, out_memory())

#RMSE calculation
rmse_model6 <- round(RMSE(validation$rating,pred),6)
new_model2<-c("Recosystem model",rmse_model6)
rmse_results<-rbind(rmse_results,new_model2)
knitr::kable(rmse_results)
```

| method | rmse |
|---|---|
| movie effect | 0.943909 |
| movie + user effect | 0.865349 |
| movie + user + time effect | 0.865315 |
| movie + user + time + genre effect | 0.864914 |
| Regularized Movie + User effect Model | 0.86482 |
| Recosystem model | 0.791295 |

```
# The following code takes a few minutes to run
# opts_tune = r$tune(train_data,
#                                 opts = list(dim      = c(10, 20, 30),
#                                 costp_l2 = c(0.01, 0.1),
#                                 costq_l2 = c(0.01, 0.1),
#                                 costp_l1 = 0,
#                                 costq_l1 = 0,
#                                 lrate    = c(0.01, 0.1),
#                                 nthread  = 4,
#                                 niter    = 10,
```

```
#                                 verbose  = TRUE))
# r$train(train_data, opts = c(opts_tune$min,
#                                 niter = 100, nthread = 4))
```

## Conclusion

The recommendation system is of course a challenge to the data science community, but also to the practitioners that want to improve their ability in data analysis. Aiming to be within this group, I try to use what I've learned during the previous courses for the development of the project. The improvement of the models has been possible incorporating step by step the different effect of covariates. The model that include all the available covariates (movieId,userId,time and genre) allows almost to reach the initial goal in term of RMSE ($<$0.86490), but a further analysis on the structure of the data suggests other possible models. First of all the movie with low number of ratings can produce large bias with negative effects on model accuracy, therefore it is necessary to use regularization models in order to solve this issue. Furthermore the unsupervised methods point out correlations among group of movies and the needs of Matrix Factorizaton techniques. Using the recosystem package and Yixuan's blog explanations, it has been possible to experiment these techniques on the project data and to improve RMSE($<$0.8). I think that additional improvements of the results are possible using recommenderlab package, but it is necessary an in-depth study of the bibliography on the subject in order to choose the next steps to do.