HarvardX-PH125.9x - MovieLens Report

Camilo Lillo

29/5/2021

1. Introduction

The aims of this study is to create a movie recommendation system using the MovieLens dataset. The develop of the movieLens recommendation system consists in an exploratory data analysis, formulation and develop machine learning models on the rating of a user who watches a movie.

This document is organized as follows: First, we repeat the Rcodes to generate in the training and test set available in the course. Second, we develop an exploratory data analysis. Third, we formulated and evaluated four predictive models and finally concluded our work.

2. Playback of edx Rcodes

Below are the exd Rcodes:

```
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(data.table)){
  install.packages("data.table", repos = "http://cran.us.r-project.org")}
if(!require(ggplot2)){
  install.packages("ggplot2",    repos = "http://cran.us.r-project.org")}
if(!require(stringi)){
  install.packages("stringi",    repos = "http://cran.us.r-project.org")}
if(!require(scales)){
  install.packages("scales",     repos = "http://cran.us.r-project.org")}
if(!require(dismo)){
  install.packages("dismo",      repos = "http://cran.us.r-project.org")}
if(!require(dplyr)){
  install.packages("dplyr",      repos = "http://cran.us.r-project.org")}
if(!require(recosystem)){
  install.packages("recosystem", repos = "http://cran.us.r-project.org")}

library(tidyverse)
library(caret)
library(data.table)
library(ggplot2)
library(stringi)
library(scales)
library(dismo)
library(dplyr)
```

```
library(recosystem)

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

dl <- tempfile()
download.file("http://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")

# I use 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)`
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)
edx     <- rbind(edx, removed)

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

edx0 = edx
```

Note that we add some packages in this step. In addition, we copy de edx data frame.

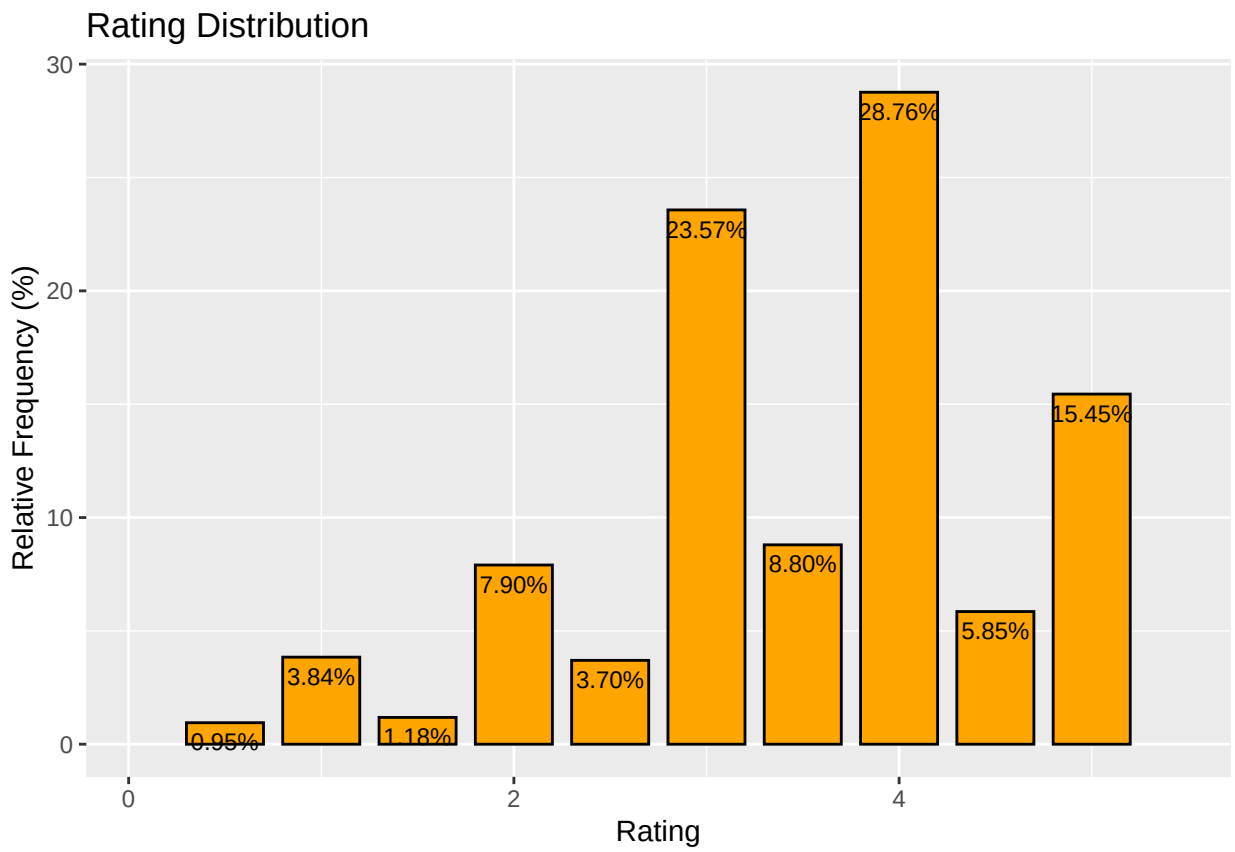
3. Exploratory data analysis

The response variable is presented below, denoted by R, which corresponds to the rating of the movie provided by the user. The support variable correspond to $a/2$ values, with $a = 1, 2, \ldots, 10$. The rating distribution is:

```
fig1 <- edx %>% group_by(rating) %>% summarize(total = length(rating)) %>%
  ggplot(aes(x = rating, y = 100*total/(sum(total)))) +
  geom_bar(stat="identity",
```

```
           position=position_dodge(), width = 0.4,
           color = "black", fill = "orange") +
  geom_text(aes(label=percent(total/(sum(total)) %>% round(1))), vjust=1.6, color="black",
            position = position_dodge(0.9), size = 3) +

  labs(title="Rating Distribution",
       x="Rating", y = "Relative Frequency (%)")
fig1
```



Note that the rating distribution is a mixture of two distributions; the integers and non-integers distributions. The rating distribution have negative asymmetric. The total number of data 9,000,055. Note that it is more likely to belong to the integers distribution. These two populations are likely to be difficult to separate when modeling. Thus, we will model without considering clusters.

For other hand, an interesting analysis corresponds to observing the distribution of average rating by user. If we assume that the distribution of the rating of each user is Gaussian, it is possible to consider confidence intervals (CI) for the average rating of each user (denoted by b_u), denoted by:

$$CI(b_u)_{95\%} = \tilde{b}_u \pm 1.96\frac{\tilde{\sigma}_u}{\sqrt{n_u}},$$

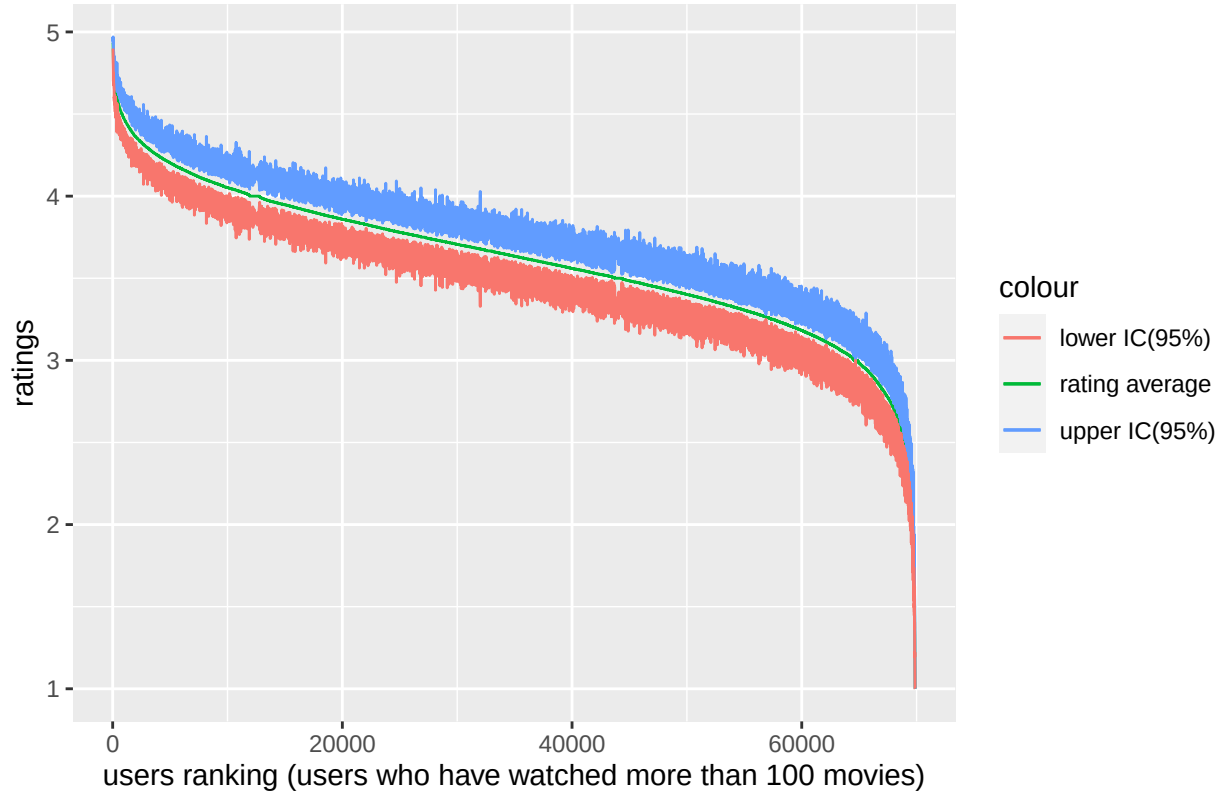
where $\tilde{b}_u$ is the average rating of the user u. In addition, $\tilde{\sigma}_u$ and n_u are the estimated standard deviation of the rating by user and the number of movies that the user rated, respectively.

The following graph corresponds to the distribution of the average rating with respect to the user and its respective confidence interval of 95% (userId variable). To robustness the analysis, users who had watched at least 100 movies were used.

```r
tab_userId = edx %>% group_by(userId) %>%
  summarize(avg = mean(rating),
            sds = sd(rating),
            cv  = 100*sd(rating)/mean(rating),
            len = length(rating))

tab_userId = tab_userId[order(-tab_userId$avg),]
tab_userId = tab_userId %>% mutate(sds = ifelse(is.na(sds), 0, sds),
                                   cv  = ifelse(is.na(cv), 0, cv))
tab_userId$rank = c(1:NROW(tab_userId))
tab_userId = tab_userId %>% mutate(icsup = avg + 1.96*(sds/sqrt(len)))
tab_userId = tab_userId %>% mutate(icinf = avg - 1.96*(sds/sqrt(len)))

fig2 = tab_userId %>% filter(len > 100) %>% ggplot(aes(rank)) +
  geom_line(aes(y = avg, colour = "rating average")) +
  geom_line(aes(y = icsup, colour = "upper IC(95%)")) +
  geom_line(aes(y = icinf, colour = "lower IC(95%)")) +
  labs(title="", y = "ratings",
       x = "users ranking (users who have watched more than 100 movies)")
fig2
```

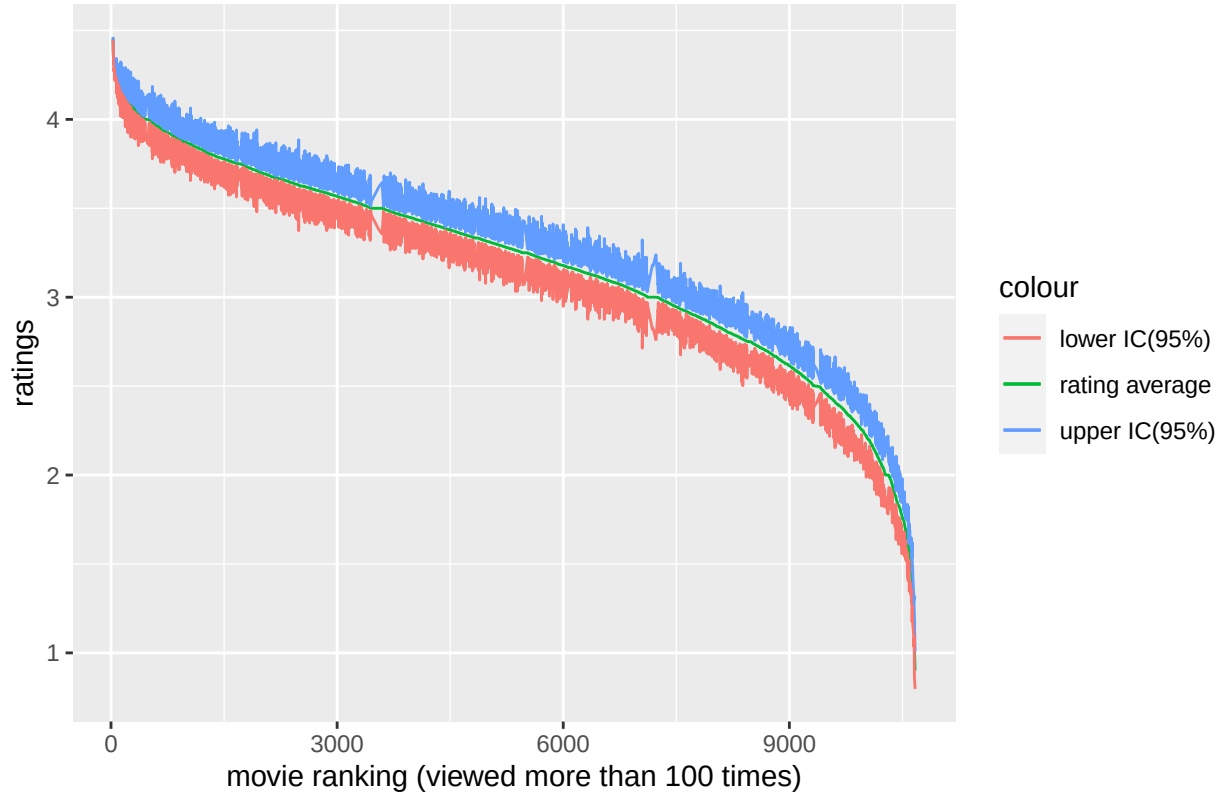


By observing the confidence intervals, it is possible to see that there is a significant difference between users. Will the same happen when grouped by movie? Below is a similar analysis to the previous one, but grouped by movie.

```
tab_movieId = edx %>% group_by(title) %>%
  summarize(avg = mean(rating),
            sds = sd(rating),
            cv  = 100*sd(rating)/mean(rating),
            len = length(rating))

tab_movieId = tab_movieId[order(-tab_movieId$avg),]
tab_movieId = tab_movieId %>% mutate(sds = ifelse(is.na(sds), 0, sds),
                                     cv  = ifelse(is.na(cv), 0, cv))
tab_movieId$rank = c(1:NROW(tab_movieId))
tab_movieId = tab_movieId %>% mutate(icsup = avg + 1.96*(sds/sqrt(len)))
tab_movieId = tab_movieId %>% mutate(icinf = avg - 1.96*(sds/sqrt(len)))

fig3 = tab_movieId %>% filter(len > 100) %>% ggplot(aes(rank)) +
  geom_line(aes(y = avg, colour = "rating average")) +
  geom_line(aes(y = icsup, colour = "upper IC(95%)")) +
  geom_line(aes(y = icinf, colour = "lower IC(95%)")) +
  labs(title="", x = "movie ranking (viewed more than 100 times)", y = "ratings")
fig3
```



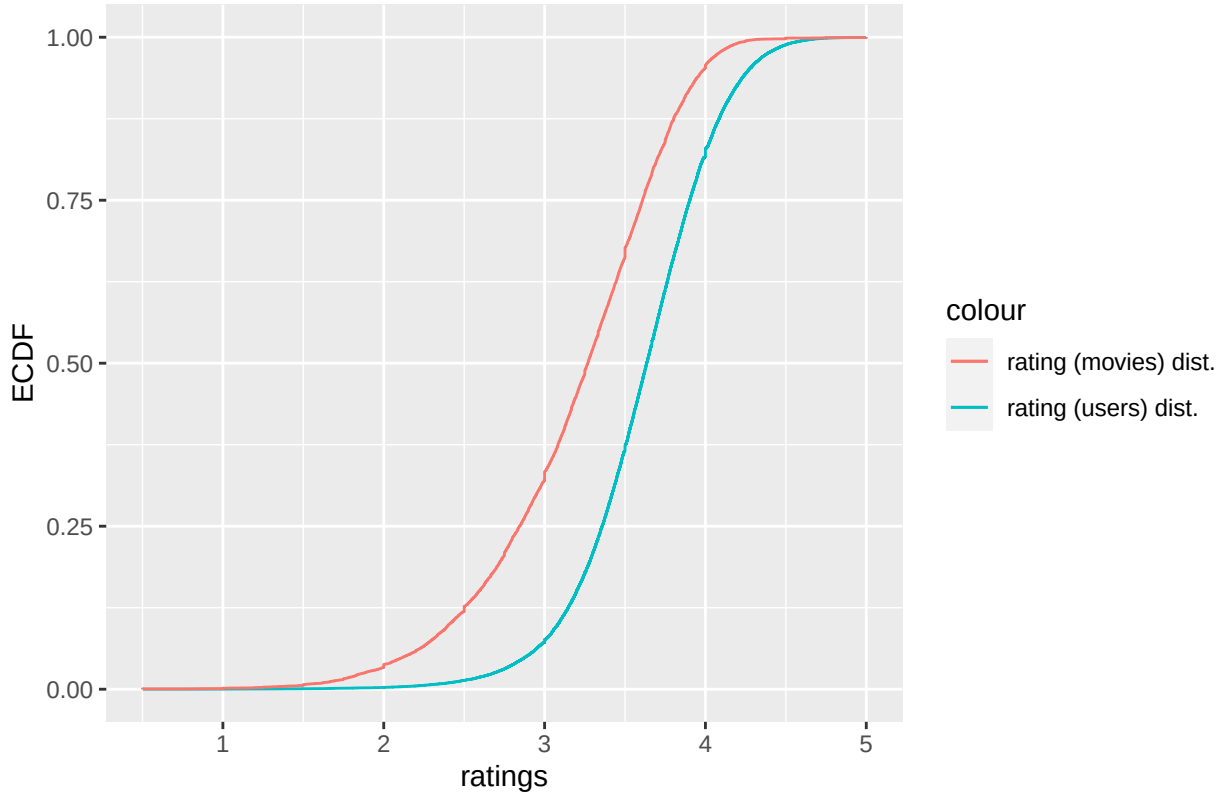
Note that results grouped by movies are similar to results grouped by user. It is possible to see that there is a significant difference between movies.

The distribution of the average rating of users will be compared to the average rating of a movie, an empirical cumulative distribution function is graphed.

```
tab_userId  =  tab_userId[order(tab_userId$avg),]
tab_movieId = tab_movieId[order(tab_movieId$avg),]

tab_userId$k  = (c(1:NROW(tab_userId) ) - 0.5)/NROW(tab_userId )
tab_movieId$k = (c(1:NROW(tab_movieId)) - 0.5)/NROW(tab_movieId)

fig4 =
  ggplot() +
  geom_line(data=tab_userId,  aes(x = avg, y = k, colour = "rating (users) dist."))  +
  geom_line(data=tab_movieId, aes(x = avg, y = k, colour = "rating (movies) dist.")) +
  labs(title="", x = "ratings", y = "ECDF")
fig4
```



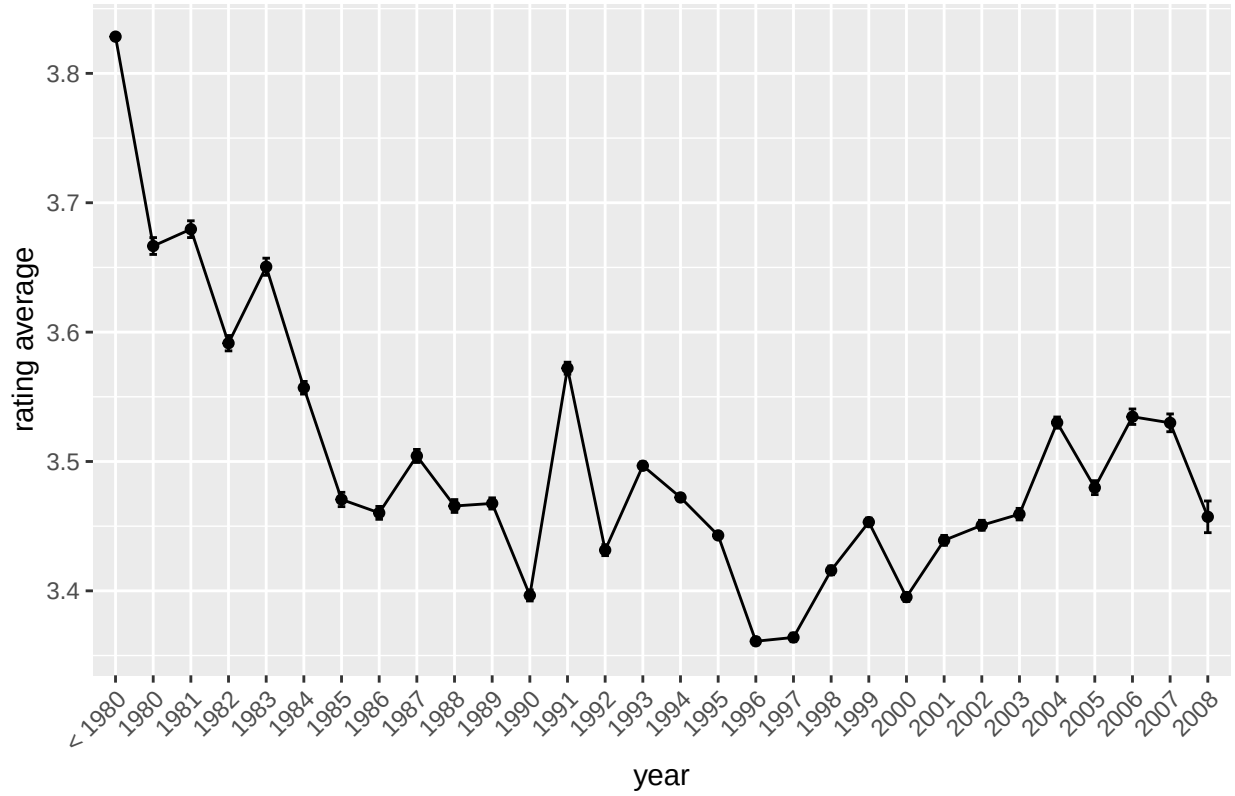
Notice that the distributions are different. First, the distribution of the average rating per user has higher statistics of central tendency. This analysis could be evidence of interaction between movie and user, since when isolating the user, or the movie, different distributions in trend and shape are obtained (for example, the asymmetry of the distribution). We will address this later.

On the other hand, we will analyze two aspects associated with the movie. The year of release and the genre of the movie.

```
edx$year = as.numeric(stri_reverse(substr(stri_reverse(edx$title), 2, 5)))
edx = edx %>% mutate(year = ifelse(year < 1980, "< 1980", year))
tab_years2 = edx %>%
  group_by(year) %>%
  summarize(avg = mean(rating),
```

```
            sds = sd(rating),
            cv  = 100*sd(rating)/mean(rating),
            len = length(rating))

fig5.2 = tab_years2 %>% ggplot(aes(x=year, y=avg)) +
  geom_line(aes(y=avg, group = 1), col="black") +
  geom_point(aes(y=avg, group = 1), col="black") +
  geom_errorbar(aes(ymin=avg-(1.96*sds/sqrt(len)),
                    ymax=avg+(1.96*sds/sqrt(len))), width=.2,
                position=position_dodge(0.05)) +
  labs(title="", x = "year", y = "rating average") +
  theme(axis.text.x = element_text(angle = 45, vjust = 1, hjust = 1))
fig5.2
```

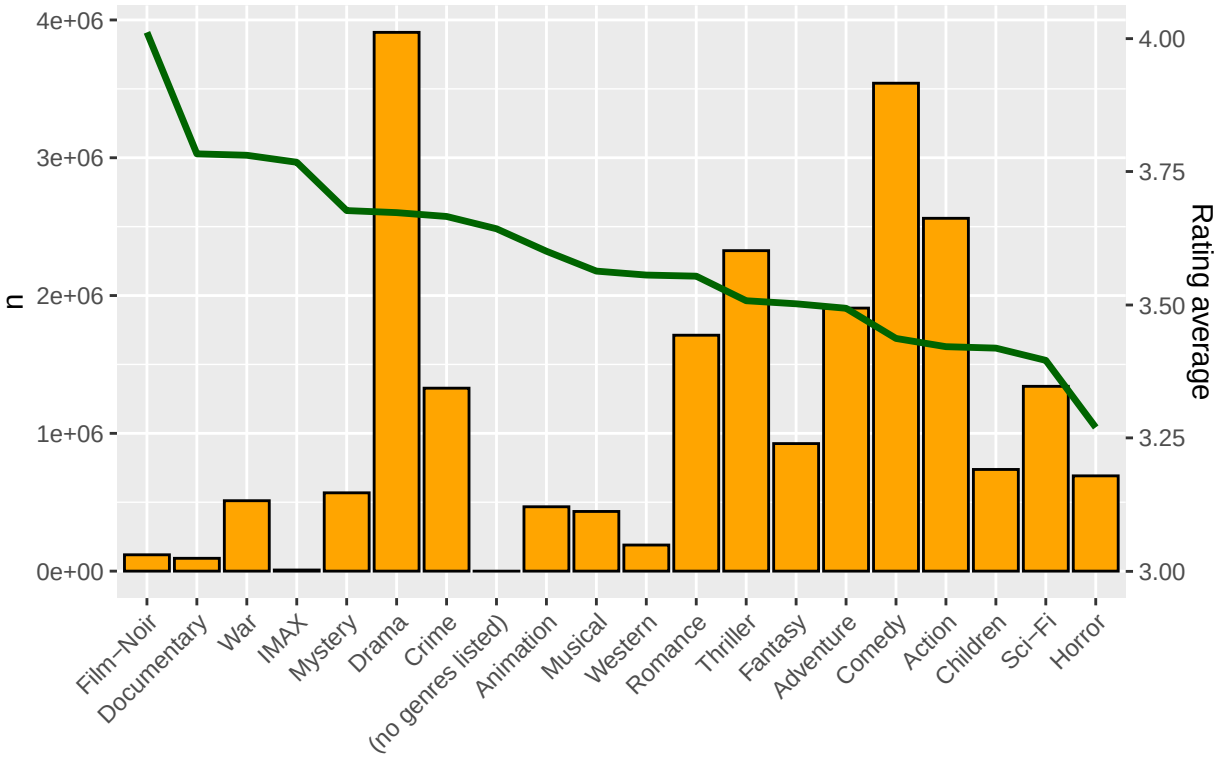



```
tab_genres = edx %>% separate_rows(genres, sep = "\\|") %>%
  group_by(genres) %>%
  summarize(avg = mean(rating),
            sds = sd(rating),
            cv  = 100*sd(rating)/mean(rating),
            len = length(rating))

b <- max(tab_genres$len)/(max(tab_genres$avg) - 3)
a <- b*(0 - 3)

fig6 = tab_genres %>% ggplot(aes(x=reorder(genres, -avg), y=len)) +
```

```
  geom_bar(col="black", stat="identity", fill = "orange") +
  geom_line(aes(y=avg * b + a, group = 1), col = "darkgreen", size = 1.2) +
  scale_y_continuous(name="n", sec.axis=sec_axis(~(. - a)/b, name = "Rating average")) +
  labs(title="", x = "") +
  theme(axis.text.x = element_text(angle = 45, vjust = 1, hjust = 1))
fig6
```



It can be seen that both the year and the gender change the average of the distribution. This could be an indication that these covariates could be inputs from the predictive model for the rating.

However, the classical numerical models related to recommender systems that only include user/movie effects within the formulation will be tested first. These models are presented below.

4. Modeling

In this work we will address the most common models of recommendation systems, based on Koren et al. (2009). Which are presented below:

$$\mathrm{m}_1 : r_{iu} = \mu + e_{iu},$$

$$\mathrm{m}_2 : r_{iu} = \mu + b_i + e_{iu},$$

$$\mathrm{m}_3 : r_{iu} = \mu + b_i + b_u + e_{iu},$$

$$\mathrm{m}_4 : r_{iu} = \mathbf{q}_i^\top \mathbf{p}_u + e_{iu},$$

where r_{iu} correspond to the rating of the i-th movie rated by the u-th user, μ is the global average, b_i is the movie bias, b_u is the user bias. In addition, $\mathbf{q}_i$ and p_u are parameter vectors that correspond to user/movie interaction. Finally, e_{iu} is the random error, where $e_{iu} \sim N(0, 1)$.

The m_1 model is a global average model, m_2 correspond to a movie effect model, m_3 is a movie and user effects model. Finally, m_4 adds the movie/user interaction to the model formulation. We will not go into the details about the inputs of the m_4 model; for more details, see Chin et. al (2015).

In order to compare the performance of these models, we use a cross-validation K-folds, with $K = 10$, using the `dismo` package. This allows us to compare exactly the same subsets (or folds) for each model, allowing a fairer comparison between the four proposed models.

The root mean square error ($RMSE$) will be used to make the comparison. The $RMSE$ is given by:

$$RMSE = \sqrt{\sum_{u,i}(\hat{r}_{ui} - r_{ui})^2/n},$$

where, $\hat{r}_{ui}$ correspond to the predicted rating value of a movie i rated by an u user.

Below is the Rcode for the models m_1, m_2 and m_3:

```
KF = 10
set.seed(2020-12-31, sample.kind = "Rounding")
edx$id_cv = kfold(1:NROW(edx), k = KF)
edx_val   = numeric()
for(u in 1:KF){
  m1_folds = edx %>% filter(id_cv != u) %>%
    summarise(avg = mean(rating))

  edx_test = edx %>% filter(id_cv == u)
  edx_test = edx_test[,c("userId", "movieId", "rating", "id_cv")]
  edx_test$pred_m1 = m1_folds

  edx_val = rbind(edx_val, edx_test, fill = TRUE)
}

RMSE_m1 = RMSE(pred = edx_val$pred_m1, obs = edx_val$rating)
print(RMSE_m1)
```

```
## [1] 1.060332
```

```
KF = 10
set.seed(2020-12-31, sample.kind = "Rounding")

# ratings model for movie effects

edx$movieId = as.factor(edx$movieId)
edx_val_m2  = numeric()
for(u in 1:KF){
  xb <- mean(edx$rating)
  m2_folds = edx %>% filter(id_cv != u) %>%
    group_by(movieId) %>%
    summarise(b_i = mean(rating - xb))
  edx_test = edx %>% filter(id_cv == u)
  edx_test$pred_m2 <- xb + edx_test %>%
    left_join(m2_folds, by = "movieId") %>%
    pull(b_i)
  edx_val_m2 = rbind(edx_val_m2, edx_test, fill = TRUE)
}
```

```r
edx_val_m2 = edx_val_m2[,c("userId", "movieId", "pred_m2")]
edx_val_m2 = na.omit(edx_val_m2)
edx_val_m2$movieId = as.numeric(as.character(edx_val_m2$movieId))

edx_val  = edx_val[,-1]
edx_val  = left_join(edx_val, edx_val_m2, by = c("userId", "movieId"))

RMSE_m2 = RMSE(pred = edx_val$pred_m2, obs = edx_val$rating, na.rm = TRUE)
print(RMSE_m2)
```

```
## [1] 0.9436378
```

```r
KF = 10
set.seed(2020-12-31, sample.kind = "Rounding")

# add user effect

edx$movieId = as.factor(edx$movieId)
edx_val_m3  = numeric()
for(u in 1:KF){

  xb <- mean(edx$rating)
  m3_folds = edx %>% filter(id_cv != u) %>%
    group_by(movieId) %>%
    summarise(b_i = mean(rating - xb))

  m3_add_user_in_folds = edx %>% filter(id_cv != u) %>% left_join(m3_folds, "movieId") %>%
    group_by(userId) %>% summarise(b_u = mean(rating - xb - b_i))

  edx_test = edx %>% filter(id_cv == u)

  edx_test$pred_m3 <- edx_test %>%
    left_join(m3_folds, by = "movieId") %>%
    left_join(m3_add_user_in_folds, by = "userId") %>%
    mutate(rating_pred = xb + b_i + b_u) %>%
    pull(rating_pred)

  edx_val_m3 = rbind(edx_val_m3, edx_test, fill = TRUE)
}

edx_val_m3 = edx_val_m3[,c("userId", "movieId", "pred_m3")]
edx_val_m3 = na.omit(edx_val_m3)

edx_val$userId    = as.character(edx_val$userId)
edx_val_m3$userId = as.character(edx_val_m3$userId)

edx_val$movieId    = as.character(edx_val$movieId)
edx_val_m3$movieId = as.character(edx_val_m3$movieId)

edx_val   = left_join(edx_val, edx_val_m3, by = c("userId", "movieId"))
RMSE_m3 = RMSE(pred = edx_val$pred_m3, obs = edx_val$rating, na.rm = TRUE)
print(RMSE_m3)
```

```
## [1] 0.8656021
```

For the m_4 model, we use the `recosystem` package, which performs cross-validation to determine certain inputs that the model uses. Then we will do our cross validation as follows: (1) we will determine the parameters of m_4 for the entire edx set. (2) We will use the 10 previously determined folds to evaluate the configuration of step (1) and compare the RMSE with the rest of the models. So the following Rcode introduces the parameter setting (don't run it, this takes too long).

```
## not run!
set.seed(2021-05-28, sample.kind = "Rounding")

train_edx <-  with(edx0, data_memory(user_index = userId,
                                     item_index = movieId,
                                     rating = rating))
test_edx  <-  with(validation, data_memory(user_index = userId,
                                           item_index = movieId,
                                           rating = rating))
# r    =  recosystem::Reco()
# opts =  r$tune(train_edx, opts = list(dim = c(20, 30, 40),
#                                       lrate = seq(0.025, 0.1, 0.025),
#                                       costp_l2 = c(0.05, 0.075, 0.1),
#                                       costq_l2 = c(0.001, 0.005, 0.01, 0.015),
#                                       nthread  = 8, niter = 10))
```

Once the parameters are determined. The following Rcode provides the 10-fold cross-validation evaluation used to compare the models in this work.

```
optim_par = list()
optim_par[["dim"]]      = 30
optim_par[["costp_l1"]] = 0
optim_par[["costp_l2"]] = 0.075
optim_par[["costq_l1"]] = 0
optim_par[["costq_l2"]] = 0.015
optim_par[["lrate"]]    = 0.05
optim_par[["loss_fun"]] = 0.7987279

edx$movieId = as.factor(edx$movieId)
edx_val_m4  = numeric()
for(u in 1:KF){

  m4_folds = edx %>% filter(id_cv != u)
  edx_test = edx %>% filter(id_cv == u)

  mfold <-  with(m4_folds, data_memory(user_index = userId,
                                       item_index = movieId,
                                       rating = rating))
  tfold  <-  with(edx_test, data_memory(user_index = userId,
                                        item_index = movieId,
                                        rating = rating))
  r    =  recosystem::Reco()
  r$train(mfold, opts = c(optim_par, nthread = 8, niter = 20, verbose = FALSE))

  pred_rating_folds <-  r$predict(tfold, out_memory())
```

```
  edx_test$pred_m4 = pred_rating_folds

  edx_val_m4 = rbind(edx_val_m4,
                     edx_test[,c("userId", "movieId", "pred_m4")],
                     fill = TRUE)

}

edx_val_m4 = edx_val_m4[,-1]

edx_val$userId    = as.character(edx_val$userId)
edx_val_m4$userId = as.character(edx_val_m4$userId)

edx_val$movieId    = as.character(edx_val$movieId)
edx_val_m4$movieId = as.character(edx_val_m4$movieId)

edx_val   = left_join(edx_val, edx_val_m4, by = c("userId", "movieId"))
RMSE_m4 = RMSE(pred = edx_val$pred_m4, obs = edx_val$rating, na.rm = TRUE)
print(RMSE_m4)
```

```
## [1] 0.7880337
```

Finally, we have the following RMSE comparison to select the best model:

Table 1: Cross validation 10-folds - performance comparison

	Global average	Movie effect	Movie and user effects	Movie and user effects and the interaction movie/user
RMSE	1.06	0.94	0.87	0.79

The model with the lowest RMSE consists of model m_4. Then, using this last model, the prediction will be made in the validation set of the course. We have the following Rcode:

```
r    =  recosystem::Reco()
r$train(train_edx, opts = c(optim_par, nthread = 8, niter = 20, verbose = FALSE))

rating_pred_validation <-  r$predict(test_edx, out_memory())
```

Before evaluating the RMSE, we will make a small correction. The numerical model does not consider the prediction to be between 0.5 and 5. Then the following Rcode corrects the high and low values and calculates the RMSE of the validation set.

```
rating_pred_validation2 = rating_pred_validation
rating_pred_validation2[rating_pred_validation2 > 5.0] = 5.0
rating_pred_validation2[rating_pred_validation2 < 0.5] = 0.5

RMSE_val = RMSE(rating_pred_validation2, validation$rating)
print(RMSE_val)
```

```
## [1] 0.7832367
```

The RMSE of the validation set is 0.78.

5. Conclusion

This work consists of generating a recommendation system for movieLens. Four classic models associated with recommender systems have been tested. A model based on global mean, user bias and movie bias, as well as their interaction, show better performance compared to RMSE, which is 0.78 in a validation set.

In EDA, some variables such as the year the movie was released or the genre of the movie could be key to further increasing the performance of the model.

6. References

Chin, W. S., Zhuang, Y., Juan, Y. C., and Lin, C. J. (2015, May). *A learning-rate schedule for stochastic gradient methods to matrix factorization.* In Pacific-Asia Conference on Knowledge Discovery and Data Mining (pp. 442–455). Springer, Cham.

Koren, Y., Bell, R., and Volinsky, C. (2009). Matrix factorization techniques for recommender systems. *Computer*, **42**(8), 30–37.