Movielens Project

Sergio Grueso

9/10/2019

# Code required to create validation and edx datasets (training and test sets):

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

## Loading required package: tidyverse

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

## v ggplot2 3.2.1 v purrr 0.3.2  
## v tibble 2.1.3 v dplyr 0.8.3  
## v tidyr 1.0.0 v stringr 1.4.0  
## v readr 1.3.1 v forcats 0.4.0

## -- Conflicts -------------------------------------------------------------------- tidyverse\_conflicts() --  
## x dplyr::filter() masks stats::filter()  
## x dplyr::lag() masks stats::lag()

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

## Loading required package: caret

## Loading required package: lattice

##   
## Attaching package: 'caret'

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

if(!require(data.table)) install.packages("data.table", repos = "http://cran.us.r-project.org")

## Loading required package: 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

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

## Loading required package: 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 object is masked from 'package:base':  
##   
## date

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")  
movies <- as.data.frame(movies) %>% mutate(movieId = as.numeric(levels(movieId))[movieId],  
 title = as.character(title),  
 genres = as.character(genres))  
movielens <- left\_join(ratings, movies, by = "movieId")  
  
set.seed(1, sample.kind="Rounding")

## 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,]  
validation <- temp %>%   
 semi\_join(edx, by = "movieId") %>%  
 semi\_join(edx, by = "userId")  
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)

# With a simple summary of the edx and validation dataset we can see their dimensions and what variables they contain:

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 838983707 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|Adventure|Sci-Fi" ...

dim(edx)

## [1] 9000055 6

str(validation)

## 'data.frame': 999999 obs. of 6 variables:  
## $ userId : int 1 1 1 2 2 2 3 3 4 4 ...  
## $ movieId : num 231 480 586 151 858 ...  
## $ rating : num 5 5 5 3 2 3 3.5 4.5 5 3 ...  
## $ timestamp: int 838983392 838983653 838984068 868246450 868245645 868245920 1136075494 1133571200 844416936 844417070 ...  
## $ title : chr "Dumb & Dumber (1994)" "Jurassic Park (1993)" "Home Alone (1990)" "Rob Roy (1995)" ...  
## $ genres : chr "Comedy" "Action|Adventure|Sci-Fi|Thriller" "Children|Comedy" "Action|Drama|Romance|War" ...

dim(validation)

## [1] 999999 6

# We can quickly check if there is any missing values with anyNA() function which indicates if there is any elements missings (FALSE means no elements missing).

anyNA(edx)

## [1] FALSE

anyNA(validation)

## [1] FALSE

#The analysis starts by looking how the variables interact with the variable we want to predict (rating) in order to see if there is any pattern we need to include in our model

# We can quickly compute the top ten most rated films on our date set with this piece of code:

edx %>% group\_by(title) %>% summarize(n = n()) %>% arrange(desc(n)) %>% head(10)

## # A tibble: 10 x 2  
## title n  
## <chr> <int>  
## 1 Pulp Fiction (1994) 31362  
## 2 Forrest Gump (1994) 31079  
## 3 Silence of the Lambs, The (1991) 30382  
## 4 Jurassic Park (1993) 29360  
## 5 Shawshank Redemption, The (1994) 28015  
## 6 Braveheart (1995) 26212  
## 7 Fugitive, The (1993) 25998  
## 8 Terminator 2: Judgment Day (1991) 25984  
## 9 Star Wars: Episode IV - A New Hope (a.k.a. Star Wars) (1977) 25672  
## 10 Apollo 13 (1995) 24284

# How many movies have been rated only once (total of 126 movies):

edx %>% group\_by(title) %>% mutate(n = n()) %>% filter(n == 1)

## # A tibble: 126 x 7  
## # Groups: title [126]  
## userId movieId rating timestamp title genres n  
## <int> <dbl> <dbl> <int> <chr> <chr> <int>  
## 1 826 64153 2.5 1230750043 Devil's Chair, The (~ Horror 1  
## 2 3457 3561 1 1051371256 Stacy's Knights (198~ Drama 1  
## 3 5227 5616 3.5 1219467370 Mesmerist, The (2002) Comedy|Fan~ 1  
## 4 5947 6941 2.5 1073321135 Just an American Boy~ Documentary 1  
## 5 6905 60880 4 1222805003 Family Game, The (Ka~ Comedy|Dra~ 1  
## 6 7304 31547 3.5 1230321725 Lessons of Darkness ~ Documentar~ 1  
## 7 7304 38435 3.5 1230589804 Forty Shades of Blue~ Drama 1  
## 8 8041 61970 2 1222779824 Moonbase (1998) Sci-Fi 1  
## 9 9212 63312 4 1226684912 Krabat (2008) Drama|Fant~ 1  
## 10 10057 58520 4 1230245584 Mala Noche (1985) Drama 1  
## # ... with 116 more rows

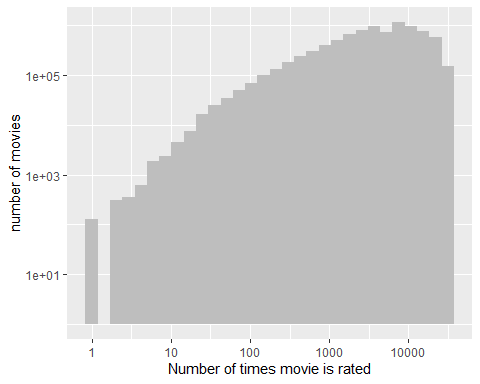
# And the general distribution of the number of rates:

edx %>% group\_by(title) %>% mutate(n = n()) %>%  
 ggplot(aes(n)) +   
 geom\_histogram(fill = "grey") +  
 scale\_x\_log10() +   
 scale\_y\_log10() +   
 xlab("Number of times movie is rated") + ylab("number of movies")

## `stat\_bin()` using `bins = 30`. Pick better value with `binwidth`.

## Warning: Transformation introduced infinite values in continuous y-axis

## Warning: Removed 1 rows containing missing values (geom\_bar).



# For example we can look if all the users rate with the same frecuency by comparing the number of rates per user.

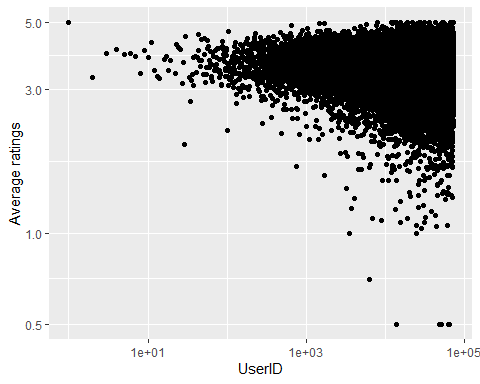
edx %>% group\_by(userId) %>% summarize(n = n()) %>%  
 ggplot(aes(n)) +  
 geom\_histogram(fill = "grey") +  
 scale\_x\_log10() +   
 scale\_y\_log10() +   
 xlab("Number of rates") + ylab("UserID")

## `stat\_bin()` using `bins = 30`. Pick better value with `binwidth`.



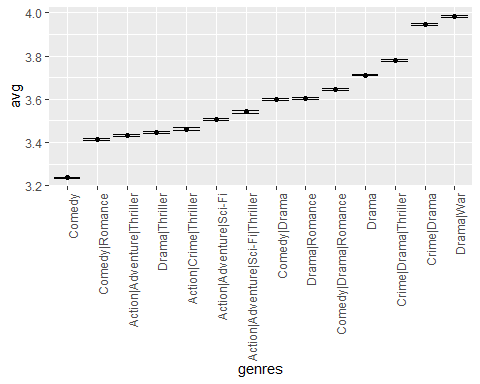
# We can plot the average rating of every user

edx %>% group\_by(userId) %>% summarize(avg= mean(rating)) %>%  
 ggplot(aes(userId, avg)) +   
 geom\_point() +  
 scale\_x\_log10() +   
 scale\_y\_log10() +   
 xlab("UserID") + ylab("Average ratings")



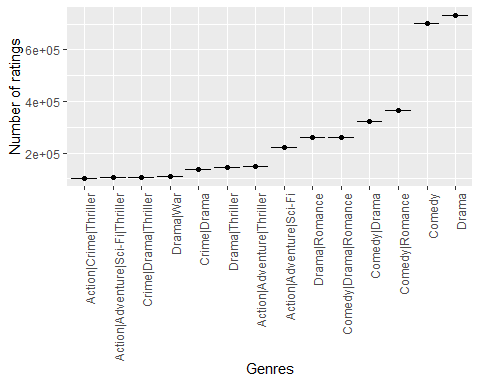
# Average rates of genres with more than 100000 rates

edx %>% group\_by(genres) %>%  
 summarize(n = n(), avg = mean(rating), se = sd(rating)/sqrt(n())) %>%  
 filter(n >= 100000) %>%   
 mutate(genres = reorder(genres, avg)) %>%  
 ggplot(aes(x = genres, y = avg, ymin = avg - 2\*se, ymax = avg + 2\*se)) +   
 geom\_point() +  
 geom\_errorbar() +   
 theme(axis.text.x = element\_text(angle = 90, hjust = 1))



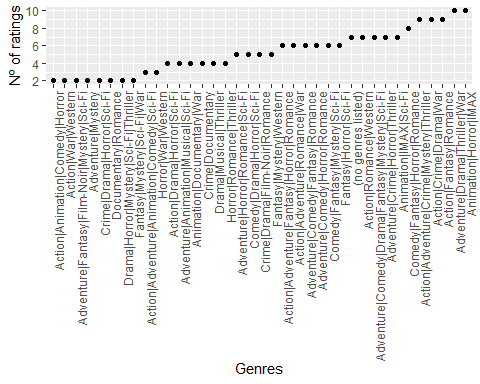
# Most rated genres

edx %>% group\_by(genres) %>%  
 summarize(n = n(), se = sd(rating)/sqrt(n())) %>%  
 filter(n >= 100000) %>%   
 mutate(genres = reorder(genres, n)) %>%  
 ggplot(aes(x = genres, y = n, ymin = n - 2\*se, ymax = n + 2\*se)) +   
 geom\_point() +  
 geom\_errorbar() +   
 theme(axis.text.x = element\_text(angle = 90, hjust = 1)) +  
 xlab("Genres") + ylab("Number of ratings")



# Least rated genres

edx %>% group\_by(genres) %>%  
 summarize(n = n()) %>%  
 filter(n <= 10) %>%   
 mutate(genres = reorder(genres, n)) %>%  
 ggplot(aes(x = genres, y = n)) +   
 geom\_point() +  
   
 theme(axis.text.x = element\_text(angle = 90, hjust = 1)) +  
 xlab("Genres") + ylab("Nº of ratings")



## model RMSE function

RMSE <- function(true\_ratings, predicted\_ratings){  
sqrt(mean((true\_ratings - predicted\_ratings)^2))}

## RMSE calculation

# model\_rmse <- RMSE(predicted\_ratings, validation$rating)

## parameter 1

mu <- mean(edx$rating)   
movie\_avgs <- edx %>%   
 group\_by(movieId) %>%   
 summarize(b\_1 = mean(rating - mu))

# parameter 2

user\_avgs <- edx %>%   
 left\_join(movie\_avgs, by='movieId') %>%  
 group\_by(userId) %>%  
 summarize(b\_2 = mean(rating - mu - b\_1))

# parameter 3:

time\_avgs <- edx %>%   
 left\_join(movie\_avgs, by='movieId') %>%  
 left\_join(user\_avgs, by= 'userId') %>%  
 mutate(timestamp = timestamp/60/60/24) %>%  
 mutate(timestamp = round(timestamp, digits = 0)) %>%  
 group\_by(timestamp) %>%  
 summarize(b\_3 = mean(rating - mu - b\_1 - b\_2))

# parameter 4

genres\_avgs <- edx %>%   
 left\_join(movie\_avgs, by='movieId') %>%  
 left\_join(user\_avgs, by= 'userId') %>%  
 mutate(timestamp = timestamp/60/60/24) %>%  
 mutate(timestamp = round(timestamp, digits = 0)) %>%  
 left\_join(time\_avgs, by= 'timestamp') %>%  
 group\_by(genres) %>%  
 summarize(b\_4 = mean(rating - mu - b\_1 - b\_2 - b\_3))

# final predictions

predicted\_ratings <- validation %>%  
 mutate(timestamp = timestamp/60/60/24) %>%  
 mutate(timestamp = round(timestamp, digits = 0)) %>%  
 left\_join(movie\_avgs, by='movieId') %>%  
 left\_join(user\_avgs, by='userId') %>%  
 left\_join(time\_avgs, by='timestamp') %>%  
 left\_join(genres\_avgs, by= 'genres') %>%  
 mutate(pred = mu + b\_1 + b\_2 + b\_3 + b\_4) %>%   
 pull(pred)  
  
model\_rmse <- RMSE(predicted\_ratings, validation$rating)  
model\_rmse

## [1] 0.8644346