MovieLens\_EDA

Art\_Shablinski

26/06/2020

#### Loading and attaching the required packages using pacman:

pacman::p\_load(pacman, corrgram, lubridate, rmarkdown, rvest, XML, shiny, tidyverse, tidytext, stringr, splitstackshape, wordcloud, modelr)

#### Read the csv files in the folder:

setwd("E:/DATA/MovieLens/ml-20m")  
getwd()

## [1] "E:/DATA/MovieLens/ml-20m"

memory.limit(size=56000)

## [1] 56000

movies <- read.csv(file = 'movies.csv', header = T, na.strings = c(""))  
ratings <- read.csv(file = 'ratings.csv', header = T, na.strings = c(""))

#### Initial Analysis:

# User Statistics:

print("UserId 5-Number Summary:")

## [1] "UserId 5-Number Summary:"

summary(ratings$userId)

## Min. 1st Qu. Median Mean 3rd Qu. Max.   
## 1 34395 69141 69046 103637 138493

# Movie Statistics:

print("UserId 5-Number Summary:")

## [1] "UserId 5-Number Summary:"

summary(ratings$movieId)

## Min. 1st Qu. Median Mean 3rd Qu. Max.   
## 1 902 2167 9042 4770 131262

# Ratings Statistics:

print("UserId 5-Number Summary:")

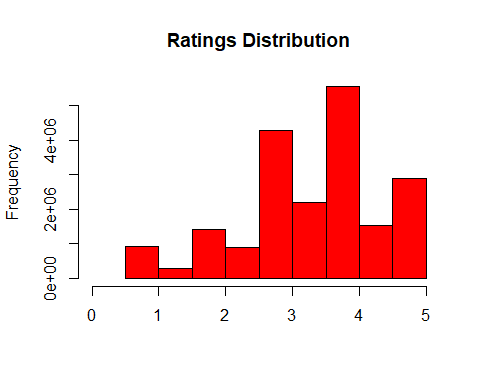
## [1] "UserId 5-Number Summary:"

summary(ratings$rating)

## Min. 1st Qu. Median Mean 3rd Qu. Max.   
## 0.500 3.000 3.500 3.526 4.000 5.000

# Distribution of our output variable (label):

hist(ratings$rating,  
 xlim = c(0, 5),  
 breaks = 10,  
 main = "Ratings Distribution",  
 xlab = "",  
 col = "red")



#### Cleaning Data:

glimpse(ratings)

## Rows: 20,000,263  
## Columns: 4  
## $ userId <int> 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, ...  
## $ movieId <int> 2, 29, 32, 47, 50, 112, 151, 223, 253, 260, 293, 296, 318...  
## $ rating <dbl> 3.5, 3.5, 3.5, 3.5, 3.5, 3.5, 4.0, 4.0, 4.0, 4.0, 4.0, 4....  
## $ timestamp <int> 1112486027, 1112484676, 1112484819, 1112484727, 111248458...

ratings <- ratings %>%  
 mutate(timestamp = as\_datetime(timestamp))  
  
summary(ratings)

## userId movieId rating   
## Min. : 1 Min. : 1 Min. :0.500   
## 1st Qu.: 34395 1st Qu.: 902 1st Qu.:3.000   
## Median : 69141 Median : 2167 Median :3.500   
## Mean : 69046 Mean : 9042 Mean :3.526   
## 3rd Qu.:103637 3rd Qu.: 4770 3rd Qu.:4.000   
## Max. :138493 Max. :131262 Max. :5.000   
## timestamp   
## Min. :1995-01-09 11:46:44   
## 1st Qu.:2000-08-20 18:55:45   
## Median :2004-12-20 15:18:06   
## Mean :2004-11-20 02:32:01   
## 3rd Qu.:2008-11-02 16:11:57   
## Max. :2015-03-31 06:40:02

movies <- movies %>%  
 # trim whitespaces  
 mutate(title = str\_trim(title)) %>%  
 # separate year into a separate column from the 'title' variable for exploratory analysis  
 extract(title, c("title\_tmp", "year"), regex = "^(.\*) \\(([0-9 \\-]\*)\\)$", remove = F) %>%  
 # for series we will take debut dates instead  
 mutate(year = if\_else(str\_length(year) > 4, as.integer(str\_split(year, "-", simplify = T)[1]), as.integer(year))) %>%  
 # replace title NA's with original title  
 mutate(title = if\_else(is.na(title\_tmp), title, title\_tmp)) %>%  
 # drop title\_tmp column  
 select(-title\_tmp) %>%  
 # turn (no genres listed) to NA  
 mutate(genres = if\_else(genres == "(no genres listed)", `is.na<-`(genres), genres))

## Warning in replace\_with(out, !condition, false, fmt\_args(~false), glue("length  
## of {fmt\_args(~condition)}")): NAs introduced by coercion

#### Glimpse at Movies Data Frame:

head(movies)

## movieId title year  
## 1 1 Toy Story 1995  
## 2 2 Jumanji 1995  
## 3 3 Grumpier Old Men 1995  
## 4 4 Waiting to Exhale 1995  
## 5 5 Father of the Bride Part II 1995  
## 6 6 Heat 1995  
## genres  
## 1 Adventure|Animation|Children|Comedy|Fantasy  
## 2 Adventure|Children|Fantasy  
## 3 Comedy|Romance  
## 4 Comedy|Drama|Romance  
## 5 Comedy  
## 6 Action|Crime|Thriller

tail(movies)

## movieId title year  
## 27273 131252 Forklift Driver Klaus: The First Day on the Job 2001  
## 27274 131254 Kein Bund fÃ¼r's Leben 2007  
## 27275 131256 Feuer, Eis & Dosenbier 2002  
## 27276 131258 The Pirates 2014  
## 27277 131260 Rentun Ruusu 2001  
## 27278 131262 Innocence 2014  
## genres  
## 27273 Comedy|Horror  
## 27274 Comedy  
## 27275 Comedy  
## 27276 Adventure  
## 27277 <NA>  
## 27278 Adventure|Fantasy|Horror

glimpse(movies)

## Rows: 27,278  
## Columns: 4  
## $ movieId <int> 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, ...  
## $ title <chr> "Toy Story", "Jumanji", "Grumpier Old Men", "Waiting to Exh...  
## $ year <int> 1995, 1995, 1995, 1995, 1995, 1995, 1995, 1995, 1995, 1995,...  
## $ genres <chr> "Adventure|Animation|Children|Comedy|Fantasy", "Adventure|C...

#### Glimpse at Ratings Data Frame:

head(movies)

## movieId title year  
## 1 1 Toy Story 1995  
## 2 2 Jumanji 1995  
## 3 3 Grumpier Old Men 1995  
## 4 4 Waiting to Exhale 1995  
## 5 5 Father of the Bride Part II 1995  
## 6 6 Heat 1995  
## genres  
## 1 Adventure|Animation|Children|Comedy|Fantasy  
## 2 Adventure|Children|Fantasy  
## 3 Comedy|Romance  
## 4 Comedy|Drama|Romance  
## 5 Comedy  
## 6 Action|Crime|Thriller

tail(movies)

## movieId title year  
## 27273 131252 Forklift Driver Klaus: The First Day on the Job 2001  
## 27274 131254 Kein Bund fÃ¼r's Leben 2007  
## 27275 131256 Feuer, Eis & Dosenbier 2002  
## 27276 131258 The Pirates 2014  
## 27277 131260 Rentun Ruusu 2001  
## 27278 131262 Innocence 2014  
## genres  
## 27273 Comedy|Horror  
## 27274 Comedy  
## 27275 Comedy  
## 27276 Adventure  
## 27277 <NA>  
## 27278 Adventure|Fantasy|Horror

glimpse(movies)

## Rows: 27,278  
## Columns: 4  
## $ movieId <int> 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, ...  
## $ title <chr> "Toy Story", "Jumanji", "Grumpier Old Men", "Waiting to Exh...  
## $ year <int> 1995, 1995, 1995, 1995, 1995, 1995, 1995, 1995, 1995, 1995,...  
## $ genres <chr> "Adventure|Animation|Children|Comedy|Fantasy", "Adventure|C...

#### Further Data Exploration:

# Number of movies per year/decade  
movies\_per\_year <- movies %>%  
 na.omit() %>%  
 select(movieId, year) %>%  
 group\_by(year) %>%  
 summarise(count = n()) %>%  
 arrange(year)

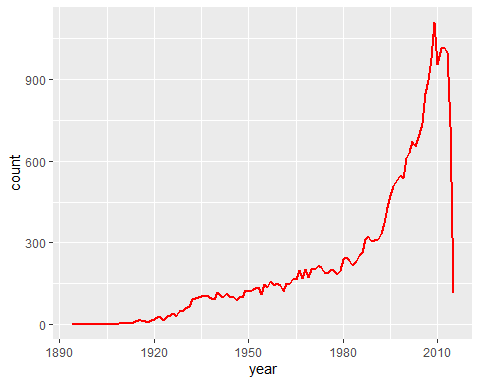
## `summarise()` ungrouping output (override with `.groups` argument)

print(movies\_per\_year)

## # A tibble: 116 x 2  
## year count  
## <int> <int>  
## 1 1894 2  
## 2 1895 2  
## 3 1896 2  
## 4 1898 2  
## 5 1899 1  
## 6 1900 1  
## 7 1901 1  
## 8 1902 1  
## 9 1903 1  
## 10 1905 1  
## # ... with 106 more rows

#### Plot movies per year:

movies\_per\_year %>%  
 ggplot(aes(x = year, y = count)) +  
 geom\_line(color="red", size=0.8)



#### Most popular movie genres:

genres\_df <- movies %>%  
 separate\_rows(genres, sep = "\\|") %>%  
 group\_by(genres) %>%  
 summarise(number = n()) %>%  
 arrange(desc(number))

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

print(genres\_df)

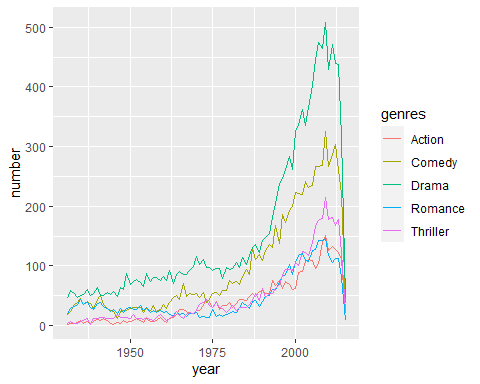
## # A tibble: 20 x 2  
## genres number  
## <chr> <int>  
## 1 Drama 13344  
## 2 Comedy 8374  
## 3 Thriller 4178  
## 4 Romance 4127  
## 5 Action 3520  
## 6 Crime 2939  
## 7 Horror 2611  
## 8 Documentary 2471  
## 9 Adventure 2329  
## 10 Sci-Fi 1743  
## 11 Mystery 1514  
## 12 Fantasy 1412  
## 13 War 1194  
## 14 Children 1139  
## 15 Musical 1036  
## 16 Animation 1027  
## 17 Western 676  
## 18 Film-Noir 330  
## 19 <NA> 246  
## 20 IMAX 196

#### Genres popularity per year:

genres\_popularity <- movies %>%  
 na.omit() %>% # omit missing values  
 select(movieId, year, genres) %>% # select columns we are interested in  
 separate\_rows(genres, sep = "\\|") %>% # separate genres into rows  
 mutate(genres = as.factor(genres)) %>% # turn genres in factors  
 group\_by(year, genres) %>% # group data by year and genre  
 summarise(number = n()) %>% # count  
 complete(year = full\_seq(year, 1), genres, fill = list(number = 0)) # add missing years/genres

## `summarise()` regrouping output by 'year' (override with `.groups` argument)

genres\_popularity %>%  
 filter(year > 1930) %>%  
 filter(genres %in% c("Drama", "Comedy", "Thriller", "Romance", "Action")) %>%  
 ggplot(aes(x = year, y = number)) +  
 geom\_line(aes(color=genres), size=0.7) +   
 scale\_fill\_brewer(palette = "Paired")



#### To avoid introducing bias and unnecessary complexity we will drop timestamp column:

ratings$timestamp <- NULL

### Making sure critical variables in the relevant dataframes are intact:

#movies %>%  
# count(movieId, sort = TRUE)  
# ratings %>%  
# count(userId, sort = TRUE)

### Final Check for missing values in the key variables required for vector embeddings:

sum(is.na(movies$userId))

## [1] 0

sum(is.na(movies$movieId))

## [1] 0

sum(is.na(ratings$rating))

## [1] 0

### Saving final data frame accordingly:

write.csv(movies, file = "movies\_df.csv", row.names = F)  
write.csv(ratings, file = "ratings\_df.csv", row.names = F)