movielens capstone harvardx data science

tuncay dogan

April 22, 2019

This is a capstone Movielens project for HarvardX Data Science.

MovieLens_Capstone_Harvardx_DataScience_Tuncay_Dogan

Introduction

Data analysis is more than just crunching numbers and visualizing them. It is, in fact, a storytelling. With the right mindset and enough data, it is feasible to create a digital footprint of human behavior. That's something that I strongly believe in. Human actions, inadvertently, leave traces of prefences, which can then be converted into assumptions about the future demand on the applicable products. In here, the decision-making process, by reading outcomes of well-analyzed data, by the management, can be simplified as well as increased in efficiency. All these then feed into the profits and the team with the most truthful story wins the Bayesian game.

There will be some extra data exploration than supposed to, to make me learn better and use it as a reference. So, please bare with me on this.

Overview

First, I will explore the data to understand the intuition of it, so that i can work with the variables.

Second, I will provide some graphs to further grasp the relationship between variables.

Third, I will try to reduce RMSE by using some techniques that are listed below.

Summary

Goal of the Movielens Recommendation Systems study:

By using the movielens dataset, I will try to build an algorithm that takes in independent variables and trains the model by using already known dependent variable, in this case it is the movie ratings, and then I hope to predict dependent variable, movie ratings, by using the model created with independent variables in hand.

To accomplish the goal, we split the data into two parts: train set (edx) and validation set (validation).

Before I dive in, let's see some details of the project provided by Movielens. > link to Movielens 10M Data Description

release year refers to the release date of the movie and the ratings by release reflect the rates the movie received by that year, on average

rate year refers to when rates are collected

rate_release_dif refers to the span of rating collection (rate year - release year), which can range from years to decades

age_of_movie refers to how old the movie is. It is calculated as this year minus the release year of the movie

Data Prep:

```
# The data didn't require cleaning.
# After checking possible missing values, time frame anomalies, and seeing whether the data is intac
# I converted timestamps into dates, separated genres, and extracted the movie release years and ra
# I also created weekday and age_of_movie and rate_month variables to grasp the dataset better.
# By doing so, I was able to compare the rate year vs release year rating differences and user votir
```

Other exploratory data analyses I've done are:

```
# seeing distinct values
# how old a movie is and for how long reviews being collected
# count of genres listed
# proportion of each genre
# top 20 movies with highest number of ratings
# ratings given from most to least and their proportions
# movie genre count by year and by count order
# ratings by release year and rate year (defined below)
# mean ratings of each genre and standard deviations
# visual inspection vs data: which genre to produce?
# what week day the most people go to movies aand what are the quietest days? (Quora Thread)
#     "Why are movies released on Thursdays?"
# correlation outputs and inference
```

Visualization part includes:

```
# rating distribution
# genres rating distribution
# rate year rating distribution
# release year rating distribution (observed two clusters: 1980 dropoff point)
#     what happened in 1980?
#         "The History of the Hollywood Movie Industry"
# user count rate distribution  (log2)
# movie count rate distribution (log2)
```

RMSE:

```
# formula and explanation
# algorithm
#     n() from dplyr library
#     ddply from plyr library
#     ridge and lasso
#     ranger
#     regularization
#     kmeans
#     svm
#     xgboost
#     randomforest
#     slope one
```

```
    #     sparse matrix
    #     OLS and GLS
    # graphs
```

```
library(tidyverse)
```

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

```
## v ggplot2 3.1.0       v purrr   0.3.2
## v tibble  2.0.1       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 'tidyr' was built under R version 3.5.3
```

```
## Warning: package 'purrr' was built under R version 3.5.3
```

```
## Warning: package 'dplyr' was built under R version 3.5.3
```

```
## -- 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(ggplot2)
library(MLmetrics)
```

```
##
## Attaching package: 'MLmetrics'
```

```
## The following objects are masked from 'package:caret':
##
##     MAE, RMSE
```

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

```
#library(nlme)
#library(plyr) # conflict with dplyr, be careful
#library(dplyr)
#library(magrittr)
#library(olsrr)
#library(psych)
#library(plotrix)
#library(MASS)
#library(broom)
#library(glmnet)
#library(stringr)

#library(rmarkdown)
```

```
#library(e1071)
#library(randomForest)
```

```
#movielens <- read.csv("test.csv") # this is a portion I used before working with the full movielens da
                # this makes it easier to re-run and loads faster, since full data is quiet big.
```

Reading the Movielens Data

validation set is created below

```
#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")

# 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 <- read.table(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")
```

Creating additional columns for exploratory data analysis

```
this_year <- as.numeric(format(Sys.time(), "%Y"))
    # convert timestamp to date format
movielens <- movielens %>% mutate(timestamp2 = structure(movielens$timestamp[],class=c('POSIXt','POSIXc
    # extract movie release year from the title
movielens <- movielens %>% mutate(releaseyear = as.numeric(str_extract(str_extract(title, "[/(]\\d{4}[/
    # find how old the movie is
movielens <- movielens %>% mutate(age_of_movie=this_year-movielens$releaseyear)
    # find the year the movie is rated by the user(s)
movielens <- movielens %>% mutate(rateyear = as.numeric(as.character(format(as.Date(timestamp2), "%Y"))
    # find the month the movie is rated
movielens <- movielens %>% mutate(ratemonth_numeric = as.numeric(as.character(format(as.Date(timestamp2
movielens <- movielens %>% mutate(ratemonth = format(as.Date(timestamp2), "%B"))
    # which week day the movie is rated by the user(s)
movielens <- movielens %>% mutate(weekday = format(as.Date(timestamp2), "%A"))
    # rate_release_dif = gives difference between rate year and release year
movielens <- movielens %>% mutate(rate_release_dif=movielens$rateyear - movielens$releaseyear)
    # separate genres
movielens2 <- separate_rows(movielens,genres,sep="\\|")
    # assign numeric values to separated genres
movielens2 <- movielens2 %>% mutate(genres_numeric=as.numeric(as.factor(movielens2$genres)))
```

Exploratory Data Analysis

```
head(movielens,1)
```

```
##   userId movieId rating timestamp            title         genres
## 1      1     122      5 838985046 Boomerang (1992) Comedy|Romance
##            timestamp2 releaseyear age_of_movie rateyear ratemonth_numeric
## 1 1996-08-02 04:24:06        1992           27     1996                8
##   ratemonth weekday rate_release_dif
## 1    August  Friday                4
```

```
head(movielens2,1)
```

```
##   userId movieId rating timestamp            title genres
## 1      1     122      5 838985046 Boomerang (1992) Comedy
##            timestamp2 releaseyear age_of_movie rateyear ratemonth_numeric
## 1 1996-08-02 04:24:06        1992           27     1996                8
##   ratemonth weekday rate_release_dif genres_numeric
## 1    August  Friday                4              6
```

```
#anyNA(movielens)
#anyNA(movielens2)
```

```
summary(movielens)
```

```
##      userId         movieId          rating        timestamp
##  Min.   :    1   Min.   :    1   Min.   :0.500   Min.   :7.897e+08
##  1st Qu.:18123   1st Qu.:  648   1st Qu.:3.000   1st Qu.:9.468e+08
##  Median :35741   Median : 1834   Median :4.000   Median :1.035e+09
##  Mean   :35870   Mean   : 4120   Mean   :3.512   Mean   :1.033e+09
##  3rd Qu.:53608   3rd Qu.: 3624   3rd Qu.:4.000   3rd Qu.:1.127e+09
##  Max.   :71567   Max.   :65133   Max.   :5.000   Max.   :1.231e+09
##     title              genres            timestamp2
##  Length:10000054    Length:10000054    Min.   :1995-01-09 03:46:49
##  Class :character   Class :character   1st Qu.:2000-01-01 14:31:20
##  Mode  :character   Mode  :character   Median :2002-10-24 09:21:21
##                                        Mean   :2002-09-21 04:05:54
##                                        3rd Qu.:2005-09-14 18:51:10
##                                        Max.   :2009-01-04 21:02:16
##   releaseyear    age_of_movie       rateyear    ratemonth_numeric
##  Min.   :1915   Min.   : 11.00   Min.   :1995   Min.   : 1.000
##  1st Qu.:1987   1st Qu.: 21.00   1st Qu.:2000   1st Qu.: 4.000
##  Median :1994   Median : 25.00   Median :2002   Median : 7.000
##  Mean   :1990   Mean   : 28.78   Mean   :2002   Mean   : 6.786
##  3rd Qu.:1998   3rd Qu.: 32.00   3rd Qu.:2005   3rd Qu.:10.000
##  Max.   :2008   Max.   :104.00   Max.   :2009   Max.   :12.000
##   ratemonth           weekday          rate_release_dif
##  Length:10000054    Length:10000054    Min.   :-2.00
##  Class :character   Class :character   1st Qu.: 2.00
##  Mode  :character   Mode  :character   Median : 7.00
##                                        Mean   :11.98
##                                        3rd Qu.:16.00
##                                        Max.   :93.00
```

```
table(movielens2$genres)
```

```
##
```

```
## (no genres listed)            Action         Adventure
##                  7           2845349           2121074
##          Animation          Children            Comedy
##             519112            820149           3934068
##              Crime       Documentary             Drama
##            1474957            103454           4344198
##            Fantasy         Film-Noir            Horror
##            1028482            131592            768225
##               IMAX           Musical           Mystery
##               9080            481174            630944
##            Romance            Sci-Fi          Thriller
##            1901883           1490489           2584435
##                War           Western
##             568063            210459
```

```
# of course, some movies fall in to a couple of genre categories,
#but for the sake of understanding the intituion, this tool is good enough.
```

```
str(movielens2)
```

```
## 'data.frame':    25967194 obs. of  15 variables:
##  $ userId           : int  1 1 1 1 1 1 1 1 1 1 ...
##  $ movieId          : num  122 122 185 185 185 231 292 292 292 292 ...
##  $ rating           : num  5 5 5 5 5 5 5 5 5 5 ...
##  $ timestamp        : int  838985046 838985046 838983525 838983525 838983525 838983392 838983421 838
##  $ title            : chr  "Boomerang (1992)" "Boomerang (1992)" "Net, The (1995)" "Net, The (1995)"
##  $ genres           : chr  "Comedy" "Romance" "Action" "Crime" ...
##  $ timestamp2       : POSIXt, format: "1996-08-02 04:24:06" "1996-08-02 04:24:06" ...
##  $ releaseyear      : num  1992 1992 1995 1995 1995 ...
##  $ age_of_movie     : num  27 27 24 24 24 25 24 24 24 24 ...
##  $ rateyear         : num  1996 1996 1996 1996 1996 ...
##  $ ratemonth_numeric: num  8 8 8 8 8 8 8 8 8 8 ...
##  $ ratemonth        : chr  "August" "August" "August" "August" ...
##  $ weekday          : chr  "Friday" "Friday" "Friday" "Friday" ...
##  $ rate_release_dif : num  4 4 1 1 1 2 1 1 1 1 ...
##  $ genres_numeric   : num  6 16 2 7 18 6 2 9 17 18 ...
```

```
#describe(movielens)
```

```
length(unique(movielens$movieId))
```

```
## [1] 10677
```

```
n_distinct(movielens$userId)
```

```
## [1] 69878
```

```
min(movielens$releaseyear)
```

```
## [1] 1915
```

```
max(movielens$releaseyear)
```

```
## [1] 2008
```

```
min(movielens$rateyear)
```

```
## [1] 1995
```

```r
max(movielens$rateyear)
```

```
## [1] 2009
```

```r
# proportion of each genre in movielens set
 p1<- data.frame(sort(round(prop.table(table(movielens2$genres))*100,1)))%>%arrange(desc(Freq))
p1
```

```
##                  Var1 Freq
## 1               Drama 16.7
## 2              Comedy 15.2
## 3              Action 11.0
## 4            Thriller 10.0
## 5           Adventure  8.2
## 6             Romance  7.3
## 7               Crime  5.7
## 8              Sci-Fi  5.7
## 9             Fantasy  4.0
## 10           Children  3.2
## 11             Horror  3.0
## 12            Mystery  2.4
## 13                War  2.2
## 14          Animation  2.0
## 15            Musical  1.9
## 16            Western  0.8
## 17          Film-Noir  0.5
## 18        Documentary  0.4
## 19 (no genres listed)  0.0
## 20               IMAX  0.0
```

```r
# top 20 movies with the highest number of ratings given in descending order

titleratingcount <- movielens %>% group_by(title) %>%
summarize(count = n()) %>%
top_n(20,count) %>%
arrange(desc(count))

titleratingcount
```

```
## # A tibble: 20 x 2
##    title                                                     count
##    <chr>                                                     <int>
##  1 Pulp Fiction (1994)                                       34864
##  2 Forrest Gump (1994)                                       34457
##  3 Silence of the Lambs, The (1991)                          33668
##  4 Jurassic Park (1993)                                      32631
##  5 Shawshank Redemption, The (1994)                          31126
##  6 Braveheart (1995)                                         29154
##  7 Fugitive, The (1993)                                      28951
##  8 Terminator 2: Judgment Day (1991)                         28948
##  9 Star Wars: Episode IV - A New Hope (a.k.a. Star Wars) (1977) 28566
## 10 Apollo 13 (1995)                                          27035
## 11 Batman (1989)                                             26996
## 12 Toy Story (1995)                                          26449
## 13 Independence Day (a.k.a. ID4) (1996)                      26042
```

```
## 14 Dances with Wolves (1990)                                  25912
## 15 Schindler's List (1993)                                    25777
## 16 True Lies (1994)                                           25381
## 17 Star Wars: Episode VI - Return of the Jedi (1983)          25098
## 18 12 Monkeys (Twelve Monkeys) (1995)                         24397
## 19 Usual Suspects, The (1995)                                 24037
## 20 Fargo (1996)                                               23794
```

```
#most given ratings in order from most to least

ratingcount <- movielens %>% group_by(rating) %>%
    summarize(count = n()) %>%
    arrange(desc(count))

ratingcount
```

```
## # A tibble: 10 x 2
##    rating   count
##     <dbl>   <int>
##  1    4   2875850
##  2    3   2356676
##  3    5   1544812
##  4    3.5  879764
##  5    2    790306
##  6    4.5  585022
##  7    1    384180
##  8    2.5  370178
##  9    1.5  118278
## 10    0.5   94988
```

```
# total rating counts for each rating category and its proportion

sum=sum(ratingcount$count)
ratingcount2 <- ratingcount%>%mutate(p_count=(ratingcount$count)/sum)
ratingcount2
```

```
## # A tibble: 10 x 3
##    rating   count p_count
##     <dbl>   <int>   <dbl>
##  1    4   2875850 0.288
##  2    3   2356676 0.236
##  3    5   1544812 0.154
##  4    3.5  879764 0.0880
##  5    2    790306 0.0790
##  6    4.5  585022 0.0585
##  7    1    384180 0.0384
##  8    2.5  370178 0.0370
##  9    1.5  118278 0.0118
## 10    0.5   94988 0.00950
```

```
# movie genre count by year in year descending order

moviecountperyear <- movielens2 %>%
select(movieId, releaseyear, genres) %>%
group_by(releaseyear, genres) %>%
summarise(count = n()) %>% arrange(desc(releaseyear))
```

```
head(moviecountperyear,10)
```

```
## # A tibble: 10 x 3
## # Groups:   releaseyear [1]
##    releaseyear genres      count
##          <dbl> <chr>       <int>
##  1        2008 Action      14337
##  2        2008 Adventure    9165
##  3        2008 Animation    2869
##  4        2008 Children     3068
##  5        2008 Comedy      13647
##  6        2008 Crime        6090
##  7        2008 Documentary   498
##  8        2008 Drama       10585
##  9        2008 Fantasy      3067
## 10        2008 Horror        725
```

```
# movie genre count by year in count descending order

moviecountperyear2 <- movielens2 %>%
select(movieId, releaseyear, genres) %>%
group_by(releaseyear, genres) %>%
summarise(count = n()) %>% arrange(desc(count))

head(moviecountperyear2,10)
```

```
## # A tibble: 10 x 3
## # Groups:   releaseyear [4]
##    releaseyear genres    count
##          <dbl> <chr>     <int>
##  1        1994 Comedy   382450
##  2        1994 Drama    371813
##  3        1995 Drama    368642
##  4        1995 Comedy   312825
##  5        1996 Comedy   285500
##  6        1995 Action   285041
##  7        1996 Drama    267592
##  8        1996 Thriller 253007
##  9        1995 Thriller 242377
## 10        1993 Drama    238811
```

```
# ratings by the release year of the movies

ratingsbyreleaseyear <- movielens %>% group_by(releaseyear) %>% summarize(mean_rating = mean(rating))
head(ratingsbyreleaseyear,10)
```

```
## # A tibble: 10 x 2
##    releaseyear mean_rating
##          <dbl>       <dbl>
##  1        1915        3.25
##  2        1916        3.77
##  3        1917        3.81
##  4        1918        3.65
##  5        1919        3.31
```

```
##  6        1920        3.93
##  7        1921        3.86
##  8        1922        3.91
##  9        1923        3.77
## 10        1924        3.94
```

```
# ratings by the rate year of the movies

mean_rating_by_rate_year <- movielens %>% group_by(rateyear) %>% summarize(mean_rating =mean(rating))
mean_rating_by_rate_year
```

```
## # A tibble: 15 x 2
##    rateyear mean_rating
##       <dbl>       <dbl>
##  1     1995        3.67
##  2     1996        3.54
##  3     1997        3.59
##  4     1998        3.51
##  5     1999        3.62
##  6     2000        3.58
##  7     2001        3.54
##  8     2002        3.47
##  9     2003        3.47
## 10     2004        3.43
## 11     2005        3.44
## 12     2006        3.47
## 13     2007        3.47
## 14     2008        3.54
## 15     2009        3.46
```

```
# this shows the mean ratings of each genre.

#movielens %>% summarize(mean(rating), sd(rating))

# if RMSE = sd, model somewhat predicts the mean
# if RMSE < sd, model shows ability to learn, depends on how much it is lower
# if RMSE > sd, model didnt even learn to guess mean correctly

# if overall accuracy between training and testing differs a lot, this can be due to overtrain that cau


rating_mean_sd_genre <- movielens2 %>% group_by(genres) %>% summarize(mean(rating), sd(rating))
rating_mean_sd_genre
```

```
## # A tibble: 20 x 3
##    genres             `mean(rating)` `sd(rating)`
##    <chr>                       <dbl>        <dbl>
##  1 (no genres listed)           3.64        1.11
##  2 Action                       3.42        1.07
##  3 Adventure                    3.49        1.05
##  4 Animation                    3.60        1.02
##  5 Children                     3.42        1.09
##  6 Comedy                       3.44        1.07
##  7 Crime                        3.67        1.01
##  8 Documentary                  3.78        1.00
##  9 Drama                        3.67        0.995
```

```
## 10 Fantasy                    3.50        1.07
## 11 Film-Noir                  4.01        0.886
## 12 Horror                     3.27        1.15
## 13 IMAX                       3.76        1.03
## 14 Musical                    3.56        1.06
## 15 Mystery                    3.68        1.000
## 16 Romance                    3.55        1.03
## 17 Sci-Fi                     3.40        1.09
## 18 Thriller                   3.51        1.03
## 19 War                        3.78        1.01
## 20 Western                    3.56        1.02
```

```
mean_sd_by_movieid <- movielens%>% group_by(movieId) %>% summarize(mean=mean(rating), sd=sd(rating),se=s
head(mean_sd_by_movieid)
```

```
## # A tibble: 6 x 4
##   movieId  mean    sd    se
##     <dbl> <dbl> <dbl> <dbl>
## 1   31692  2.25  2.47 1.24
## 2   33160  2.75  2.47 1.24
## 3   58898  3.38  2.14 0.534
## 4   26059  3     2.12 1.06
## 5   60983  3.5   2.12 1.06
## 6    6766  2.17  2.08 0.694
```

```
mean_sd_by_userid <- movielens%>% group_by(userId) %>% summarize(mean=mean(rating), sd=sd(rating),se=sd
head(mean_sd_by_userid)
```

```
## # A tibble: 6 x 4
##   userId  mean    sd     se
##    <int> <dbl> <dbl>  <dbl>
## 1  53601  2.75  2.31 0.115
## 2  59073  2.75  2.31 0.115
## 3   9269  2.86  2.30 0.110
## 4  58191  2.52  2.30 0.115
## 5  22236  3.34  2.18 0.0873
## 6  21871  2.03  2.11 0.0622
```

```
movielens%>% group_by(movieId) %>%
summarize(n=n(),howold=2019-first(releaseyear),title=title[1],mean=mean(rating)) %>%
mutate (hmray=n/howold)%>%
top_n(10,hmray) %>% #hmray=how many rates a year
arrange(desc(mean))
```

```
## # A tibble: 10 x 6
##    movieId     n howold title                                 mean hmray
##      <dbl> <int>  <dbl> <chr>                                <dbl> <dbl>
##  1     318 31126     25 Shawshank Redemption, The (1994)      4.46 1245.
##  2    2571 23229     20 Matrix, The (1999)                    4.21 1161.
##  3     593 33668     28 Silence of the Lambs, The (1991)      4.20 1202.
##  4     296 34864     25 Pulp Fiction (1994)                   4.16 1395.
##  5     110 29154     24 Braveheart (1995)                     4.08 1215.
##  6     356 34457     25 Forrest Gump (1994)                   4.01 1378.
##  7     457 28951     26 Fugitive, The (1993)                  4.01 1114.
##  8     150 27035     24 Apollo 13 (1995)                      3.89 1126.
##  9     480 32631     26 Jurassic Park (1993)                  3.66 1255.
```

```
## 10     780 26042     23 Independence Day (a.k.a. ID4) (1996)  3.38 1132.
```

If I am a movie maker, producing which genre of movie may provide the highest rating and possibly profit???

Of course, this question requires data for the production cost of each movie, and ticket sales and prices, along with digital content incomes, and so on. But, for simplicity, I assume'em all constant across the platform.

From visual inspection, I see that making War, Fantasy, and Crime movies tend to rate higher and since there are not many of those produced, which can be seen in proportions or in movie genre count by year section, these kinds of movies wont get lost in the jungle easily.

That's not to say that market demand analysis shouldn't be made. Each generation display different characteristics and therefore desire different products, although there are visible common denominators.

Let's try to show whetherthe visual inspection is true

Frequency table shows (on average, each user has n_freq rating for that genre) that due to high rating receival of comedy, drama, action, thriller, movie makers tend to produce more of those (which can be seen in count values as well), and therefore their mean tend to regress toward the total mean faster than the ones with low frequency, such as documentary.

Although the visual inspection turned out to be somewhat wrong, due to box office gross sums impact on production, now we learn that producers seem to make more of high grossing movies to maximize their profits which happen to fall into those categories...

> link to source of paraphrase

```r
# frequency of users voting(rating) for each genre

a <- data.frame(table(movielens2$genres))
names(a) <- c("genre", "genre sum")
c <- a[2]/(n_distinct(movielens2$userId))
names(c) <- "freq"
d <- cbind(a,c)
d %>% arrange(desc(freq))
```

```
##                genre genre sum         freq
## 1              Drama   4344198 6.216832e+01
## 2             Comedy   3934068 5.629909e+01
## 3             Action   2845349 4.071881e+01
## 4           Thriller   2584435 3.698496e+01
## 5          Adventure   2121074 3.035396e+01
## 6            Romance   1901883 2.721719e+01
## 7             Sci-Fi   1490489 2.132987e+01
## 8              Crime   1474957 2.110760e+01
## 9            Fantasy   1028482 1.471825e+01
## 10          Children    820149 1.173687e+01
## 11            Horror    768225 1.099380e+01
## 12           Mystery    630944 9.029222e+00
## 13               War    568063 8.129354e+00
```

```
## 14          Animation    519112 7.428833e+00
## 15            Musical    481174 6.885915e+00
## 16            Western    210459 3.011806e+00
## 17          Film-Noir    131592 1.883168e+00
## 18        Documentary    103454 1.480495e+00
## 19               IMAX      9080 1.299408e-01
## 20 (no genres listed)         7 1.001746e-04
```

since many people prefer to go to theatre and many movies are released on fridays,

Why are movies released on Thursdays?

it's plausible to observe a higher rating on fridays and weekends.

seeing higher mean rating on mondays may be because professional movie critiques are only ready by then,

as well as people give it some time to digest the movie.

mid weekdays (tuesday, wednesday, and thursday) offer the lowest average rating.

most people work the mid-weekdays, even if time zone differences accross the globe is taken into consideration,

assuming that data collection is not time-zone variant

```
weekday_mean_sd <- movielens%>% group_by(weekday) %>% summarize(mean=mean(rating), sd=sd(rating)) %>% a
weekday_mean_sd
```

```
## # A tibble: 7 x 3
##   weekday    mean    sd
##   <chr>     <dbl> <dbl>
## 1 Saturday   3.53  1.06
## 2 Sunday     3.52  1.06
## 3 Monday     3.52  1.06
## 4 Friday     3.51  1.06
## 5 Tuesday    3.51  1.06
## 6 Wednesday  3.50  1.06
## 7 Thursday   3.50  1.06
```

Correlation between year of rating collected and how old a movie is

in time, rating accumulate, therefore it is expected that older movies to have more rating

on the other hand, new generation is more tech savvy,

so the newly produced movies that attract young generation to receive more reviews and rate.

as it can be seen, older the movie, higher its rating.(cor(rating,age_of_movie)=.1143)

also, if the rate collection year is further into the future than its release year, it's got a positive impact on rating

cor(rating,rate_release_dif)=.1039

```
corr <- movielens2 %>% select(rating,userId,movieId,releaseyear,age_of_movie,rateyear)
cor(corr)
```

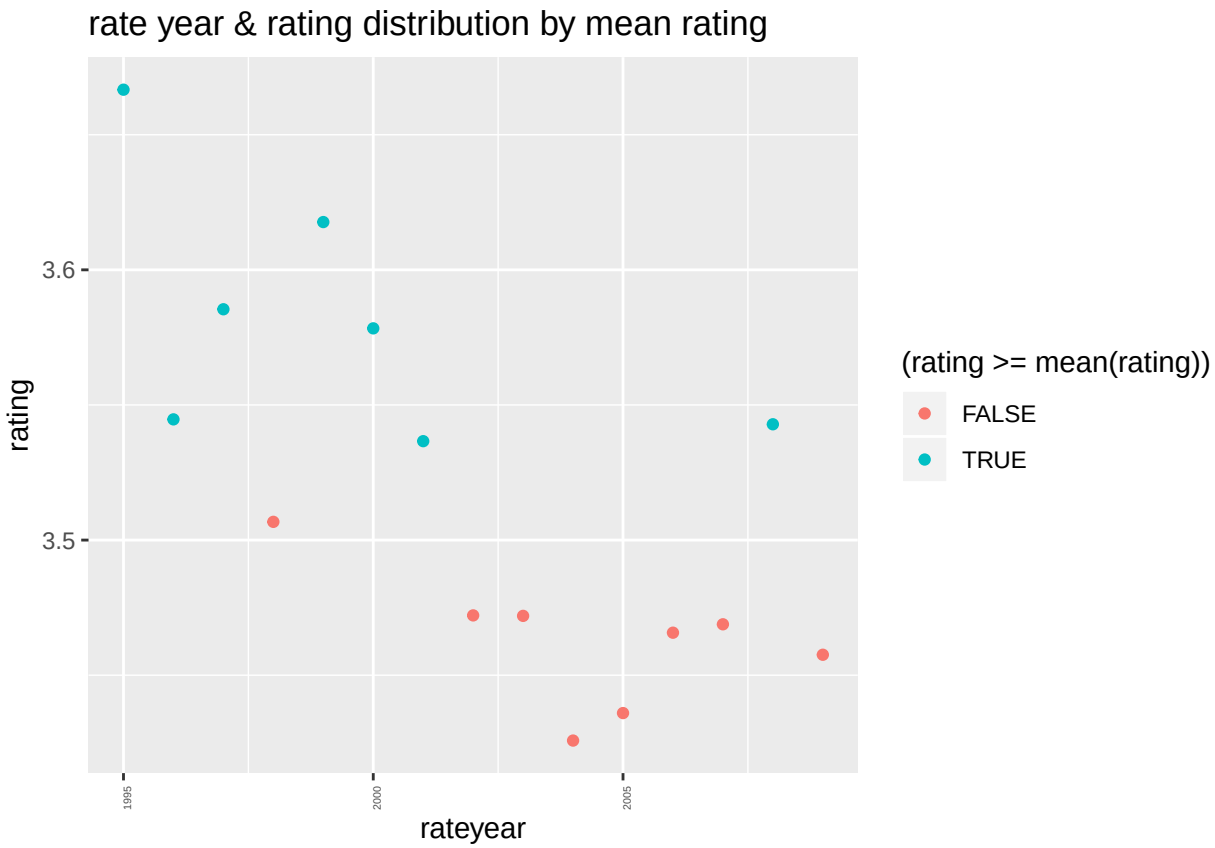
```
##                    rating        userId      movieId   releaseyear
## rating        1.000000000  0.0027416665 -0.006862508 -0.1131128726
## userId        0.002741666  1.0000000000  0.004419056  0.0004581293
## movieId      -0.006862508  0.0044190555  1.000000000  0.2571778395
## releaseyear  -0.113112873  0.0004581293  0.257177839  1.0000000000
## age_of_movie  0.113112873 -0.0004581293 -0.257177839 -1.0000000000
## rateyear     -0.035905327  0.0159283095  0.374035353  0.1100008633
##                age_of_movie    rateyear
## rating        0.1131128726 -0.03590533
## userId       -0.0004581293  0.01592831
## movieId      -0.2571778395  0.37403535
## releaseyear  -1.0000000000  0.11000086
## age_of_movie  1.0000000000 -0.11000086
## rateyear     -0.1100008633  1.00000000
```

Data Visualization

In this graph, i sort the ratings by the rate year. As it is clear from the graph that, given enough time, ratings get closer to the mean value. But, the businesses, due to ever advancing technology and fast adopting and fast consuming society, may not pay much attention to this, because although the rates regress to the mean in the long run, the most profits are realized in the short and in some instances medium run. As Keynes said: "in the long run, we are all dead."

```
#print(paste((n_distinct(movielens$rateyear)), "unique dates"))
#157 unique dates

movielens %>% group_by(rateyear) %>%
summarize(rating = mean(rating)) %>%
ggplot(aes(rateyear, rating, color=(rating>=mean(rating)))) +
geom_point() +
theme(text = element_text(),axis.text.x = element_text(size = 4,angle = 90, hjust = 0, vjust = 1, face =
ggtitle("rate year & rating distribution by mean rating")
```



it looks like movie ratings across the year differ between the rates of movies by release year and the rates of movies by rate year.

release year refers to the release date of the movie and the ratings by release reflect the rates the movie received by that year, on average

rate year refers to when rates are collected since the release time, which spans over several years/decades in many instances.

average rating should be about 3.52

ratings which are collected over a period time, rather the just the release year rating, reflect the mean rating better. In fact, though the earliest movie in the dataset dates back to 1915, rating collection starts at 1995.

it also proves the point of 'regress to the mean' concept.

additionally, by looking at the release year rate, movies that had been produced between roughly 1920 and 1970 received better ratings when they were released, compared to the 1990s and 2000s.

categorically, what genres of movies pushed the overall rating higher than mean from 1920 to 1970??? (answer is given below)

also, by looking at the release year rating graph, it is almost as if there are two clusters, which is separated by the year 1980 that acts like a cutoff point, in where sudden drop can be observed. For simplicity, i cut off the two clusters by mean rating.

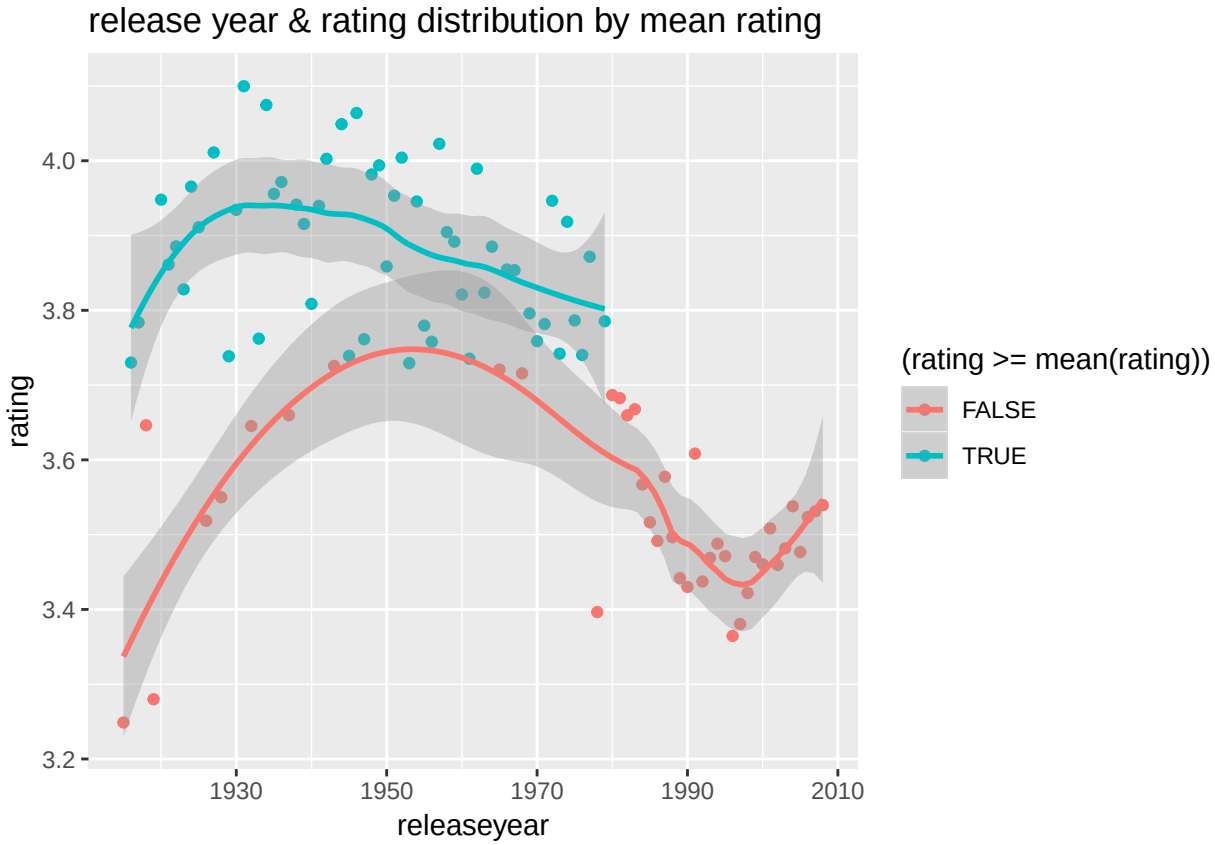
"In the 1980's, the past creativity of the film industry became homogenized and overly marketable. Designed only for audience appeal, most 1980's feature films were considered generic and few became classics. This decade is recognized as the introduction of high concept films that could be easily described in 25 words or less, which made the movies of this time more marketable, understandable, and culturally accessible."....

> link to source of quotation

```
#print(paste((n_distinct(validation3$releaseyear)), "unique dates"))
#94 unique dates

movielens2 %>% group_by(releaseyear) %>%
summarize(rating = mean(rating)) %>%
ggplot(aes(releaseyear, rating,color=(rating>=mean(rating)))) +
geom_point() +
geom_smooth(span=0.5)+
ggtitle("release year & rating distribution by mean rating")
```

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



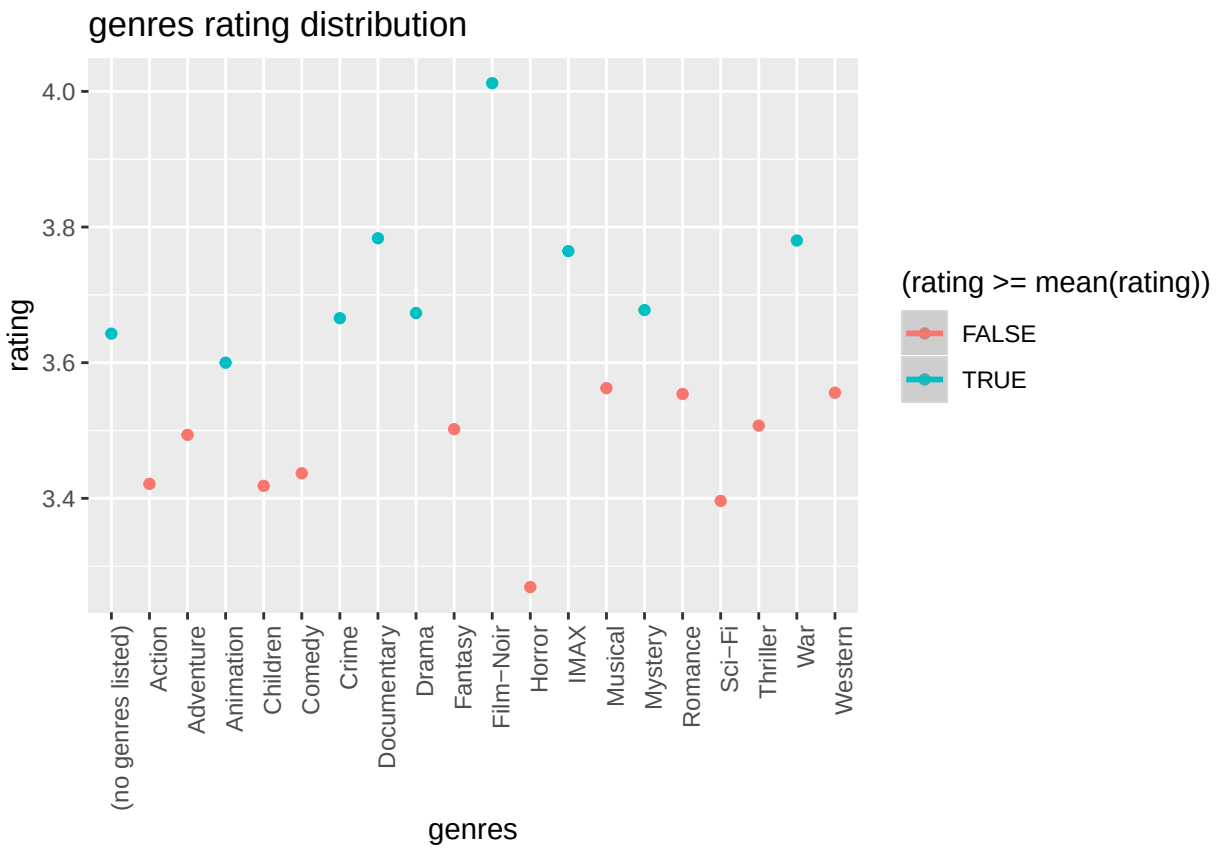
In this graph, we can observe that some genres perform better in ratings than others, such as documentary (about 0.4% of total movies in this dataset), drama (about 16.7%), animation

(2%). Even though comedy, action, and thriller genres take a larger portion of the dataset, they were rated lower than the average. which says that movies belong to these genres produce quantity but not quality.

```
#print(paste((n_distinct(validation3$releaseyear)), "unique dates"))
#94 unique dates

movielens2 %>% group_by(genres) %>%
summarize(rating = mean(rating)) %>%
ggplot(aes(genres, rating,color=(rating>=mean(rating)))) +
geom_point() +
geom_smooth(span=0.2)+
theme(axis.text.x = element_text(angle = 90, hjust = 1))+
ggtitle("genres rating distribution")
```

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



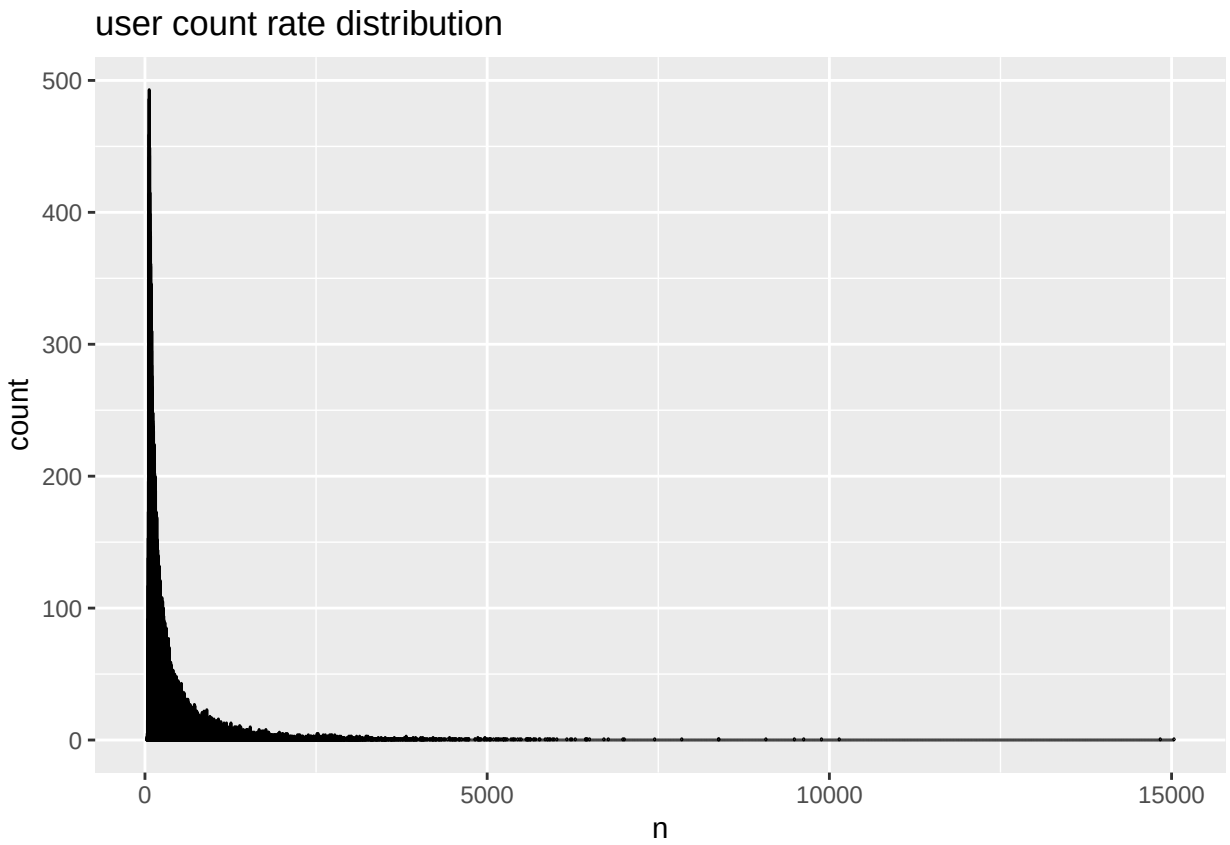
This graph shows the ratings distrubution. As it can be clearly seen that the more whole number ratings were given than the fraction ratings.

```
movielens2 %>%
  ggplot(aes(rating)) +
  geom_histogram(binwidth=0.2, fill="black",color="black") +
  ggtitle("Rating Distribution seq(0, 5, 0.5)")
```

```
## Warning: Computation failed in `stat_bin()`:
## cannot allocate vector of size 198.1 Mb
```

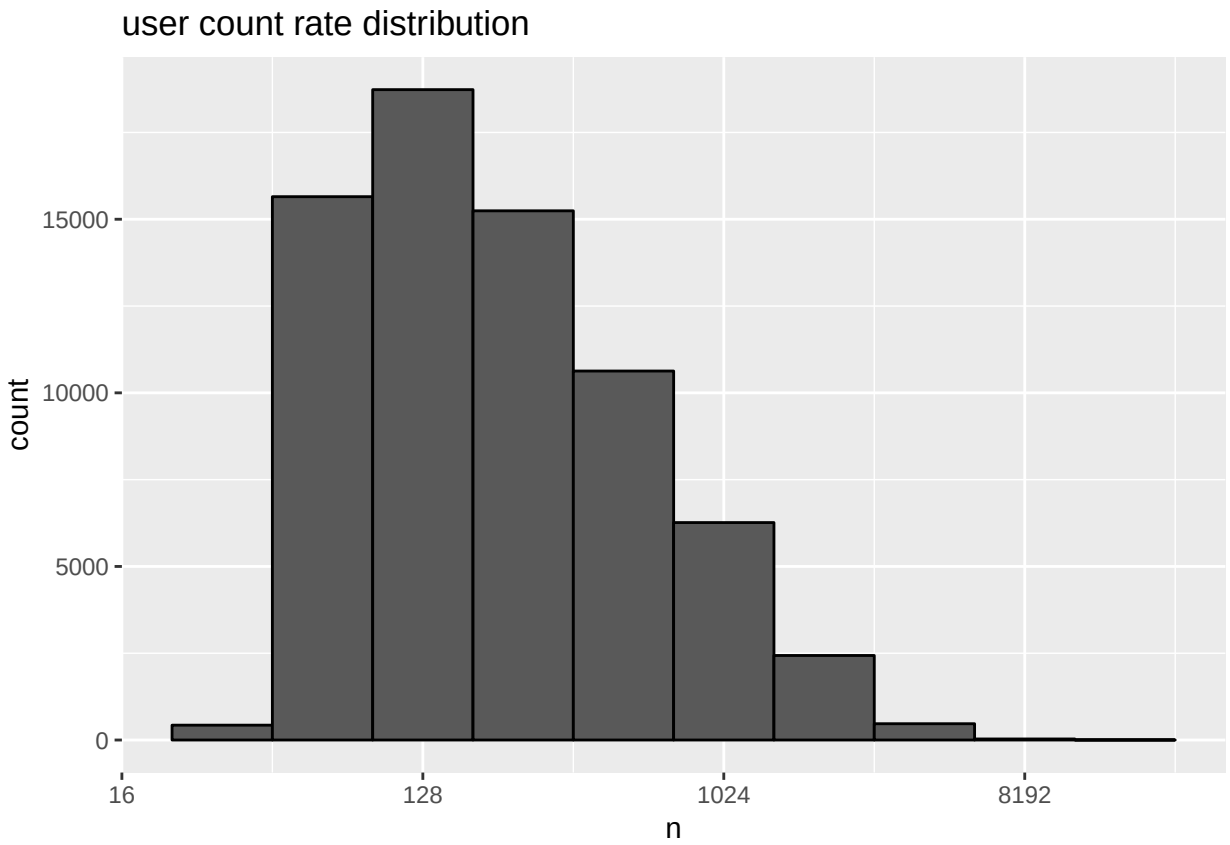

```
# most users rated movies in single digits.

movielens2 %>%
  count(userId) %>%
  ggplot(aes(n)) +
  geom_histogram(binwidth = 1, color = "black") +
  #scale_x_continuous(trans="log2")+
  ggtitle("user count rate distribution")
```



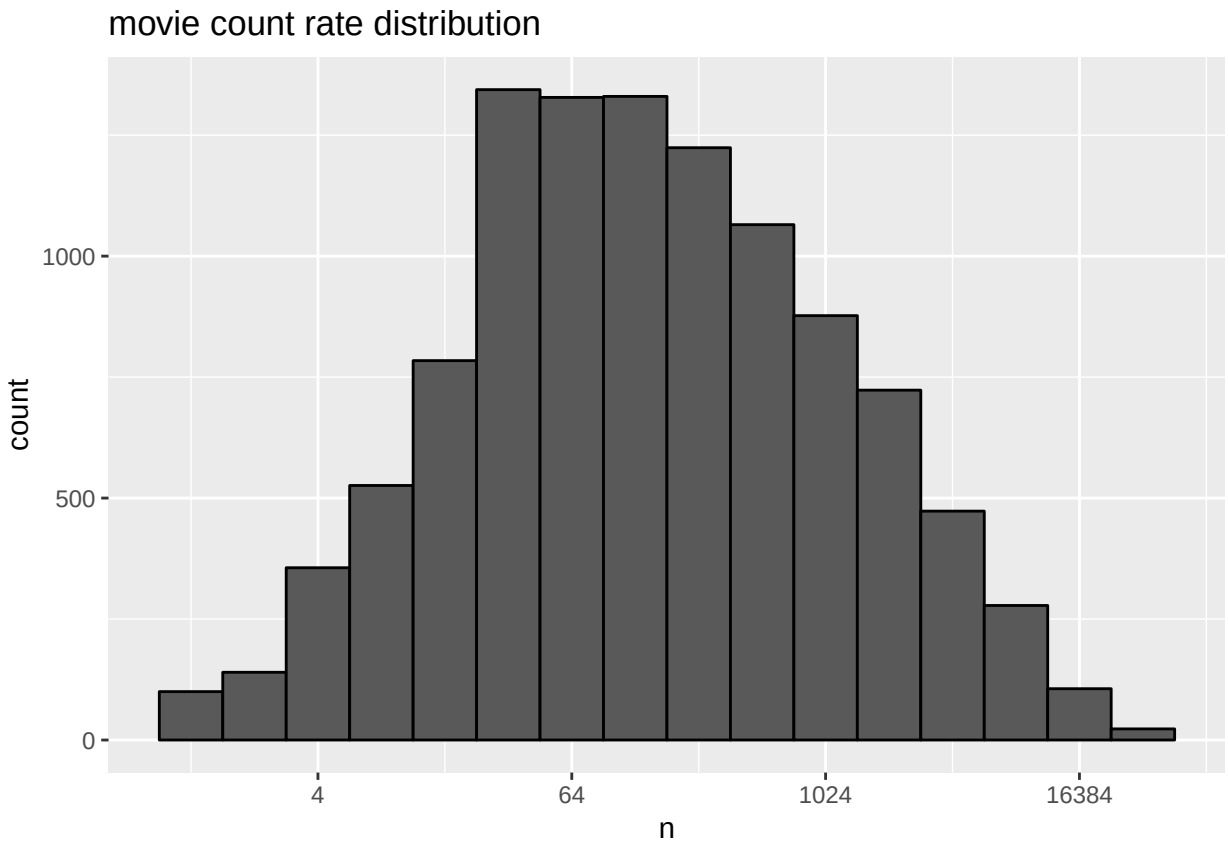

```
# most users rated movies in single digits. To make better inference, i log the the n to visualize more

movielens2 %>%
  count(userId) %>%
  ggplot(aes(n)) +
  geom_histogram(binwidth = 1, color = "black") +
  scale_x_continuous(trans="log2")+
  ggtitle("user count rate distribution")
```




```
# movies ratings count distribution

movielens %>%
  count(movieId) %>%
  ggplot(aes(n)) +
  geom_histogram(binwidth = 1, color = "black") +
  scale_x_continuous(trans="log2")+
  ggtitle("movie count rate distribution")
```



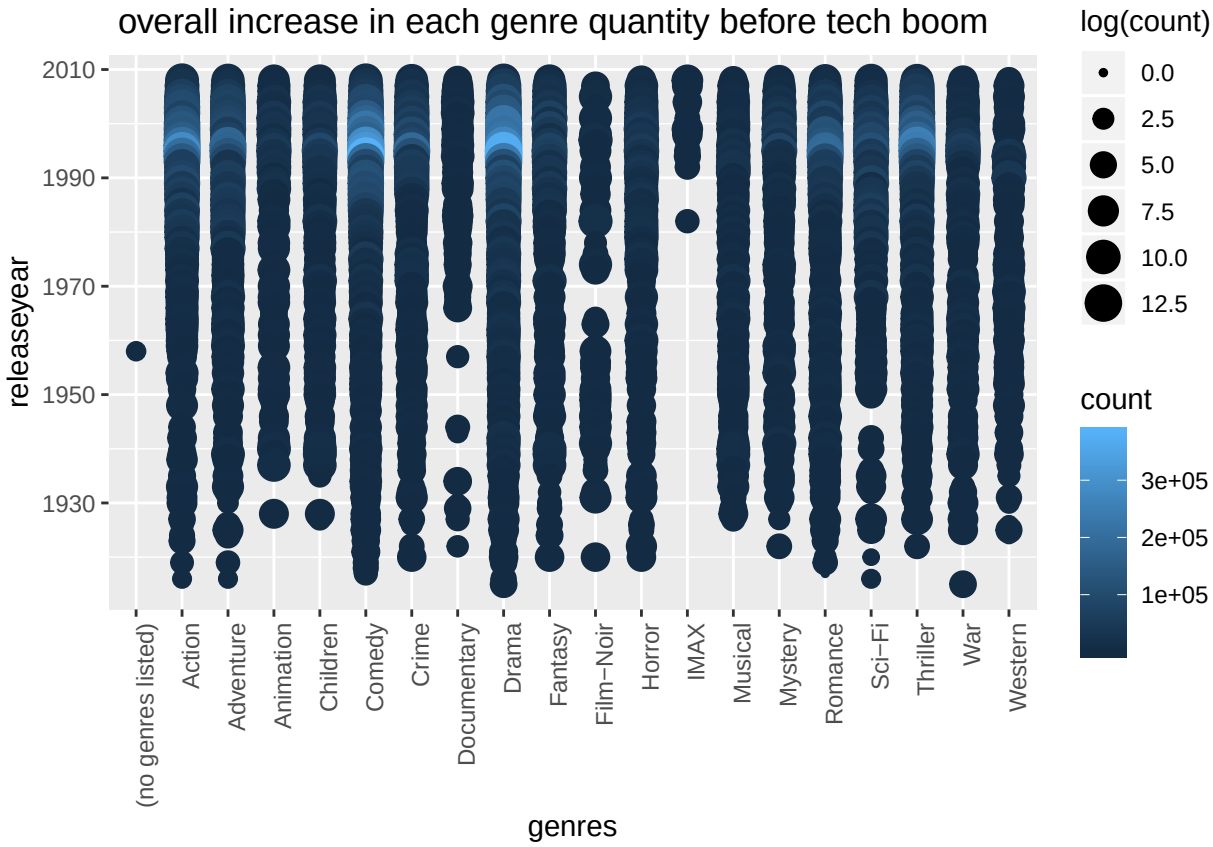
this shows that mid 1990s, almost every genre had an increase in count, meaning more produced films

this is just before the tech boom

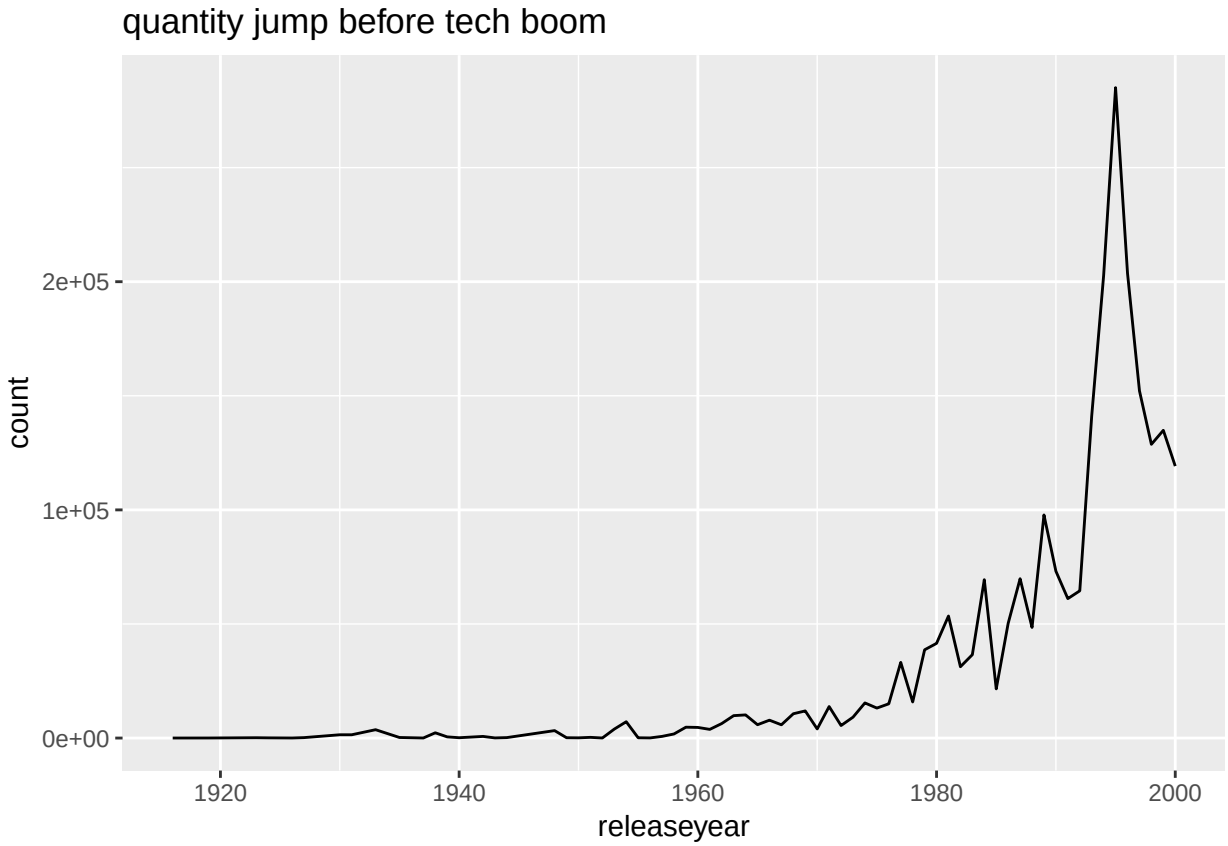
people's expectation were high

that leads to more consumption of leisure time

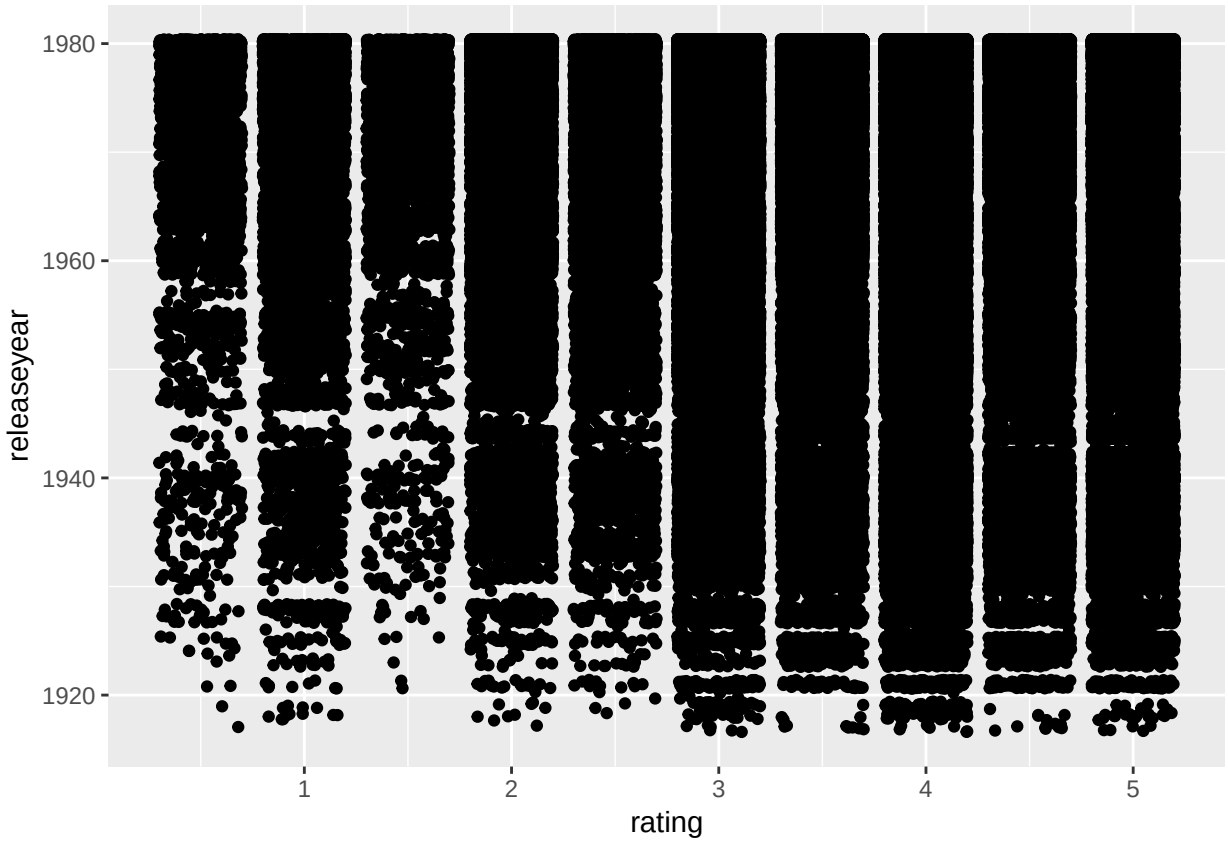
```
moviecountperyear %>%
    ggplot(aes(genres,releaseyear, color=count)) +
    geom_point(aes(size=log(count))) +
    theme(axis.text.x = element_text(angle = 90, hjust = 1))+
    ggtitle(" overall increase in each genre quantity before tech boom")
```




```
# this is another way of seeing increase in movie quantity just before the tech boom
moviecountperyear %>%
    filter(releaseyear <= 2000 & genres=="Action") %>%
    ggplot(aes(releaseyear, count)) +
    geom_line() +
    ggtitle("quantity jump before tech boom")
```



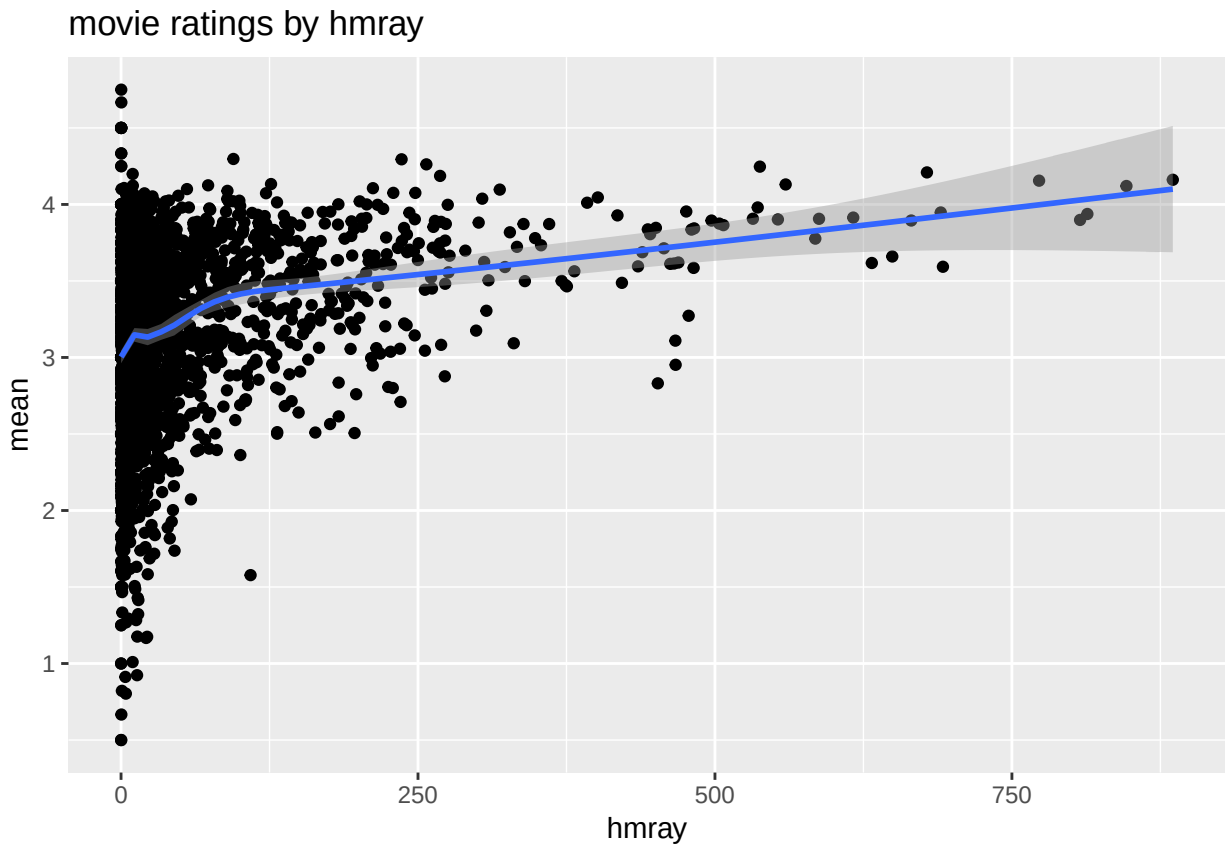

```
movielens2 %>%
    filter(releaseyear <= 1980 & genres %in% c("Comedy")) %>%
    ggplot(aes(rating,releaseyear))+
    geom_jitter()
```




```
# compare this to >=1980, it's clear to see that after 1980, movie industry started mass producing,
# therefore the quantity overcame the quality,
# thus the distribution of ratings almost did even out.
# before 1980, most comedy movies were rated between 3 and 5.
# but after 1980, range became 1 and 5
# the same idea can be applied to other genres
```

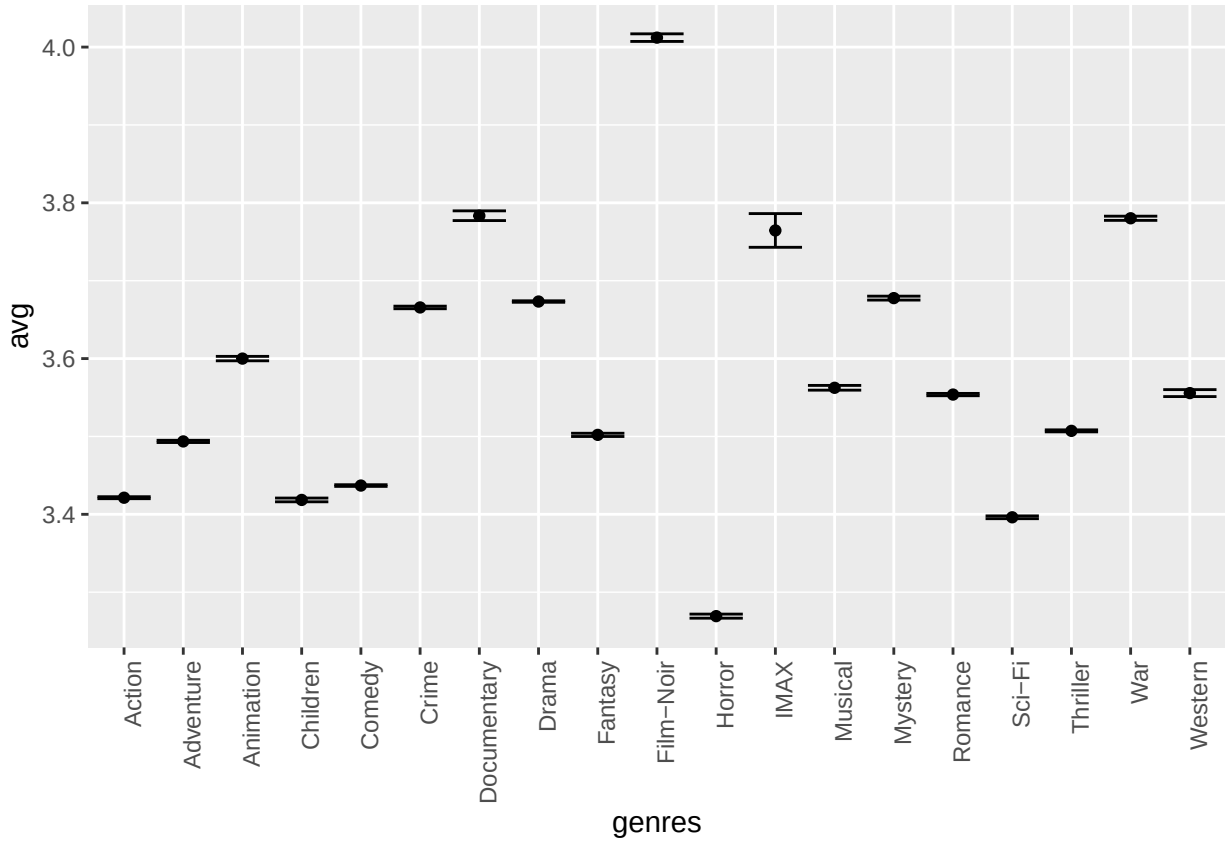
```
movielens %>%
filter(releaseyear >= 2000) %>%
group_by(movieId) %>%
summarize(n=n(),howold=2019-first(releaseyear),title=title[1],mean=mean(rating)) %>%
mutate (hmray=n/howold)%>% #hmray=how many rates a year
ggplot(aes(hmray, mean)) +
geom_point() +
geom_smooth() +
ggtitle("movie ratings by hmray")
```

```
## `geom_smooth()` using method = 'gam' and formula 'y ~ s(x, bs = "cs")'
```




```
# over time, ratings stabilize at around the mean
```

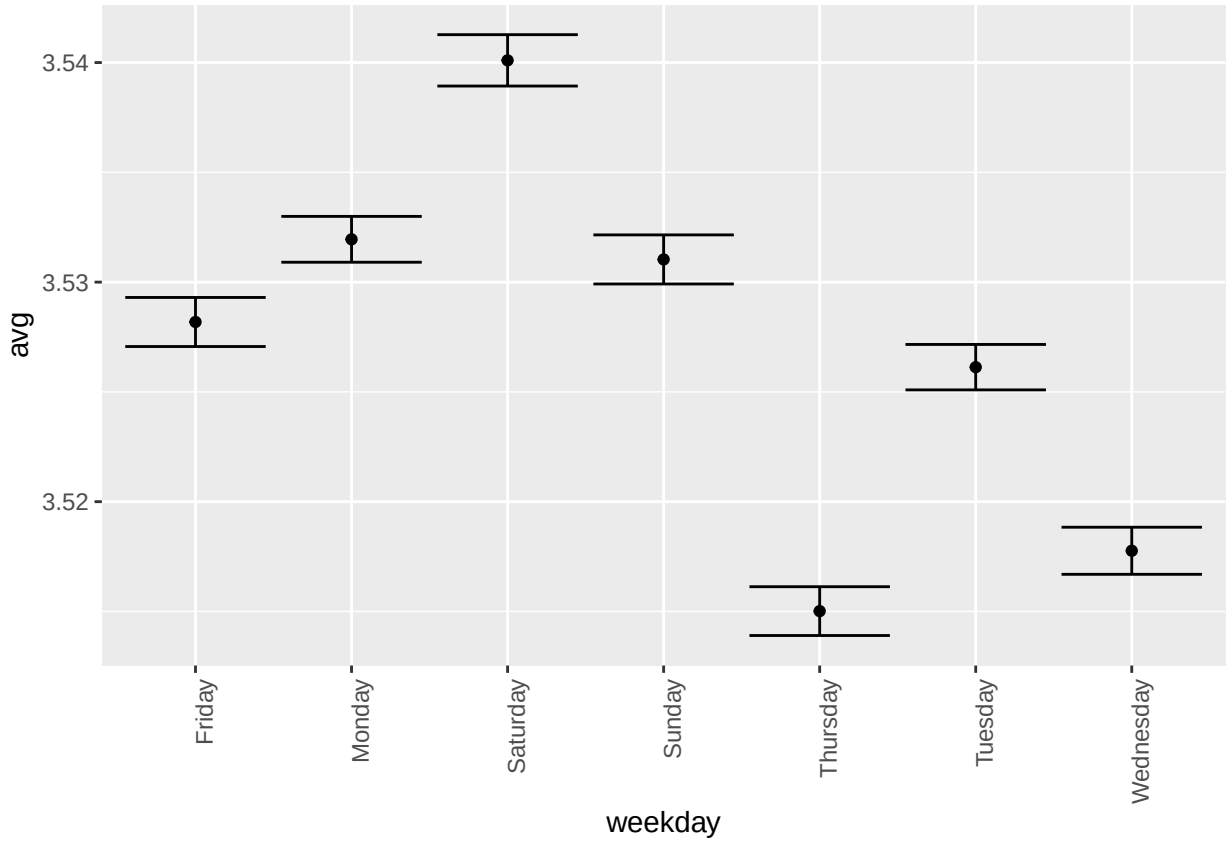
```
movielens2 %>% group_by(genres) %>%
    summarize(n = n(), avg = mean(rating), se = sd(rating)/sqrt(n())) %>%
    filter(n >= 100) %>%
    mutate(reorder(genres, avg)) %>%
    ggplot(aes(x = genres, y = avg, ymin = avg - 2*se, ymax = avg + 2*se)) +
    geom_point() +
    geom_errorbar() +
    theme(axis.text.x = element_text(angle = 90, hjust = 1))
```




```
    ggtitle("average movie ratings by genre")
```

```
## $title
## [1] "average movie ratings by genre"
##
## attr(,"class")
## [1] "labels"
```

```
movielens2 %>% group_by(weekday) %>%
    summarize(n = n(), avg = mean(rating), se = sd(rating)/sqrt(n())) %>%
    filter(n >= 1000) %>%
    mutate(reorder(weekday, avg)) %>%
    ggplot(aes(x = weekday, y = avg, ymin = avg - 2*se, ymax = avg + 2*se)) +
    geom_point() +
    geom_errorbar() +
    theme(axis.text.x = element_text(angle = 90, hjust = 1))
```

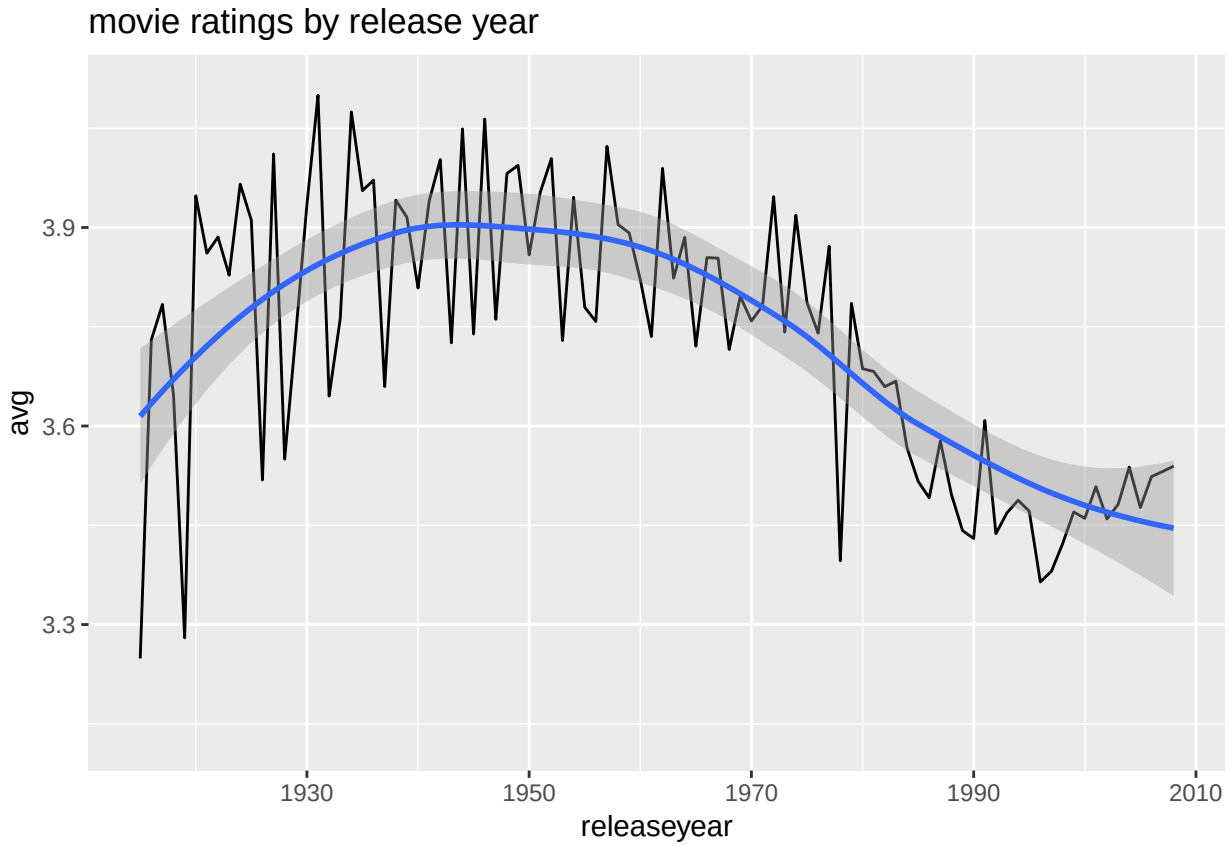



```
    ggtitle("average movie ratings by week day")
```

```
## $title
## [1] "average movie ratings by week day"
##
## attr(,"class")
## [1] "labels"
```

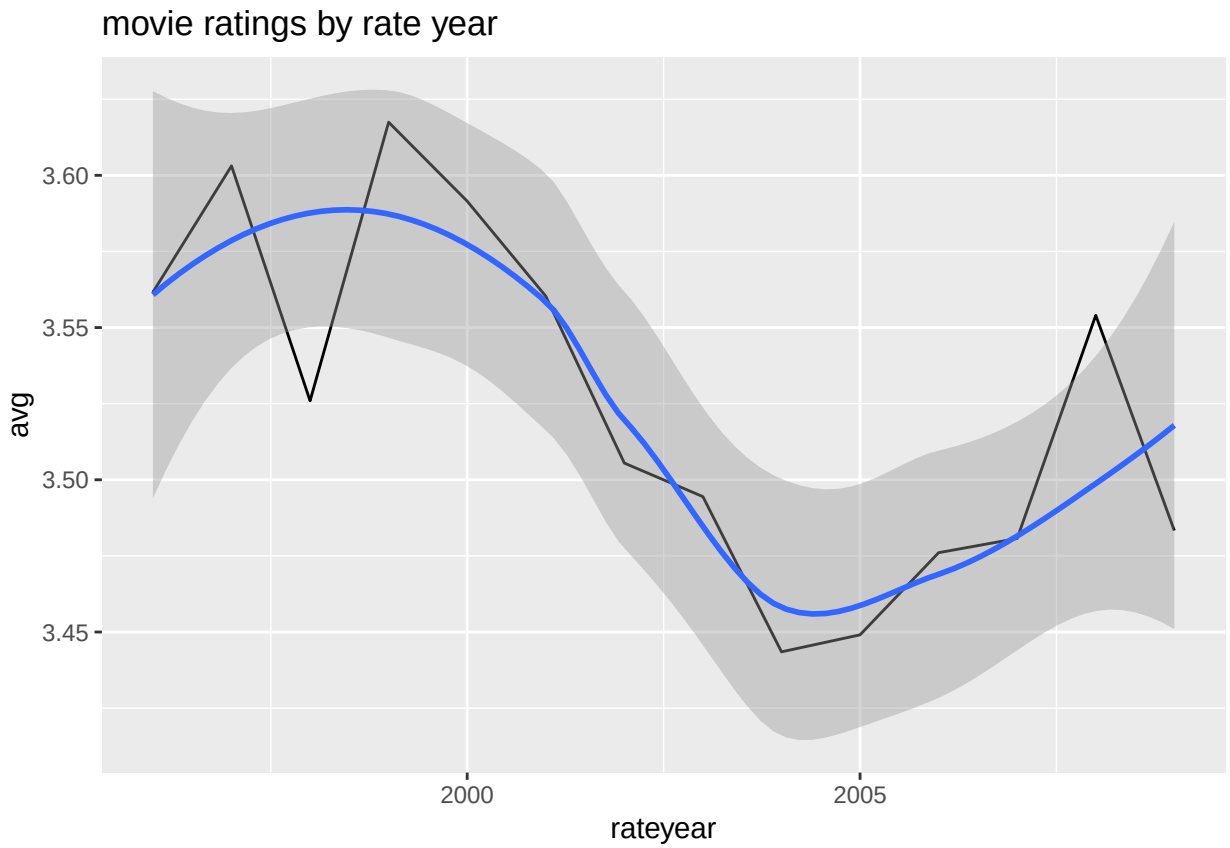
```
movielens2 %>% group_by(releaseyear) %>%
    summarize(n = n(), avg = mean(rating), se = sd(rating)/sqrt(n())) %>%
    filter(n >= 5) %>%
    mutate(reorder(releaseyear, avg)) %>%
    ggplot(aes(x = releaseyear, y = avg, ymin = avg - 2*se, ymax = avg + 2*se)) +
    geom_line() +
    geom_smooth() +
    ggtitle("movie ratings by release year")
```

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




```
movielens2 %>% group_by(rateyear) %>%
    summarize(n = n(), avg = mean(rating), se = sd(rating)/sqrt(n())) %>%
    filter(n >= 10) %>%
    mutate(reorder(rateyear, avg)) %>%
    ggplot(aes(x = rateyear, y = avg, ymin = avg - 2*se, ymax = avg + 2*se)) +
    geom_line() +
    geom_smooth() +
    ggtitle("movie ratings by rate year")
```

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



Creating a Validation set

```
# Validation set will be 10% of MovieLens data

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)
```

```
## Joining, by = c("userId", "movieId", "rating", "timestamp", "title", "genres", "timestamp2", "releas
```

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

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

```
length(validation$rating)
```

```
## [1] 999999
```

```
length(edx$rating)
```

```
## [1] 9000055
```

```
head(validation,1)
```

```
##   userId movieId rating timestamp                title genres
## 1      1     231      5 838983392 Dumb & Dumber (1994) Comedy
##            timestamp2 releaseyear age_of_movie rateyear ratemonth_numeric
## 1 1996-08-02 03:56:32        1994           25     1996                 8
##   ratemonth weekday rate_release_dif
## 1    August  Friday                2
```

RMSE

RMSE:

$$\text{RMSE} = \sqrt{\text{MSE}} = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(\hat{y}_i - y_i)^2}$$

sqrt(mean(pred - obs)^2)

> link to source of RMSE equation

if RMSE = sd, model somewhat predicts the mean

if RMSE < sd, model shows ability to learn, depends on how much it is lower

if RMSE > sd, model didnt even learn to guess mean correctly

if overall accuracy between training and testing differs a lot, this can be due to overtrain that causes over fitting.

there are users that rate only one movie, i.e. userId==“1”

there are other users that rate certain genre movies higher, i.e. userId==“10”

therefore, each user carries a certain bias in his/her decision toward certain movies and genres

RMSE = sqrt(1/n (r_hat_u_m - r_u_m))

r_hat_u_m = predicted rating for movie m and user u.

r_u_m = test set rating for movie m and user u.

b_u = bias for user u

b_m = this provides an amount to diminish the effect of user bias

ratings from each user was centered around zero by removing the mean and then divide the sum of centered means by the count of values within.

ratings for each movie was centered around zero by removing the mean and then divide the sum of centered means by the count of values within.

in other words, it could be seen as deriving standard errors out of each sub group of userID and movieId.

I have tried n() function of dplyr library to divide it by the frequency of each sub-group of userId and movieId, so that I can reduce the impact of variations.

second, I tried ddply functionality of plyr library, again, to have a better estimate

```
mu <- mean(edx$rating)
paste0("sd is ", sd(validation$rating))
```

```
## [1] "sd is 1.06120225268297"
```

```
paste0("baseline RMSE is ", RMSE(mu,validation$rating))
```

```
## [1] "baseline RMSE is 1.06120181029262"
```

by using ddply functionality of plyr library

```
#### fyi: running plyr and dplyr library at the same time causes an error.
# ```R
# cdata_movieId <- ddply(edx, c("movieId"), summarise,
#                 N = length(movieId),
#                 group_rating = mean(rating),
#                 group_se_by_movie = (group_rating - mean(edx$rating))/N
# )
# cdata_movieId <- cdata_movieId %>% arrange(desc(group_se_by_movie))
# head(cdata_movieId)
# anyNA(cdata_movieId)
# showmissing <- cdata_movieId[!complete.cases(cdata_movieId),]
# ```
#
#
#
#
# ```R
# cdata_userId <- ddply(edx, c("userId"), summarise,
#                 N = length(userId),
#                 group_rating = mean(rating),
#                 group_se_by_user = (group_rating - mean(edx$rating))/N
# )
# cdata_userId <- cdata_userId %>% arrange(desc(group_se_by_user))
# head(cdata_userId)
# anyNA(cdata_userId)
# showmissing <- cdata_userId[!complete.cases(cdata_userId),]
# ```
```

```
#
#
# ```R
# MyMerge <- function(x, y){
#   df <- merge(x, y, by= "userId", all.x= TRUE, all.y= TRUE)
#   return(df)
# }
# cdata_merged_edx <- Reduce(MyMerge, list(edx,cdata_userId))
# head(cdata_merged_edx) %>% arrange(desc(group_se_by_user))
# ```
#
#
#
# ```R
# MyMerge <- function(x, y){
#   df <- merge(x, y, by= "movieId", all.x= TRUE, all.y= TRUE)
#   return(df)
# }
# cdata_merged_edx <- Reduce(MyMerge, list(cdata_merged_edx,cdata_movieId))
# head(cdata_merged_edx)  %>% arrange(desc(group_se_by_movie))
# ```
#
#
# ```R
# cdata_merged_edx <- cdata_merged_edx %>% mutate(adj_rating = mean(cdata_merged_edx$group_rating.x) + g
# head(cdata_merged_edx)
# ```
#
#
# ```R
# paste0("edx RMSE is: ",RMSE(cdata_merged_edx$rating,cdata_merged_edx$adj_rating))
# ```
#
#
# ```R
# cdata_movieId <- ddply(validation, c("movieId"), summarise,
#                N = length(movieId),
#                group_rating = mean(rating),
#                group_se_by_movie = (group_rating - mean(validation$rating))/N
# )
# cdata_movieId <- cdata_movieId %>% arrange(desc(group_se_by_movie))
# head(cdata_movieId)
# anyNA(cdata_movieId)
# showmissing <- cdata_movieId[!complete.cases(cdata_movieId),]
# ```
#
#
# ```R
# cdata_userId <- ddply(validation, c("userId"), summarise,
#                N = length(userId),
#                group_rating = mean(rating),
#                group_se_by_user = (group_rating - mean(validation$rating))/N
# )
```

```
# cdata_userId <- cdata_userId %>% arrange(desc(group_se_by_user))
# head(cdata_userId)
# anyNA(cdata_userId)
# showmissing <- cdata_userId[!complete.cases(cdata_userId),]
# ```
#
#
# ```R
# MyMerge <- function(x, y){
#   df <- merge(x, y, by= "userId", all.x= TRUE, all.y= TRUE)
#   return(df)
# }
# cdata_merged_validation <- Reduce(MyMerge, list(validation,cdata_userId))
# head(cdata_merged_validation) %>% arrange(desc(group_se_by_user))
# ```
#
#
# ```R
# MyMerge <- function(x, y){
#   df <- merge(x, y, by= "movieId", all.x= TRUE, all.y= TRUE)
#   return(df)
# }
# cdata_merged_validation <- Reduce(MyMerge, list(cdata_merged_validation,cdata_movieId))
# head(cdata_merged_validation)  %>% arrange(desc(group_se_by_movie))
# ```
#
#
# ```R
# cdata_merged_validation <- cdata_merged_validation %>% mutate(adj_rating = mean(cdata_merged_validati
# head(cdata_merged_validation)
# ```
#
#
# ```R
# paste0("validation RMSE is: ",RMSE(cdata_merged_validation$rating,cdata_merged_validation$adj_rating)
# ```
```

by using n() functionality of dyplr library

```
mu <- mean(edx$rating)

bi <- edx %>%
group_by(movieId) %>%
summarize(bi= sum(rating - mu)/(n()))
```

```
bu <- edx %>%
group_by(userId) %>%
summarize(bu= sum(rating - mu)/(n()))
```

```
predicted_ratings3 <- edx %>%
    left_join(bi, by = "movieId") %>%
    left_join(bu, by = "userId") %>%
    mutate(pred = mu+bu+bi) %>% .$pred
```

```
#plot(predicted_ratings3)
#plot(movielens$rating)
```

```
paste0("edx set RMSE is: ",RMSE(predicted_ratings3,edx$rating))
```

```
## [1] "edx set RMSE is: 0.876753351274179"
```

by using n() functionality of dyplr library

```
mu <- mean(validation$rating)

bi <- validation %>%
group_by(movieId) %>%
summarize(bi= sum(rating - mu)/(n()))
```

```
bu <- validation %>%
group_by(userId) %>%
summarize(bu= sum(rating - mu)/(n()))
```

```
predicted_ratings4 <- validation %>%
    left_join(bi, by = "movieId") %>%
    left_join(bu, by = "userId") %>%
    mutate(pred = mu+bu+bi) %>% .$pred
```

```
#plot(predicted_ratings4)
#plot(movielens$rating)
```

```
paste0("validation set RMSE is: ",RMSE(predicted_ratings4,validation$rating))
```

```
## [1] "validation set RMSE is: 0.853425092420938"
```

Conclusion

Recommender systems are parts of our daily lives. Understanding the preferences of each individual to better suit them with the products is the goal of every profit maximizing business. Therefore, the importance of it is undeniable. From a movie to watch on any given day, to what to wear at events, to what to reads, and on, recommendation tools play such a critical role in shaping the demand creation. Given the current structure of social media and their genius ability to collect data about the people's choices in such detail enable marketing to reach an incredibly accurate efficiency levels. By filtering those choice in a content base or collaborative, we can match consumers with the right products at the right time.

For this assignment, I have tried multiple models to reduce RMSE. Some models, unfortunately, due to RAM constraints, didn't produce results. Because of this, just to see whether they work, I exported a 1% of the data to check the code and it works fine. But, on the full data, my laptop is unable to produce any results and warn me for being unable to "allocate a vector size of 7.3 Gb."

On the partial data, the models that I specified are able to reduce RMSE between 1% and 5%, but that stills keeps me above the specified threshold.

To overcome this problem, I applied the model that's been created by Hans Bystrom at Stanford University. In his paper of "Movie Recommendations from User Ratings", he stated this equation: bui = mu + bu + bi.

After obtaining the bu and bi values that are stated in this paper by using the n() function from dpylr and ddply function from plyr, I was finally able to reduce RMSE even further and eventually pass the target.

Though more work is required to get a better picture of ways of reducing RMSE, my preliminary results are able to reduce RMSE below the asked threshold.

RMSE for Validation set is 0.8534

RMSE for edx set is .8767

Baseline RMSE is 1.061