Capstone Project: Recommendation System using MovieLens Dataset

Bruno Ribeiro

Abstract: Recommendation systems are a common tool among e-commerce, social media and content-based websites. Companies like Amazon and Netflix try to predict what rating a specific user would give to a specific item in order to recommended that item to that user. This project's goal was to build a Movie Recommendation System based on the rating a specific user would give to a specific movie, using the MovieLens dataset with 10,000,000 entries. The performance of the algorithm was measured based on the residual mean squared error - RMSE. To the algorithm to be acceptable, it should reach a RMSE $\leq$ 0.87750 in our validation set. Our model was a linear regression considering the "Global Effects" and reached a RMSE $\approx$ 0.8641207 in the validation set. Since 0.8641207 < 0.87750, the model that we built met the parameters to be considered acceptable.

1. Introduction

Recommendation systems, also know as recommender systems or recommendation engines, consist in a Machine Learning technique that analyzes available data to make suggestions for something that a user might be interested in, such as a book, a movie or a song. Just as an example, a search engine is one type of recommendation engine, responding to search queries with pages of results that are the search engine's best suggestions for websites that satisfy the user's query, based on the search term plus other data, such as location and trending topics.

These engines are common among e-commerce, social media and content-based websites. Companies like Amazon and Netflix, that have many customers, are able to collect massive datasets that can be used to predict if a particular user will like a specific item. Usually, these companies try to predict what *rating* user will give to a specific item and if a high rating is predicted then the item is recommended to that user.

Recommendation systems are a useful alternative to search algorithms since they help users discover items they might not have found otherwise. Since they are utilized in a variety of areas like movies, music, news, books, research articles, search queries, social tags, and products in general, they are one of the most successful and widespread application of machine learning technologies in business.

This project's goal was to build a Movie Recommendation System, based on the rating a specific user would give to a specific movie, and it was a requirement to achieve a passing grade in the Data Science Professional Certificate Program, provided by Harvard University through edX.

In this project we've done, primarily, do the following key steps:

1. Analysed the gathered data;
2. Defined the predictors that would be used to predict the rating a user would give to a movie;
3. Trained and evaluated a machine learning algorithm;
4. Validated the algorithm that was built.

1.1. The Data

The data used in this project was obtained from GroupLens Research and it consists of 10,000,054 ratings and 95,580 tags applied to 10,681 movies by 71,567 users of the online movie recommender service MovieLens, that were collected over various periods of time. As stated by GroupLens, users were selected at random for inclusion and all of them had rated at least 20 movies. Also, their ids have been anonymized.

The downloaded data was a *zip* file that contained three *dat extension* files encoded as UTF-8:

1. ***ratings.dat***: all ratings are contained in this file. Each line of this file represents one rating of one movie by one user, and has the following format: UserID::MovieID::Rating::Timestamp. The lines within this file are ordered first by UserID, then, within user, by MovieID. Ratings are made on a 5-star scale, with half-star increments. Timestamps represent seconds since midnight Coordinated Universal Time (UTC) of January 1, 1970.

2. ***movies.dat***: contains the movies information. Each line of this file represents one movie, and has the following format: MovieID::Title::Genres. MovieID is the real MovieLens id. Movie titles, by policy, should be entered identically to those found in the Internet Movie Database - IMDB - , including year of release. However, they are entered manually, so errors and inconsistencies may exist. Genres are a pipe-separated list.

3. ***tags.dat***: Each line of this file represents one tag applied to one movie by one user, and has the following format: UserID::MovieID::Tag::Timestamp. The lines within this file are ordered first by UserID, then, within user, by MovieID. Tags are user generated metadata about movies. Each tag is typically a single word, or short phrase. The meaning, value and purpose of a particular tag is determined by each user. Timestamps represent seconds since midnight Coordinated Universal Time (UTC) of January 1, 1970.

There were three more files. A Unix shell script that can be used to split the ratings data for five-fold cross-validation of rating predictions called ***split_ratings.sh***, which depends on a second script, writen in Perl, called ***allbut.pl***. The third file was a README file.

For this project, both ***movie.dat*** and ***ratings.dat*** files were merged into a single dataset containing 10,000,054 of 6 variables: userId, movieId, rating, timestamp, title and genres. The remaining files were not used.

Table 1: The dataset used

userId	movieId	rating	timestamp	title	genres
1	122	5	838985046	Boomerang (1992)	Comedy|Romance
1	185	5	838983525	Net, The (1995)	Action|Crime|Thriller
1	292	5	838983421	Outbreak (1995)	Action|Drama|Sci-Fi|Thriller
1	316	5	838983392	Stargate (1994)	Action|Adventure|Sci-Fi
1	329	5	838983392	Star Trek: Generations (1994)	Action|Adventure|Drama|Sci-Fi
1	355	5	838984474	Flintstones, The (1994)	Children|Comedy|Fantasy

2. Analysis and Methods

The dataset was downloaded from GroupLens website and later split into two datasets using the script described in Appendix 2. The first one, called edx, was created using 90% of the dataset entries, while the other, called validation, was built using the remaining 10%. In order to simulate a real-world application, the algorithm was developed using only the edx set and then used to predict the ratings existing in the validation set, *as they are unknown.*

Table 2: Datasets dimensions

	edx	validation
Rows	9000055	999999
Columns	6	6

However, two columns were added to the dataset. The first one was the "year", which was an information merged with the "title" and referes to the year the movie premiered. The second column added was the "era". Since different periods of cinema have differente appeals, for analysis purposes, this information was added. Also, the timestamp column was transformed into a date format. After these modifications, the dataset was ready for the algorithm to start to be built. To make easier to read, the columns were reordered.

Table 3: The final edx dataset

userId	movieId	timestamp	title	year	era	genres	rating
1	122	1996-08-02	Boomerang	1992	90's	Comedy|Romance	5
1	185	1996-08-02	Net, The	1995	90's	Action|Crime|Thriller	5
1	292	1996-08-02	Outbreak	1995	90's	Action|Drama|Sci-Fi|Thriller	5
1	316	1996-08-02	Stargate	1994	90's	Action|Adventure|Sci-Fi	5
1	329	1996-08-02	Star Trek: Generations	1994	90's	Action|Adventure|Drama|Sci-Fi	5
1	355	1996-08-02	Flintstones, The	1994	90's	Children|Comedy|Fantasy	5

To build the algorithm, the edx dataset was split into a train set and a test set, containing 90% and 10% of the dataset, respectively. The algorithm was built using the train set and its performance was measured against the test set. Once all the parameters were defined and the performance considered satisfactory, the whole edx dataset was used to train the algorithm and the predictions for the validation set were made.

The performance of the algorithm was measured based on the residual mean squared error - RMSE:

$$\text{RMSE} = \sqrt{\frac{1}{N} \sum_{u,i} \left(\hat{y}_{u,i} - y_{u,i}\right)^2} \qquad (1)$$

where $\hat{y}_{u,i}$ is the predicted rating, $y_{u,i}$ is the actual rating and N is the number of user/movie combinations and the sum occurring over all these combinations.

2.1 The Model

The model used for this project, in essence, was a linear regression considering the "Global Effects". This consists, mainly, in decomposing the rating into minor parts in order to identify some predictors. Initialy, the predictors considered were:

- A **baseline rating**: here was used the average of all ratings and called μ;
- A **movie-specific effect**: roughly speaking, refers to the tendency of a movie being rated higher or lower than it is expected. Denoted here as b_i;
- A **user-specific effect**: consists in the same tendency as the movie-specific effect, but for users. Here it will be denoted as b_u.

That said, the rating prediction would be given by:

$$Y_{u,i} = \mu + b_i + b_u + \epsilon_{u,i} \qquad (2)$$

where $\epsilon_{u,i}$ is the error or an unknown parameter.

Several prediction models were tested until the final model could be reached, each of them with a "new effect" included. The final model used in this project is given by:

$$Y_{u,i} = \mu + b_i + b_u + b_y + b_g + b_t + \epsilon_{u,i} \qquad (3)$$

where b_e is an era effect related to when the movie premiered, b_g is a genre effect (it was considered only the main genre of the movie) and b_t is a time effect. It's simply the difference between the year the movie premiered and the year it was rated.

However, to avoid noisy estimates, that could lead to large errors, we penalized large estimates that came from small sample sizes. In other words, we used a regularization to add a penalty for large values of the predictors to the sum of squares equation that we wanted to minimize. That said, we were minimizing:

$$\frac{1}{N}\sum_{u,i}\left(y_{i,u} - \mu - b_i - b_u - b_e - b_g - b_t\right)^2 + \lambda\left(\sum_i b_i^2 + \sum_u b_u^2 + \sum_y b_e^2 + \sum_g b_g^2 + \sum_t b_t^2\right) \qquad (4)$$

where λ is the penalty term chosen by cross-validation. In this work, by cross-validation, it was chosen to be $\lambda = 5$.

The level of acceptance for the RMSE, i.e. the goal, was set to be ≤ 0.87750.

2.2 Hardware Specifications

This project was done on a computer with an Intel Core i7-6700 CPU @ 3.40GHz and 32Gb DDR4 @ 2133MHz of RAM, running Ubuntu 18.04.1 LTS. All the scripts and algorithms were built on R Language, version 3.5.2. The whole session information, including the packages used, can be found in Appendix 1.

3. Results

3.1 Data Exploration

After the adjustments, the edx dataset was composed of 9,000,055 entries of 7 variables. There was no missing values and, also, no zero ratings. It contained 10,677 movies, from the year of 1915 to 2008, and 69,878 users. The ratings distribution shows that the most common rates are 3 and 4 and, also, that half star ratings seems to be less common than whole star ratings.

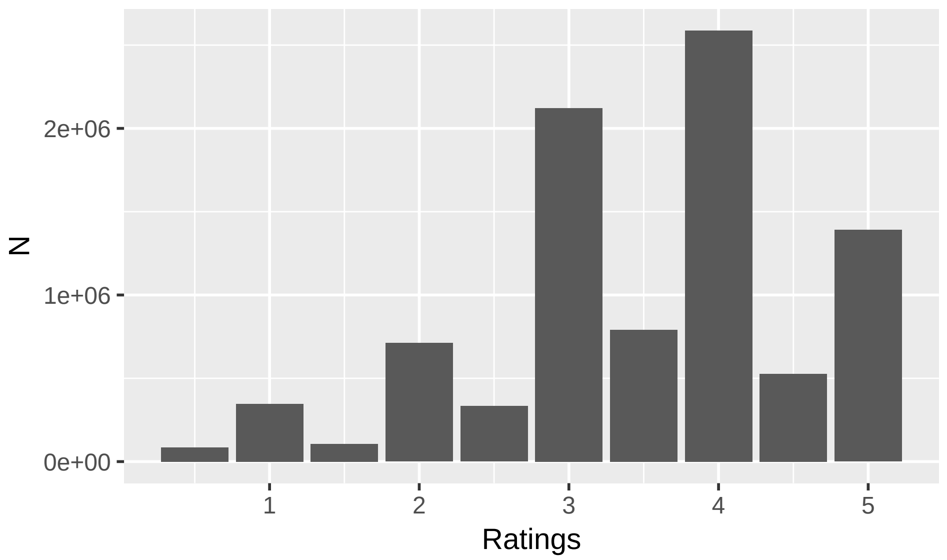

Figure 1: Ratings distribution

A simple analysis has shown us that the most watched movies "off all times" were Pulp Fiction (31,362), Forrest Gump (31,079) and The Silence of the Lambs (30,382), and that Drama (3,910,127) is the most watched genre, followed by Comedy (3,540,930) and Action (2,560,545).

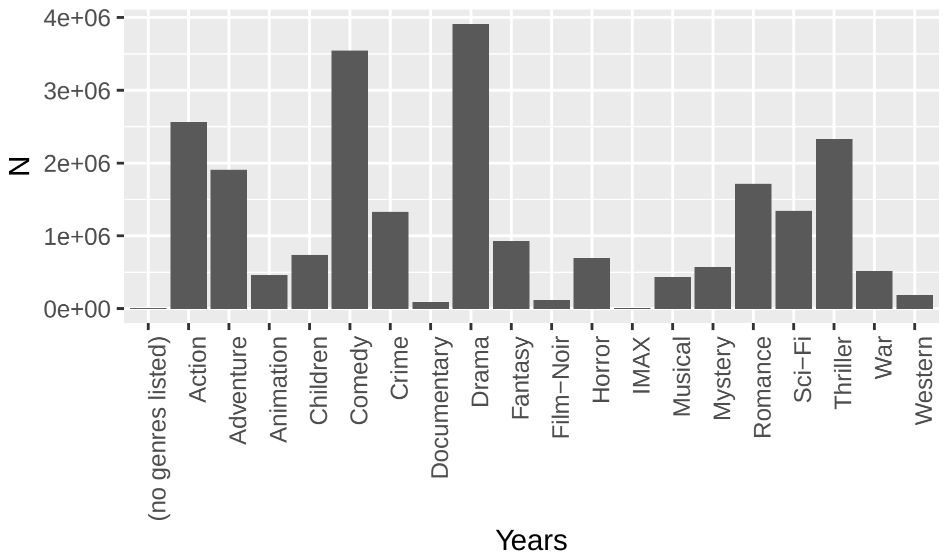

Figure 2: Number of views by genre

A quick look in how the movies watched were distributed through the "era of the cinema", we could see that the 90's was the period with more movies watched, 4,518,239. However, a look at Figure 4, below, has shown us that the rating of the 90's was below the average while the movies from the 50's, for example, were quite well rated. This could be explained by the number of movies watched during the periods, as shown in Figure 3.

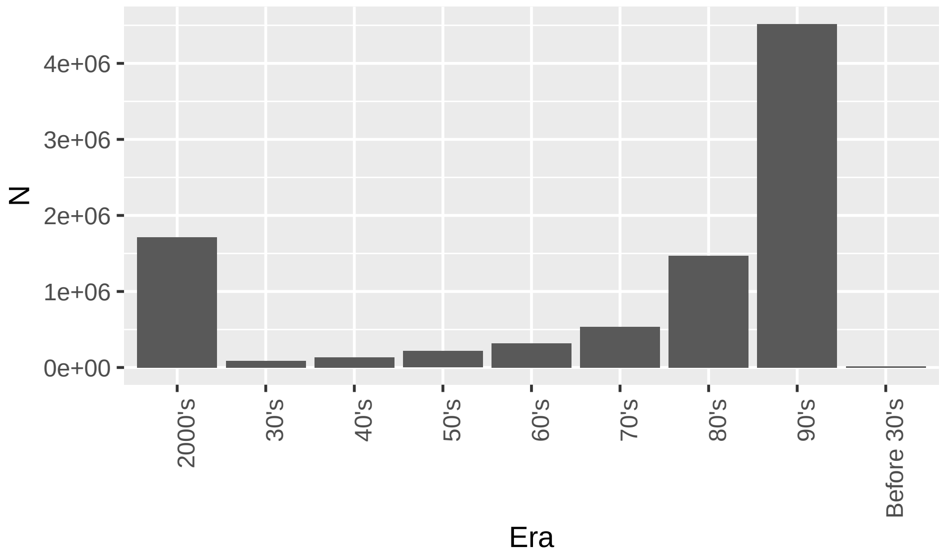

Figure 3: Number of views by cinema era

Intuitively, it is known that some genres, as some movies, are rated higher than others and that it also represents some of the users preferences. Other parameters like the year the movie was produced, the time between when a movie premiered and when a user watched it can also have some effects on the ratings.

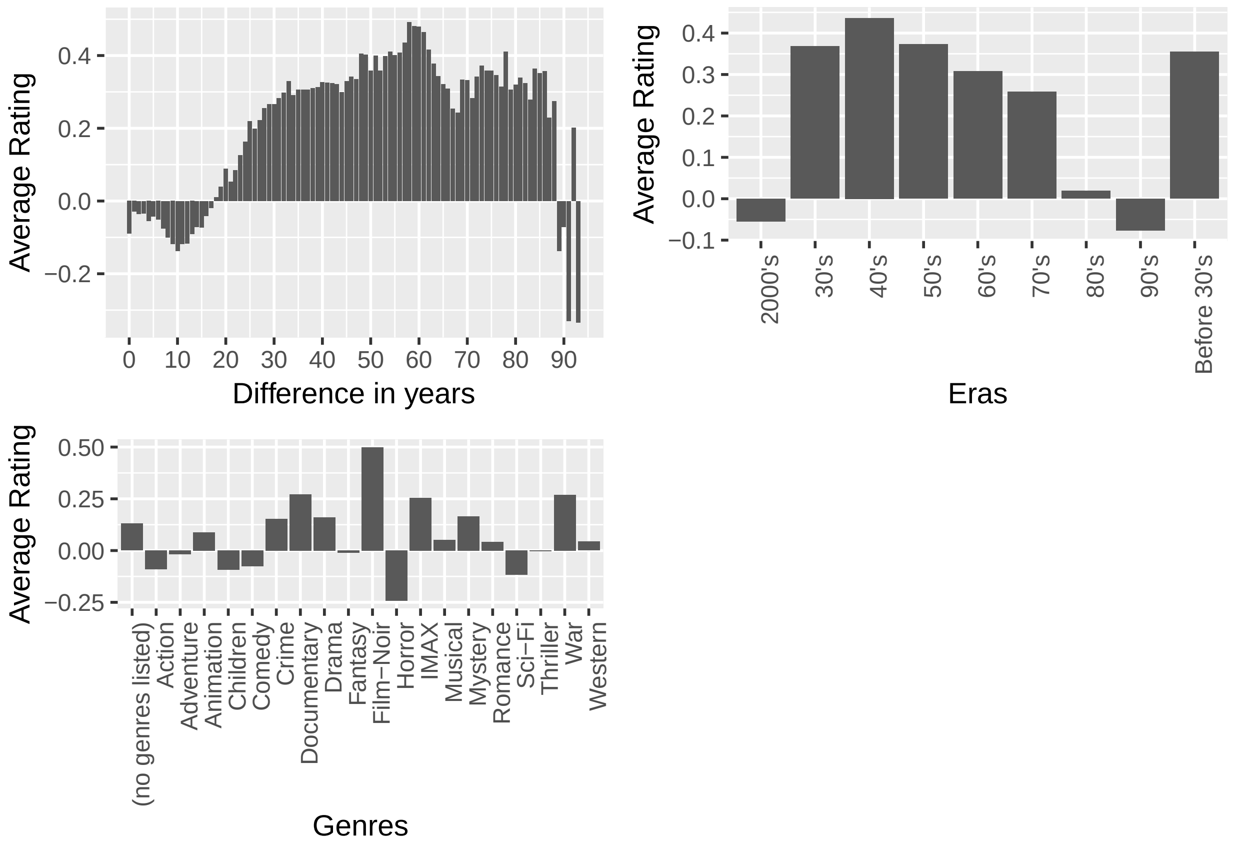

Figure 4: Genres, difference in years and year of production deviation from the mean

Figure 4, above, shows how some parameters deviates from the mean. It suggests that understanding these parameters and estimating these *biases* allow us to perform better predictions about the rating a specific user would give to a movie.

3.2 Machine Learning Algorithm

As it can be seen in Table 4, below, as we included new *biases* there was some significant improvements in our prediction. The first model to be used agaist our test set considered only the baseline rating, i.a. the μ. Using only the average of all ratings we could achieve a RMSE of ≈ 1.06. Once we included the movie and the user parameters our RMSE improves significantly ≈ 0.865.

Since the year the movie premiered, the difference between the year it premiered and the year the user watched and the genres seems to affect the rating, these *biases* were estimated and used to compose the model. Once all the biases were estimated and the regularization penalty was defined, our RMSE turns out to be ≈ 0.864.

Considering that our "goal" was to achieve a RMSE of ≤ 0.87750, the model was *acceptable* to be tested against our validation set for our final RMSE.

Table 4: Model comparison during the modelling process

Method	RMSE
Average	1.0600069
Average + Movie Bias	0.9435103
Average + User and Movie Biases	0.8656449
Average + Era, User and Movie Biases	0.8654363
Average + Genres, Era, User and Movie Biases	0.8653302
Average + Time, Genres, Era, User and Movie Biases	0.8650002
Regularized, Average + Time, Genres, Era, User and Movie Biases	0.8646655

After defining the parameters (biases) that would be considered into our model and the penalty term value, $\lambda = 5$, the model was tested against our "real-world" dataset. As shown in Table 5, below, our model applied in the "real-world" dataset was quite consistent with the results obtained in our training. That said, our model was able to get a **final RMSE of $\approx$ 0.86412**, which was below our goal of 0.87750 and, therefore, acceptable.

Table 5: Training and Validation RMSEs

Method	Training RMSE	Validation RMSE
Average	1.0600069	1.0612018
Average + Movie Bias	0.9435103	0.9438763
Average + User and Movie Biases	0.8656449	0.8647160
Average + Era, User and Movie Biases	0.8654363	0.8645410
Average + Genres, Era, User and Movie Biases	0.8653302	0.8644371
Average + Time, Genres, Era, User and Movie Biases	0.8650002	0.8641207
Regularized, Average + Time, Genres, Era, User and Movie Biases	0.8646655	NA

* There's no value for the regularized RMSE on the validation set because it was already calculated with the regularization parameter.

4. Conclusion

As previously said, recommendation systems are a common engine among e-commerce, social media and content-based websites and consists of a Machine Learning technique that analyzes available data to make suggestions for something that a user might be interested in. The main objective of this project was to build a Movie Recommendation System algorithm to predict which rating a specific user would give to a specific movie. For such, it was used the MovieLens dataset with 10,000,000 entries.

The requirement for the acceptance of the algorithm was to reach a RMSE < 0.87750. In other words, since the RMSE can be seem as a *'standard deviation'*, it should predict the rating with deviance of, at most, $Y_{u,i} \pm 0.87750$. The model that we build was able to meet the established standard by reaching a **RMSE of** $\approx$ **0.86412**, i.e. the true rating should be $Y_{u,i} \pm 0.86412$.

It's important to note that the baseline rating (μ), the movie bias (b_i) and the user bias (b_u) are the most important parameters to be estimated, since these three parameters already provide a very good estimative of the user rating. Although there are other algorithms and techniques that can be used to make such predictions, any improvement in the RMSE after the estimation of the three parameters above becomes so small and involves so much computation that, for this project, it was not considered. That's because, as said above, this work aim was to build a Movie Recommendation System with a RMSE < 0.87750.

Appendix 1: Session Info

```
## R version 3.5.2 (2018-12-20)
## Platform: x86_64-pc-linux-gnu (64-bit)
## Running under: Ubuntu 18.04.1 LTS
##
## Matrix products: default
## BLAS: /usr/lib/x86_64-linux-gnu/blas/libblas.so.3.7.1
## LAPACK: /usr/lib/x86_64-linux-gnu/lapack/liblapack.so.3.7.1
##
## locale:
##  [1] LC_CTYPE=pt_BR.UTF-8       LC_NUMERIC=C
##  [3] LC_TIME=pt_PT.UTF-8        LC_COLLATE=pt_BR.UTF-8
##  [5] LC_MONETARY=pt_PT.UTF-8    LC_MESSAGES=pt_BR.UTF-8
##  [7] LC_PAPER=pt_PT.UTF-8       LC_NAME=C
##  [9] LC_ADDRESS=C               LC_TELEPHONE=C
## [11] LC_MEASUREMENT=pt_PT.UTF-8 LC_IDENTIFICATION=C
##
## attached base packages:
## [1] stats     graphics  grDevices utils     datasets  methods   base
##
## other attached packages:
##  [1] gridExtra_2.3   caret_6.0-81    lattice_0.20-38 lubridate_1.7.4
##  [5] forcats_0.3.0   stringr_1.3.1   dplyr_0.7.8     purrr_0.2.5
##  [9] readr_1.2.1     tidyr_0.8.2     tibble_1.4.2    ggplot2_3.1.0
## [13] tidyverse_1.2.1
##
## loaded via a namespace (and not attached):
##  [1] Rcpp_1.0.0         class_7.3-14       assertthat_0.2.0
##  [4] rprojroot_1.3-2    digest_0.6.18      ipred_0.9-8
##  [7] foreach_1.4.4      R6_2.3.0           cellranger_1.1.0
## [10] plyr_1.8.4         backports_1.1.2    stats4_3.5.2
## [13] evaluate_0.12      highr_0.7          httr_1.3.1
## [16] pillar_1.3.0       rlang_0.3.0.1      lazyeval_0.2.1
## [19] readxl_1.1.0       rstudioapi_0.8     data.table_1.11.8
## [22] rpart_4.1-13       Matrix_1.2-15      rmarkdown_1.10
## [25] labeling_0.3       splines_3.5.2      gower_0.1.2
## [28] munsell_0.5.0      broom_0.5.0        compiler_3.5.2
## [31] modelr_0.1.2       pkgconfig_2.0.2    htmltools_0.3.6
## [34] nnet_7.3-12        tidyselect_0.2.5   prodlim_2018.04.18
## [37] codetools_0.2-15   crayon_1.3.4       withr_2.1.2
## [40] MASS_7.3-51.1      recipes_0.1.4      ModelMetrics_1.2.2
## [43] grid_3.5.2         nlme_3.1-137       jsonlite_1.5
## [46] gtable_0.2.0       magrittr_1.5       scales_1.0.0
## [49] cli_1.0.1          stringi_1.2.4      reshape2_1.4.3
## [52] bindrcpp_0.2.2     timeDate_3043.102  xml2_1.2.0
## [55] generics_0.0.2     lava_1.6.4         iterators_1.0.10
## [58] tools_3.5.2        glue_1.3.0         hms_0.4.2
## [61] survival_2.43-3    yaml_2.2.0         colorspace_1.3-2
## [64] rvest_0.3.2        knitr_1.20         bindr_0.1.1
## [67] haven_2.0.0
```

Appendix 2: Script for Creating the Datasets

The following code was used to generate the datasets. The *edx set* was used to develop the algorithm and, for a final test, the algorithm was used to predict movie ratings in the *validation set* as if they were unknown.

```r
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")

# 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)
edx <- rbind(edx, removed)

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

# Save the datasets

write_csv(edx, "data/raw_edx.csv")
write_csv(validation, "data/raw_validation.csv")
```

Appendix 3: Data Wrangling

```r
# Load the necessary libraries to work
library(tidyverse)
library(caret)
library(lubridate)

# Load the datasets into R space
edx <- read_csv("data/raw_edx.csv")
validation <- read.csv("data/raw_validation.csv")

dim(edx)
dim(validation)

summary(edx)
summary(validation)

str(edx)
str(validation)

head(edx)

# Need to separate the year from the title and correct the timestamp to a Date format.
# To make easier to work in both datasets, we'll  merge the two datasets to work on both
# of them at the same time. To not forget the dimensions of both sets, we added a
# "temporary flag" into both of them: 0 for the edx and 1 for the validation.

flag <- c(rep(0, nrow(edx)), rep(1, nrow(validation)))

complete <- rbind(edx, validation)
complete <- complete %>%
  mutate(flag=flag)

dim(complete)

# Create and year vector to store the years from the title and then add before "genres"

year <- str_extract_all(complete$title, "\\([0-9]{4}\\)") %>%
  str_extract_all("[0-9]{4}") %>%
  unlist() %>%
  as.numeric()

complete <- complete %>%
  add_column(year, .before="genres")

# Now we remove the year from the "title"
complete <- complete %>%
  mutate(title=str_remove_all(title, " \\(|[0-9]{4}\\)"))

# Convert the timestamp to date

complete <- complete %>%
  mutate(timestamp=as.POSIXct(timestamp, origin="1970-01-1")) %>%
  mutate(timestamp=as.Date(timestamp)) # Since the time doesn't matter, we keep only
                                       # the date.


# Since different periods of cinema have different "appeals", we will create a new
# variable, called "era". It referes to the period of the cinema: 30's, 40's, 50's... etc.

complete <- complete %>% # This will turn the years into 10, 20, 30, 40...and 2000 if
                         # above 2000
  mutate(period=ifelse(year<2000, floor((year-1900)/10)*10 , 2000))

complete <- add_column(complete, era=NA)

complete$era[complete$period<30] <- "Before 30's"
complete$era[complete$period==30] <- "30's"
complete$era[complete$period==40] <- "40's"
complete$era[complete$period==50] <- "50's"
complete$era[complete$period==60] <- "60's"
complete$era[complete$period==70] <- "70's"
complete$era[complete$period==80] <- "80's"
complete$era[complete$period==90] <- "90's"
complete$era[complete$period>90] <- "2000's"

# Let's reorder the dataset just to make it easier to read and remove the period column

complete <- complete %>%
  select(userId, movieId, timestamp, title, year, era, genres, rating, flag,  -period)

# Split the dataset again

edx <- complete %>%
  filter(flag==0) %>%
  select(-flag)

validation <- complete %>%
  filter(flag==1) %>%
  select(-flag)

# Save the new datasets

write_csv(edx, "data/edx.csv")
write_csv(validation, "data/validation.csv")

# No need to keep the complete and validation datasets in the enviroment. Clean the
# "garbage" and return some memory to OS
rm(complete, validation, flag, year)
gc()
```

Appendix 4: Data Exploration

```r
# General info
summary(edx)
dim(edx)

edx %>%
  select(movieId, userId) %>%
  summarize('unique movies'=n_distinct(movieId), 'unique users'=n_distinct(userId))

# Ratings distribution
edx %>%
  group_by(rating) %>%
  summarize(N=n()) %>%
  arrange(desc(N))

# Plot ratings distibution
edx %>%
  group_by(rating) %>%
  summarize(N=n()) %>%
  ggplot(aes(x=rating, y=N))+
  geom_histogram(stat = "identity")+
  labs(title="Ratings Distribution", x="Ratings", y="N")

# Views by genres
edx %>%
  separate_rows(genres, sep="\\|") %>%
  group_by(genres) %>%
  summarize(N=n()) %>%
  arrange(desc(N))

# Plot # of views by genre
edx %>%
  separate_rows(genres, sep="\\|") %>%
  group_by(genres) %>%
  summarize(N=n()) %>%
  ggplot(aes(x=genres, y=N))+
  geom_bar(stat = "identity")+
  theme(axis.text.x = element_text(angle = 90, hjust = 1))+
  labs(title="View by Genre", x="Years", y="N")

# Number of times movies were watched
edx %>%
  group_by(title) %>%
  summarize(N=n()) %>%
  arrange(desc(N))

# Views by era of the cinema
edx %>%
  group_by(era) %>%
  summarize(N=n()) %>%
  arrange(desc(N))
```

```r
edx %>%
  group_by(era) %>%
  summarize(N=n()) %>%
  ggplot(aes(x=era, y=N))+
  geom_bar(stat = "identity")+
  theme(axis.text.x = element_text(angle = 90, hjust = 1))+
  labs(title="Views by Cinema Era", x="Years", y="N")

# Most watched movies of each era
most_watched <- c()
period <- c("Before 30's", "30's", "40's", "50's", "60's", "70's", "80's", "90's",
            "2000's")
for (i in 1:length(period)){

  most_watched <- rbind(most_watched,
        edx %>%
          filter(era==period[i]) %>%
          group_by(title) %>%
          summarize(N=n()) %>%
          top_n(1)
        )}
most_watched <- cbind(most_watched,era=period)

# Extract the average rating to make some observation of the rating behaviour
mu <- mean(edx$rating)

# Average ratings considering the "distance" between movie premiere and being watched
edx %>%
  mutate(Difference=abs(year(timestamp)-year)) %>%
  select(Difference, rating) %>%
  ggplot(aes(x=Difference, y=(rating-mu)))+
  geom_bar(stat="summary", fun.y="mean")+
  scale_x_continuous(breaks = seq(0,93,10))+
  labs(x="Difference in years", y="Average Rating")

# Average ratings considering eras
edx %>%
  group_by(era) %>%
  ggplot(aes(x=era, y=(rating-mu)))+
  geom_bar(position = "dodge", stat="summary", fun.y="mean")+
  theme(axis.text.x = element_text(angle = 90, hjust = 1))+
  labs(x="Era", y="Average Rating")

# Average ratings considering genres
edx %>%
  separate_rows(genres, sep="\\|") %>%
  ggplot(aes(x=genres, y=(rating-mu)))+
  geom_bar(position = "dodge", stat="summary", fun.y="mean")+
  theme(axis.text.x = element_text(angle = 90, hjust = 1))+
  scale_y_continuous(breaks=seq(-0.5, 0.5, 0.25))+
  labs(x="Genres", y="Average Rating")
```

Appendix 5: Modelling

```r
# Since the title of the movie is not relevant fot he modelling, we'll drop it.
edx <- edx %>%
  select(-title)

# Split the dataset into train and test sets
set.seed(755)
test_index <- createDataPartition(y = edx$rating, times = 1, p = 0.1, list = FALSE)
train_set <- edx[-test_index,]
test_set <- edx[test_index,]

# Make sure to don't include users and movies in the test set that do not appear in
# the training set
test_set <- test_set %>%
  semi_join(train_set, by = "movieId") %>%
  semi_join(train_set, by = "userId")

# Remove not necessary objects
rm(test_index)

# Release the memory
gc()

# Define the naseline rating, here it will be the average
mu <- train_set %>%
  summarize(mu=mean(rating)) %>%
  pull()

# Test the average against the ratings to define a starting point and store de RMSEs
# for comparison

rmse <- data.frame("Method"="Average", "RMSE"=RMSE(mu, test_set$rating))
knitr::kable(rmse)

#########################################################################################

# Estimating the movie bias

movie_avg <- train_set %>%
  group_by(movieId) %>%
  summarize(b_i=mean(rating-mu))

# Predictions
y_hat <- test_set %>%
  left_join(movie_avg, by="movieId") %>%
  mutate(prediction=mu+b_i) %>%
  .$prediction

# Store the new RMSE
rmse <- rmse %>%
  rbind(data.frame("Method"="Average + Movie Bias", "RMSE"=RMSE(y_hat, test_set$rating)))
```

```r
########################################################################################

# Estimating the user bias
usr_avg <- train_set %>%
  left_join(movie_avg, by="movieId") %>%
  group_by(userId) %>%
  summarize(b_u=mean(rating-mu-b_i))

# Predictions
y_hat <- test_set %>%
  left_join(movie_avg, by="movieId") %>%
  left_join(usr_avg, by="userId") %>%
  mutate(prediction=mu+b_i+b_u) %>%
  .$prediction


# Note that the range of y_hat is below 0 and above 5.
# Since there are no values below 0.5 in the dataset, everything below 0 will become
# 0.5 and everything above 5 will become 5.

y_hat[y_hat<0] <- 0.5
y_hat[y_hat>5] <- 5

# Store the new RMSE
rmse <- rmse %>%
  rbind(data.frame("Method"="Average + User and Movie Biases",
                   "RMSE"=RMSE(y_hat, test_set$rating)))

# Since genres, the difference between time it was watched and time it was produced so
# as the era seem to have some effect in the rating, this values will also be estimated

########################################################################################

# Estimating the Era bias

era_avg <- train_set %>%
  left_join(movie_avg, by="movieId") %>%
  left_join(usr_avg, by="userId") %>%
  group_by(era) %>%
  summarize(b_e=mean(rating-mu-b_i-b_u))

# Predictions
y_hat <- test_set %>%
  left_join(movie_avg, by="movieId") %>%
  left_join(usr_avg, by="userId") %>%
  left_join(era_avg, by="era") %>%
  mutate(prediction=mu+b_i+b_u+b_e) %>%
  .$prediction

# "Correct" the range
y_hat[y_hat<0] <- 0.5
y_hat[y_hat>5] <- 5
```

```r
# Store the new RMSE
rmse <- rmse %>%
  rbind(data.frame("Method"="Average + Era, User and Movie Biases",
                   "RMSE"=RMSE(y_hat, test_set$rating)))

##########################################################################################

# Estimating the genres bias. For simplicity, here, it'll only be considered the first
# (principal) genre of the movie.

genres_avg <- train_set %>%
  separate(genres, "genres", sep="\\|", extra="drop") %>%
  left_join(movie_avg, by="movieId") %>%
  left_join(usr_avg, by="userId") %>%
  left_join(era_avg, by="era") %>%
  group_by(genres) %>%
  summarize(b_g=mean(rating-mu-b_i-b_u-b_e))

# Predictions
y_hat <- test_set %>%
  separate(genres, "genres", sep="\\|", extra="drop") %>%
  left_join(movie_avg, by="movieId") %>%
  left_join(usr_avg, by="userId") %>%
  left_join(era_avg, by="era") %>%
  left_join(genres_avg, by="genres") %>%
  mutate(prediction=mu+b_i+b_u+b_e+b_g) %>%
  .$prediction

y_hat[y_hat<0] <- 0.5
y_hat[y_hat>5] <- 5

# Store the new RMSE
rmse <- rmse %>%
  rbind(data.frame("Method"="Average + Genres, Era, User and Movie Biases",
                   "RMSE"=RMSE(y_hat, test_set$rating)))

##########################################################################################

# Estimating a "time" bias. It's the difference between the movie premiere and when it
# was watched.

time_diff_avg <- train_set %>%
  separate(genres, "genres", sep="\\|", extra="drop") %>%
  mutate(timeDiff=abs(year(timestamp)-year)) %>%
  left_join(movie_avg, by="movieId") %>%
  left_join(usr_avg, by="userId") %>%
  left_join(era_avg, by="era") %>%
  left_join(genres_avg, by="genres") %>%
  group_by(timeDiff) %>%
  summarize(b_t=mean(rating-mu-b_i-b_u-b_e-b_g))

# Predictions
y_hat <- test_set %>%
  separate(genres, "genres", sep="\\|", extra="drop") %>%
  mutate(timeDiff=abs(year(timestamp)-year)) %>%
  left_join(movie_avg, by="movieId") %>%
  left_join(usr_avg, by="userId") %>%
  left_join(era_avg, by="era") %>%
  left_join(genres_avg, by="genres") %>%
  left_join(time_diff_avg, by="timeDiff") %>%
  mutate(prediction=mu+b_i+b_u+b_e+b_g+b_t) %>%
  .$prediction

# "Correct" the range
y_hat[y_hat<0] <- 0.5
y_hat[y_hat>5] <- 5


# Store the new RMSE
rmse <- rmse %>%
  rbind(data.frame("Method"="Average + Time, Genres, Era, User and Movie Biases",
                   "RMSE"=RMSE(y_hat, test_set$rating)))
```

Appendix 6: Regularization

```
# To define the value to pick for regularization, we'll perform a cross-validation
# to verify which value fits better to minimize the model error.

# Define the values to test
lambdas <- seq(0, 10 , 0.2)

# Build the function to cross-validate.
# This function will take a few minutes running
rmses <- sapply(lambdas, function(l){

  mu <- mean(train_set$rating)

  b_i <- train_set %>%
    group_by(movieId) %>%
    summarize(b_i = sum(rating - mu)/(n()+l))

  b_u <- train_set %>%
    left_join(b_i, by="movieId") %>%
    group_by(userId) %>%
    summarize(b_u = sum(rating - b_i - mu)/(n()+l))

  b_e <- train_set %>%
    left_join(b_i, by="movieId") %>%
    left_join(b_u, by="userId") %>%
    group_by(era) %>%
    summarize(b_e=sum(rating-mu-b_i-b_u)/(n()+l))

  b_g <- train_set %>%
    separate(genres, "genres", sep="\\|", extra="drop") %>%
    left_join(b_i, by="movieId") %>%
    left_join(b_u, by="userId") %>%
    left_join(b_e, by="era") %>%
    group_by(genres) %>%
    summarize(b_g=sum(rating-mu-b_i-b_u-b_e)/(n()+l))

  b_t <- train_set %>%
    separate(genres, "genres", sep="\\|", extra="drop") %>%
    mutate(timeDiff=abs(year(timestamp)-year)) %>%
    left_join(b_i, by="movieId") %>%
    left_join(b_u, by="userId") %>%
    left_join(b_e, by="era") %>%
    left_join(b_g, by="genres") %>%
    group_by(timeDiff) %>%
    summarize(b_t=sum(rating-mu-b_i-b_u-b_e-b_g)/(n()+l))

  y_hat <- test_set %>%
    separate(genres, "genres", sep="\\|", extra="drop") %>%
    mutate(timeDiff=abs(year(timestamp)-year)) %>%
    left_join(b_i, by = "movieId") %>%
    left_join(b_u, by = "userId") %>%
    left_join(b_e, by= "era") %>%
```

```r
    left_join(b_g, by= "genres") %>%
    left_join(b_t, by= "timeDiff") %>%
    mutate(prediction=mu+b_i+b_u+b_e+b_g+b_t) %>%
    .$prediction

  return(RMSE(y_hat, test_set$rating))
})

# Pick the lambda value with the minor rmse
lambdas[which.min(rmses)]

rmse <- rmse %>%
  rbind(data.frame("Method"="Regularized, Average + Time, Genres, Era, User and
                   Movie Biases", "RMSE"=RMSE(y_hat, test_set$rating)))

knitr::kable(rmse)

write_csv(rmse, 'support_data/train_rmses.csv')
```

Appendix 7: Validation

```r
# Clean the environment and load the datasets again
rm(list=ls())
gc()

edx <- read_csv('data/edx.csv')
validation <- read_csv('data/validation.csv')

edx <- edx %>%
  select(-title)

validation <- validation %>%
  select(-title)

# Define lambra as 5
l <- 5

# Only the average
mu <- mean(edx$rating)

rmse <- data.frame('Method'='Average', RMSE=RMSE(mu, validation$rating))


# Movie bias

b_i <- edx %>%
  group_by(movieId) %>%
  summarize(b_i = sum(rating - mu)/(n()+l))

# Predictions

y_hat <- validation %>%
  left_join(b_i, by="movieId") %>%
  mutate(prediction=mu+b_i) %>%
  .$prediction

# Store the RMSE
rmse <- rmse %>%
  rbind(data.frame("Method"="Average + Movie Bias",
                   "RMSE"=RMSE(y_hat, validation$rating)))


# User bias
b_u <- edx  %>%
  left_join(b_i, by="movieId") %>%
  group_by(userId) %>%
  summarize(b_u = sum(rating - b_i - mu)/(n()+l))

# Predictions
y_hat <- validation %>%
  left_join(b_i, by="movieId") %>%
  left_join(b_u, by="userId") %>%
  mutate(prediction=mu+b_i+b_u) %>%
  .$prediction

# "Correect" the range
y_hat[y_hat<0] <- 0.5
y_hat[y_hat>5] <- 5

# Store the RMSE
rmse <- rmse %>%
  rbind(data.frame("Method"="Average + User and Movie Biases",
                   "RMSE"=RMSE(y_hat, validation$rating)))

# Era bias
b_e <- edx%>%
  left_join(b_i, by="movieId") %>%
  left_join(b_u, by="userId") %>%
  group_by(era) %>%
  summarize(b_e=sum(rating-mu-b_i-b_u)/(n()+1))

# Predictions
y_hat <- validation %>%
  left_join(b_i, by="movieId") %>%
  left_join(b_u, by="userId") %>%
  left_join(b_e, by="era") %>%
  mutate(prediction=mu+b_i+b_u+b_e) %>%
  .$prediction

# "Correect" the range
y_hat[y_hat<0] <- 0.5
y_hat[y_hat>5] <- 5

# Store the RMSE
rmse <- rmse %>%
  rbind(data.frame("Method"="Average + Era, User and Movie Biases",
                   "RMSE"=RMSE(y_hat, validation$rating)))

# Genre bias
b_g <- edx %>%
  separate(genres, "genres", sep="\\|", extra="drop") %>%
  left_join(b_i, by="movieId") %>%
  left_join(b_u, by="userId") %>%
  left_join(b_e, by="era") %>%
  group_by(genres) %>%
  summarize(b_g=sum(rating-mu-b_i-b_u-b_e)/(n()+1))

# Predictions
y_hat <- validation %>%
  separate(genres, "genres", sep="\\|", extra="drop") %>%
  left_join(b_i, by = "movieId") %>%
  left_join(b_u, by = "userId") %>%
  left_join(b_e, by= "era") %>%
  left_join(b_g, by= "genres") %>%
  mutate(prediction=mu+b_i+b_u+b_e+b_g) %>%
  .$prediction

# "Correct" the range
y_hat[y_hat<0] <- 0.5
y_hat[y_hat>5] <- 5

# Store the new RMSE
rmse <- rmse %>%
  rbind(data.frame("Method"="Average + Genres, Era, User and Movie Biases",
                   "RMSE"=RMSE(y_hat, validation$rating)))


# Time bias
b_t <- edx %>%
  separate(genres, "genres", sep="\\|", extra="drop") %>%
  mutate(timeDiff=abs(year(timestamp)-year)) %>%
  left_join(b_i, by="movieId") %>%
  left_join(b_u, by="userId") %>%
  left_join(b_e, by="era") %>%
  left_join(b_g, by="genres") %>%
  group_by(timeDiff) %>%
  summarize(b_t=sum(rating-mu-b_i-b_u-b_e-b_g)/(n()+l))

#Prediction
y_hat <- validation %>%
  separate(genres, "genres", sep="\\|", extra="drop") %>%
  mutate(timeDiff=abs(year(timestamp)-year)) %>%
  left_join(b_i, by = "movieId") %>%
  left_join(b_u, by = "userId") %>%
  left_join(b_e, by= "era") %>%
  left_join(b_g, by= "genres") %>%
  left_join(b_t, by= "timeDiff") %>%
  mutate(prediction=mu+b_i+b_u+b_e+b_g+b_t) %>%
  .$prediction

# "Correct" the range
y_hat[y_hat<0] <- 0.5
y_hat[y_hat>5] <- 5


# Store the new RMSE
rmse <- rmse %>%
  rbind(data.frame("Method"="Average + Time, Genres, Era, User and Movie Biases",
                   "RMSE"=RMSE(y_hat, validation$rating)))


# Comparison between training and validation RMSES

rmse <- rmse %>%
  rbind(data.frame("Method"="Regularized, Average + Time, Genres, Era, User and
                   Movie Biases", "RMSE"=NA))

train_rmses <- read_csv('support_data/train_rmses.csv')
```

```r
rmses <- cbind(train_rmses, rmse[,2])
colnames(rmses) <- c("Method", "Training RMSE", "Validation RMSE")

write_csv(rmses, 'support_data/train_val_rmses.csv')
```

Appendix 8: Support Data

```
#Ratings distribution

ratings_distribution <- edx %>%
  group_by(rating) %>%
  summarize(N=n()) %>%
  arrange(desc(N))

write_csv(ratings_distribution, 'support_data/ratings_distribution.csv')

#Views by Genres


# Views by genres
genres_views <- edx %>%
  separate_rows(genres, sep="\\|") %>%
  group_by(genres) %>%
  summarize(N=n()) %>%
  arrange(desc(N))

write_csv(genres_views, 'support_data/genres_views.csv')


#Views by era

era_views <- edx %>%
  group_by(era) %>%
  summarize(N=n()) %>%
  arrange(desc(N))

write_csv(era_views, 'support_data/era_views.csv')

#Difference between years (production and watched) and ratings
year_diff_avg_rating <- edx %>%
  mutate(Difference=abs(year(timestamp)-year)) %>%
  select(Difference, rating)

write_csv(year_diff_avg_rating, 'support_data/year_diff_avg_rating.csv')

# Era average rating
era_avg_rating <- edx %>%
  group_by(era) %>%
  select(era, rating)

write_csv(era_avg_rating, 'support_data/era_avg_rating.csv')

# Genres average rating

genres_avg_rating <- edx %>%
  separate_rows(genres, sep="\\|") %>%
  select(genres, rating)
```

```r
write_csv(genres_avg_rating, 'support_data/genres_avg_rating.csv')


p1 <- ratings_year_diff %>%
  ggplot(aes(x=Difference, y=(rating-mu)))+
  geom_bar(stat="summary", fun.y="mean")+
  scale_x_continuous(breaks = seq(0,93,10))+
  labs(x="Difference in years", y="Average Rating")

p2 <- era_avg_rating %>%
  ggplot(aes(x=era, y=(rating-mu)))+
  geom_bar(position = "dodge", stat="summary", fun.y="mean")+
  theme(axis.text.x = element_text(angle = 90, hjust = 1))+
  labs(x="Era", y="Average Rating")


p3 <- genres_avg_rating %>%
  ggplot(aes(x=genres, y=(rating-mu)))+
  geom_bar(position = "dodge", stat="summary", fun.y="mean")+
  theme(axis.text.x = element_text(angle = 90, hjust = 1))+
  scale_y_continuous(breaks=seq(-0.5, 0.5, 0.25))+
  labs(x="Genres", y="Average Rating")

grid.arrange(p1,p2,p3, nrow=2, ncol=2)
```