HarvardX Data Science Capstone: MovieLens

Francis 'Kapi' Capistrano

04/12/2023

1. Introduction

This capstone project entails the development of a movie recommendation system using an open data set on movie ratings. The **MovieLens** data set is maintained by GroupLens Research at the University of Minnesota. This project, in particular, uses a version of the data set that contains over 10 million ratings applied to more than 10,000 movies by nearly 70,000 users of MovieLens. The goal is to create a machine learning model that is robust enough, as indicated by a root mean squared error (RMSE) that is as low as possible. The challenge of this project is that there are only six variables in the data set.

1.1 Data preparation

Although my data setup largely follows the course-provided procedure for creating the training (*edx*) and validation or final hold-out sets from the MovieLens 10M data, some modifications were made to address the constraint of having limited variables. Below is the first part of the data preparation code as provided in the course:

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

```
## Loading required package: tidyverse
```

```
## ── Attaching core tidyverse packages ──────────────────────── tidyverse 2.0.0 ──
## ✔ dplyr     1.1.0     ✔ readr     2.1.4
## ✔ forcats   1.0.0     ✔ stringr   1.5.0
## ✔ ggplot2   3.4.1     ✔ tibble    3.1.8
## ✔ lubridate 1.9.2     ✔ tidyr     1.3.0
## ✔ purrr     1.0.1
## ── Conflicts ────────────────────────────────────────── tidyverse_conflicts() ──
## ✖ dplyr::filter() masks stats::filter()
## ✖ dplyr::lag()    masks stats::lag()
## ℹ Use the ]8;;http://conflicted.r-lib.org/ conflicted package]8;; to force all conflicts to become errors
```

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

```
## Loading required package: caret
## Loading required package: lattice
##
## Attaching package: 'caret'
##
## The following object is masked from 'package:purrr':
##
##     lift
```

```
library(tidyverse)
library(caret)
library(knitr)
library(gridExtra)
```

```
##
## Attaching package: 'gridExtra'
##
## The following object is masked from 'package:dplyr':
##
##     combine
```

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

options(timeout = 120)

dl <- "ml-10M100K.zip"
if(!file.exists(dl))
  download.file("https://files.grouplens.org/datasets/movielens/ml-10m.zip", dl)

ratings_file <- "ml-10M100K/ratings.dat"
if(!file.exists(ratings_file))
  unzip(dl, ratings_file)

movies_file <- "ml-10M100K/movies.dat"
if(!file.exists(movies_file))
  unzip(dl, movies_file)

ratings <- as.data.frame(str_split(read_lines(ratings_file), fixed("::"), simplify = TRU
E),
                         stringsAsFactors = FALSE)
colnames(ratings) <- c("userId", "movieId", "rating", "timestamp")
ratings <- ratings %>%
  mutate(userId = as.integer(userId),
         movieId = as.integer(movieId),
         rating = as.numeric(rating),
         timestamp = as.integer(timestamp))

movies <- as.data.frame(str_split(read_lines(movies_file), fixed("::"), simplify = TRUE),
                        stringsAsFactors = FALSE)
colnames(movies) <- c("movieId", "title", "genres")
movies <- movies %>%
  mutate(movieId = as.integer(movieId))
```

Before joining the two data sets containing movies (with *movieID*, *title*, and *genre* fields) and ratings (with *userID*, *movieID*, *rating*, and *timestamp* fields), the variables were expanded as follows:

First, for the *movies* data set, a variable to represent the year of the movie's release was added by extracting the same from the *title* variable. Moreover, the *genres* variable contains multiple information in each observation. While this variable is retained for testing, a series of logical or "dummy" variables were created for each of the 19 genres available (excluding "no genre provided").

Second the *ratings* data set, the *timestamp* field is in integer form (in seconds since midnight of January 1, 1970, UTC) that is not readily analyzable. An additional field *ratingdate* was added to transform the *timestamp* data in YYYY-MM-DD format. Later on, permutations of this will be tested in the modeling (e.g., year only, month only,

etc.).Key characteristics of the data.

```
# Before joining the ratings and movies data sets, I first tried to:
## 1) modify the "movies" data set by (a) creating a new variable "movieyear" (b) turning
"genres" into a series of logical variables

movies <- movies %>%
  mutate(movieyear = str_extract(title,"\\([1|2][0|9][0-9][0-9]\\)")) %>%
  mutate(movieyear = str_extract(movieyear, "[0-9][0-9][0-9][0-9]")) %>%
  mutate(moviedecade = ifelse(as.integer(movieyear)<1920,"1910s",
                         ifelse(as.integer(movieyear)<1930,"1920s",
                         ifelse(as.integer(movieyear)<1940,"1930s",
                         ifelse(as.integer(movieyear)<1950,"1940s",
                         ifelse(as.integer(movieyear)<1960,"1950s",
                         ifelse(as.integer(movieyear)<1970,"1960s",
                         ifelse(as.integer(movieyear)<1980,"1970s",
                         ifelse(as.integer(movieyear)<1990,"1980s",
                         ifelse(as.integer(movieyear)<2000,"1990s","2000s")))))))))

genrelist <- movies %>% separate_rows(genres, sep = "\\|") %>%
  group_by(genres) %>%
  summarize(count = n()) %>%
  arrange(genres)
genrelist <- genrelist$genres

movies <- movies %>% mutate(nogenres = str_detect(genres,genrelist[1]),
                            action = str_detect(genres,genrelist[2]),
                            adventure = str_detect(genres,genrelist[3]),
                            animation = str_detect(genres,genrelist[4]),
                            children = str_detect(genres,genrelist[5]),
                            comedy = str_detect(genres,genrelist[6]),
                            crime = str_detect(genres,genrelist[7]),
                            documentary = str_detect(genres,genrelist[8]),
                            drama = str_detect(genres,genrelist[9]),
                            fantasy = str_detect(genres,genrelist[10]),
                            filmnoir = str_detect(genres,genrelist[11]),
                            horror = str_detect(genres,genrelist[12]),
                            imax = str_detect(genres,genrelist[13]),
                            musical = str_detect(genres,genrelist[14]),
                            mystery = str_detect(genres,genrelist[15]),
                            romance = str_detect(genres,genrelist[16]),
                            scifi = str_detect(genres,genrelist[17]),
                            thriller = str_detect(genres,genrelist[18]),
                            war = str_detect(genres,genrelist[19]),
                            western = str_detect(genres,genrelist[20]))

## 2) modify the "ratings" data set by adding a "ratingdate" variable
ratings <- ratings %>% mutate(ratingdate = as.Date.POSIXct(timestamp))
```

Finally, as I joined the two data sets, created my final hold-out test set, and cleaned up my working environment, I removed all other temporary data sets except the *movielens*, *movies*, and *ratings* data sets for reference.

```
#Time to join!
movielens <- left_join(ratings, movies, by = "movieId")

# Final hold-out test set will be 10% of MovieLens data
set.seed(1, sample.kind="Rounding")
```

```
## Warning in set.seed(1, sample.kind = "Rounding"): non-uniform 'Rounding' sampler
## used
```

```
test_index <- createDataPartition(y = movielens$rating, times = 1, p = 0.1, list = FALSE)
edx <- movielens[-test_index,]
temp <- movielens[test_index,]

# Make sure userId and movieId in final hold-out test set are also in edx set
final_holdout_test <- temp %>%
  semi_join(edx, by = "movieId") %>%
  semi_join(edx, by = "userId")

# Add rows removed from final hold-out test set back into edx set
removed <- anti_join(temp, final_holdout_test)
```

```
## Joining with `by = join_by(userId, movieId, rating, timestamp, ratingdate,
## title, genres, movieyear, moviedecade, nogenres, action, adventure, animation,
## children, comedy, crime, documentary, drama, fantasy, filmnoir, horror, imax,
## musical, mystery, romance, scifi, thriller, war, western)`
```

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

# Kept some of the data frames for use later
rm(dl, test_index, temp, removed, movies_file, ratings_file)
```

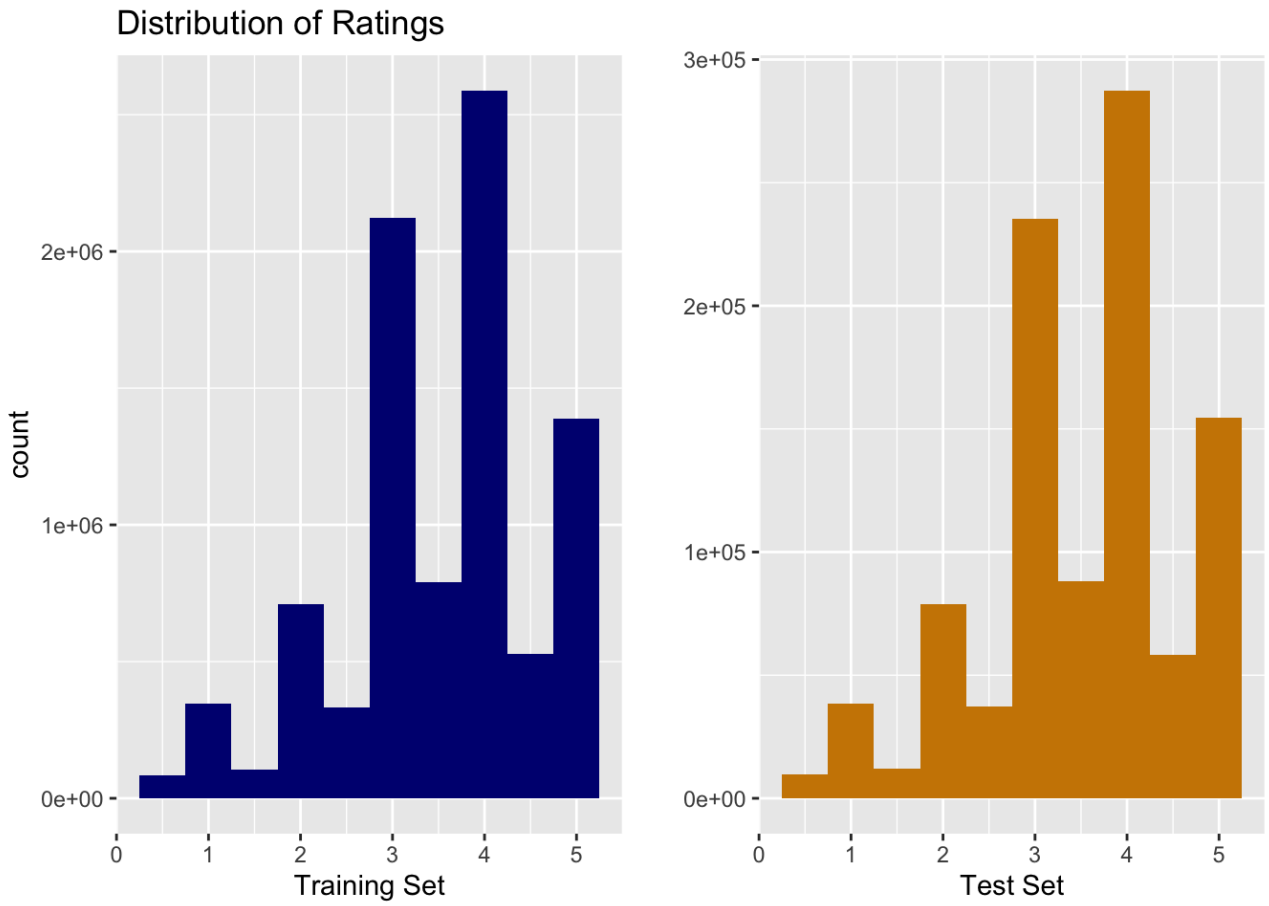
2. Exploratory Data Analysis

Our main variable of interest is the *rating* a value between 0.5 to 5, in intervals of 0.5, that are assigned to each movie (identified by the *movieId* and *title* variables) by users (identified by the *userId* variable). The full data set that we constructed (*movielens*) consists of 10,000,054 observations pertaining to individual ratings provided by 69,878 users to 10,677 movies. Of this data set, 10% of observations are randomly selected for the validation set (*final_holdout_test*) and the rest as the training set *edx*. As could be gleaned from the table and histograms below, the partitioning resulted in balanced characteristics between the training and test sets.

```
## Basic Descriptives: Movie Ratings

descriptives_movielens <- movielens %>% summarize(title = "Full MovieLens 10M Data Set",
                                     NoMovies = n_distinct(movieId),
                                     NoRaters = n_distinct(userId),
                                     RatersPerMovie = round(n_distinct(userId)/n_distinct
(movieId), 0),
                                     AveRating = round(mean(rating),3),
                                     SDRating = round(sd(rating),3),
                                     AveRatingDate = mean(ratingdate))
descriptives_train <- edx %>% summarize(title = "EDX Training Data Set",
                                  NoMovies = n_distinct(movieId),
                                  NoRaters = n_distinct(userId),
                                  RatersPerMovie = round(n_distinct(userId)/n_distinct(mov
ieId), 0),
                                  AveRating = round(mean(rating),3),
                                  SDRating = round(sd(rating),3),
                                  AveRatingDate = mean(ratingdate))
descriptives_holdout <- final_holdout_test %>% summarize(title = "Final Holdout Test Set",
                                                         NoMovies = n_distinct(movieId),
                                                         NoRaters = n_distinct(userId),
                                                         RatersPerMovie = round(n_distinct
(userId)/n_distinct(movieId), 0),
                                                         AveRating = round(mean(rating),
3),
                                                         SDRating = round(sd(rating),3),
                                                         AveRatingDate = mean(ratingdate))
descriptives <- bind_rows(descriptives_movielens, descriptives_train, descriptives_holdou
t)
rm(descriptives_movielens, descriptives_train, descriptives_holdout)
kable(t(descriptives[2:7]), align = "c", col.names = c("Full MovieLens 10M Data Set", "EDX
Training Data Set", "Final Holdout Test Set"))
```

	Full MovieLens 10M Data Set	EDX Training Data Set	Final Holdout Test Set
NoMovies	10677	10677	9809
NoRaters	69878	69878	68534
RatersPerMovie	7	7	7
AveRating	3.512	3.512	3.512
SDRating	1.060	1.060	1.061
AveRatingDate	2002-09-20	2002-09-21	2002-09-19

```
edx_hist <- edx %>% ggplot(aes(rating)) + geom_histogram(binwidth = 0.5, fill = "navy") +
labs(title = "Distribution of Ratings",x = "Training Set")
holdout_hist <- final_holdout_test %>% ggplot(aes(rating)) + geom_histogram(binwidth = 0.
5, fill = "orange3") + labs(x = "Test Set", y="", title = "")
grid.arrange(edx_hist, holdout_hist, ncol = 2)
```



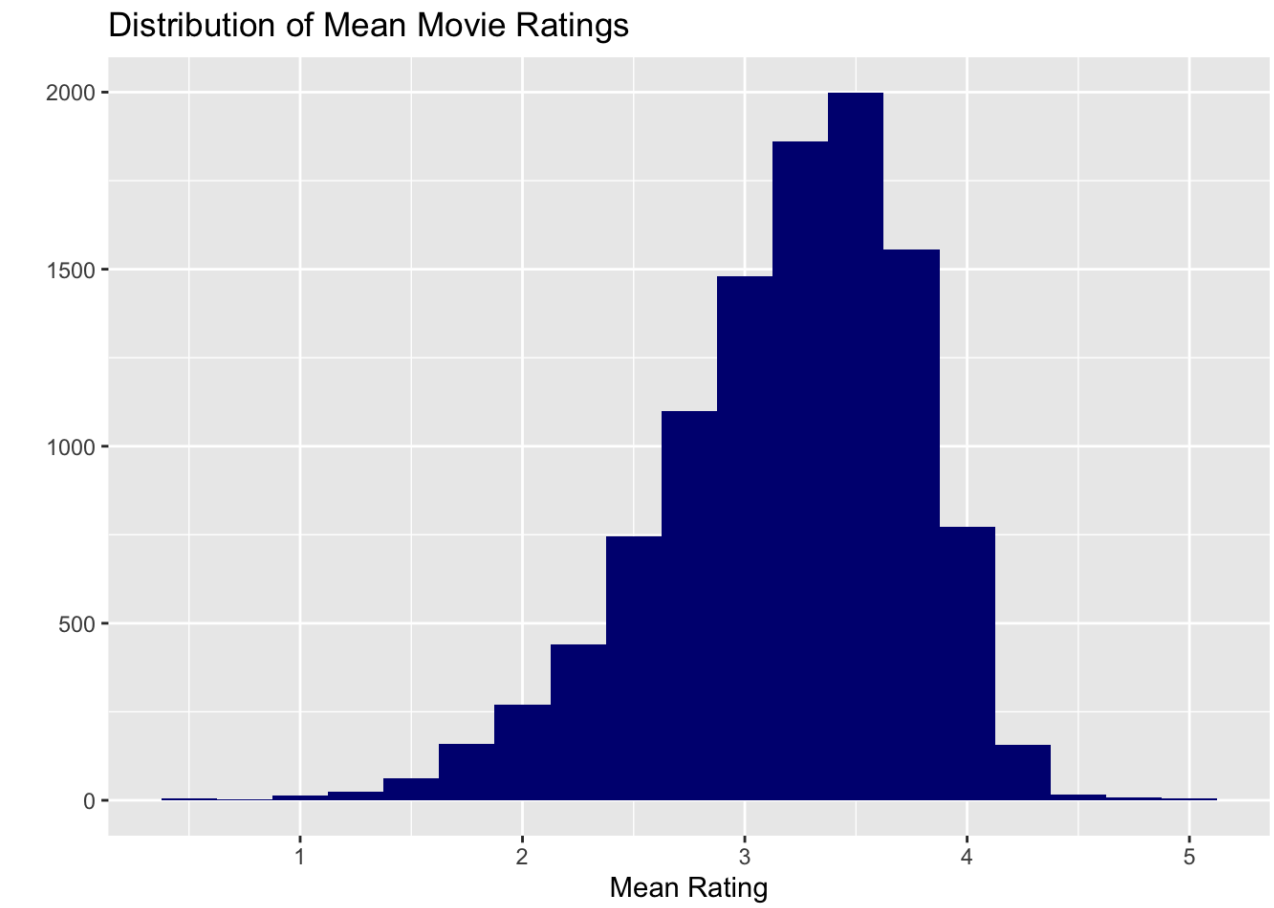

```
rm(edx_hist, holdout_hist)
```

The data set constructed contains other variables that could characterize each observation. These include the period of the movie's release (*movieyear* and *moviedecade*, both of which were extracted from the *title* of the movie) as well as the genre of the movie. As the original variable *genres* contained multiple tags or classifications per observation, logical variables for each of the 19 genres were generated. These are explored in the following subsections.

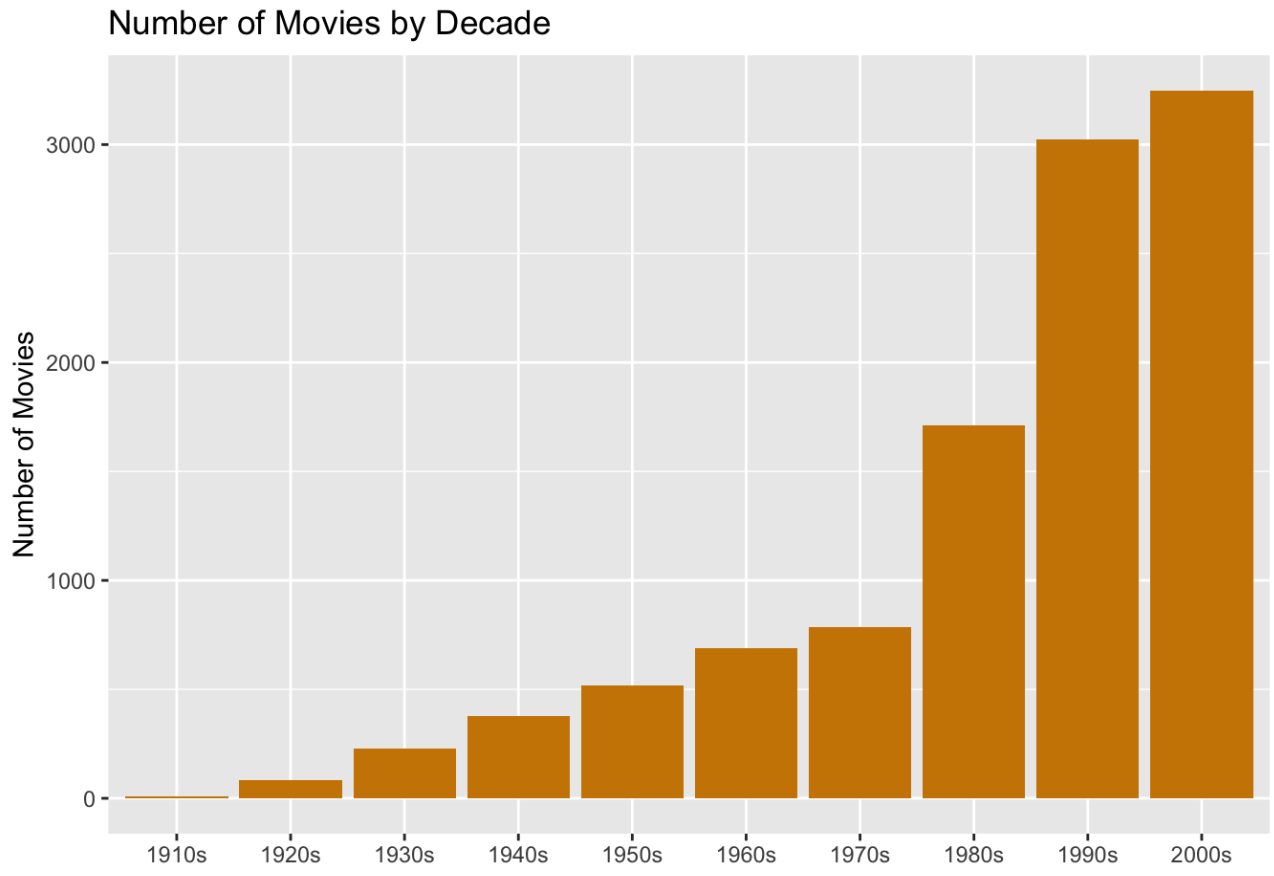
2.1 Exploring Movie and User Effects

The previous graph shows that the most ratings given are 3s and 4s, indicating that users tend to rate movies more positively than the midpoint. The mean rating is 3.512, with a standard deviation of about 1.1 points. How much do movies and users vary ratings from the mean? On the former, the charts below illustrate a few observations. First, the distribution of mean ratings per movie tends to be negatively skewed. Second, there also tend to be more movies released in the latter decades. Finally, more movies tend to be released under a limited set of genres, with drama and comedy being the clear front runners, and with thriller, romance, and action as distant followers.

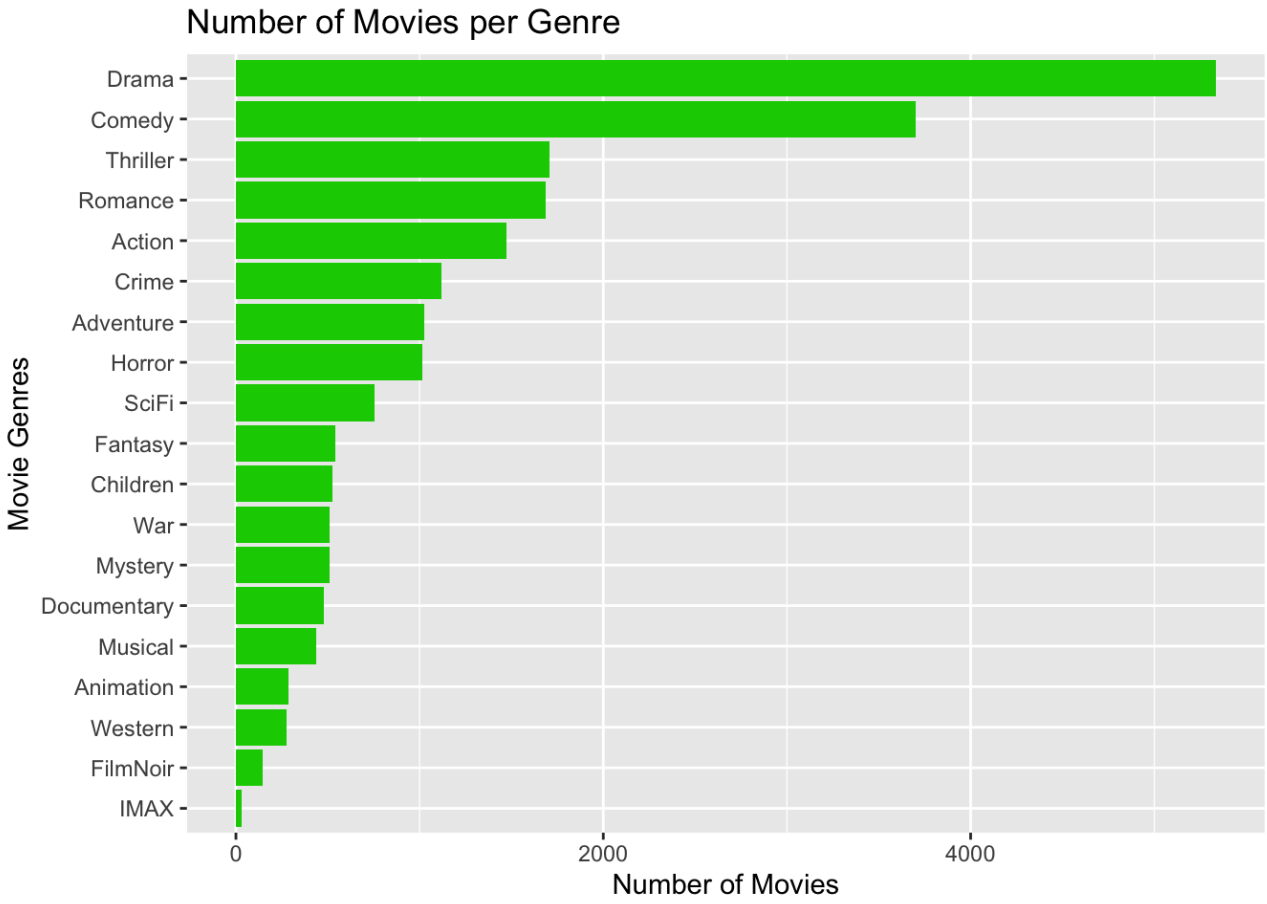
```
## Exploring Movie and User Effects on Ratings

#Movie Effects

edx %>% group_by(movieId) %>% summarize(meanRating = mean(rating)) %>%
  ggplot(aes(meanRating)) + geom_histogram(binwidth = 0.25, fill = "navy") +
  labs(x = "Mean Rating", y = "", title = "Distribution of Mean Movie Ratings")
```




```
edx %>% group_by(moviedecade) %>% summarize(noMovies = n_distinct(movieId)) %>%
  ggplot(aes(moviedecade, noMovies)) + geom_col(fill = "orange3") +
  labs(x = "", y = "Number of Movies", title = "Number of Movies by Decade")
```

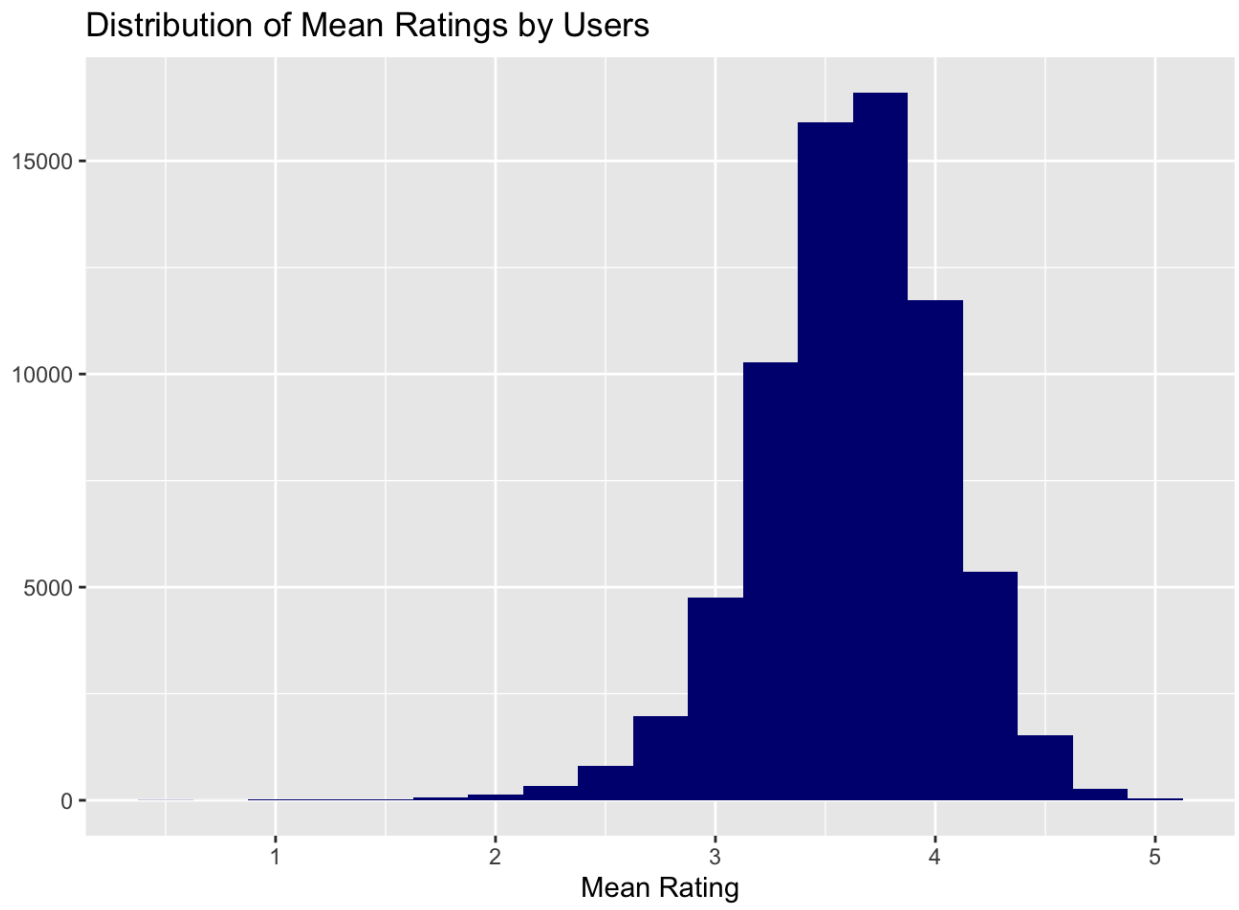


```
edx %>% group_by(movieId) %>% summarize(Action = ifelse(sum(action)>0,1,0),
                                        Adventure = ifelse(sum(adventure)>0,1,0),
                                        Animation = ifelse(sum(animation)>0,1,0),
                                        Children = ifelse(sum(children)>0,1,0),
                                        Comedy = ifelse(sum(comedy)>0,1,0),
                                        Crime = ifelse(sum(crime)>0,1,0),
                                        Documentary = ifelse(sum(documentary)>0,1,0),
                                        Drama = ifelse(sum(drama)>0,1,0),
                                        Fantasy = ifelse(sum(fantasy)>0,1,0),
                                        FilmNoir = ifelse(sum(filmnoir)>0,1,0),
                                        Horror = ifelse(sum(horror)>0,1,0),
                                        IMAX = ifelse(sum(imax)>0,1,0),
                                        Musical = ifelse(sum(musical)>0,1,0),
                                        Mystery = ifelse(sum(mystery)>0,1,0),
                                        Romance = ifelse(sum(romance)>0,1,0),
                                        SciFi = ifelse(sum(scifi)>0,1,0),
                                        Thriller = ifelse(sum(thriller)>0,1,0),
                                        War = ifelse(sum(war)>0,1,0),
                                        Western = ifelse(sum(western)>0,1,0)) %>%
  summarize(Action = sum(Action), Adventure = sum(Adventure), Animation = sum(Animation),
            Children = sum(Children), Comedy = sum(Comedy), Crime = sum(Crime),
            Documentary = sum(Documentary), Drama = sum(Drama), Fantasy = sum(Fantasy),
            FilmNoir = sum(FilmNoir), Horror = sum(Horror), IMAX = sum(IMAX),
            Musical = sum(Musical), Mystery = sum(Mystery), Romance = sum(Romance),
            SciFi = sum(SciFi), Thriller = sum(Thriller), War = sum(War), Western = sum(We
stern)) %>%
  t() %>% as.data.frame() %>% rownames_to_column(var = "Genre") %>% rename(noMovies = V1)
%>%
  ggplot(aes(noMovies, fct_reorder(Genre, noMovies))) + geom_col(fill="green3") +
  labs(x = "Number of Movies", y = "Movie Genres", title = "Number of Movies per Genre")
```

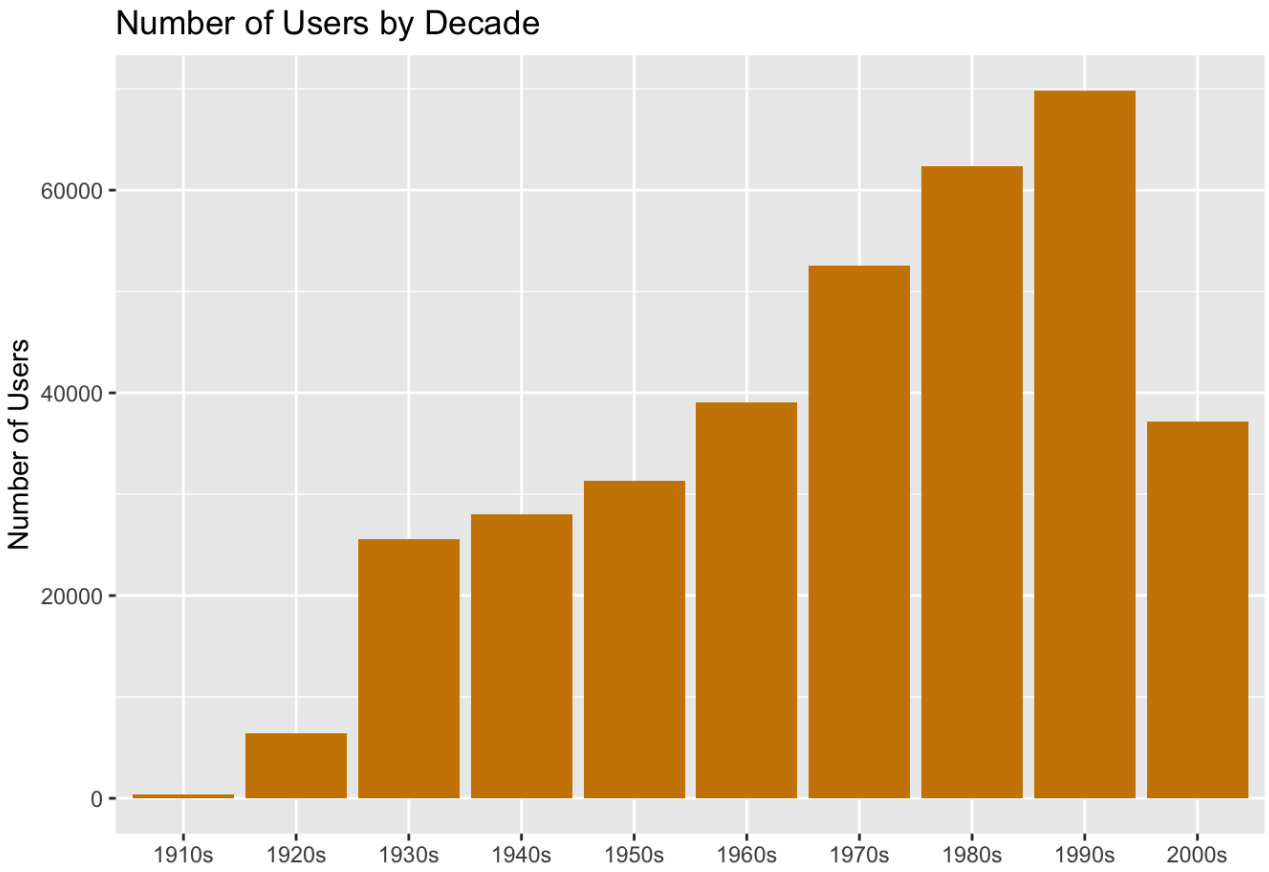


The data on users also seem to show similar behaviors but with key nuances. First, the distribution of average ratings per user also seems to be skewed negatively. There is also an increasing trend in the number of users per decade, but except in the 2000s where there's a clear drop. Finally, drama, comedy, action, thriller, and romance are also the most rated genres, although the number of users per genre seems to be a bit more balanced than the number of movies per genre. These facts will be useful to remember in the modeling exercises later.

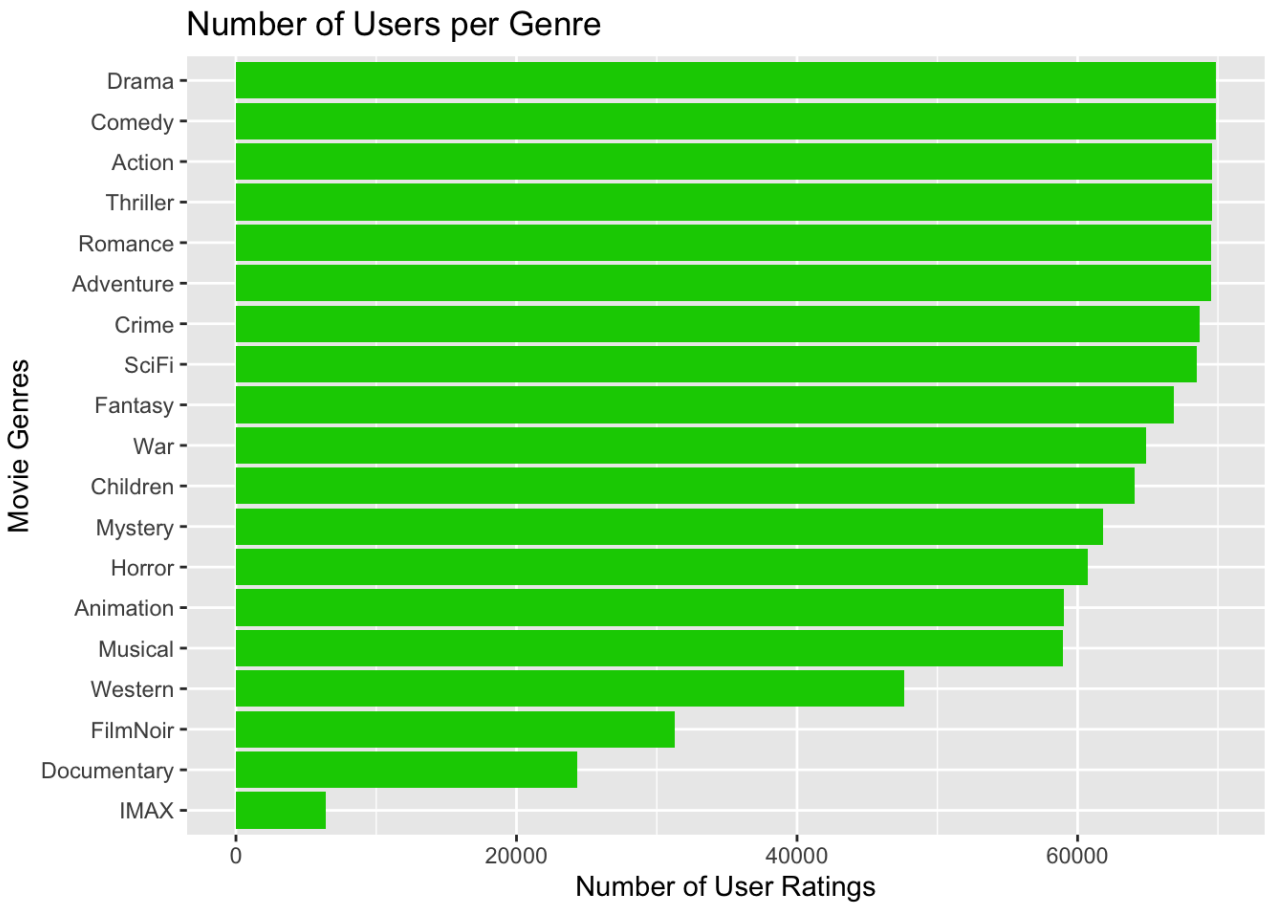
```
#User Effects

edx %>% group_by(userId) %>% summarize(meanRating = mean(rating)) %>%
  ggplot(aes(meanRating)) + geom_histogram(binwidth = 0.25, fill = "navy") +
  labs(x = "Mean Rating", y = "", title = "Distribution of Mean Ratings by Users")
```




```
edx %>% group_by(moviedecade) %>% summarize(noUsers = n_distinct(userId)) %>%
  ggplot(aes(moviedecade, noUsers)) + geom_col(fill = "orange3") +
  labs(x = "", y = "Number of Users", title = "Number of Users by Decade")
```


Number of Users by Decade
Number of Users
60000
40000
20000
0
1910s
1920s
1930s
1940s
1950s
1960s
1970s
1980s
1990s
2000s

```
edx %>% group_by(userId) %>% summarize(Action = ifelse(sum(action)>0,1,0),
                                       Adventure = ifelse(sum(adventure)>0,1,0),
                                       Animation = ifelse(sum(animation)>0,1,0),
                                       Children = ifelse(sum(children)>0,1,0),
                                       Comedy = ifelse(sum(comedy)>0,1,0),
                                       Crime = ifelse(sum(crime)>0,1,0),
                                       Documentary = ifelse(sum(documentary)>0,1,0),
                                       Drama = ifelse(sum(drama)>0,1,0),
                                       Fantasy = ifelse(sum(fantasy)>0,1,0),
                                       FilmNoir = ifelse(sum(filmnoir)>0,1,0),
                                       Horror = ifelse(sum(horror)>0,1,0),
                                       IMAX = ifelse(sum(imax)>0,1,0),
                                       Musical = ifelse(sum(musical)>0,1,0),
                                       Mystery = ifelse(sum(mystery)>0,1,0),
                                       Romance = ifelse(sum(romance)>0,1,0),
                                       SciFi = ifelse(sum(scifi)>0,1,0),
                                       Thriller = ifelse(sum(thriller)>0,1,0),
                                       War = ifelse(sum(war)>0,1,0),
                                       Western = ifelse(sum(western)>0,1,0)) %>%
  summarize(Action = sum(Action), Adventure = sum(Adventure), Animation = sum(Animation),
            Children = sum(Children), Comedy = sum(Comedy), Crime = sum(Crime),
            Documentary = sum(Documentary), Drama = sum(Drama), Fantasy = sum(Fantasy),
            FilmNoir = sum(FilmNoir), Horror = sum(Horror), IMAX = sum(IMAX),
            Musical = sum(Musical), Mystery = sum(Mystery), Romance = sum(Romance),
            SciFi = sum(SciFi), Thriller = sum(Thriller), War = sum(War), Western = sum(We
stern)) %>%
  t() %>% as.data.frame() %>% rownames_to_column(var = "Genre") %>% rename(noMovies = V1)
%>%
  ggplot(aes(noMovies, fct_reorder(Genre, noMovies))) + geom_col(fill="green3") +
  labs(x = "Number of User Ratings", y = "Movie Genres", title = "Number of Users per Genr
e")
```

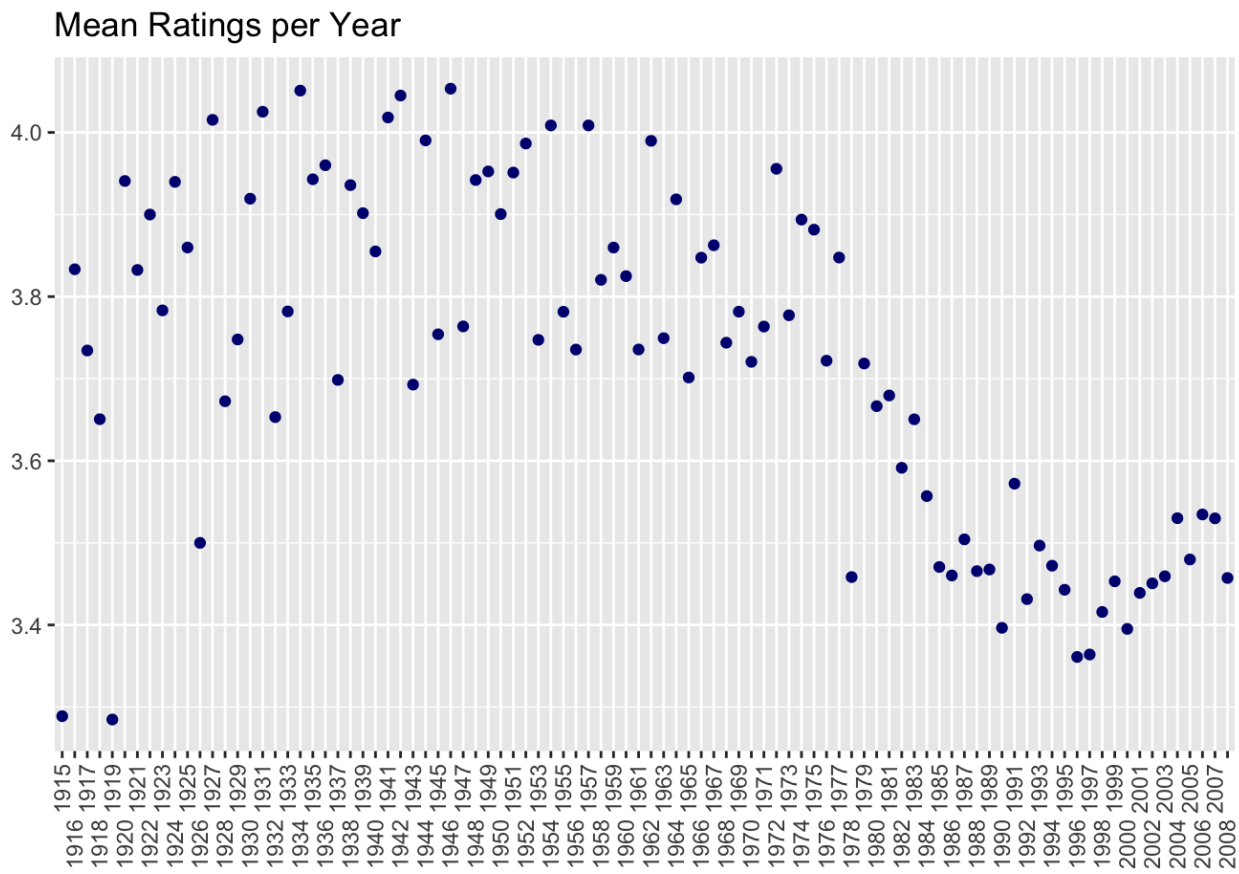


2.2 Ratings per Year and Decade

How do movie ratings change through the years? The scatter plot below implies a slightly decreasing trend in the mean ratings given to all movies per year. After grouping by decade, the decreasing trend is not as evident: rather, one could glean that average ratings peak around the 1920s to 1950s. To recall, the number of movies released and number of rating users per decade were still low during these times.

```
## Ratings per Year and Decade

# Per year

edx %>% group_by(movieyear) %>% summarize(meanRating = mean(rating)) %>%
  ggplot(aes(movieyear, meanRating)) + geom_point(color = "navy") + geom_smooth() +
  labs(title = "Mean Ratings per Year", x = "", y = "") +
  guides(x = guide_axis(angle = 90, n.dodge = 2))
```

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

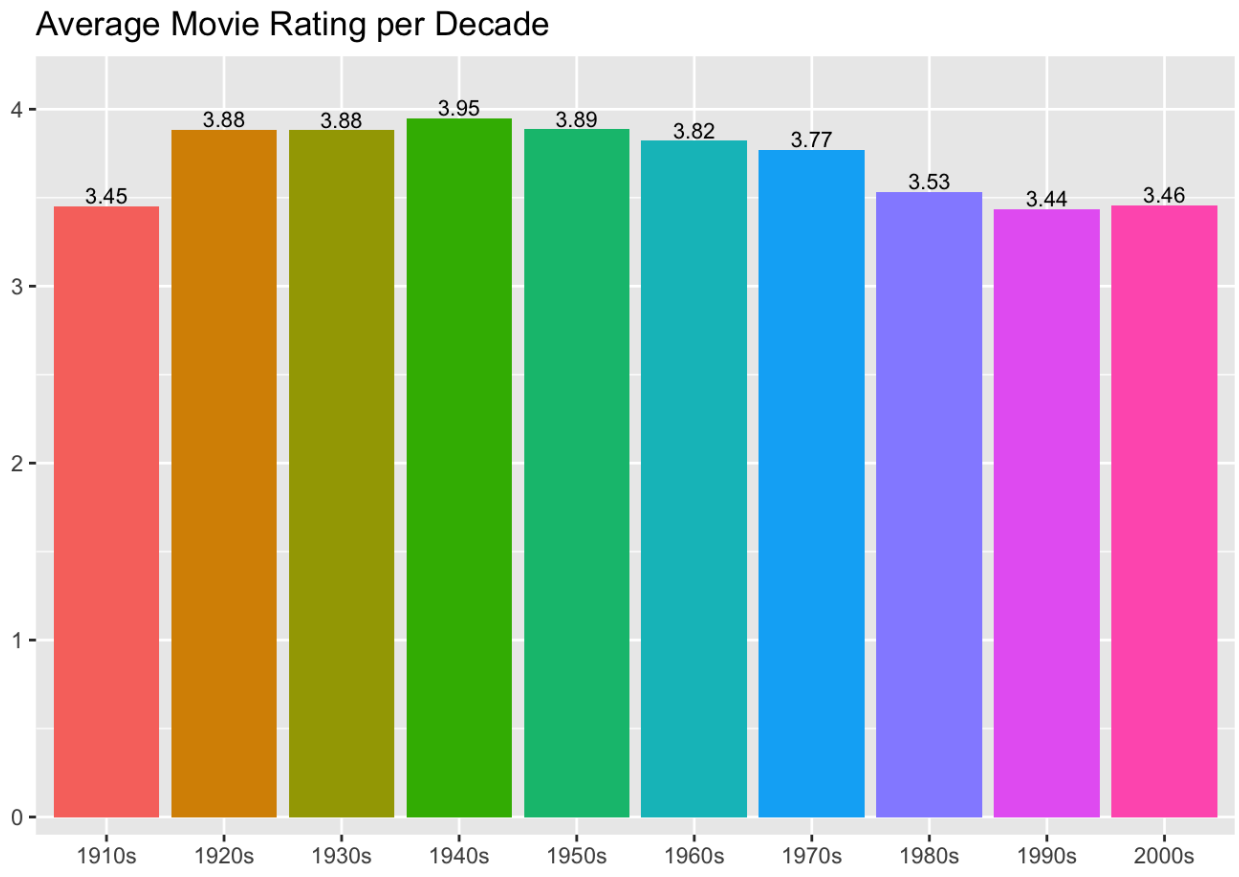



```
# Per decade

stats_decade <- edx %>% group_by(MovieDecade = moviedecade) %>% summarize(NoMovies = n_dis
tinct(movieId),  NoRaters = n_distinct(userId), RatersPerMovie = round(n_distinct(userId)/
n_distinct(movieId), 0), AveRating = round(mean(rating),3),  SDRating = round(sd(rating),
3), AveRatingDate = mean(ratingdate))
stats_decade %>% kable(digits = 2, align = "c")
```

MovieDecade	NoMovies	NoRaters	RatersPerMovie	AveRating	SDRating	AveRatingDate
1910s	11	414	38	3.45	1.12	2004-07-21
1920s	83	6443	78	3.88	0.98	2003-11-10

MovieDecade	NoMovies	NoRaters	RatersPerMovie	AveRating	SDRating	AveRatingDate
1930s	230	25586	111	3.88	0.96	2002-08-31
1940s	377	28048	74	3.95	0.94	2002-09-18
1950s	520	31364	60	3.89	0.94	2002-09-14
1960s	690	39024	57	3.82	0.96	2002-12-22
1970s	784	52584	67	3.77	1.01	2002-11-25
1980s	1712	62394	36	3.53	1.05	2002-11-28
1990s	3022	69832	23	3.44	1.08	2001-07-08
2000s	3248	37138	11	3.46	1.04	2005-08-17

```
stats_decade %>% ggplot(aes(fct_reorder(MovieDecade, MovieDecade), AveRating, fill = Movie
Decade)) +
  geom_col() + geom_text(aes(label = round(AveRating,2)), vjust = -0.2, size = 3) +
  labs(x = "", y = "", title = "Average Movie Rating per Decade") +
  coord_cartesian(ylim = c(0.1,4.1)) +
  theme(legend.position = "none")
```



2.3 Ratings per Genre

It is interesting to note that, on average, certain movie genres garnered higher ratings than the others. The highest-rated genre is Film Noir, followed by Documentaries, War, and IMAX movies. In contrast, Horror is the lowest-rated genre, followed by Sci-Fi, Children, and Action.

```
## Descriptive Statistics per Genre

stats_action <- edx %>% filter(action == TRUE) %>% summarize(NoMovies = n_distinct(movieI
d),
                                                             AveReleaseYear = round(mean(a
s.integer(movieyear)),0),
                                                             NoRaters = n_distinct(userI
d),
                                                             RatersPerMovie = round(n_dist
inct(userId)/n_distinct(movieId), 0),
                                                             AveRating = mean(rating),
                                                             SDRating = sd(rating),
                                                             AveRatingDate = mean(ratingda
te))
stats_adventure <- edx %>% filter(adventure == TRUE) %>% summarize(NoMovies = n_distinct(m
ovieId),
                                                             AveReleaseYear = round(mean(a
s.integer(movieyear)),0),
                                                             NoRaters = n_distinct(userI
d),
                                                             RatersPerMovie = round(n_dist
inct(userId)/n_distinct(movieId), 0),
                                                             AveRating = mean(rating),
                                                             SDRating = sd(rating),
                                                             AveRatingDate = mean(ratingda
te))
stats_animation <- edx %>% filter(animation == TRUE) %>% summarize(NoMovies = n_distinct(m
ovieId),
                                                             AveReleaseYear = round(mean(a
s.integer(movieyear)),0),
                                                             NoRaters = n_distinct(userI
d),
                                                             RatersPerMovie = round(n_dist
inct(userId)/n_distinct(movieId), 0),
                                                             AveRating = mean(rating),
                                                             SDRating = sd(rating),
                                                             AveRatingDate = mean(ratingda
te))
stats_children <- edx %>% filter(children == TRUE) %>% summarize(NoMovies = n_distinct(mov
ieId),
                                                             AveReleaseYear = round(mean(a
s.integer(movieyear)),0),
                                                             NoRaters = n_distinct(userI
d),
                                                             RatersPerMovie = round(n_dist
inct(userId)/n_distinct(movieId), 0),
                                                             AveRating = mean(rating),
                                                             SDRating = sd(rating),
                                                             AveRatingDate = mean(ratingda
te))
stats_comedy <- edx %>% filter(comedy == TRUE) %>% summarize(NoMovies = n_distinct(movieI
d),
                                                             AveReleaseYear = round(mean(a
s.integer(movieyear)),0),
                                                             NoRaters = n_distinct(userI
d),
                                                             RatersPerMovie = round(n_dist
inct(userId)/n_distinct(movieId), 0),
```

```
                                                    AveRating = mean(rating),
                                                    SDRating = sd(rating),
                                                    AveRatingDate = mean(ratingda
te))
stats_crime <- edx %>% filter(crime == TRUE) %>% summarize(NoMovies = n_distinct(movieId),
                                                    AveReleaseYear = round(mean(a
s.integer(movieyear)),0),
                                                    NoRaters = n_distinct(userI
d),
                                                    RatersPerMovie = round(n_dist
inct(userId)/n_distinct(movieId), 0),
                                                    AveRating = mean(rating),
                                                    SDRating = sd(rating),
                                                    AveRatingDate = mean(ratingda
te))
stats_documentary <- edx %>% filter(documentary == TRUE) %>% summarize(NoMovies = n_distin
ct(movieId),
                                                    AveReleaseYear = round(mean(a
s.integer(movieyear)),0),
                                                    NoRaters = n_distinct(userI
d),
                                                    RatersPerMovie = round(n_dist
inct(userId)/n_distinct(movieId), 0),
                                                    AveRating = mean(rating),
                                                    SDRating = sd(rating),
                                                    AveRatingDate = mean(ratingda
te))
stats_drama <- edx %>% filter(drama == TRUE) %>% summarize(NoMovies = n_distinct(movieId),
                                                    AveReleaseYear = round(mean(a
s.integer(movieyear)),0),
                                                    NoRaters = n_distinct(userI
d),
                                                    RatersPerMovie = round(n_dist
inct(userId)/n_distinct(movieId), 0),
                                                    AveRating = mean(rating),
                                                    SDRating = sd(rating),
                                                    AveRatingDate = mean(ratingda
te))
stats_fantasy <- edx %>% filter(fantasy == TRUE) %>% summarize(NoMovies = n_distinct(movie
Id),
                                                    AveReleaseYear = round(mean(a
s.integer(movieyear)),0),
                                                    NoRaters = n_distinct(userI
d),
                                                    RatersPerMovie = round(n_dist
inct(userId)/n_distinct(movieId), 0),
                                                    AveRating = mean(rating),
                                                    SDRating = sd(rating),
                                                    AveRatingDate = mean(ratingda
te))
stats_filmnoir <- edx %>% filter(filmnoir == TRUE) %>% summarize(NoMovies = n_distinct(mov
ieId),
                                                    AveReleaseYear = round(mean(a
s.integer(movieyear)),0),
                                                    NoRaters = n_distinct(userI
d),
                                                    RatersPerMovie = round(n_dist
inct(userId)/n_distinct(movieId), 0),
                                                    AveRating = mean(rating),
```

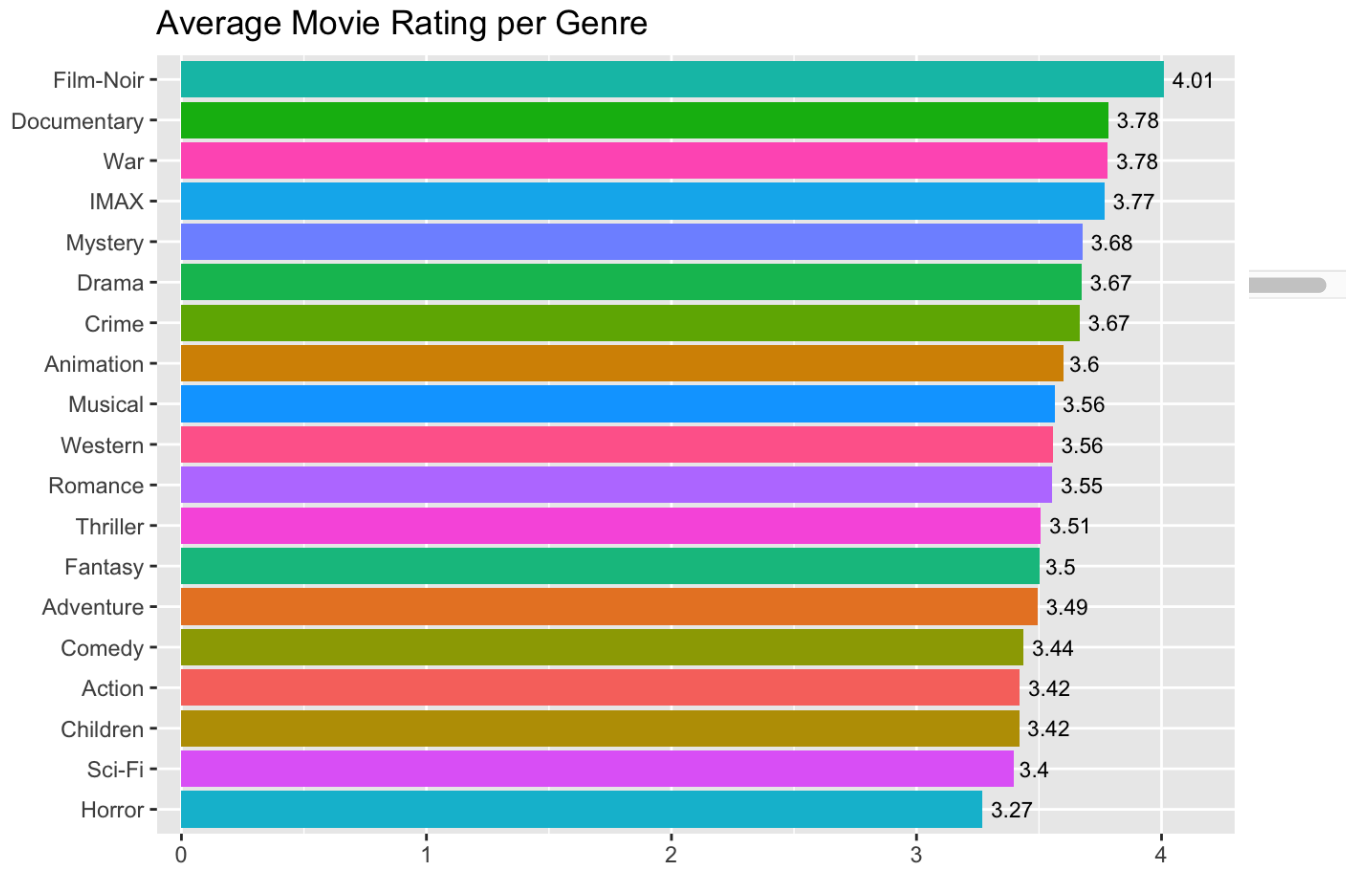
```
                                                                SDRating = sd(rating),
                                                                AveRatingDate = mean(ratingda
te))
stats_horror <- edx %>% filter(horror == TRUE) %>% summarize(NoMovies = n_distinct(movieI
d),
                                                                AveReleaseYear = round(mean(a
s.integer(movieyear)),0),
                                                                NoRaters = n_distinct(userI
d),
                                                                RatersPerMovie = round(n_dist
inct(userId)/n_distinct(movieId), 0),
                                                                AveRating = mean(rating),
                                                                SDRating = sd(rating),
                                                                AveRatingDate = mean(ratingda
te))
stats_imax <- edx %>% filter(imax == TRUE) %>% summarize(NoMovies = n_distinct(movieId),
                                                                AveReleaseYear = round(mean(a
s.integer(movieyear)),0),
                                                                NoRaters = n_distinct(userI
d),
                                                                RatersPerMovie = round(n_dist
inct(userId)/n_distinct(movieId), 0),
                                                                AveRating = mean(rating),
                                                                SDRating = sd(rating),
                                                                AveRatingDate = mean(ratingda
te))
stats_musical <- edx %>% filter(musical == TRUE) %>% summarize(NoMovies = n_distinct(movie
Id),
                                                                AveReleaseYear = round(mean(a
s.integer(movieyear)),0),
                                                                NoRaters = n_distinct(userI
d),
                                                                RatersPerMovie = round(n_dist
inct(userId)/n_distinct(movieId), 0),
                                                                AveRating = mean(rating),
                                                                SDRating = sd(rating),
                                                                AveRatingDate = mean(ratingda
te))
stats_mystery <- edx %>% filter(mystery == TRUE) %>% summarize(NoMovies = n_distinct(movie
Id),
                                                                AveReleaseYear = round(mean(a
s.integer(movieyear)),0),
                                                                NoRaters = n_distinct(userI
d),
                                                                RatersPerMovie = round(n_dist
inct(userId)/n_distinct(movieId), 0),
                                                                AveRating = mean(rating),
                                                                SDRating = sd(rating),
                                                                AveRatingDate = mean(ratingda
te))
stats_romance <- edx %>% filter(romance == TRUE) %>% summarize(NoMovies = n_distinct(movie
Id),
                                                                AveReleaseYear = round(mean(a
s.integer(movieyear)),0),
                                                                NoRaters = n_distinct(userI
d),
                                                                RatersPerMovie = round(n_dist
inct(userId)/n_distinct(movieId), 0),
                                                                AveRating = mean(rating),
```

```
                                                                SDRating = sd(rating),
                                                                AveRatingDate = mean(ratingda
te))
stats_scifi <- edx %>% filter(scifi == TRUE) %>% summarize(NoMovies = n_distinct(movieId),
                                                                AveReleaseYear = round(mean(a
s.integer(movieyear)),0),
                                                                NoRaters = n_distinct(userI
d),
                                                                RatersPerMovie = round(n_dist
inct(userId)/n_distinct(movieId), 0),
                                                                AveRating = mean(rating),
                                                                SDRating = sd(rating),
                                                                AveRatingDate = mean(ratingda
te))
stats_thriller <- edx %>% filter(thriller == TRUE) %>% summarize(NoMovies = n_distinct(mov
ieId),
                                                                AveReleaseYear = round(mean(a
s.integer(movieyear)),0),
                                                                NoRaters = n_distinct(userI
d),
                                                                RatersPerMovie = round(n_dist
inct(userId)/n_distinct(movieId), 0),
                                                                AveRating = mean(rating),
                                                                SDRating = sd(rating),
                                                                AveRatingDate = mean(ratingda
te))
stats_war <- edx %>% filter(war == TRUE) %>% summarize(NoMovies = n_distinct(movieId),
                                                                AveReleaseYear = round(mean(a
s.integer(movieyear)),0),
                                                                NoRaters = n_distinct(userI
d),
                                                                RatersPerMovie = round(n_dist
inct(userId)/n_distinct(movieId), 0),
                                                                AveRating = mean(rating),
                                                                SDRating = sd(rating),
                                                                AveRatingDate = mean(ratingda
te))
stats_western <- edx %>% filter(western == TRUE) %>% summarize(NoMovies = n_distinct(movie
Id),
                                                                AveReleaseYear = round(mean(a
s.integer(movieyear)),0),
                                                                NoRaters = n_distinct(userI
d),
                                                                RatersPerMovie = round(n_dist
inct(userId)/n_distinct(movieId), 0),
                                                                AveRating = mean(rating),
                                                                SDRating = sd(rating),
                                                                AveRatingDate = mean(ratingda
te))

stats_genre <- bind_rows(stats_action, stats_adventure, stats_animation, stats_children, s
tats_comedy,
                         stats_crime, stats_documentary, stats_drama,stats_fantasy, stats_
filmnoir,
                         stats_horror, stats_imax, stats_musical, stats_mystery, stats_rom
ance,
                         stats_scifi, stats_thriller, stats_war, stats_western)
stats_genre <- bind_cols(Genre = genrelist[2:20],stats_genre)
rm(stats_action, stats_adventure, stats_animation, stats_children, stats_comedy, stats_cri
```

```
me,
   stats_documentary, stats_drama, stats_fantasy, stats_filmnoir, stats_horror, stats_ima
x,
   stats_musical, stats_mystery, stats_romance, stats_scifi, stats_thriller, stats_war, st
ats_western)
stats_genre <- as_tibble(stats_genre)

stats_genre %>% arrange(desc(AveRating)) %>% kable(digits = 2, align = "c")
```

Genre	NoMovies	AveReleaseYear	NoRaters	RatersPerMovie	AveRating	SDRating	AveRatingDate
Film-Noir	148	1975	31270	211	4.01	0.89	2003-02-12
Documentary	481	1997	24295	51	3.78	1.00	2003-11-19
War	510	1986	64892	127	3.78	1.01	2002-08-25
IMAX	29	2003	6393	220	3.77	1.03	2005-11-15
Mystery	509	1989	61845	122	3.68	1.00	2003-03-11
Drama	5336	1990	69866	13	3.67	1.00	2002-09-26
Crime	1117	1992	68691	61	3.67	1.01	2002-12-11
Animation	286	1989	59018	206	3.60	1.02	2003-02-03
Musical	436	1978	58918	135	3.56	1.06	2002-06-25
Western	275	1984	47648	173	3.56	1.02	2002-03-11
Romance	1685	1990	69530	41	3.55	1.03	2002-06-11
Thriller	1705	1993	69567	41	3.51	1.03	2002-09-22
Fantasy	543	1990	66833	123	3.50	1.07	2003-04-11
Adventure	1025	1991	69521	68	3.49	1.05	2002-11-21
Comedy	3703	1991	69864	19	3.44	1.07	2002-09-22
Action	1473	1992	69607	47	3.42	1.07	2002-11-03
Children	528	1987	64059	121	3.42	1.09	2002-08-26
Sci-Fi	754	1990	68469	91	3.40	1.09	2002-11-27
Horror	1013	1988	60695	60	3.27	1.15	2003-01-07

```
stats_genre %>% ggplot(aes(AveRating, fct_reorder(Genre, AveRating), fill = Genre)) + geom
_col() +
  geom_text(aes(label = round(AveRating,2)), hjust = -0.2, size = 3) +
  labs(x = "", y = "", title = "Average Movie Rating per Genre") +
  coord_cartesian(xlim = c(0.1,4.1)) +
  theme(legend.position = "none")
```

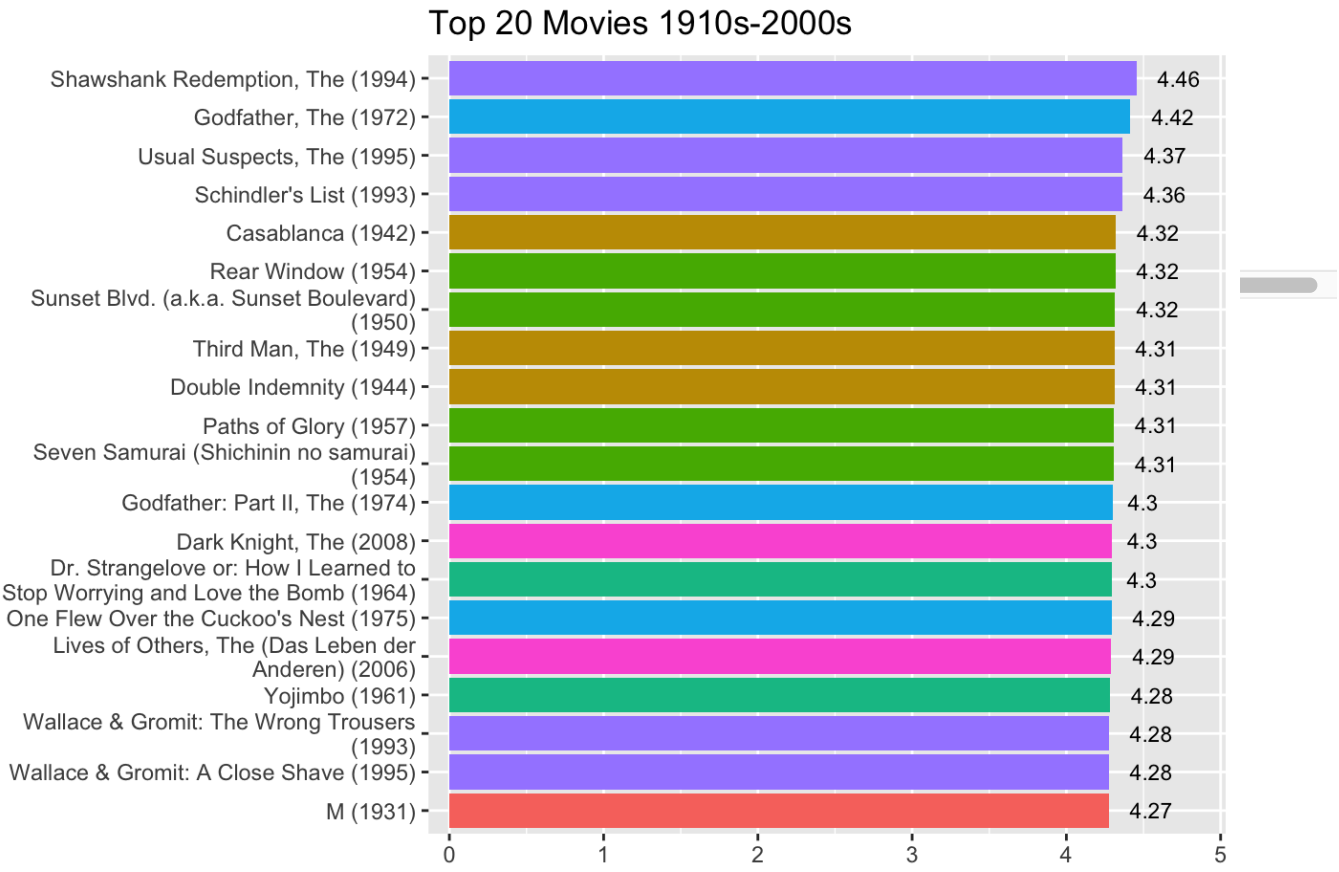


2.4 Top Rated Movies

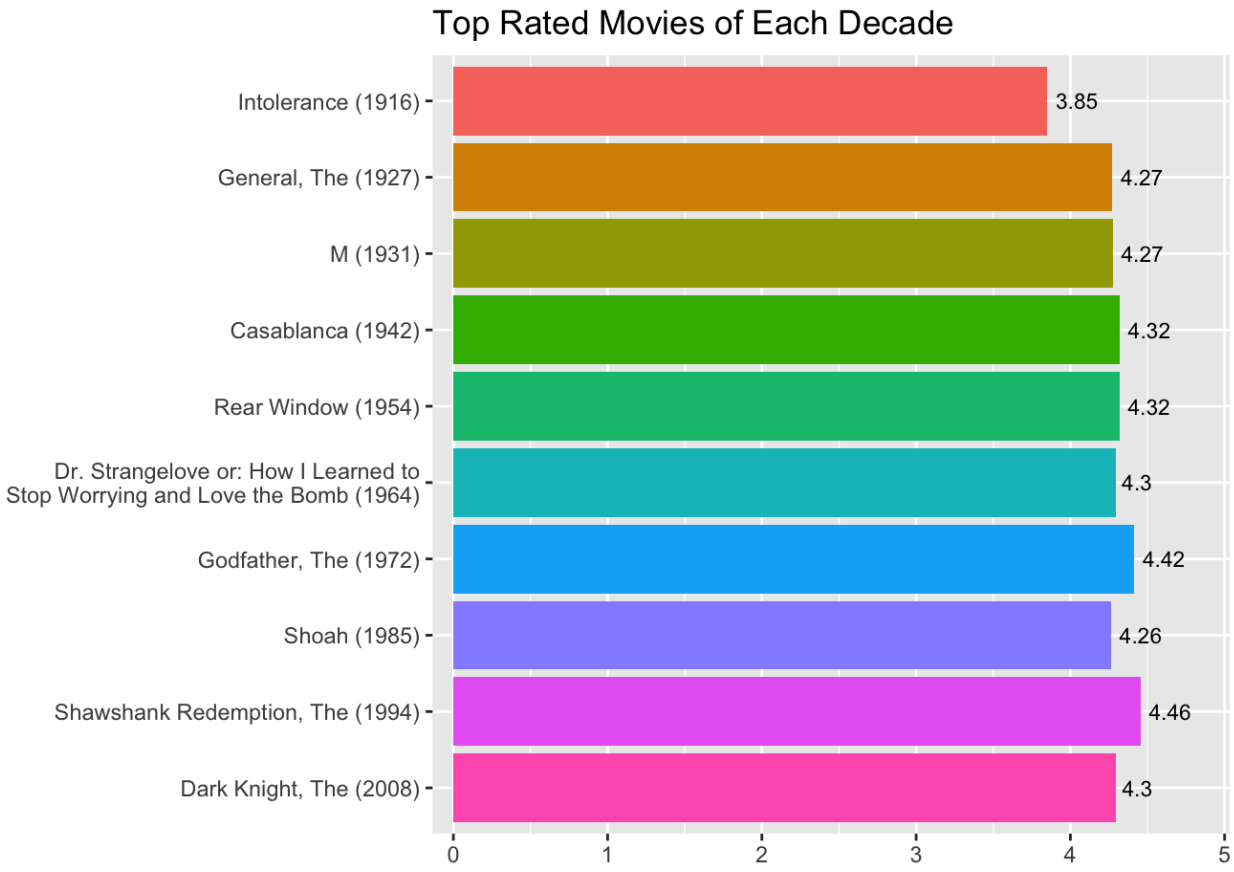
It is also of interest to note which of the movies covered in the data set were the most highly rated. However, there are several movies that only had a handful user ratings in the data set, and these distort the listing as many of them had either very high or very low ratings. To present more robust rankings, we limited the movies to those which had at least 10 user ratings. I note that 10 is a arbitrarily defined minimum, and later in the machine learning section a model that optimizes this minimum will be trained.

The frontunner is Shawshank Redemption: a drama released in 1994 that garnered an average rating of 4.46 points. The following graphics show the top 30 rated movies of all time, per decade, and per genre.

```
## Top Rated Movies

# Top All Time (Only of Movies with 10 or more Raters)

stats_movie <- edx %>% group_by(movieId) %>%
  summarize(AveRating = mean(rating), NoRaters = n_distinct(userId)) %>%
  inner_join(movies, by = join_by(movieId))

top_all_time <- stats_movie[1:7] %>% filter(NoRaters >= 10) %>% arrange(desc(AveRating)) %
>% head(top_all_time, n = 20) %>% select(-movieyear)

top_all_time %>% ggplot(aes(AveRating, fct_reorder(title, AveRating), fill = moviedecade))
+ geom_col() +
  geom_text(aes(label = round(AveRating,2)), hjust = -0.5, size = 3) +
  labs(title = "Top 20 Movies 1910s-2000s", x = "", y = "") +
  scale_y_discrete(labels = function(y) str_wrap(y, width = 40)) +
  coord_cartesian(xlim = c(0.1,4.8)) + theme(legend.position = "none")
```

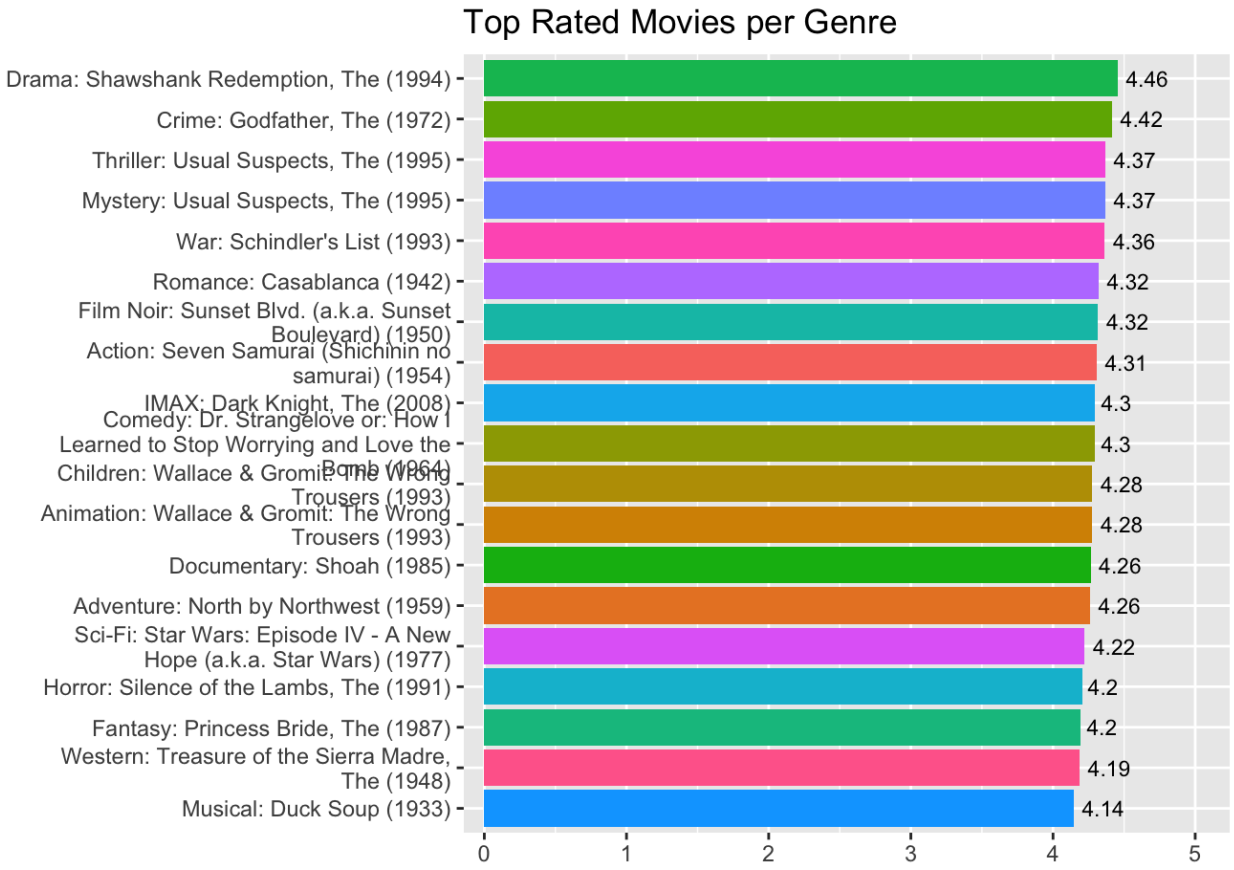

Top 20 Movies 1910s-2000s
Shawshank Redemption, The (1994)
Godfather, The (1972)
Usual Suspects, The (1995)
Schindler's List (1993)
Casablanca (1942)
Rear Window (1954)
Sunset Blvd. (a.k.a. Sunset Boulevard) (1950)
Third Man, The (1949)
Double Indemnity (1944)
Paths of Glory (1957)
Seven Samurai (Shichinin no samurai) (1954)
Godfather: Part II, The (1974)
Dark Knight, The (2008)
Dr. Strangelove or: How I Learned to Stop Worrying and Love the Bomb (1964)
One Flew Over the Cuckoo's Nest (1975)
Lives of Others, The (Das Leben der Anderen) (2006)
Yojimbo (1961)
Wallace & Gromit: The Wrong Trousers (1993)
Wallace & Gromit: A Close Shave (1995)
M (1931)
4.46
4.42
4.37
4.36
4.32
4.32
4.32
4.31
4.31
4.31
4.31
4.3
4.3
4.3
4.29
4.29
4.28
4.28
4.28
4.27
0
1
2
3
4
5

```r
# Top Rated Movies Per Decade
top_10s <- stats_movie[1:7] %>% filter(NoRaters >= 10 & moviedecade == "1910s") %>% arrang
e(desc(AveRating)) %>% head(n = 1)
top_20s <- stats_movie[1:7] %>% filter(NoRaters >= 10 & moviedecade == "1920s") %>% arrang
e(desc(AveRating)) %>% head(n = 1)
top_30s <- stats_movie[1:7] %>% filter(NoRaters >= 10 & moviedecade == "1930s") %>% arrang
e(desc(AveRating)) %>% head(n = 1)
top_40s <- stats_movie[1:7] %>% filter(NoRaters >= 10 & moviedecade == "1940s") %>% arrang
e(desc(AveRating)) %>% head(n = 1)
top_50s <- stats_movie[1:7] %>% filter(NoRaters >= 10 & moviedecade == "1950s") %>% arrang
e(desc(AveRating)) %>% head(n = 1)
top_60s <- stats_movie[1:7] %>% filter(NoRaters >= 10 & moviedecade == "1960s") %>% arrang
e(desc(AveRating)) %>% head(n = 1)
top_70s <- stats_movie[1:7] %>% filter(NoRaters >= 10 & moviedecade == "1970s") %>% arrang
e(desc(AveRating)) %>% head(n = 1)
top_80s <- stats_movie[1:7] %>% filter(NoRaters >= 10 & moviedecade == "1980s") %>% arrang
e(desc(AveRating)) %>% head(n = 1)
top_90s <- stats_movie[1:7] %>% filter(NoRaters >= 10 & moviedecade == "1990s") %>% arrang
e(desc(AveRating)) %>% head(n = 1)
top_00s <- stats_movie[1:7] %>% filter(NoRaters >= 10 & moviedecade == "2000s") %>% arrang
e(desc(AveRating)) %>% head(n = 1)

top_per_decade <- bind_rows(top_10s, top_20s, top_30s, top_40s, top_50s,
                            top_60s, top_70s, top_80s, top_90s, top_00s)
rm(top_10s, top_20s, top_30s, top_40s, top_50s,
   top_60s, top_70s, top_80s, top_90s, top_00s)

top_per_decade %>% ggplot(aes(AveRating, fct_reorder(title, moviedecade, .desc = TRUE), fi
ll = moviedecade)) + geom_col() +
  geom_text(aes(label = round(AveRating,2)), hjust = -0.2, size = 3) +
  labs(title = "Top Rated Movies of Each Decade", x="", y = "") +
  coord_cartesian(xlim = c(0.1,4.8)) +
  scale_y_discrete(labels = function(y) str_wrap(y, width = 40)) +
  theme(legend.position = "none")
```


Top Rated Movies of Each Decade
Intolerance (1916)
General, The (1927)
M (1931)
Casablanca (1942)
Rear Window (1954)
Dr. Strangelove or: How I Learned to Stop Worrying and Love the Bomb (1964)
Godfather, The (1972)
Shoah (1985)
Shawshank Redemption, The (1994)
Dark Knight, The (2008)
3.85
4.27
4.27
4.32
4.32
4.3
4.42
4.26
4.46
4.3
0
1
2
3
4
5

```
## Top Rated Movies per Genre

top_action <- stats_movie %>% filter(NoRaters >= 10 & action == TRUE) %>% arrange(desc(Ave
Rating)) %>% head(n = 1) %>% mutate(genretop = "Action")
top_adventure <- stats_movie %>% filter(NoRaters >= 10 & adventure == TRUE) %>% arrange(de
sc(AveRating)) %>% head(n = 1) %>% mutate(genretop = "Adventure")
top_animation <- stats_movie %>% filter(NoRaters >= 10 & animation == TRUE) %>% arrange(de
sc(AveRating)) %>% head(n = 1) %>% mutate(genretop = "Animation")
top_children <- stats_movie %>% filter(NoRaters >= 10 & children == TRUE) %>% arrange(desc
(AveRating)) %>% head(n = 1) %>% mutate(genretop = "Children")
top_comedy <- stats_movie %>% filter(NoRaters >= 10 & comedy == TRUE) %>% arrange(desc(Ave
Rating)) %>% head(n = 1) %>% mutate(genretop = "Comedy")
top_crime <- stats_movie %>% filter(NoRaters >= 10 & crime == TRUE) %>% arrange(desc(AveRa
ting)) %>% head(n = 1) %>% mutate(genretop = "Crime")
top_documentary <- stats_movie %>% filter(NoRaters >= 10 & documentary == TRUE) %>% arrang
e(desc(AveRating)) %>% head(n = 1) %>% mutate(genretop = "Documentary")
top_drama <- stats_movie %>% filter(NoRaters >= 10 & drama == TRUE) %>% arrange(desc(AveRa
ting)) %>% head(n = 1) %>% mutate(genretop = "Drama")
top_fantasy <- stats_movie %>% filter(NoRaters >= 10 & fantasy == TRUE) %>% arrange(desc(A
veRating)) %>% head(n = 1) %>% mutate(genretop = "Fantasy")
top_filmnoir <- stats_movie %>% filter(NoRaters >= 10 & filmnoir == TRUE) %>% arrange(desc
(AveRating)) %>% head(n = 1) %>% mutate(genretop = "Film Noir")
top_horror <- stats_movie %>% filter(NoRaters >= 10 & horror == TRUE) %>% arrange(desc(Ave
Rating)) %>% head(n = 1) %>% mutate(genretop = "Horror")
top_imax <- stats_movie %>% filter(NoRaters >= 10 & imax == TRUE) %>% arrange(desc(AveRati
ng)) %>% head(n = 1) %>% mutate(genretop = "IMAX")
top_musical <- stats_movie %>% filter(NoRaters >= 10 & musical == TRUE) %>% arrange(desc(A
veRating)) %>% head(n = 1) %>% mutate(genretop = "Musical")
top_mystery <- stats_movie %>% filter(NoRaters >= 10 & mystery == TRUE) %>% arrange(desc(A
veRating)) %>% head(n = 1) %>% mutate(genretop = "Mystery")
top_romance <- stats_movie %>% filter(NoRaters >= 10 & romance == TRUE) %>% arrange(desc(A
veRating)) %>% head(n = 1) %>% mutate(genretop = "Romance")
top_scifi <- stats_movie %>% filter(NoRaters >= 10 & scifi == TRUE) %>% arrange(desc(AveRa
ting)) %>% head(n = 1) %>% mutate(genretop = "Sci-Fi")
top_thriller <- stats_movie %>% filter(NoRaters >= 10 & thriller == TRUE) %>% arrange(desc
(AveRating)) %>% head(n = 1) %>% mutate(genretop = "Thriller")
top_war <- stats_movie %>% filter(NoRaters >= 10 & war == TRUE) %>% arrange(desc(AveRatin
g)) %>% head(n = 1) %>% mutate(genretop = "War")
top_western <- stats_movie %>% filter(NoRaters >= 10 & western == TRUE) %>% arrange(desc(A
veRating)) %>% head(n = 1) %>% mutate(genretop = "Western")

top_per_genre <- bind_rows(top_action, top_adventure, top_animation, top_children, top_com
edy,
                           top_crime, top_documentary, top_drama, top_fantasy, top_filmnoi
r,
                           top_horror, top_imax, top_musical, top_mystery, top_romance,
                           top_scifi, top_thriller, top_war, top_western) %>% mutate(genre
title = str_c(genretop, ": ", title))
rm(top_action, top_adventure, top_animation, top_children, top_comedy,
   top_crime, top_documentary, top_drama, top_fantasy, top_filmnoir,
   top_horror, top_imax, top_musical, top_mystery, top_romance,
   top_scifi, top_thriller, top_war, top_western)

top_per_genre %>% ggplot(aes(AveRating, fct_reorder(genretitle, AveRating), fill = genreto
p)) + geom_col() +
  geom_text(aes(label = round(AveRating,2)), hjust = -0.2, size = 3) +
  labs(title = "Top Rated Movies per Genre", x ="", y = "") +
  coord_cartesian(xlim = c(0.1,5)) +
```

```
  scale_y_discrete(labels = function(y) str_wrap(y, width = 40)) +
  theme(legend.position = "none")
```



3. Machine Learning Analysis

In this section of the paper, we attempt to construct predictive models for the movie ratings. In the algorithm development, however, we first further partition the *edx* data set into training and testing sets, reserving the *final holdout test* set for the final RMSE generation in the conclusion.

After doing so, we use the mean rating of 3.512 as our baseline model. We then proceed to replicate modeling based on movie and user effects that was shown in the HarvardX Machine Learning course. Additionally, we will train and test linear models using the additional descriptors produced earlier, i.e., year and genre. Will the use of these models improve on the baseline RMSE of 1.0604?

```
test_index <- createDataPartition(edx$rating, p = 0.1, list = FALSE)

edx_train <- edx[test_index,]
edx_test <- edx[-test_index,]

RMSEs <- tibble()

meanRating <- mean(edx_train$rating)
baselineRMSE <- RMSE(meanRating,edx_test$rating)
RMSEs <- tibble(Model = "Baseline Using Mean Rating", RMSE = baselineRMSE)

kable(RMSEs, digits = 4)
```

Model	RMSE
Baseline Using Mean Rating	1.0602

3.1 Movie and User Effects

In this section, we replicate the movie and user effects modeling approach implemented in the HarvardX Machine Learning course on a smaller version of the MovieLens data set. Movie and User effects were first estimated separately and then together, leading to some improvements in the RMSE. The model that accounts for both Movie and User effects yielded the best RMSE at around 0.8859. Regularization was subsequently implemented to "penalize" small numbers of observations per user and per movie. Unfortunately, regularization provided a very minuscule improvement to the model.

```
# Movie Effects

movieMeanRatings <- edx_train %>% group_by(movieId) %>%
  summarize(AveRating = mean(rating)) %>%
  mutate(movieEffect = AveRating - meanRating)

edx_test <- edx_test %>%
  left_join(movieMeanRatings) %>%
  mutate(movieEffect = ifelse(is.na(movieEffect), 0, movieEffect)) %>%
  mutate(movieEffectPred = movieEffect + meanRating)
```

```
## Joining with `by = join_by(movieId)`
```

```
movieEffectsModelRMSE <- RMSE(edx_test$movieEffectPred, edx_test$rating)
RMSEs <- bind_rows(RMSEs,tibble(Model = "Movie Effects Model", RMSE = movieEffectsModelRMS
E))

# User Effects

userMeanRatings <- edx_train %>%
  left_join(movieMeanRatings) %>%
  group_by(userId) %>%
  summarize(userEffect = mean(rating - meanRating - movieEffect))
```

```
## Joining with `by = join_by(movieId)`
```

```
edx_test <- edx_test %>%
  left_join(userMeanRatings) %>%
  mutate(userEffect = ifelse(is.na(userEffect),0,userEffect)) %>%
  mutate(userEffectPred = userEffect + meanRating)
```

```
## Joining with `by = join_by(userId)`
```

```
userEffectsModelRMSE <- RMSE(edx_test$userEffectPred, edx_test$rating)
RMSEs <- bind_rows(RMSEs, tibble(Model = "User Effects Model", RMSE = userEffectsModelRMS
E))

# Movie & User Effects

edx_test <- edx_test %>%
  mutate(movieuserEffectPred = movieEffect + userEffect + meanRating)

movieuserEffectsModelRMSE <- RMSE(edx_test$movieuserEffectPred, edx_test$rating)
RMSEs <- bind_rows(RMSEs, tibble(Model = "Movie & User Effects Model", RMSE = movieuserEff
ectsModelRMSE))

## Regularization
lambdas <- seq(0, 10, 0.25)
regularizedRMSEs <- sapply(lambdas, function(lambda){
    Movies <- edx_train %>%
          group_by(movieId) %>%
          summarize(bMovie = sum(rating - meanRating)/(n()+lambda))
     Users <- edx_train %>%
          left_join(Movies, by="movieId") %>%
          group_by(userId) %>%
          summarize(bUser = sum(rating - bMovie - meanRating)/(n()+lambda))
     predicted_ratings <- edx_test %>%
          left_join(Movies, by = "movieId") %>%
          left_join(Users, by = "userId") %>%
         mutate(bMovie = ifelse(is.na(bMovie), 0, bMovie),
                 bUser = ifelse(is.na(bUser),0,bUser)) %>%
          mutate(pred = meanRating + bMovie + bUser)
     return(RMSE(predicted_ratings$pred, edx_test$rating))
})

min(regularizedRMSEs)
```

```
## [1] 0.8858617
```

```
best_lambda = lambdas[which.min(regularizedRMSEs)]
best_lambda
```

```
## [1] 4.75
```

```
RMSEs <- bind_rows(RMSEs, tibble(Model="Regularized Movie + User Effect Model",
                                        RMSE = min(regularizedRMSEs)))

kable(RMSEs, digits = 4)
```

Model	RMSE
Baseline Using Mean Rating	1.0602
Movie Effects Model	0.9491
User Effects Model	1.0264
Movie & User Effects Model	0.9076

Model	RMSE
Regularized Movie + User Effect Model	0.8859

3.2 Linear Models on Movie Period and Genre

I tried to implement additional linear models that use decade, year, and genre as factors. On their own, the linear models performed only slightly better than the baseline model and worse than the movie and user effects. The linear models yielded an RMSE less than 1 only when the movie and user fixed effects were added to the final linear regression model.

```
# Regress Against Decade Dummies

edx_train <- edx_train %>% mutate(dummy = TRUE, moviedecade2 = str_c("d",moviedecade)) %>%
  spread(key=moviedecade2, value=dummy, fill=FALSE)
edx_test <- edx_test %>% mutate(dummy = TRUE, moviedecade2 = str_c("d",moviedecade)) %>%
  spread(key=moviedecade2, value=dummy, fill=FALSE)

lm_fit_decades <- lm(rating ~ d1920s + d1930s + d1940s + d1950s +
                       d1960s + d1970s + d1980s + d1990s + d2000s, data = edx_train)
#alternative specification: lm(rating ~ as.factor(moviedecade), data = edx)

p_hat <- predict(lm_fit_decades, edx_test)
edx_test <- edx_test %>% bind_cols(as_tibble(p_hat)) %>% rename(lmDecadesPred = value)
rm(p_hat)

lmDecadesModelRMSE <- RMSE(edx_test$lmDecadesPred, edx_test$rating)
RMSEs <- bind_rows(RMSEs, tibble(Model = "Linear Model on Decade Dummies", RMSE = lmDecade
sModelRMSE))

# Regress Against Year

#First, let me use "year" as a numeric value

edx_train <- edx_train %>% mutate(movieyear2 = as.numeric(movieyear))
edx_test <- edx_test %>% mutate(movieyear2 = as.numeric(movieyear))

lm_fit_years <- lm(rating ~ movieyear2, data = edx_train)

p_hat <- predict(lm_fit_years, edx_test)
edx_test <- edx_test %>% bind_cols(as_tibble(p_hat)) %>% rename(lmYearsPred = value)
rm(p_hat)

lmYearsModelRMSE <- RMSE(edx_test$lmYearsPred, edx_test$rating)
RMSEs <- bind_rows(RMSEs, tibble(Model = "Linear Model on Year as Numeric", RMSE = lmYears
ModelRMSE))

#Can I attempt at a linear model with years transformed into dummies?

lm_fit_years_dummies <- lm(rating ~ as.factor(movieyear), data=edx_train)

p_hat <- predict(lm_fit_years_dummies, edx_test)
edx_test <- edx_test %>% bind_cols(as_tibble(p_hat)) %>% rename(lmYearsDumPred = value)
rm(p_hat)

lmYearsModelDumRMSE <- RMSE(edx_test$lmYearsDumPred, edx_test$rating)
RMSEs <- bind_rows(RMSEs, tibble(Model = "Linear Model on Year as Dummies", RMSE = lmYears
ModelDumRMSE))

# Regress Against Genre Dummies

lm_fit_genres <- lm(rating ~ action + adventure + animation + children + comedy +
                      crime + documentary + drama + fantasy + filmnoir + horror +
                      imax + musical + mystery + romance + scifi + thriller + war +
                      western, data = edx_train)

p_hat <- predict(lm_fit_genres, edx_test)
edx_test <- edx_test %>% bind_cols(as_tibble(p_hat)) %>% rename(lmGenresPred = value)
rm(p_hat)
```

```
lmGenresModelRMSE <- RMSE(edx_test$lmGenresPred, edx_test$rating)
RMSEs <- bind_rows(RMSEs, tibble(Model = "Linear Model on Genre Dummies", RMSE = lmGenresM
odelRMSE))

# Regress Against Decade & Genre Dummies

lm_fit_genresdecades <- lm(rating ~ action + adventure + animation + children + comedy +
                       crime + documentary + drama + fantasy + filmnoir + horror +
                       imax + musical + mystery + romance + scifi + thriller + war +
                       western + d1920s + d1930s + d1940s + d1950s +
                       d1960s + d1970s + d1980s + d1990s + d2000s, data = edx_train)

p_hat <- predict(lm_fit_genresdecades, edx_test)
edx_test <- edx_test %>% bind_cols(as_tibble(p_hat)) %>% rename(lmGenreDecadesPred = valu
e)
rm(p_hat)

lmGenreDecadesModelRMSE <- RMSE(edx_test$lmGenreDecadesPred, edx_test$rating)
RMSEs <- bind_rows(RMSEs, tibble(Model = "Linear Model on Decade & Genre Dummies", RMSE =
lmGenreDecadesModelRMSE))

# Regressions with Movie & User Fixed Effects

edx_test <- edx_test %>%
  mutate(lmGenreDecadesFEPred = lmGenreDecadesPred + movieEffect + userEffect)

lmGenreDecadesFEModelRMSE <- RMSE(edx_test$lmGenreDecadesFEPred, edx_test$rating)
RMSEs <- bind_rows(RMSEs, tibble(Model = "Final Linear Model with Movie & User Fixed Effec
ts"
                                 , RMSE = lmGenreDecadesFEModelRMSE))

kable(RMSEs, digits = 4)
```

Model	RMSE
Baseline Using Mean Rating	1.0602
Movie Effects Model	0.9491
User Effects Model	1.0264
Movie & User Effects Model	0.9076
Regularized Movie + User Effect Model	0.8859
Linear Model on Decade Dummies	1.0515
Linear Model on Year as Numeric	1.0525
Linear Model on Year as Dummies	1.0493
Linear Model on Genre Dummies	1.0412
Linear Model on Decade & Genre Dummies	1.0327
Final Linear Model with Movie & User Fixed Effects	0.9363

4. Conclusion

Beyond replicating the recommendation system models presented in the HarvardX Machine Learning course, I attempted to produce additional dimensions to the data set by extracting available information into separate variables: the year and decade of movies' release; and several dummies representing movie genres. Including these in the machine learning exercise through linear regression, however, did not produce a model that is stronger than the regularized movie & user effects model.

I had wanted to use other classification algorithms such as k-Nearest Neighbor, Generative Models, Classification and Regression Trees, and Random Forest. However, the size of the data set is not manageable on my machine, and I've encountered frequent crashes along the way.

In the end, the strongest machine learning model produced through this capstone project was the one using regularized movie and user effects, which produced an RMSE of 0.8861.

```
## Reimplementing the Final Model and Testing Against Final Holdout Set

     Movies <- edx_train %>%
          group_by(movieId) %>%
          summarize(bMovie = sum(rating - meanRating)/(n()+best_lambda))
     Users <- edx_train %>%
          left_join(Movies, by="movieId") %>%
          group_by(userId) %>%
          summarize(bUser = sum(rating - bMovie - meanRating)/(n()+best_lambda))
     predicted_ratings <-
          final_holdout_test %>%
          left_join(Movies, by = "movieId") %>%
          left_join(Users, by = "userId") %>%
          mutate(bMovie = ifelse(is.na(bMovie), 0, bMovie),
                 bUser = ifelse(is.na(bUser),0,bUser)) %>%
          mutate(pred = meanRating + bMovie + bUser)

final_RMSE <- RMSE(predicted_ratings$pred, final_holdout_test$rating)

kable(tibble(Model = "Regularized Movie & User Effects" ,RMSE = final_RMSE), digits = 4)
```

Model	RMSE
Regularized Movie & User Effects	0.8861