HarvardX - PH125.9x Data Science: Capstone

MovieLens

Julie COLLEONI

2025-05-20

Contents

1 1. Introduction

This report documents a predictive modeling project conducted as part of the HarvardX Professional Certificate in Data Science. The analysis leverages the MovieLens 10M dataset, which was sourced from edX but originally curated by GroupLens, a research lab at the University of Minnesota.

The primary objective is to develop a machine learning model capable of predicting user ratings for movies. The methodological approach includes partitioning the data into training and testing sets, with model predictions ultimately evaluated on a dedicated validation set. A key requirement of the project is that the final holdout test set must not be utilized during model training, development, or selection, but reserved exclusively for final evaluation.

The model's performance will be assessed using the Root Mean Square Error (RMSE), a metric that quantifies the average deviation between the predicted and actual ratings. The target RMSE for this project is set at less than 0.86490; a lower RMSE indicates a more precise and accurate predictive model.

In the following sections, the report details the data exploration, modeling approaches, results, and interpretation of findings.

1.1 1.1 Dataset Overview

In order to work on the data it is necessary to:

- First load the data set, make intial data wrangling with the code provided by Harvard Edx:

```
knitr::opts_chunk$set(echo=FALSE)

############################################################
# Create edx and final_holdout_test sets
############################################################

# Note: this process could take a couple of minutes
```

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

```
## Le chargement a nécessité le package : tidyverse
```

```
## -- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
## v dplyr     1.1.4     v readr     2.1.5
## v forcats   1.0.0     v stringr   1.5.1
## v ggplot2   3.5.1     v tibble    3.2.1
## v lubridate 1.9.4     v tidyr     1.3.1
## v purrr     1.0.4
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
## i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become errors
```

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

```
## Le chargement a nécessité le package : caret
## Le chargement a nécessité le package : lattice
##
## Attachement du package : 'caret'
##
## L'objet suivant est masqué depuis 'package:purrr':
##
##     lift
```

```r
if (!requireNamespace("plotly", quietly = TRUE)) install.packages("plotly")
if (!requireNamespace("webshot", quietly = TRUE)) install.packages("webshot")
if (!requireNamespace("webshot2", quietly = TRUE)) install.packages("webshot2")
```

```
## Registered S3 method overwritten by 'webshot2':
##   method        from
##   print.webshot webshot
```

```r
if (!requireNamespace("htmlwidgets", quietly = TRUE)) install.packages("htmlwidgets")
webshot::install_phantomjs()
```

```
## It seems that the version of `phantomjs` installed is greater than or equal to the requested version.
```

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

```
## Le chargement a nécessité le package : scales
##
## Attachement du package : 'scales'
##
## L'objet suivant est masqué depuis 'package:purrr':
##
##     discard
##
## L'objet suivant est masqué depuis 'package:readr':
##
##     col_factor
```

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

```
## Le chargement a nécessité le package : patchwork
```

```r
if(!require(kableExtra)) install.packages("kableExtra")
```

```
## Le chargement a nécessité le package : kableExtra
##
## Attachement du package : 'kableExtra'
##
## L'objet suivant est masqué depuis 'package:dplyr':
##
##     group_rows
```

```r
library(tidyverse)
library(caret)
library(webshot)
library(webshot2)
```

```
##
## Attachement du package : 'webshot2'
##
## Les objets suivants sont masqués depuis 'package:webshot':
##
##     appshot, resize, rmdshot, shrink, webshot
```

```r
library(htmlwidgets)
library(plotly)
```

```
##
## Attachement du package : 'plotly'
##
## L'objet suivant est masqué depuis 'package:ggplot2':
##
##     last_plot
##
## L'objet suivant est masqué depuis 'package:stats':
##
##     filter
##
## L'objet suivant est masqué depuis 'package:graphics':
##
##     layout
```

```r
library(scales)
library(patchwork)
library(kableExtra)

# MovieLens 10M dataset:
# https://grouplens.org/datasets/movielens/10m/
```

```r
# 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 = TRUE),
                         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))

movielens <- left_join(ratings, movies, by = "movieId")
```

- Then create two subsets thanks to the code also provided:
  - One will be "edx" that will be used for training, developing and the selecting the most appropriate algorith.
  - The second subset will be "final_holdout_test" that will be used for evaluating the RMSE of the selected algorithm.

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

```
## Joining with `by = join_by(userId, movieId, rating, timestamp, title, genres)`
```

- Then have a initial exploration of the data to define the size of the datasel and its main components.

```
## Number of lines in edx: 9,000,055
```

```
## Number of lines in final_holdout_test: 999,999
```

```
## Number of distinct users: 69,878
```

```
## Number of distinct movies: 10,677

## Number of distinct ratings (values): 10

## The different possible ratings (values): 0.5 1 1.5 2 2.5 3 3.5 4 4.5 5

## Number of distinct timestamps: 6,519,590

## Number of distinct movie titles: 10,676

## Number of distinct genre combinations: 797
```

The dataset used for this project, as provided by EdX, consists of *9,000,055* ratings submitted by *69,878* unique users on *10,677* distinct movies. Ratings are associated with over *6.5 millions* unique timestamps, illustrating that most ratings were recorded at unique points in time. This high level of temporal granularity offers the opportunity to analyze user behavior and trends over time; however, it also suggests that the timestamp data may need to be aggregated (for instance by year or month) to draw meaningful conclusions at the individual or cohort level.

1.2 1.2 Project Summary

The analysis follows a structured pipeline:

- **Data preparation**: Partitioning the data into a training set (`edx`) and a final hold-out validation set, ensuring that all users and movies in the validation set appear in the training set.
- **Data exploration**: Comprehensive exploration was performed to understand the distribution of ratings, to uncover trends over time, and to identify possible sources of bias related to both user and movie characteristics.
- **Model development**: A series of models of increasing complexity were developed. The process began with a simple global average model, then introduced movie-specific and user-specific effects, and finally incorporated regularization techniques to mitigate overfitting, particularly for movies or users with limited data.
- **Performance evaluation**: At each stage of model development, performance was evaluated using the Root Mean Square Error (RMSE) metric to track improvements. The final selected model was then applied to the hold-out validation set for ultimate scoring.

This report is structured into five main sections: an introduction, a methods and analysis section, a modelling section, a results section, and a conclusion with limitations.

2 2.Methods and Data analysis

2.1 2.1 Data Preparation

To begin the analysis, the MovieLens 10M dataset was downloaded and preprocessed according to the project guidelines. The raw dataset includes two main files: `ratings.dat`, which contains the ratings provided by users, and `movies.dat`, which contains metadata about the movies (titles and genres). These files were merged into a single dataset using the `movieId` as a key.

Each entry in the merged dataset includes the following variables: - `userId`: a unique identifier for the user, - `movieId`: a unique identifier for the movie, - `rating`: the rating given by the user (from 0.5 to 5.0), - `timestamp`: the date and time the rating was recorded, - `title`: the name of the movie, - `genres`: a list of genres associated with the movie.

2.1.1 2.1.1 Data Cleaning

Before exploring the data or building any models, it is important to make sure that the dataset is clean and usable. In order to build solid models, it is important to rely in clean data.

In this section, I do the following:

- Check for problems such as missing values or duplicate rows.
- Create new variables to better describe the data, such as the year of the movie or the main genre.
- Split the data into two parts: one for building the model (called `edx`) and one for testing it (called `validation`).

These steps are essential to make sure that the results will be accurate and meaningful.

2.1.1.1 i. Checking for missing values The initial step involves verifying the presence of missing values in any column of the dataset. No missing values were detected. A result of zero indicates that the dataset is complete in this regard, which is a positive outcome for the reliability of subsequent analyses.

```
##    userId   movieId    rating timestamp     title    genres
##         0         0         0         0         0         0
```

2.1.1.2 ii. Checking for duplicate rows I also verify whether the dataset contains any exact duplicate rows, which could occur if some ratings were inadvertently recorded multiple times.

```
knitr::opts_chunk$set(echo = TRUE)

#Checking for duplicate rows


sum(duplicated(edx))
```

```
## [1] 0
```

If the result is zero, it confirms that there are no duplicate entries, which is a positive sign for data quality.

2.1.1.3 iii. Checking column types It is important to ensure that each variable is stored in the appropriate data type. For example, the user ID and movie ID should be integers, ratings should be numeric, and the movie title and genres should be text (character strings).

```
knitr::opts_chunk$set(echo = TRUE)

#Checking column types


str(edx)
```

```
## 'data.frame':    9000055 obs. of  6 variables:
##  $ userId   : int  1 1 1 1 1 1 1 1 1 1 ...
##  $ movieId  : int  122 185 292 316 329 355 356 362 364 370 ...
##  $ rating   : num  5 5 5 5 5 5 5 5 5 5 ...
##  $ timestamp: int  838985046 838983525 838983421 838983392 838983392 838984474 838983653 838984885 83
##  $ title    : chr  "Boomerang (1992)" "Net, The (1995)" "Outbreak (1995)" "Stargate (1994)" ...
##  $ genres   : chr  "Comedy|Romance" "Action|Crime|Thriller" "Action|Drama|Sci-Fi|Thriller" "Action|Ac
```

2.1.1.4 iiii. Checking rating values

It is important to verify that all ratings fall within the valid range, which for this dataset is between 0.5 and 5.0. A review of unique values is conducted to verify that no out-of-range or erroneous values are present.

```
knitr::opts_chunk$set(echo = TRUE)

#Checking rating values


sort(unique(edx$rating))
```

```
##  [1] 0.5 1.0 1.5 2.0 2.5 3.0 3.5 4.0 4.5 5.0
```

2.1.1.5 iiiii. Summary of data cleaning

These checks show that the dataset is clean:

- No missing values detected.
- No duplicate rows present.
- All variables have the appropriate data types.
- Ratings are within the expected range (0.5 to 5.0).

With the dataset confirmed as clean and consistent, the next step consists of generating new variables to facilitate further analysis and modeling.

2.1.2 2.1.2 Creating new variables

The first step of the analysis involved preparing the MovieLens 10M dataset for modeling.

Each observation includes:

- `userId`: a unique identifier for the user,
- `movieId`: a unique identifier for the movie,
- `rating`: the rating value (0.5 to 5.0),
- `timestamp`: the time the rating was made (in Unix time),
- `title`: the movie name along with its release year,
- `genres`: a pipe-separated list of genres (for instance, *Action*/*Adventure*).

Once the dataset has been cleaned, additional variables can be created to enhance its usefulness. These new variables facilitate a more detailed description of the data and allow for the identification of relevant patterns. They are also expected to improve the performance of subsequent predictive modeling.

In this step, four new variables are created:

- `release_year`: the year the movie was released
- `rating_year`: the year when the rating was given
- `main_genre`: the first or main genre of the movie
- `user_frequency`: how often each user rates movies

2.1.2.1 i. The new variable: release_year

The release year refers to the year each movie was first made available, typically indicated in the title (for instance, "Titanic (1997)"). Identifying the release year allows for analysis of whether user ratings differ between older and more recent films.

```
knitr::opts_chunk$set(echo = TRUE)

library(dplyr)

edx <- edx %>%
  mutate(release_year = str_extract(title, "\\(\\d{4}\\)") %>%
           str_remove_all("[()]") %>%
           as.integer())

# Show first rows to verify
head(select(edx, title, release_year))
```

```
##                           title release_year
## 1              Boomerang (1992)         1992
## 2               Net, The (1995)         1995
## 4               Outbreak (1995)         1995
## 5               Stargate (1994)         1994
## 6 Star Trek: Generations (1994)         1994
## 7       Flintstones, The (1994)         1994
```

2.1.2.2 ii. The new variable: rating_year

This variable represents the year in which the user submitted a rating, derived from the timestamp recorded in seconds. Analyzing this information enables the study of how movie ratings have evolved over time.

```
knitr::opts_chunk$set(echo = TRUE)

library(dplyr)

# Create the new variable: rating_year
edx <- edx %>%
  mutate(rating_year = as.POSIXct(timestamp, origin = "1970-01-01") %>%
           format("%Y") %>%
           as.integer())

# Show a few rows to verify
head(select(edx, timestamp, rating_year))
```

```
##   timestamp rating_year
## 1 838985046        1996
## 2 838983525        1996
## 4 838983421        1996
## 5 838983392        1996
## 6 838983392        1996
## 7 838984474        1996
```

2.1.2.3 iii. The new variable: main_genre

Many movies are associated with multiple genres (for example: "Action|Adventure|Sci-Fi"). For analytical purposes, only the first listed genre is retained as the main genre. This approach simplifies grouping and facilitates the comparison of average ratings across different genres.

```
knitr::opts_chunk$set(echo = TRUE)

edx <- edx %>%
  mutate(main_genre = str_split(genres, "\\|", simplify = TRUE)[,1])

# Show a few rows to verify
head(select(edx, genres, main_genre))
```

```
##                          genres main_genre
## 1                Comedy|Romance     Comedy
## 2          Action|Crime|Thriller     Action
## 4  Action|Drama|Sci-Fi|Thriller     Action
## 5        Action|Adventure|Sci-Fi     Action
## 6 Action|Adventure|Drama|Sci-Fi     Action
## 7        Children|Comedy|Fantasy   Children
```

2.1.2.4 iiii. The new variable: user_frequency

User activity varies considerably, with some individuals providing many ratings and others only a few. This variable measures the average number of ratings submitted per user each year, which can help identify highly active or less active users.

```
knitr::opts_chunk$set(echo = TRUE)

user_activity <- edx %>%
  group_by(userId) %>%
  summarise(
    n_ratings = n(),
    first_rating = min(rating_year, na.rm = TRUE),
    last_rating = max(rating_year, na.rm = TRUE),
    active_years = last_rating - first_rating + 1,
    user_frequency = n_ratings / active_years,
    .groups = "drop"
  )

# Show a few users
head(user_activity)
```

```
## # A tibble: 6 x 6
##   userId n_ratings first_rating last_rating active_years user_frequency
##    <int>     <int>        <int>       <int>        <dbl>          <dbl>
## 1      1        19         1996        1996            1           19
## 2      2        17         1997        1997            1           17
## 3      3        31         2005        2006            2           15.5
## 4      4        35         1996        1996            1           35
## 5      5        74         1997        1997            1           74
## 6      6        39         2001        2001            1           39
```

With these new variables established, the dataset can now be explored in greater detail in the following section.

2.2 2.2 Data Exploration and Visualization

With the dataset now cleaned and complete, further exploration can be conducted to better understand user rating behaviors. This step helps identify meaningful patterns and informs the selection of appropriate

modeling approaches in subsequent analyses.

2.2.1 2.2.1 What do the ratings look like?

The initial analysis focuses on the distribution of ratings, examining whether users tend to be generous or strict in their assessments and whether the entire rating scale is utilized.

```
knitr::opts_chunk$set(echo = TRUE)

library(scales)
library(knitr)

# Group ratings by type and calculate percentage (rounded to whole number)
rating_summary <- edx %>%
  mutate(rating_type = ifelse(rating %% 1 == 0, "Rounded", "Half-point")) %>%
  count(rating_type, name = "Number of Ratings") %>%
  mutate(
    Percentage = round(`Number of Ratings` / sum(`Number of Ratings`) * 100)
  )

# Display formatted table
kable(rating_summary, caption = "Distribution of Rounded vs Half-point Ratings (in %)")
```

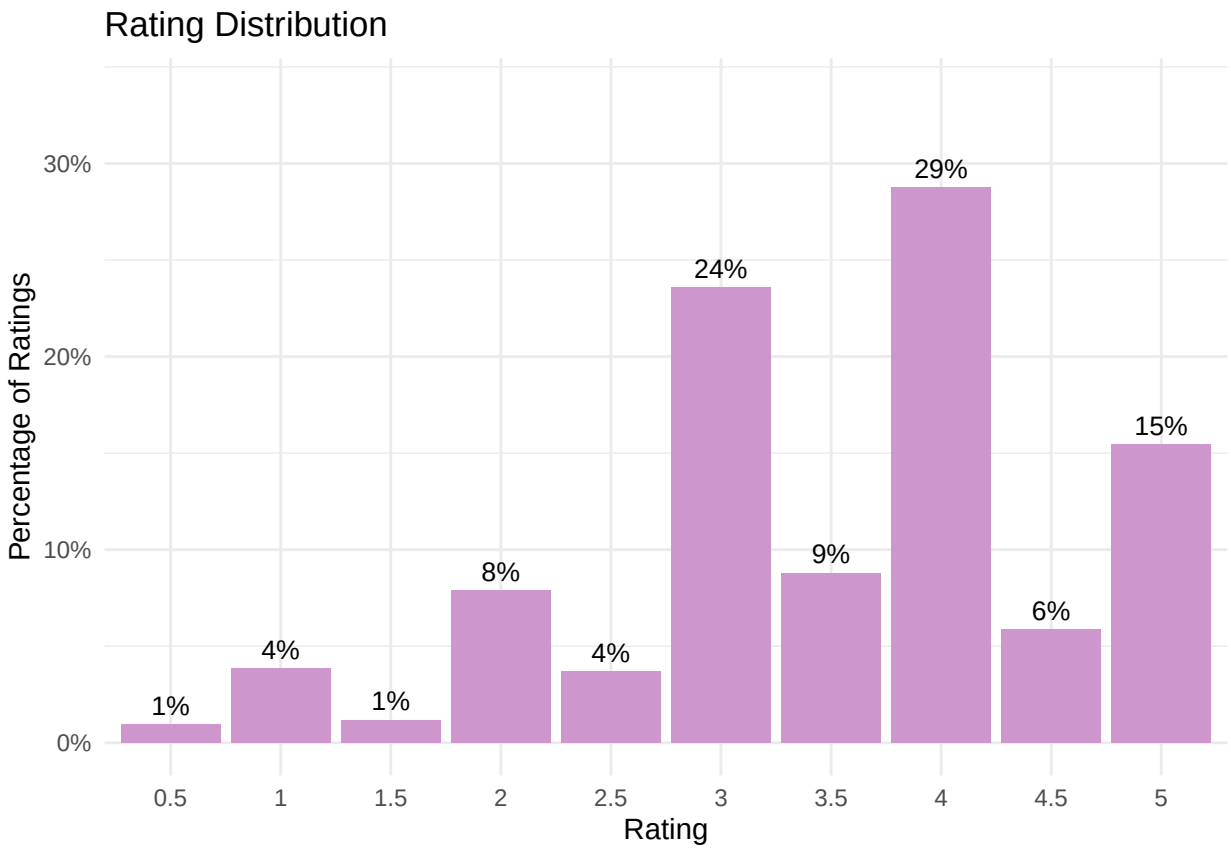
Table 1: Distribution of Rounded vs Half-point Ratings (in %)

rating_type	Number of Ratings	Percentage
Half-point	1843170	20
Rounded	7156885	80

The analysis of the table reveals that 80% of users give rounded ratings rather than half point ratings such as 1.5 and similar values.

```
knitr::opts_chunk$set(echo = TRUE)

library(ggplot2)
library(dplyr)
library(scales)

# Prepare data with percentage
rating_dist <- edx %>%
  count(rating) %>%
  mutate(
    percentage = n / sum(n),
    label = percent(percentage, accuracy = 1)
  )

# Plot
ggplot(rating_dist, aes(x = factor(rating), y = percentage)) +
  geom_col(fill = "plum3") +
  geom_text(aes(label = label), vjust = -0.5, size = 3.5) +
  scale_y_continuous(labels = percent_format(accuracy = 1), limits = c(0, max(rating_dist$percentage) +
```

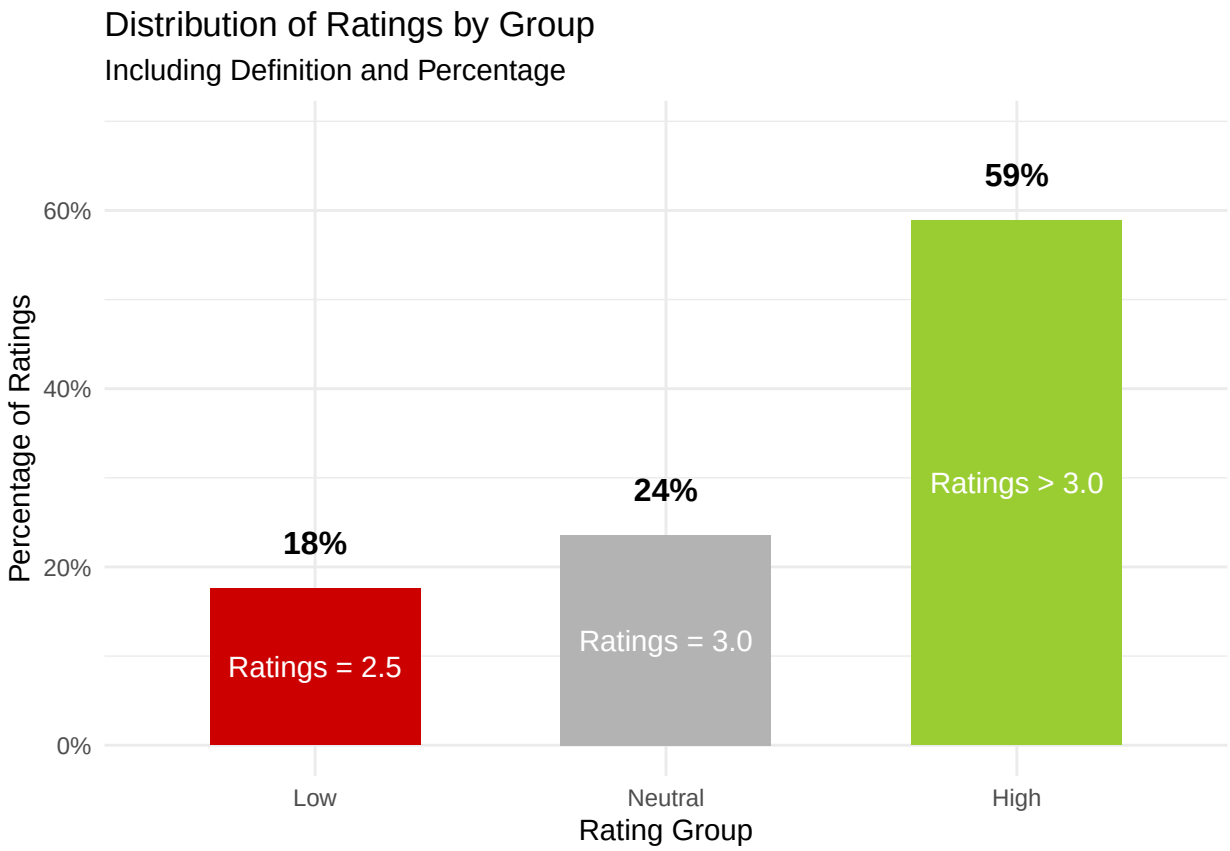
```
  labs(
    title = "Rating Distribution",
    x = "Rating",
    y = "Percentage of Ratings"
  ) +
  theme_minimal()
```



The data indicates that the most frequently given rating is 4. However, approximately 24% of users remain neutral, assigning a rating of 3. Only 15% of users appear to be fully convinced by the movie, awarding it the maximum rating of 5.

```
knitr::opts_chunk$set(echo = TRUE)

library(dplyr)
library(ggplot2)
library(scales)

# Group ratings into categories and control display order
edx_grouped <- edx %>%
  mutate(rating_group = case_when(
    rating <= 2.5 ~ "Low",
    rating == 3.0 ~ "Neutral",
    rating > 3.0  ~ "High"
  )) %>%
  mutate(rating_group = factor(rating_group, levels = c("Low", "Neutral", "High")))
```

```r
# Summarize ratings
rating_group_summary <- edx_grouped %>%
  count(rating_group) %>%
  mutate(
    percentage = n / sum(n),
    percentage_label = percent(percentage, accuracy = 1),
    explanation = case_when(
      rating_group == "Low" ~ "Ratings  2.5",
      rating_group == "Neutral" ~ "Ratings = 3.0",
      rating_group == "High" ~ "Ratings > 3.0"
    )
  )

# Plot with internal explanation and top percentage
ggplot(rating_group_summary, aes(x = rating_group, y = percentage, fill = rating_group)) +
  geom_col(width = 0.6, show.legend = FALSE) +

  # Label inside the bar - centered vertically
  geom_text(aes(y = percentage / 2, label = explanation), color = "white", size = 4) +

  # Percentage above the bar
  geom_text(aes(label = percentage_label), vjust = -1.5, size = 4.2, fontface = "bold") +

  scale_y_continuous(labels = percent_format(accuracy = 1),
                     limits = c(0, max(rating_group_summary$percentage) + 0.1)) +

  scale_fill_manual(values = c("Low" = "red3", "Neutral" = "grey70", "High" = "olivedrab3")) +

  labs(
    title = "Distribution of Ratings by Group",
    subtitle = "Including Definition and Percentage",
    x = "Rating Group",
    y = "Percentage of Ratings"
  ) +

  theme_minimal()
```

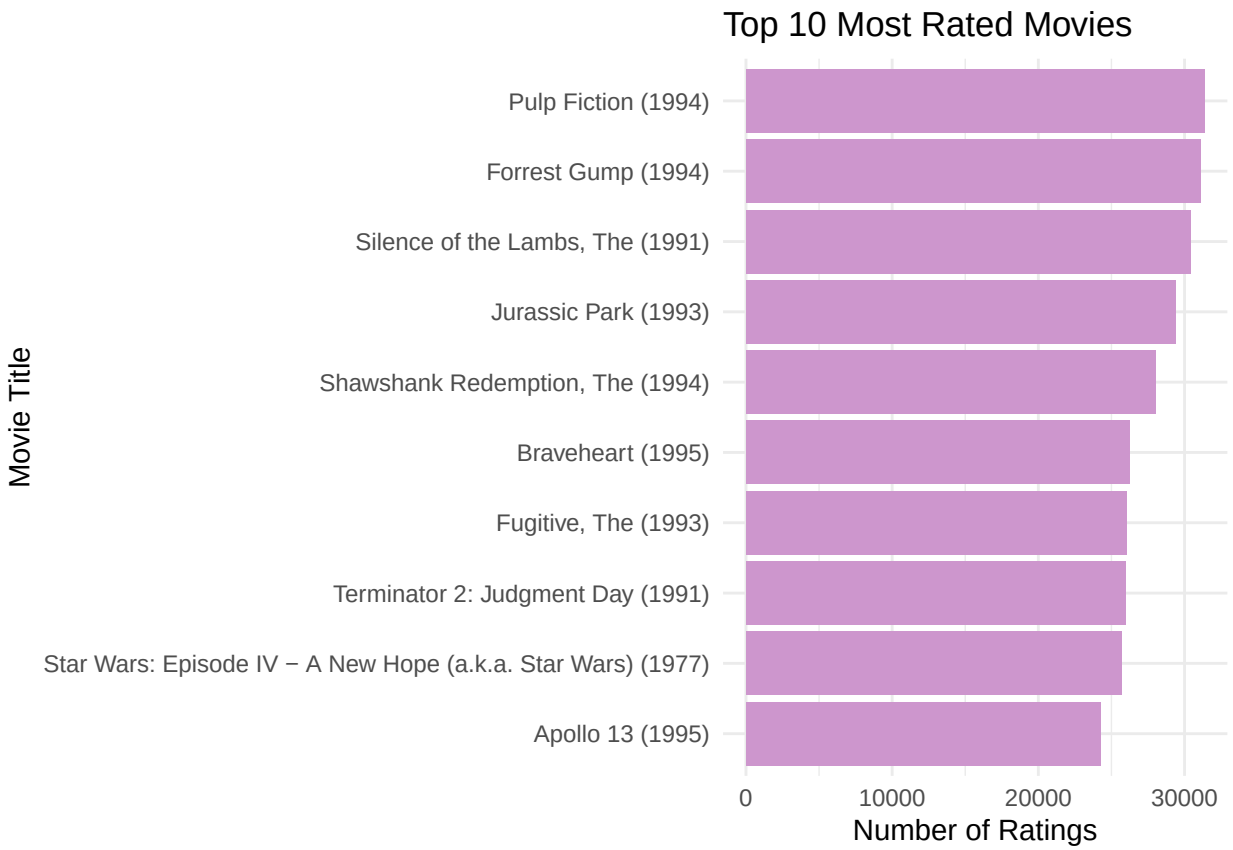


The ratings were divided into three groups: Low (ratings 2.5), Neutral (ratings = 3.0), and High (ratings > 3.0). This chart demonstrates that most users (59%) give high ratings, while low and neutral ratings are much less common. The clear difference between the groups confirms a positive bias in user behavior.

2.2.2 2.2.2 Which movies are rated the most?

An examination of which movies receive the most ratings helps determine if a few popular movies dominate the dataset.

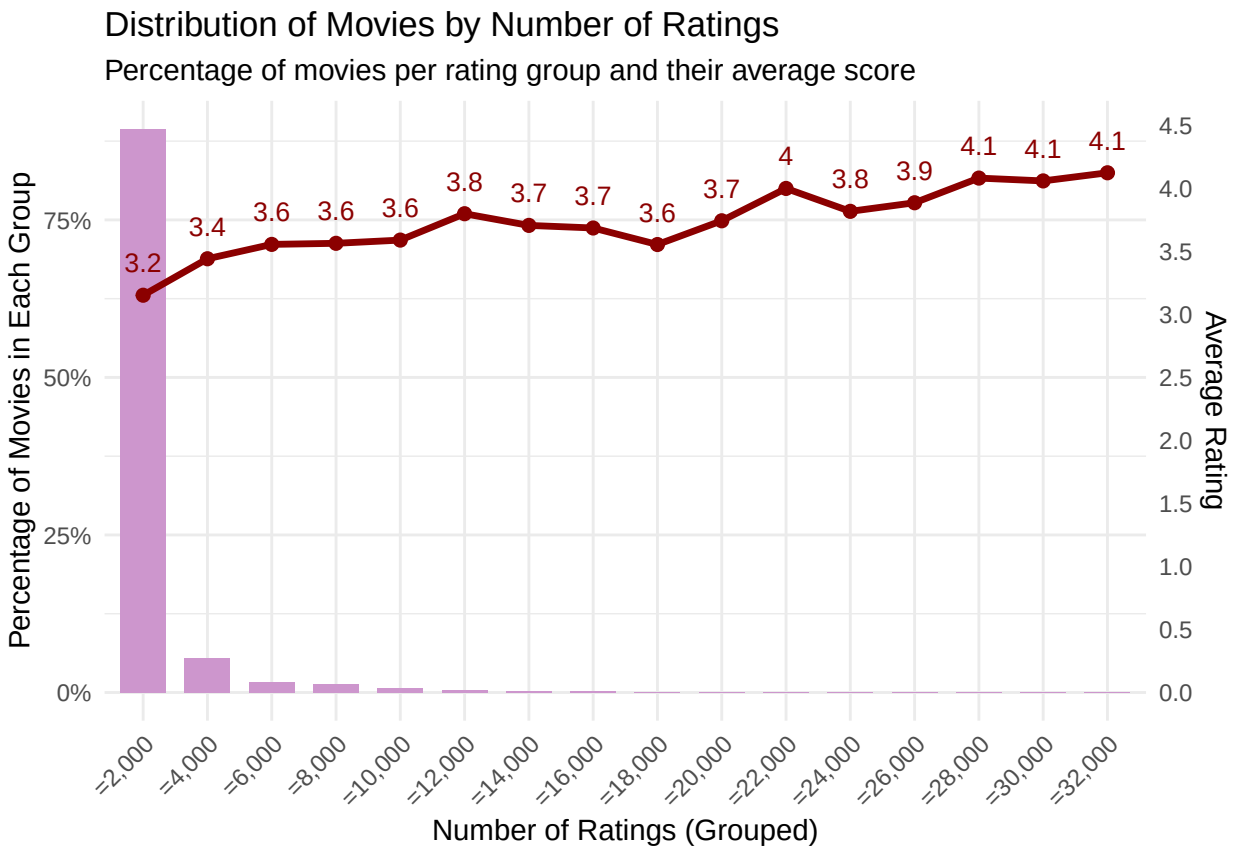
```
knitr::opts_chunk$set(echo = TRUE)

edx %>%
  count(title) %>%
  top_n(10, n) %>%
  ggplot(aes(x = reorder(title, n), y = n)) +
  geom_col(fill = "plum3") +
  coord_flip() +
  labs(title = "Top 10 Most Rated Movies", x = "Movie Title", y = "Number of Ratings") +
  theme_minimal()
```



Some movies, like Pulp Fiction, Forrest Gump or The Silence of the Lambs, have thousands of ratings. These popular movies might influence the model more significantly than others.

```
knitr::opts_chunk$set(echo = TRUE)

library(dplyr)
library(ggplot2)
library(scales)

# # Step 1: Count the number of ratings per film + their average rating
movie_stats <- edx %>%
  group_by(title) %>%
  summarise(
    n_ratings = n(),
    avg_rating = mean(rating),
    .groups = "drop"
  )

# Step 2: Create groups of 2000 ratings
breaks <- seq(0, max(movie_stats$n_ratings) + 2000, by = 2000)
labels <- paste0(" ", comma(breaks[-1]))

movie_stats <- movie_stats %>%
  mutate(
    rating_bin = cut(
      n_ratings,
```

```
      breaks = breaks,
      labels = labels,
      include.lowest = TRUE
    )
  )

# Step 3: Summary by group
rating_bin_summary <- movie_stats %>%
  group_by(rating_bin) %>%
  summarise(
    number_of_movies = n(),
    avg_rating = mean(avg_rating),
    .groups = "drop"
  ) %>%
  mutate(
    proportion = number_of_movies / sum(number_of_movies)
  )

# Step 4: Combo Chart

ggplot(rating_bin_summary, aes(x = rating_bin)) +
  geom_col(aes(y = proportion), fill = "plum3", width = 0.7) +
  geom_line(aes(y = avg_rating / 5), color = "darkred", linewidth = 1.2, group = 1) +
  geom_point(aes(y = avg_rating / 5), color = "darkred", size = 2) +
  scale_y_continuous(
    name = "Percentage of Movies in Each Group",
    labels = percent_format(accuracy = 1),
    sec.axis = sec_axis(~ . * 5, name = "Average Rating", breaks = seq(0, 5, 0.5))
  ) +
   geom_text(
    aes(y = avg_rating / 5, label = round(avg_rating, 1)),
    vjust = -1.2,
    color = "darkred",
    size = 3.5
  ) +
  labs(
    title = "Distribution of Movies by Number of Ratings",
    subtitle = "Percentage of movies per rating group and their average score",
    x = "Number of Ratings (Grouped)",
    y = "Share of Movies"
  ) +
  theme_minimal() +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))
```



This chart demonstrates that the vast majority of movies in the dataset received fewer than 2,000 ratings. Moving to more popular films (over 10,000 or 30,000 ratings), the number of movies in each group becomes very small.

While it might seem that these very popular movies dominate the dataset, they actually represent only a tiny fraction of the total. This suggests that the model should not rely too heavily on these few popular films, as they do not reflect most of the data.

Interestingly, the average rating tends to increase slightly with popularity. However, since those groups contain so few movies, the increase may be attributed to selection effects: more popular movies are often more appreciated or more mainstream.

Therefore, even though blockbusters receive more attention, the model must be built on the broader structure of thousands of less popular titles.

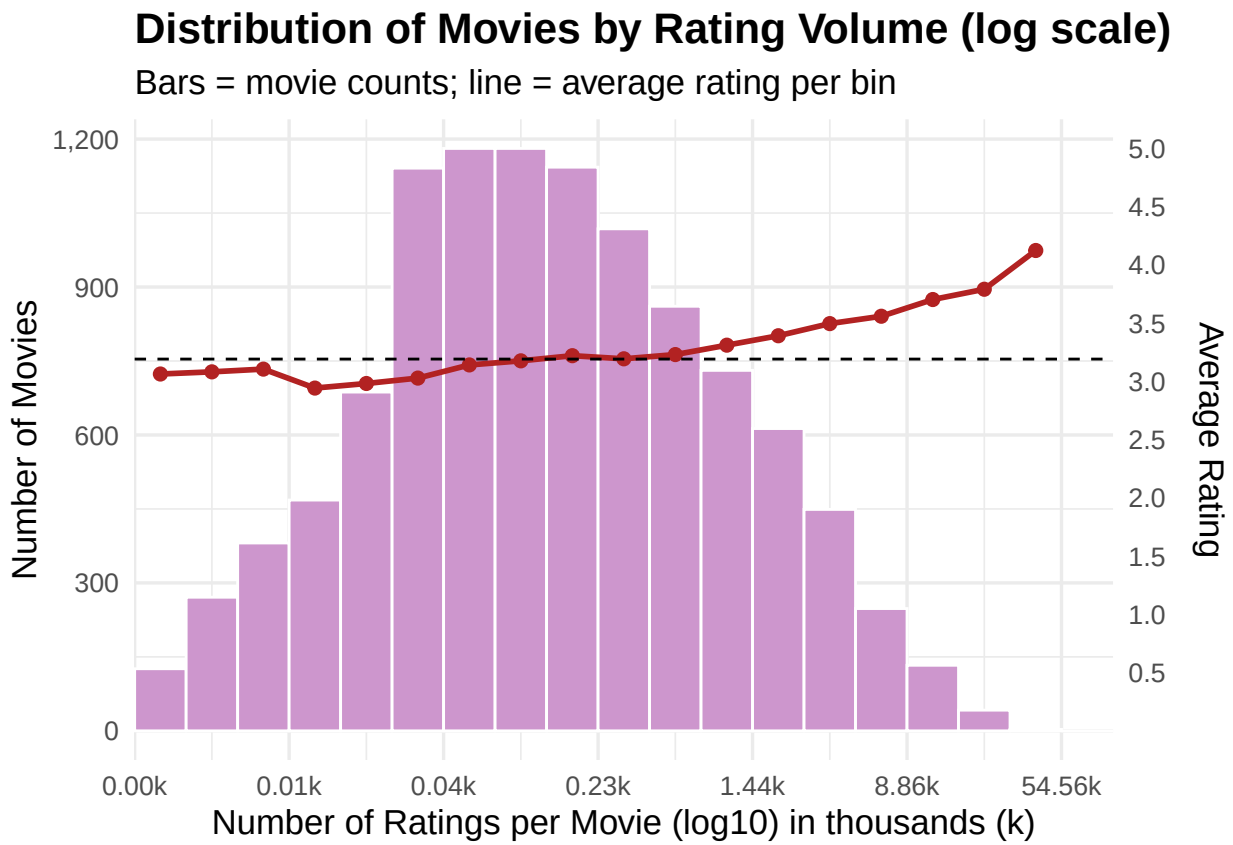
To extend this analysis further, a log scale is necessary for the number of ratings per movie.

```
knitr::opts_chunk$set(echo = TRUE)

library(dplyr)
library(ggplot2)
library(scales)

# 1. Compute per-movie stats
movie_stats <- edx %>%
  group_by(movieId) %>%
  summarise(
    n_ratings  = n(),
```

```r
    avg_rating = mean(rating),
    .groups     = "drop"
  )

# 2. Define 20 log-spaced breaks
breaks <- 10^seq(
  floor(log10(1)),
  ceiling(log10(max(movie_stats$n_ratings))),
  length.out = 20
)

# 3. Subset every 3rd break for axis labels
break_labels <- breaks[seq(1, length(breaks), by = 3)]

# 4. Bin the movies for the line/points
bin_summary <- movie_stats %>%
  mutate(bin = cut(
    n_ratings,
    breaks         = breaks,
    include.lowest = TRUE
  )) %>%
  group_by(bin) %>%
  summarise(
    movies_in_bin = n(),
    mean_rating   = mean(avg_rating),
    .groups       = "drop"
  ) %>%
  # compute numeric mid-point of each bin
  mutate(
    lower = breaks[as.integer(bin)],
    upper = breaks[as.integer(bin) + 1],
    mid   = sqrt(lower * upper)
  )

# 5. Scaling factor for the secondary axis
scale_factor <- max(bin_summary$movies_in_bin) / 5  # ratings run ~0.5-5

# 6. Plot: histogram + line + points
ggplot() +
  # A) histogram of counts
  geom_histogram(
    data   = movie_stats,
    aes(x = n_ratings, y = after_stat(count)),
    breaks = breaks,
    fill   = "plum3",
    color  = "white"
  ) +
  # B) average rating trend
  geom_line(
    data = bin_summary,
    aes(x = mid, y = mean_rating * scale_factor),
    color = "firebrick",
    size  = 1
```

```
  ) +
  geom_point(
    data = bin_summary,
    aes(x = mid, y = mean_rating * scale_factor),
    color = "firebrick",
    size  = 2
  ) +
  geom_hline(
    yintercept = mean(movie_stats$avg_rating) * scale_factor,
    linetype   = "dashed",
    color      = "black"
  ) +
  # Left axis = number of movies; Right axis = average rating
  scale_y_continuous(
    name   = "Number of Movies",
    labels = comma_format(),
    sec.axis = sec_axis(
      ~ . / scale_factor,
      name   = "Average Rating",
      breaks = seq(0.5, 5, 0.5)
    )
  ) +
  # X axis on log10, labeled in k, label only every 3rd bin
  scale_x_log10(
    name   = "Number of Ratings per Movie (log10) in thousands (k)",
    limits = range(breaks),
    expand = c(0, 0),
    breaks = break_labels,  # <--- only every 3rd tick is labeled
    labels = label_number(scale = 1/1000, suffix = "k", accuracy = 0.01)
  ) +
  labs(
    title    = "Distribution of Movies by Rating Volume (log scale)",
    subtitle = "Bars = movie counts; line = average rating per bin"
  ) +
  theme_minimal(base_size = 13) +
  theme(
    plot.title         = element_text(face = "bold"),
    axis.title.y.right = element_text(margin = margin(l = 10))
  )
```

```
## Warning: Using `size` aesthetic for lines was deprecated in ggplot2 3.4.0.
## i Please use `linewidth` instead.
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_lifecycle_warnings()` to see where this warning was
## generated.
```



The majority of movies receive a moderate number of ratings, while only a few movies have very high or very low numbers of ratings. The average rating for most movies is between around 3.3 and 3.7, but the most popular movies tend to have even higher ratings.

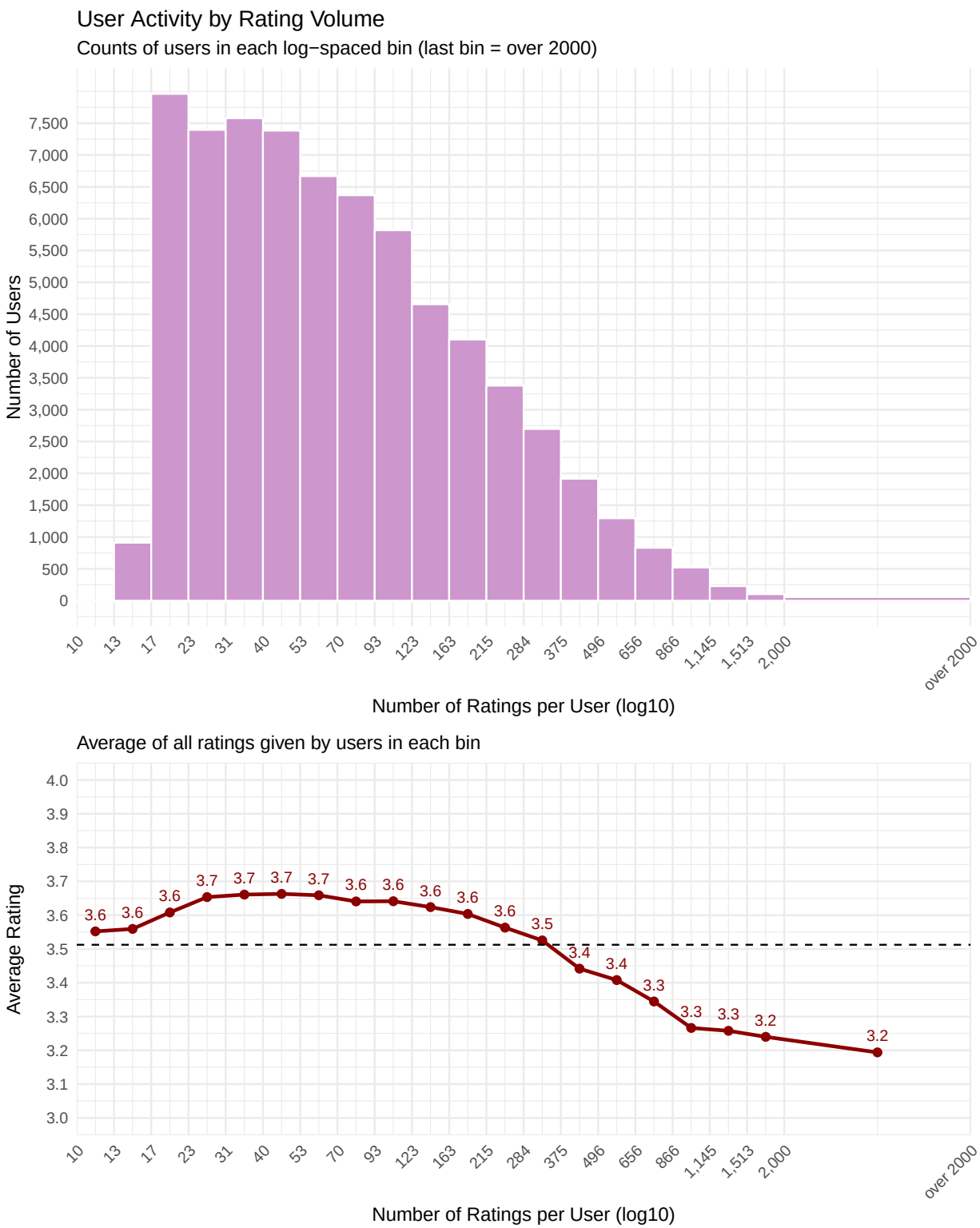
2.2.3 2.2.3 How many ratings does each user give?

An examination of user activity patterns reveals varying levels of engagement. Some users may rate many movies, while others rate only a few.

```
knitr::opts_chunk$set(echo = TRUE)

library(dplyr)
library(ggplot2)
library(scales)
library(patchwork)

# 1. Count ratings per user
user_stats <- edx %>%
  count(userId, name="n_ratings") %>%
  filter(n_ratings > 1)

# 2. Define log-spaced "main" breaks up to 2000
main_breaks <- 10^seq(
  floor(log10(min(user_stats$n_ratings))),
  log10(2000),
  length.out = 20
```

```
)

# 3. Compute an overflow break so each bin has equal log-width
ratio          <- main_breaks[2] / main_breaks[1]
max_n          <- max(user_stats$n_ratings)
n_steps        <- ceiling(log(max_n / tail(main_breaks,1)) / log(ratio))
overflow_break <- tail(main_breaks,1) * ratio^n_steps

# 4. Final breaks vector & labels
breaks <- c(main_breaks, overflow_break)
labels <- c(
  label_number(accuracy = 1, big.mark = ",")(main_breaks),
  "over 2000"
)
x_limits <- range(breaks)

# 5. Assign integer bins 1:(length(breaks)-1)
user_stats <- user_stats %>%
  mutate(bin = cut(
    n_ratings,
    breaks         = breaks,
    include.lowest = TRUE,
    labels         = FALSE
  ))

# 6a. Top summary: count per bin + numeric edges + midpoint
top_summary <- user_stats %>%
  count(bin, name="n_users") %>%
  mutate(
    lower = breaks[bin],
    upper = breaks[bin+1],
    mid   = sqrt(lower * upper)
  )

# 6b. Bottom summary: average of all ratings per bin + midpoint
bot_summary <- edx %>%
  inner_join(user_stats, by="userId") %>%
  group_by(bin) %>%
  summarise(
    avg_rating = mean(rating),
    .groups    = "drop"
  ) %>%
  inner_join(top_summary %>% select(bin, mid), by="bin")

# 7. Global average rating
global_avg <- mean(edx$rating)

# 8a. Top panel: fixed histogram with locked x-axis
p1 <- ggplot(top_summary) +
  geom_rect(aes(
    xmin = lower, xmax = upper,
    ymin = 0,     ymax = n_users
  ), fill="plum3", color="white") +
```

```r
  scale_x_log10(
    name    = "Number of Ratings per User (log10)",
    limits  = x_limits,
    expand  = c(0, 0),
    breaks  = breaks,
    labels  = labels
  ) +
  scale_y_continuous(
    name   = "Number of Users",
    breaks = seq(0, max(top_summary$n_users), by = 500),
    labels = comma_format()
  ) +
  labs(
    title    = "User Activity by Rating Volume",
    subtitle = "Counts of users in each log-spaced bin (last bin = over 2000)"
  ) +
  theme_minimal() +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))

# 8b. Bottom panel: average-rating trend with the same x-axis
p2 <- ggplot(bot_summary, aes(x = mid, y = avg_rating)) +
  geom_line(color="darkred", size=1) +
  geom_point(color="darkred", size=2) +
  geom_text(aes(label = round(avg_rating,1)),
            vjust = -1, color = "darkred", size = 3) +
  geom_hline(yintercept = global_avg,
             linetype = "dashed", color = "black") +
  scale_x_log10(
    name    = "Number of Ratings per User (log10)",
    limits  = x_limits,
    expand  = c(0, 0),
    breaks  = breaks,
    labels  = labels
  ) +
  scale_y_continuous(
    name   = "Average Rating",
    limits = c(3, 4.),
    breaks = seq(3, 4., 0.1)
  ) +
  labs(subtitle = "Average of all ratings given by users in each bin") +
  theme_minimal() +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))

# 9. Stack panels at 3:2 ratio and draw
combo <- p1 / p2 + plot_layout(heights = c(3, 2))

# Print the combined plot
combo
```



It is notable that users who provide less than approximately 30 ratings and those who provide more than 80 ratings tend to give lower ratings than users who provide between 30 and 80 ratings.

```
knitr::opts_chunk$set(echo = TRUE)


library(dplyr)
library(ggplot2)
library(scales)
library(patchwork)

# Step 1: Create of the user_stats
user_stats <- edx %>%
  count(userId, name="n_ratings") %>%
  filter(n_ratings > 1)

# Step 2: Then computate and print percentages
total_users    <- nrow(user_stats)
pct_less100    <- round(mean(user_stats$n_ratings <  100) * 100, 1)
pct_atleast100 <- round(mean(user_stats$n_ratings >= 100) * 100, 1)

cat("Users with <100 ratings:   ", pct_less100, "%\n")
```

```
## Users with <100 ratings:    65.5 %
```

```
cat("Users with  100 ratings:   ", pct_atleast100, "%\n")
```

```
## Users with  100 ratings:    34.5 %
```

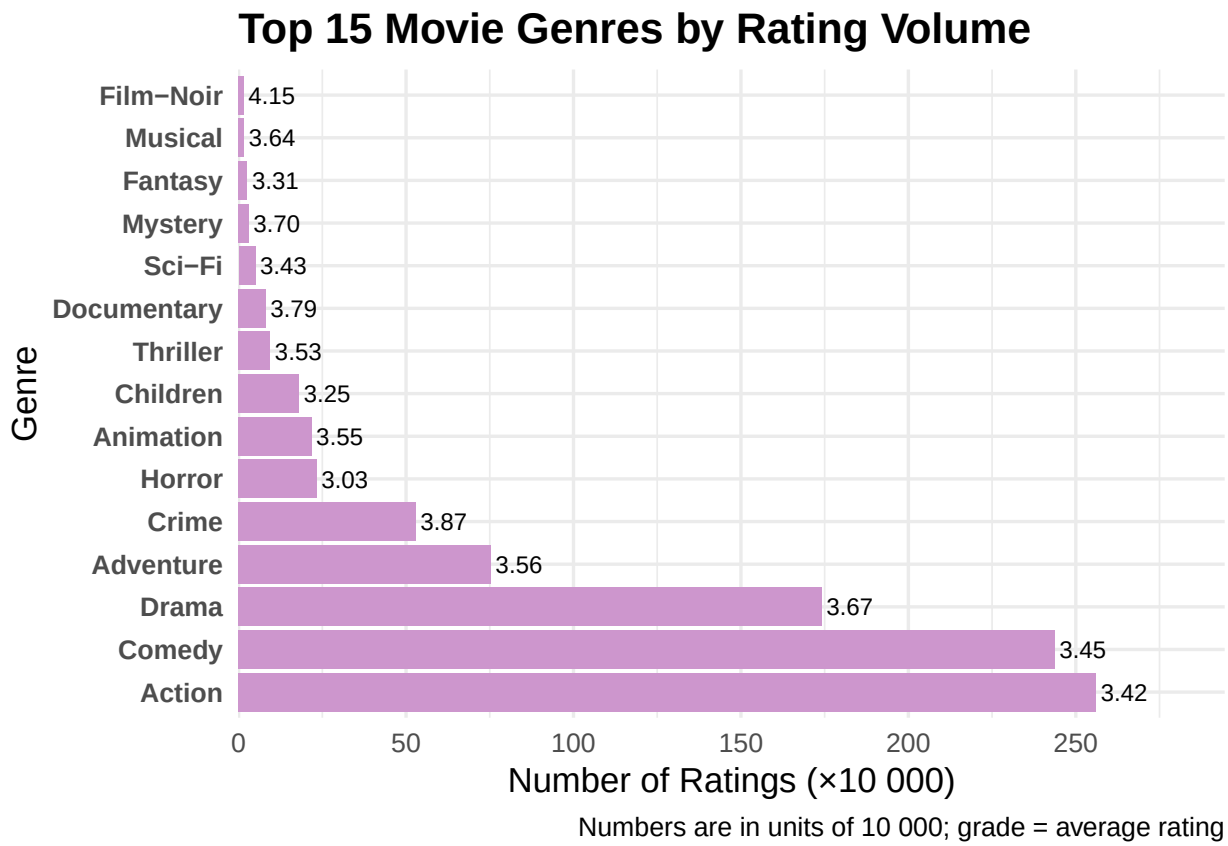
The data shows that approximately 2/3 of users give fewer than 100 ratings. A small group of users contribute hundreds or thousands of ratings. This distribution suggests that the model needs to take user activity levels into account.

2.2.4 2.2.4 Which genres are most common?

An analysis of the most frequent genres provides insight into the types of movies present in the dataset.

```
knitr::opts_chunk$set(echo = TRUE)

library(dplyr)
library(stringr)
library(ggplot2)
library(forcats)
library(scales)
library(knitr)

# 1. Compute top 15 genres by volume + median
genre_summary <- edx %>%
  mutate(main_genre = str_extract(genres, "^[^|]+")) %>%      # first genre
  group_by(main_genre) %>%
  summarise(
    n_ratings    = n(),
    avg_rating = mean(rating),
    .groups       = "drop"
```

```r
  ) %>%
  slice_max(n_ratings, n = 15) %>%                              # top 15
  mutate(
    main_genre    = fct_reorder(main_genre, n_ratings, .desc = TRUE),
    volume_10k    = n_ratings / 10000
  )

# 2. Plot
max10k <- ceiling(max(genre_summary$volume_10k) / 5) * 5

ggplot(genre_summary, aes(x = volume_10k, y = main_genre)) +
  geom_col(fill = "plum3") +
  geom_text(
    aes(label = sprintf("%.2f", avg_rating)),
    hjust = -0.1, color = "black", size = 3
  ) +
  scale_x_continuous(
    name   = "Number of Ratings (×10 000)",
    breaks = seq(0, max10k, by = 50),
    labels = label_number(accuracy = 1),
    expand = expansion(mult = c(0, 0.15))
  ) +
  labs(
    title = "Top 15 Movie Genres by Rating Volume",
    y     = "Genre",
    caption = "Numbers are in units of 10 000; grade = average rating"
  ) +
  theme_minimal(base_size = 13) +
  theme(
    plot.title = element_text(face = "bold"),
    axis.text.y = element_text(face = "bold")
  )
```



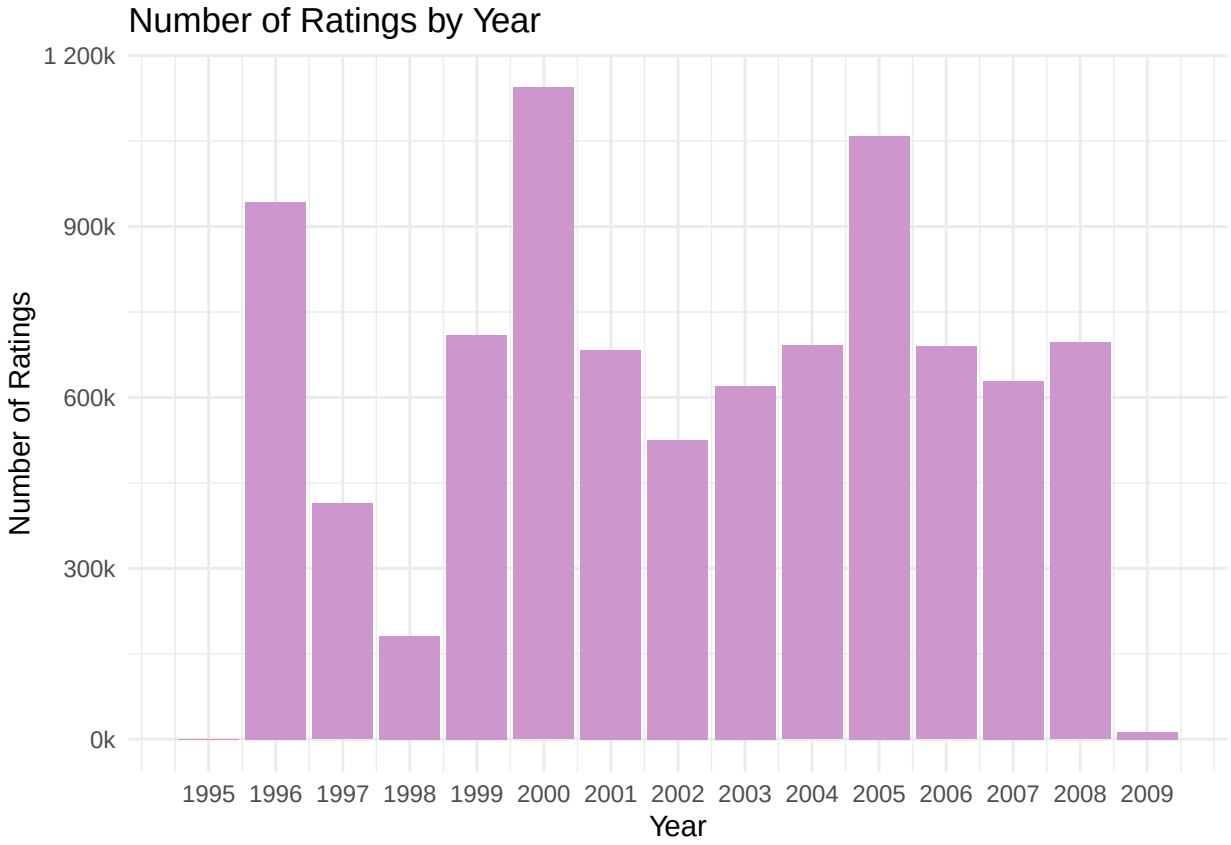
Action, Comedy and Drama appear to be the most frequently rated genres, while genres such as Musical or Film-Noir are considerably less common.

2.2.5 2.2.5 Do the number of ratings change over time?

The distribution of ratings can also be examined over time to determine whether certain years experienced higher volumes of ratings compared to others.

```
knitr::opts_chunk$set(echo = TRUE)

library(dplyr)
library(ggplot2)
library(scales)

# 1. Summarize once, store into df_year
df_year <- edx %>%
  group_by(rating_year) %>%
  summarise(count = n(), .groups = "drop") %>%
  arrange(rating_year)

# 2. Plot using df_year
ggplot(df_year, aes(x = rating_year, y = count)) +
  geom_col(fill = "plum3") +
  scale_x_continuous(
    breaks = df_year$rating_year,
    labels = df_year$rating_year
```

```
  ) +
  scale_y_continuous(
    name   = "Number of Ratings",
    labels = label_number(scale = 1/1000, suffix = "k")
  ) +
  labs(
    title = "Number of Ratings by Year",
    x     = "Year"
  ) +
  theme_minimal()
```



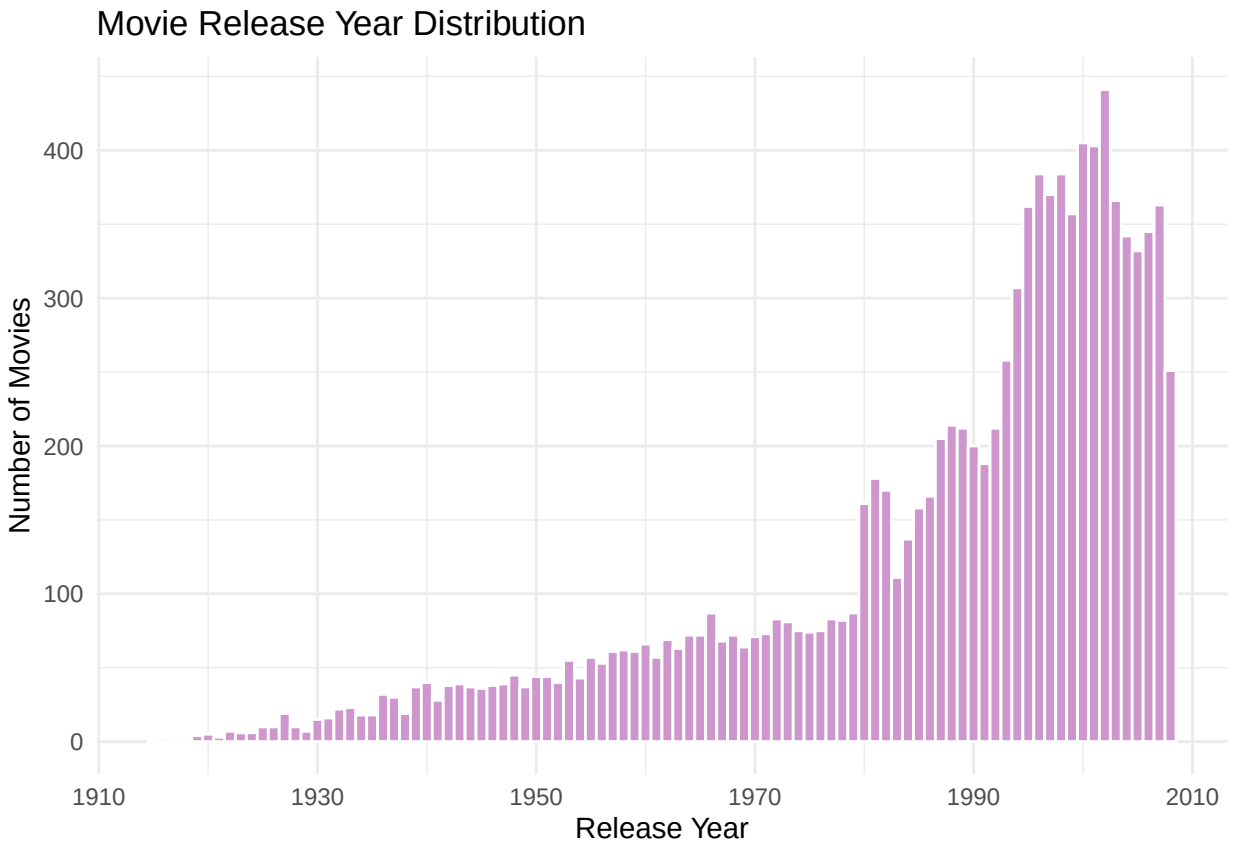
The annual number of ratings fluctuates rather than following a linear trend, and does not display a normal distribution. A significant increase is observed around the year 2000, followed by a decline in 2001–2002. The volume of ratings then rises again towards 2005, before stabilizing between 2006 and 2008. This saw-tooth pattern indicates that user activity occurred in waves rather than exhibiting steady growth. Except for the years 1995 and 2009, the number of ratings recorded each year appears sufficient to represent the general user base.

2.2.6 2.2.6 When were the movies released?

The release year of the movies is examined to determine whether the dataset primarily contains older or more recent films.

```
knitr::opts_chunk$set(echo = TRUE)
```

```r
library(dplyr)
library(ggplot2)
library(stringr)

edx %>%
  # 1) pull out the 4-digit year from the title into a new column
  mutate(
    release_year = str_extract(title, "(?<=\\()\\d{4}(?=\\))") %>%
                   as.integer()
  ) %>%
  # 2) reduce to one row per movie
  distinct(movieId, release_year) %>%
  # 3) drop any missing/extract failures (just in case)
  filter(!is.na(release_year)) %>%
  # 4) build the histogram
  ggplot(aes(x = release_year)) +
    geom_histogram(binwidth = 1, fill = "plum3", color = "white") +
    labs(
      title = "Movie Release Year Distribution",
      x     = "Release Year",
      y     = "Number of Movies"
    ) +
    theme_minimal()
```



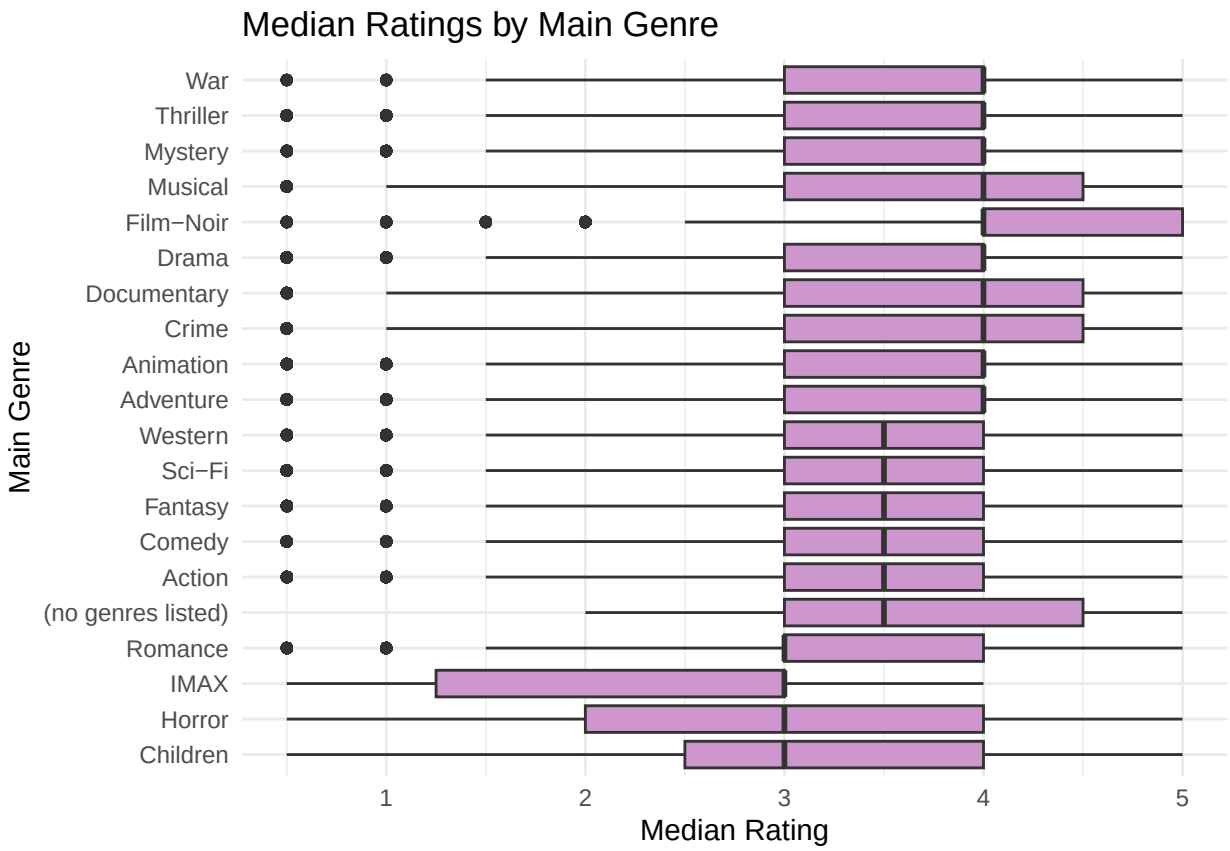
Most movies in the dataset were released between 1980 and 2005, with a small number of older classics and very few titles released after 2010. This distribution may impact predictions if user rating behavior differs

between older and newer films.

2.2.7 2.2.7 Do some genres get better ratings?

An analysis is conducted to determine whether ratings vary according to the movie's main genre.

```
knitr::opts_chunk$set(echo = TRUE)

library(dplyr)
library(ggplot2)
library(stringr)

edx %>%
  # 1. Extract the first genre into main_genre
  mutate(
    main_genre = str_extract(genres, "^[^|]+")
  ) %>%
  # 2. Drop any movies where extraction failed
  filter(!is.na(main_genre)) %>%
  # 3. Reorder genres by their median rating
  mutate(
    main_genre = fct_reorder(main_genre, rating, .fun = median)
  ) %>%
  # 4. Plot
  ggplot(aes(x = main_genre, y = rating)) +
    geom_boxplot(fill = "plum3") +
    coord_flip() +
    labs(
      title = "Median Ratings by Main Genre",
      x     = "Main Genre",
      y     = "Median Rating"
    ) +
    theme_minimal()
```

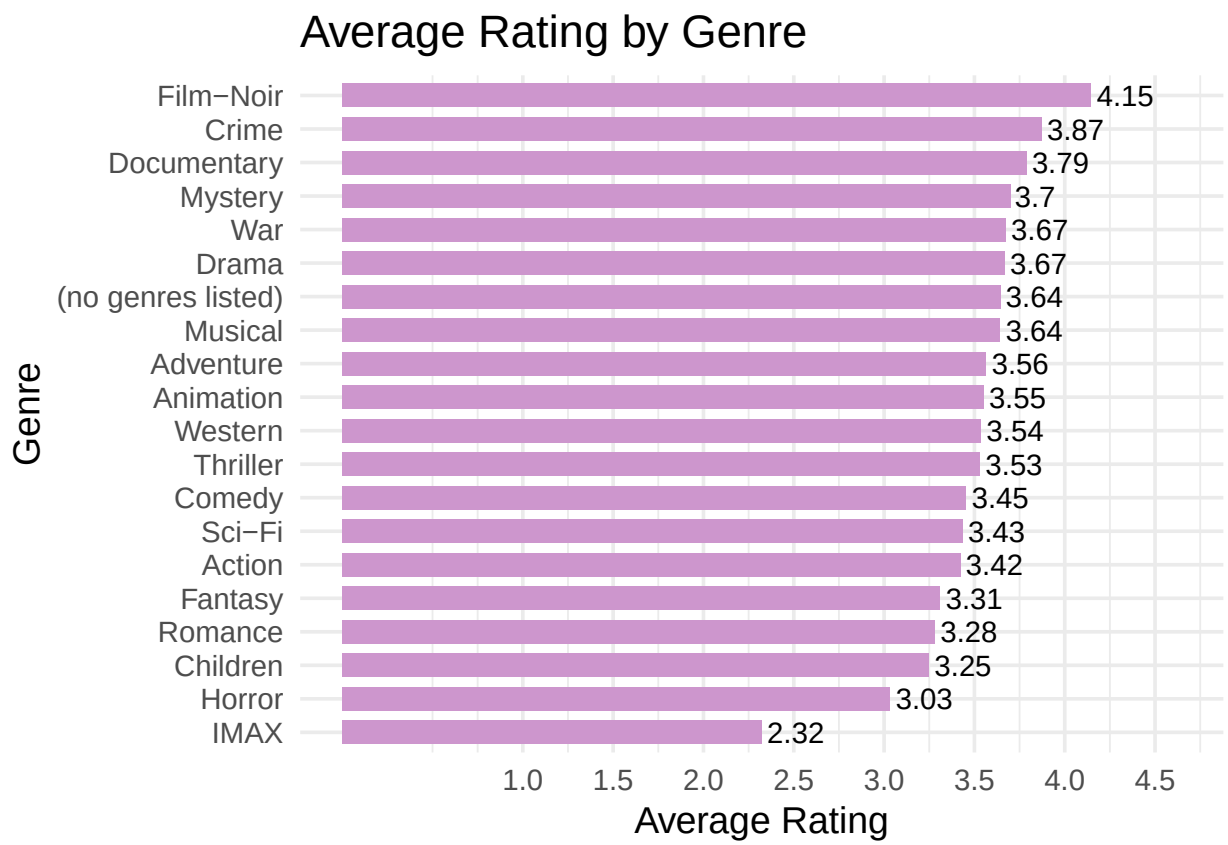


Certain genres, such as Documentary or War, tend to receive higher average ratings, while genres like Horror exhibit greater variability. These patterns indicate that genre may influence user rating behavior.

To provide a clearer understanding of genre performance, the average rating for each genre is examined.

```
knitr::opts_chunk$set(echo = TRUE)

library(dplyr)
library(ggplot2)

# Calculate the average rating by genre
avg_rating_genre <- edx %>%
  filter(!is.na(main_genre)) %>%
  group_by(main_genre) %>%
  summarise(
    avg_rating = mean(rating, na.rm = TRUE),
    count = n()
  ) %>%
  arrange(desc(avg_rating))

# Plot: Average Rating by Genre (bar chart)
ggplot(avg_rating_genre, aes(x = reorder(main_genre, avg_rating), y = avg_rating, fill = avg_rating)) +
  geom_col(show.legend = FALSE, width = 0.7, fill = "plum3") +
  geom_text(aes(label = round(avg_rating, 2)),
            hjust = -0.1, size = 4, color = "black") + # Adds value to end of bar
  coord_flip() +
  labs(
```

```
    title = "Average Rating by Genre",
    x = "Genre",
    y = "Average Rating"
  ) +
  scale_y_continuous(limits = c(0, max(avg_rating_genre$avg_rating) + 0.5), breaks = seq(1, 5, 0.5)) +
  theme_minimal(base_size = 14)
```



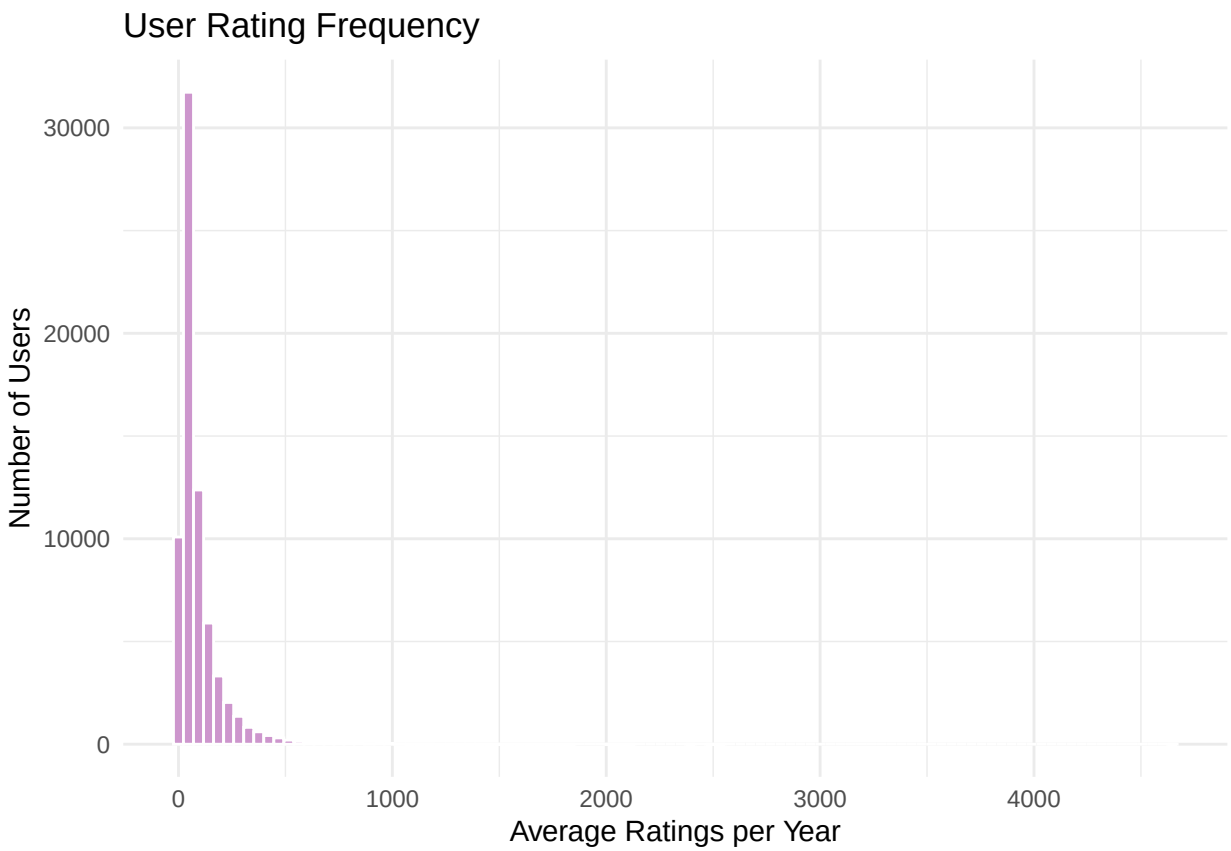
The chart reveals substantial differences in average ratings across movie genres. Film-Noir, Crime, and Documentary genres receive the highest average ratings, with Film-Noir notably standing out at 4.15. In contrast, IMAX and Horror genres register the lowest average ratings, with IMAX rated lowest at 2.32. Most genres, including Drama, Mystery, War, and Musical, have average ratings ranging from 3.3 to 3.9. Popular genres such as Comedy, Action, and Sci-Fi are rated slightly below the overall average. These findings suggest that audience preferences and appreciation vary considerably depending on genre. The particularly low rating for IMAX may be attributable to a smaller number of movies in this category or differing user expectations.

2.2.8 2.2.8 How often do users rate movies?

The frequency with which each user rates movies is examined to better understand user behavior.

```
knitr::opts_chunk$set(echo = TRUE)

user_activity %>%
  ggplot(aes(x = user_frequency)) +
  geom_histogram(bins = 100, fill = "plum3", color = "white") +
  labs(title = "User Rating Frequency",
```

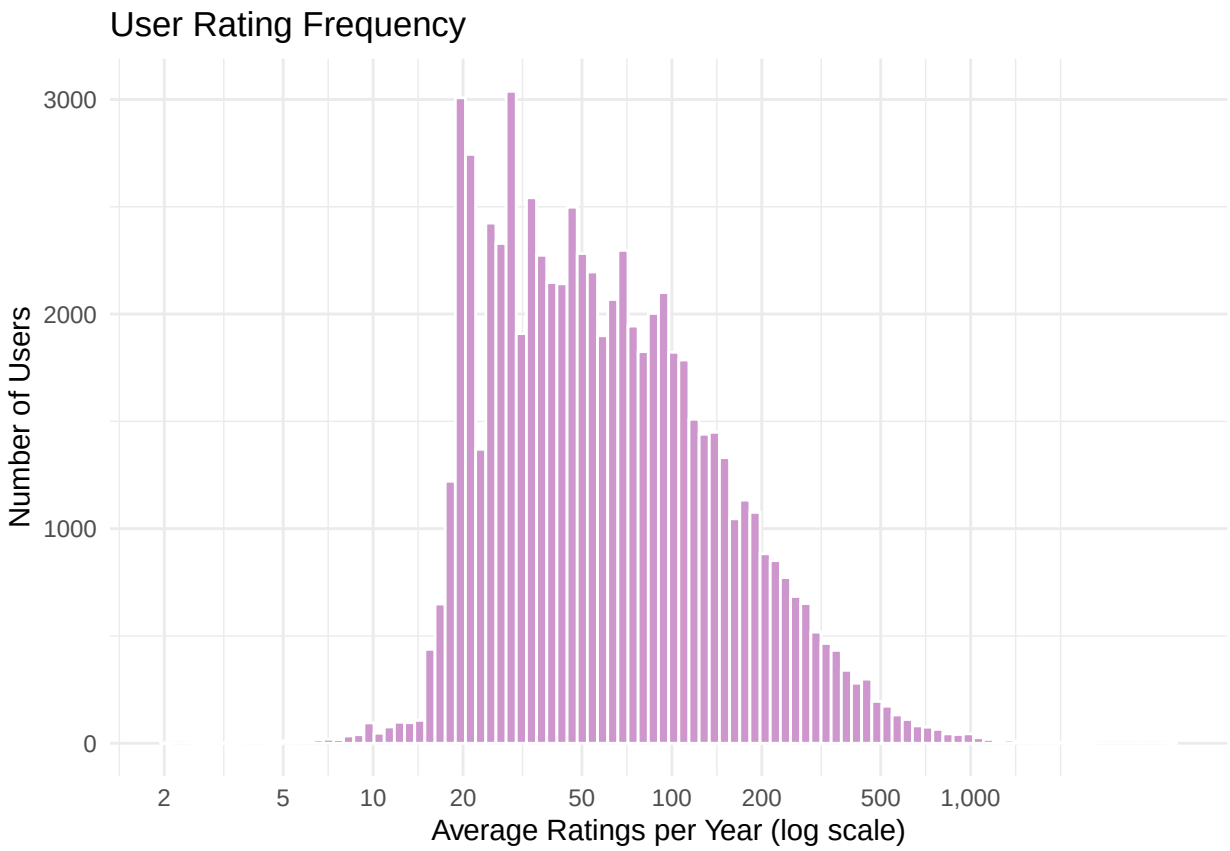
```
      x = "Average Ratings per Year",
      y = "Number of Users") +
  theme_minimal()
```



The majority of users rate fewer than 20 movies per year on average, while a small number of highly active users provide significantly more ratings. This imbalance may influence the model, as more active users have a disproportionate impact on the dataset.

To improve readability, the data is analyzed using a logarithmic x-axis.

```
knitr::opts_chunk$set(echo = TRUE)

user_activity %>%
  ggplot(aes(x = user_frequency)) +
  geom_histogram(bins = 100, fill = "plum3", color = "white") +
  scale_x_log10(              # Add this line for a log-scale x axis
    name = "Average Ratings per Year (log scale)",
    breaks = c(1, 2, 5, 10, 20, 50, 100, 200, 500, 1000),
    labels = scales::comma_format()
  ) +
  labs(
    title = "User Rating Frequency",
    y = "Number of Users"
  ) +
  theme_minimal()
```



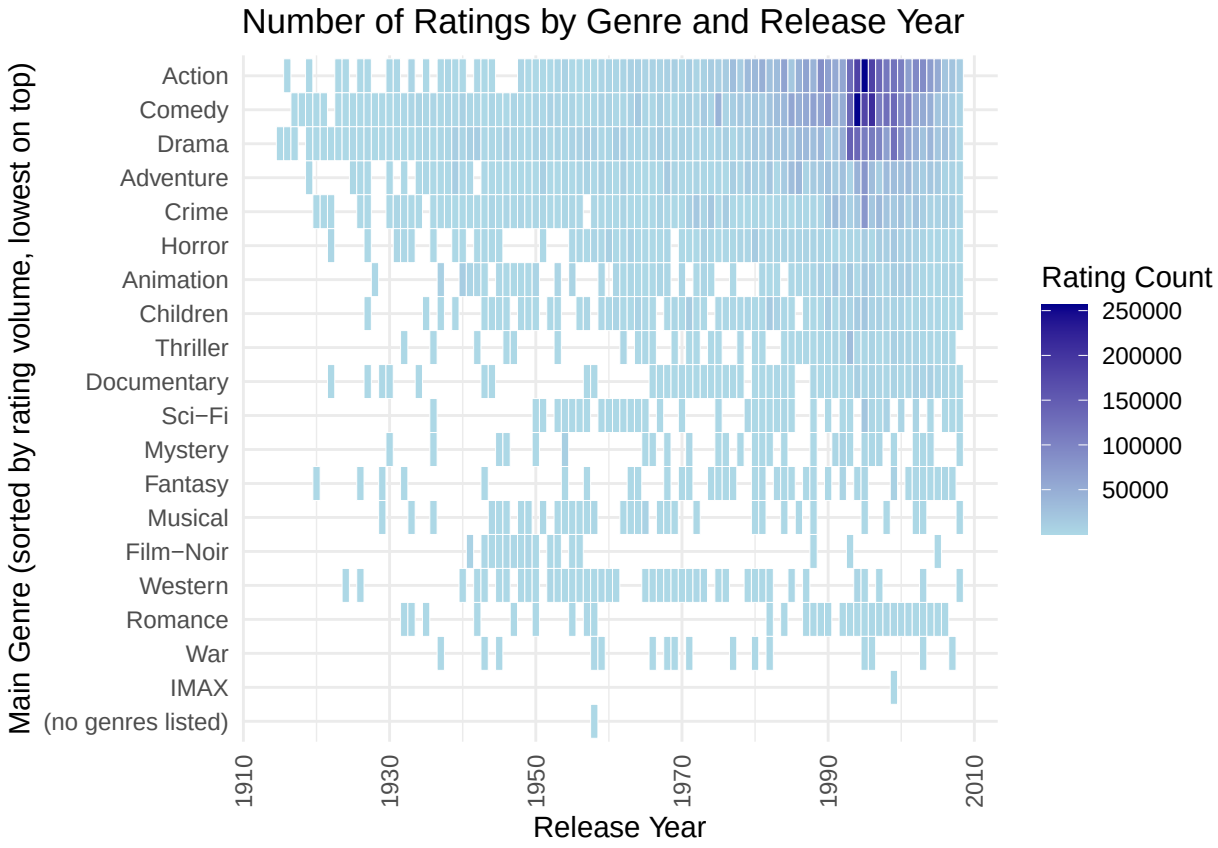
The histogram provides a clearer view of the distribution of user rating frequency when displayed on a logarithmic scale. Most users submit between 20 and 200 ratings per year, while very few users rate fewer than 10 or more than 1,000 movies annually. The use of a log scale highlights that a small proportion of highly active users account for a large number of ratings, whereas the majority of users are less active.

2.2.9 2.2.9 How many ratings were given for each genre over time?

The number of ratings for each genre is examined according to the release year of the movie. This analysis provides insight into which types of films were most frequently watched and rated during different time periods.

```
knitr::opts_chunk$set(echo = TRUE)

library(dplyr)
library(ggplot2)

# 1. Calculate number of ratings per year and genre
genre_year_volume <- edx %>%
  filter(!is.na(release_year), !is.na(main_genre)) %>%
  group_by(release_year, main_genre) %>%
  summarise(count = n(), .groups = "drop")

# 2. Calculate total volume by genre
genre_totals <- genre_year_volume %>%
  group_by(main_genre) %>%
  summarise(total_count = sum(count)) %>%
```

```
  arrange(total_count)

# 3. Factor main_genre by total_count (lowest on top, highest at bottom)
genre_year_volume <- genre_year_volume %>%
  mutate(main_genre = factor(main_genre, levels = genre_totals$main_genre))

# 4. Plot heatmap
ggplot(genre_year_volume, aes(x = release_year, y = main_genre, fill = count)) +
  geom_tile(color = "white") +
  scale_fill_gradient(low = "lightblue", high = "darkblue", na.value = "grey90") +
  labs(
    title = "Number of Ratings by Genre and Release Year",
    x = "Release Year",
    y = "Main Genre (sorted by rating volume, lowest on top)",
    fill = "Rating Count"
  ) +
  theme_minimal() +
  theme(axis.text.x = element_text(angle = 90, vjust = 0.5))
```



The heatmap displays the number of ratings for each movie genre by release year, with darker blue indicating a higher volume of ratings.

Genres such as Drama and Comedy received a significant number of ratings during the 1990s and early 2000s. In contrast, genres like Western or Film-Noir have fewer ratings overall, primarily concentrated among older films.

This analysis reveals which genres were most popular for viewing and rating in different decades. It also

highlights certain limitations within the dataset: some genres and time periods are underrepresented, which could impact the accuracy of model predictions.

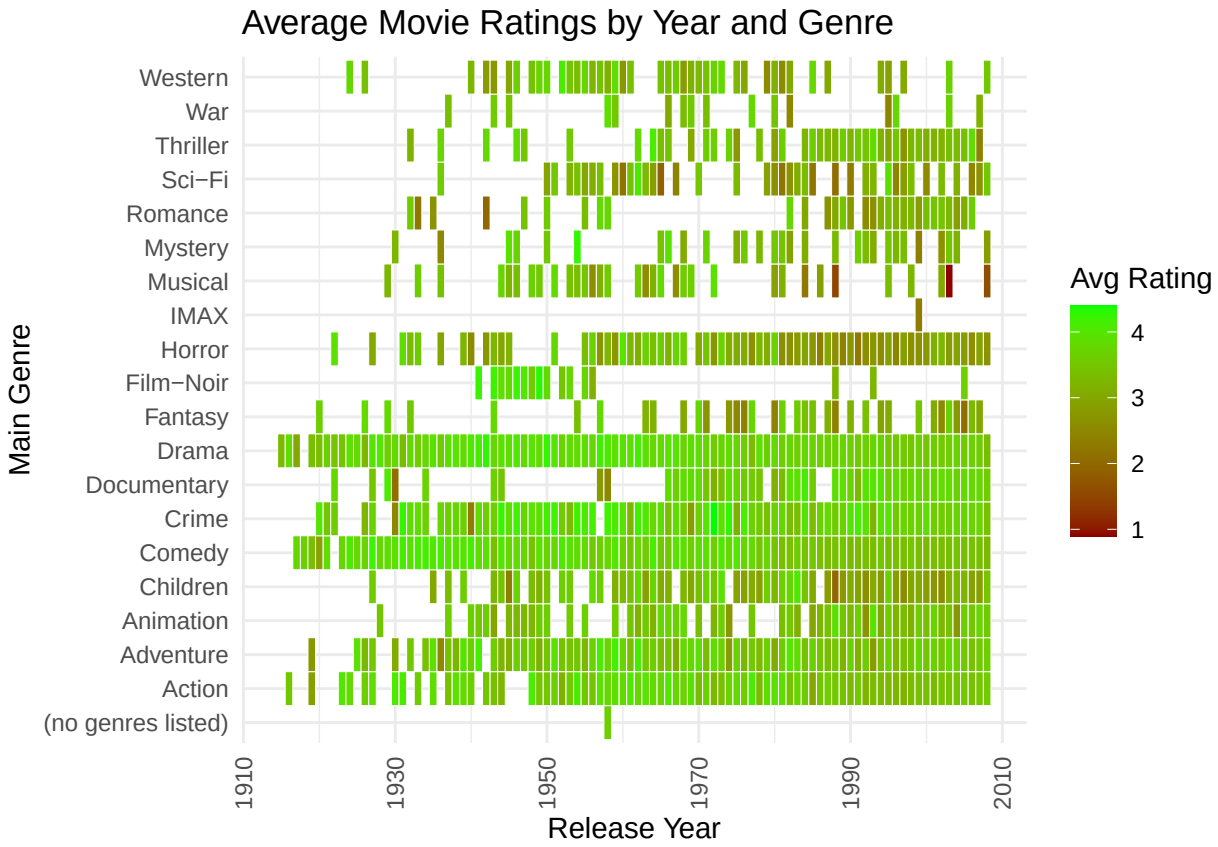
2.2.10 2.2.10 Are there trends in genre preferences over time?

To analyze how user preferences have evolved over time, the average rating for each genre by release year is examined. The resulting heatmap illustrates which genres were most highly rated in different time periods, with green indicating higher average ratings (close to 5) and dark red representing lower ratings.

The visualization reveals that some genres have increased in popularity over the years, while others have experienced a decline.

These observations provide valuable insights for model development, suggesting that both genre and release year may influence user rating behavior.

```
knitr::opts_chunk$set(echo = TRUE)

# Calculate the average grade by year and gender
genre_year_matrix <- edx %>%
  filter(!is.na(release_year), !is.na(main_genre)) %>%
  group_by(release_year, main_genre) %>%
  summarise(mean_rating = mean(rating), .groups = "drop")

# Create a heatmap: years on the x-axis, genders on the y-axis, color = average score

ggplot(genre_year_matrix, aes(x = release_year, y = main_genre, fill = mean_rating)) +
  geom_tile(color = "white") +
  scale_fill_gradient(low = "darkred", high = "green", na.value = "grey90") +
  labs(title = "Average Movie Ratings by Year and Genre",
       x = "Release Year",
       y = "Main Genre",
       fill = "Avg Rating") +
  theme_minimal() +
  theme(axis.text.x = element_text(angle = 90, vjust = 0.5))
```

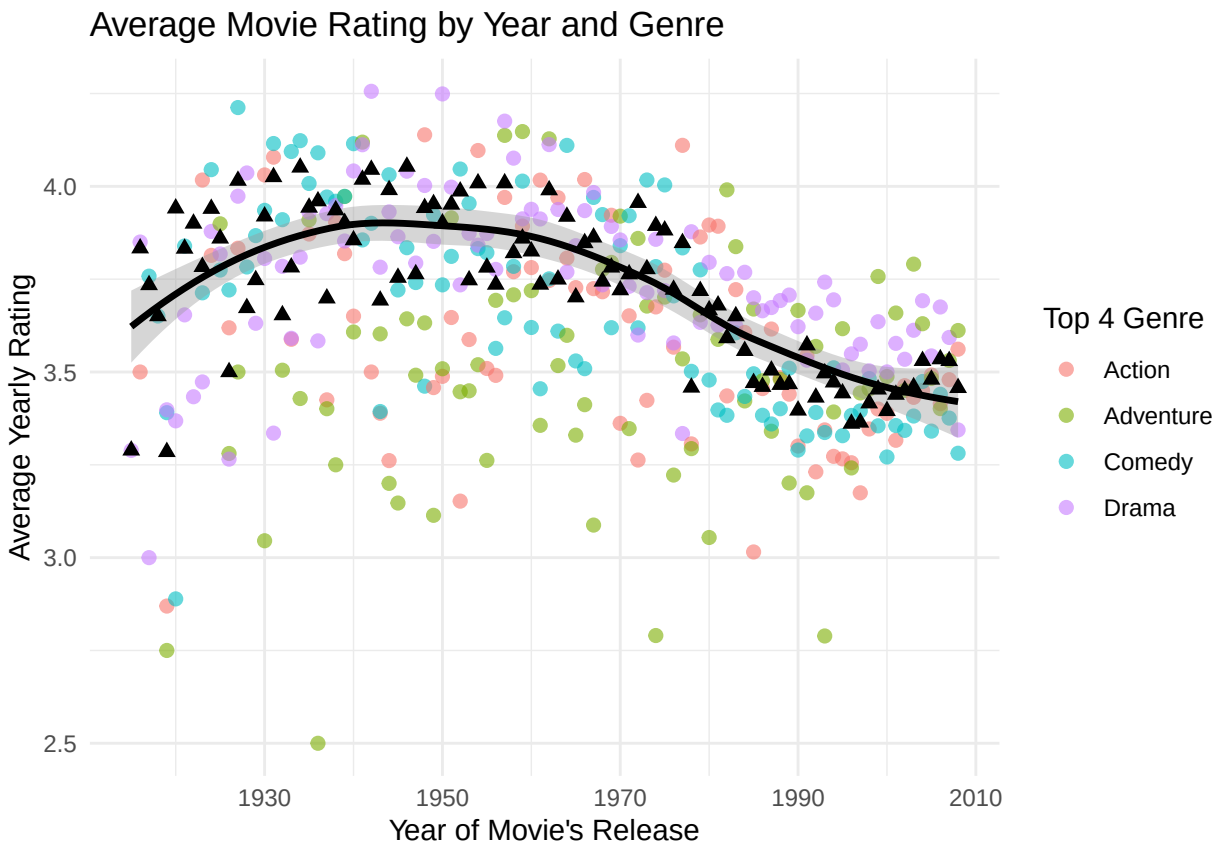


The graph illustrates how the average movie rating varies by genre and year. Genres such as Drama, Comedy, Adventure, and Action consistently feature a large number of movies each year, whereas other genres are less frequently represented. Most average ratings fall between 3 and 4, with limited variation observed over time. Certain genres, including Western and War, are more prevalent among older films and have become less common in recent years.

2.2.11 2.2.11 Is there a big disparity in average rating over time and in the top 4 genres ?

```
knitr::opts_chunk$set(echo = TRUE)

library(dplyr)
library(ggplot2)

# Calculate average rating per release year
avg_year_rating <- edx %>%
  group_by(release_year) %>%
  summarise(avg_rating = mean(rating), .groups = "drop")

# Identify top 4 genres by number of ratings
top_genres <- edx %>%
  count(main_genre, sort = TRUE) %>%
  top_n(4, n) %>%
  pull(main_genre)

# Calculate average rating per year and genre
```

```r
avg_year_genre <- edx %>%
  group_by(release_year, main_genre) %>%
  summarise(avg_rating = mean(rating), .groups = "drop") %>%
  filter(main_genre %in% top_genres)

# Create the plot
ggplot() +
  # 1. Points colored for the top movie genre (part 1)
  geom_point(
    data = avg_year_genre,
    aes(x = release_year, y = avg_rating, color = main_genre),
    size = 2, alpha = 0.6
  ) +
  # 2. Global trends smoothed (part 2)
  geom_smooth(
    data = avg_year_rating,
    aes(x = release_year, y = avg_rating),
    method = "loess", se = TRUE, color = "black", linewidth = 1.2
  ) +
  # 3. Points for the average yearly rating in black triangle
  geom_point(
    data = avg_year_rating,
    aes(x = release_year, y = avg_rating),
    color = "black", size = 2, shape = 17
  ) +
  labs(
    title = "Average Movie Rating by Year and Genre",
    x = "Year of Movie's Release",
    y = "Average Yearly Rating",
    color = "Top 4 Genre"
  ) +
  theme_minimal()
```



The graph presents the average movie rating by year and genre. Ratings tend to be higher for films released between 1930 and 1960, while a gradual decline is observed for most genres after 1970.

The overall trend, indicated by the black line, confirms this pattern. These changes may reflect shifts in user rating behavior or transformations in movie production over time.

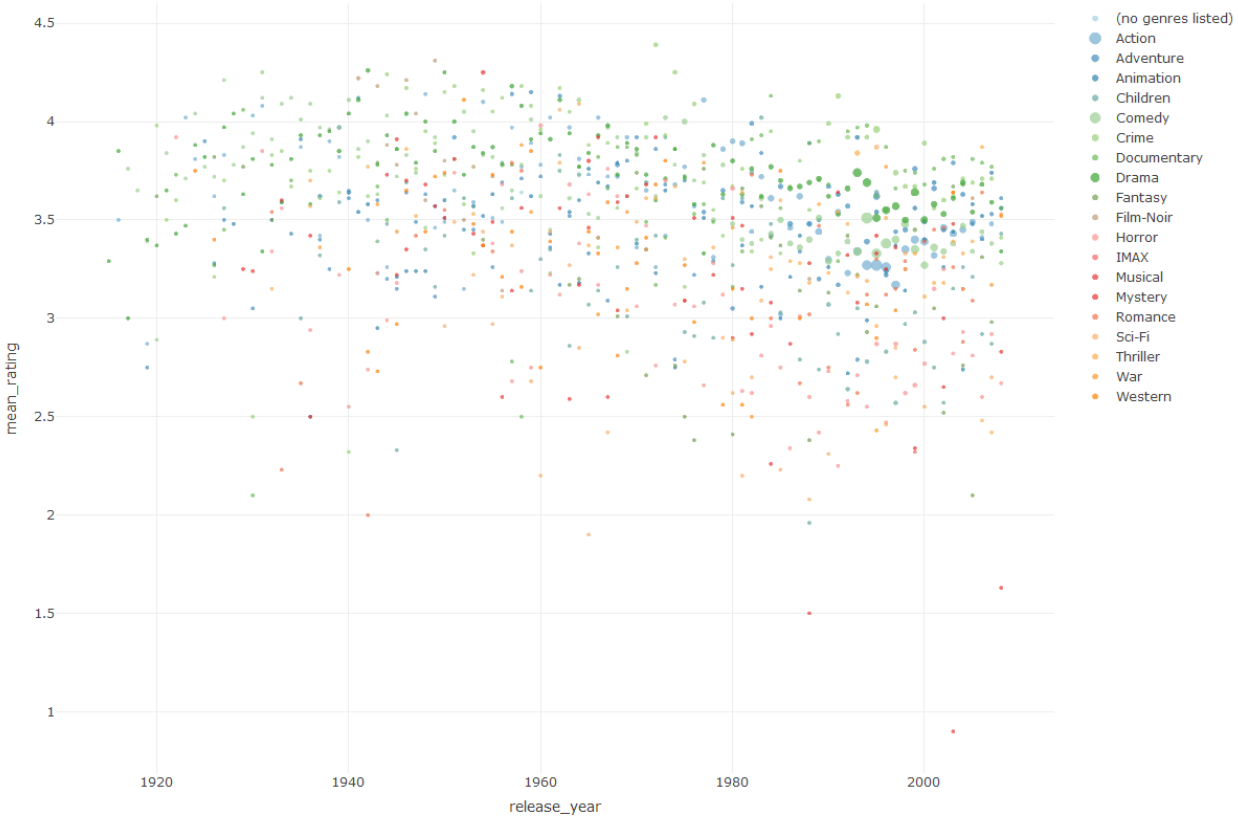
Additionally, notable differences between genres are evident each year, with certain genres such as Adventure displaying greater variability.

2.2.12 2.2.12 How average movie ratings changed over time, by genre?

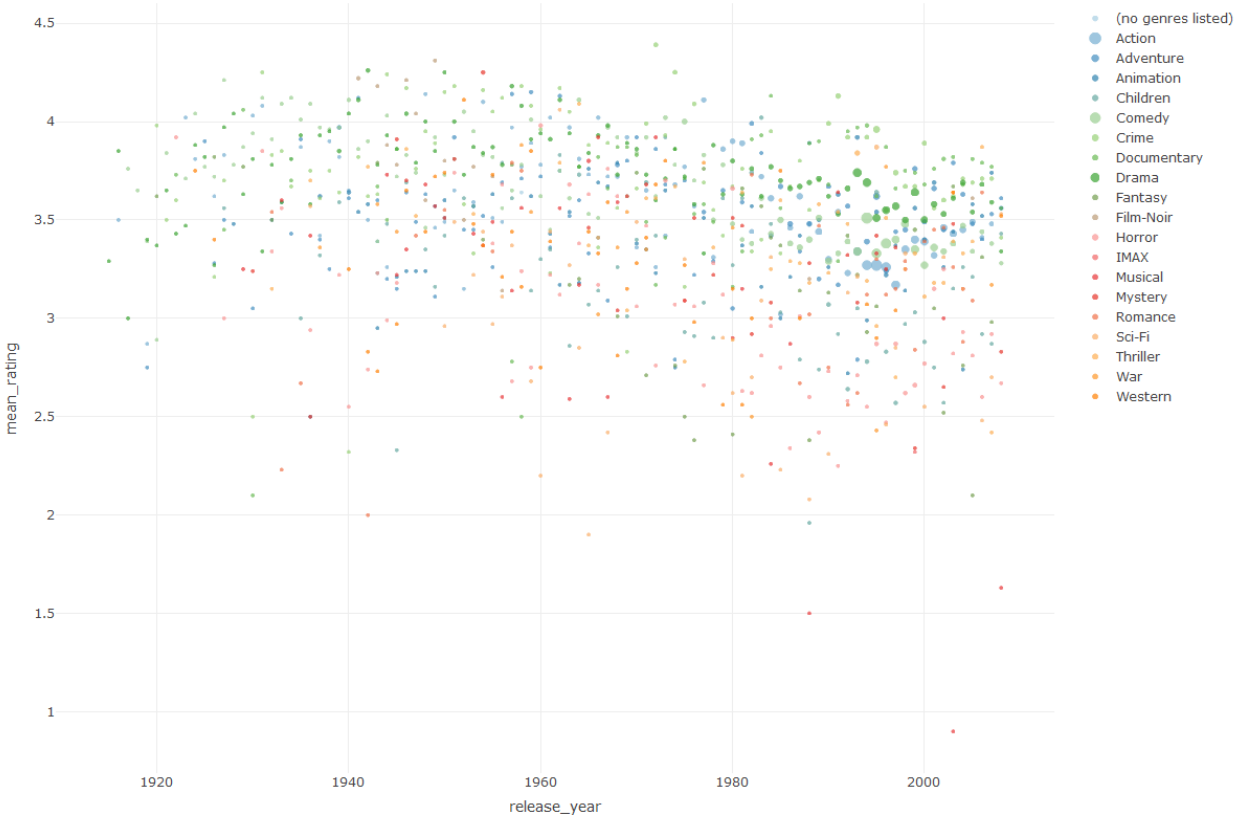
The chart displays the average rating for each genre and release year, with point size indicating the number of ratings received. Additional details for each point can be accessed by hovering over them.

```
knitr::opts_chunk$set(echo = TRUE)


library(plotly)
library(dplyr)
library(RColorBrewer)
library(webshot2)
library(webshot)
library(htmlwidgets)


# Prepare data
genre_year_summary <- edx %>%
```

```r
  filter(!is.na(main_genre), !is.na(release_year)) %>%
  group_by(release_year, main_genre) %>%
  summarise(
    mean_rating = round(mean(rating), 2),
    count = n(),
    .groups = "drop"
  )

# Build a color palette with enough unique colors
n_genres <- length(unique(genre_year_summary$main_genre))
my_colors <- colorRampPalette(brewer.pal(8, "Paired"))(n_genres) # or try "Paired", "Dark2"

# Create the plotly chart
p <- plot_ly(
  data = genre_year_summary,
  x = ~release_year,
  y = ~mean_rating,
  type = "scatter",
  mode = "markers",
  color = ~main_genre,
  colors = my_colors,
  size = ~count,
  sizes = c(5, 50),
  marker = list(
    sizemode = "area",
    opacity = 0.7
  ),
  text = ~paste(
    "Genre:", main_genre,
    "<br>Year:", release_year,
    "<br>Average Rating:", mean_rating,
    "<br>Number of Ratings:", count
  ),
  hoverinfo = "text"
)

# Save the plotly as a page HTML
htmlwidgets::saveWidget(p, "plotly_temp.html")

# Take a webshot of plotly in PNG
webshot::webshot("plotly_temp.html", "plotly_plot.png", vwidth = 1200, vheight = 800)
```




```
# Integrate it into the pdf report
knitr::include_graphics("plotly_plot.png")
```



This chart provides a combined view of both quality (average rating) and popularity (number of ratings) for each genre. For instance, certain genres may receive high ratings but have low rating volume, while others are widely rated yet receive lower average scores. Most movie genres maintain average ratings between 3 and 4 across the years.

Following 1980, the number of movies released increases for all genres, though average ratings remain relatively stable over time. Comedy, Drama, and Action emerge as the most prevalent genres.

2.2.13 2.2.13 Summary of Data Exploration

Prior to developing a prediction algorithm, thorough data exploration is essential. This process facilitates a comprehensive understanding of the dataset's structure, trends, and potential biases. Analyzing how users rate movies, across genres, release years, and varying levels of user activity, enables more informed decisions in model design and helps prevent common errors.

Exploration of the MovieLens dataset uncovers several key insights regarding user behavior, movie attributes, and rating patterns:

- The number of ratings fluctuates over the years, displaying peaks and declines rather than a linear trend.
- Sufficient ratings are available for most years, ensuring meaningful analysis. Average ratings vary over time and across genres, with certain genres consistently receiving higher or lower scores.
- The dataset provides a comprehensive overview of user engagement and movie ratings, shedding light on preferences across different periods and film categories.
- Most users rate between 20 and 200 movies per year, while a small group of highly active users contribute thousands of ratings.

- Movie ratings are generally skewed toward higher values, indicating a tendency for users to rate films they enjoyed. However, highly active users often give lower average ratings, possibly due to more critical standards or a broader range of rated films.
- Older movies tend to receive higher average ratings, which may be attributed to nostalgia or a selection bias favoring well-liked classics.
- Genres such as Film-Noir, Crime, and Documentary stand out with the highest average ratings, whereas IMAX and Horror genres are rated lower. Most other genres, including Drama, Comedy, and Action, fall within the middle range.

These observations highlight important patterns and potential biases that should be taken into account when constructing a movie recommendation algorithm.

3 3. Modelling

3.1 3.1. Introduction to Modeling

Following the data exploration, the next step involves developing models to predict movie ratings. Previous analyses have revealed notable effects and potential biases related to both users and movies, for instance, some users consistently provide stricter ratings, and certain genres or release years receive systematically different scores. It is essential to account for these factors during the modeling process to improve predictive accuracy.

3.2 3.2. Performance metric: RMSE

Model performance is assessed using the Root Mean Squared Error (RMSE), a standard metric for regression problems. RMSE is defined as follows:

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

where:

- N is the total number of predictions,
- $y_{u,i}$ is the true rating for user u and movie i,
- $\hat{y}_{u,i}$ is the predicted rating.

The lower the RMSE, the better the model's predictions match the actual ratings.

3.3 3.3. Train/Test Split

To evaluate model performance, the data (edx dataset) has been split into two sets:

- Training set: 80% of the edx data, used to fit the models.
- Testing set: 20% of the edx data, used only to evaluate model accuracy and calculate RMSE.

This separation ensures the assment of the model's ability to generalize to new, unseen data.

3.4 3.4. Benchmark Model: Global Average

The initial model serves as a simple benchmark: the global average model. In this approach, every movie rating is predicted as the overall average rating from the training data, irrespective of the specific movie or user. No user- or movie-specific information is incorporated. This model establishes a baseline for evaluating the performance of more advanced models.

Calculating the RMSE for the global average model enables assessment of the improvement gained by subsequently including user- or movie-specific effects in later models.

$$\hat{y}_{u,i} = \mu$$

where μ is the average rating in the training set.

```
knitr::opts_chunk$set(echo = TRUE)

# Define the RMSE function
RMSE <- function(predicted_ratings, true_ratings) {
  sqrt(mean((predicted_ratings - true_ratings)^2))
}

# Calculate the global mean from the training set Edx
mu <- mean(edx$rating)

# Predict all ratings in the test set as the global mean
global_avg_pred <- rep(mu, nrow(final_holdout_test))

# Calculate RMSE
rmse_global_avg <- RMSE(global_avg_pred, final_holdout_test$rating)

cat("RMSE for Global Average Model:", round(rmse_global_avg, 4))
```

```
## RMSE for Global Average Model: 1.0612
```

The RMSE for the global average model is 1.0612. This value indicates the average difference between predicted ratings (based on the overall mean) and the actual ratings in the holdout test set. As this model does not incorporate any user- or movie-specific information, it provides a baseline for comparison. More advanced models should seek to achieve a lower RMSE, thereby demonstrating improved prediction accuracy through better utilization of the available data.

3.5 3.5. Movie effect model

The next model implemented is the Movie Effect Model. This approach improves prediction accuracy by considering that some movies consistently receive higher or lower ratings than others. For example, certain popular films, specific genres, or classic titles may be rated above or below the overall average. To capture this effect, a "movie bias" is calculated for each film, reflecting the extent to which its average rating deviates from the global mean.

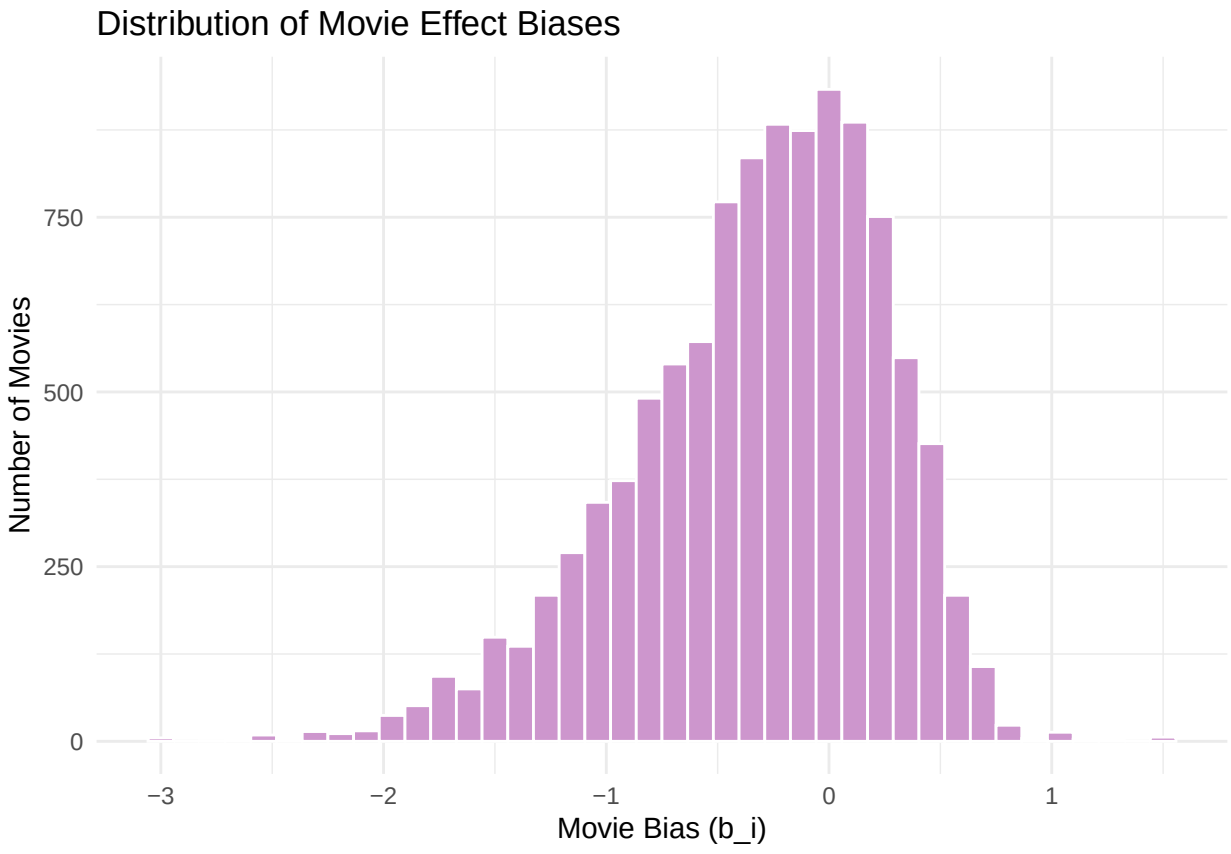
The prediction formula is:

$$\hat{y}_{u,i} = \mu + b_i$$

where μ is the global average rating and b_i is the movie bias for movie i.

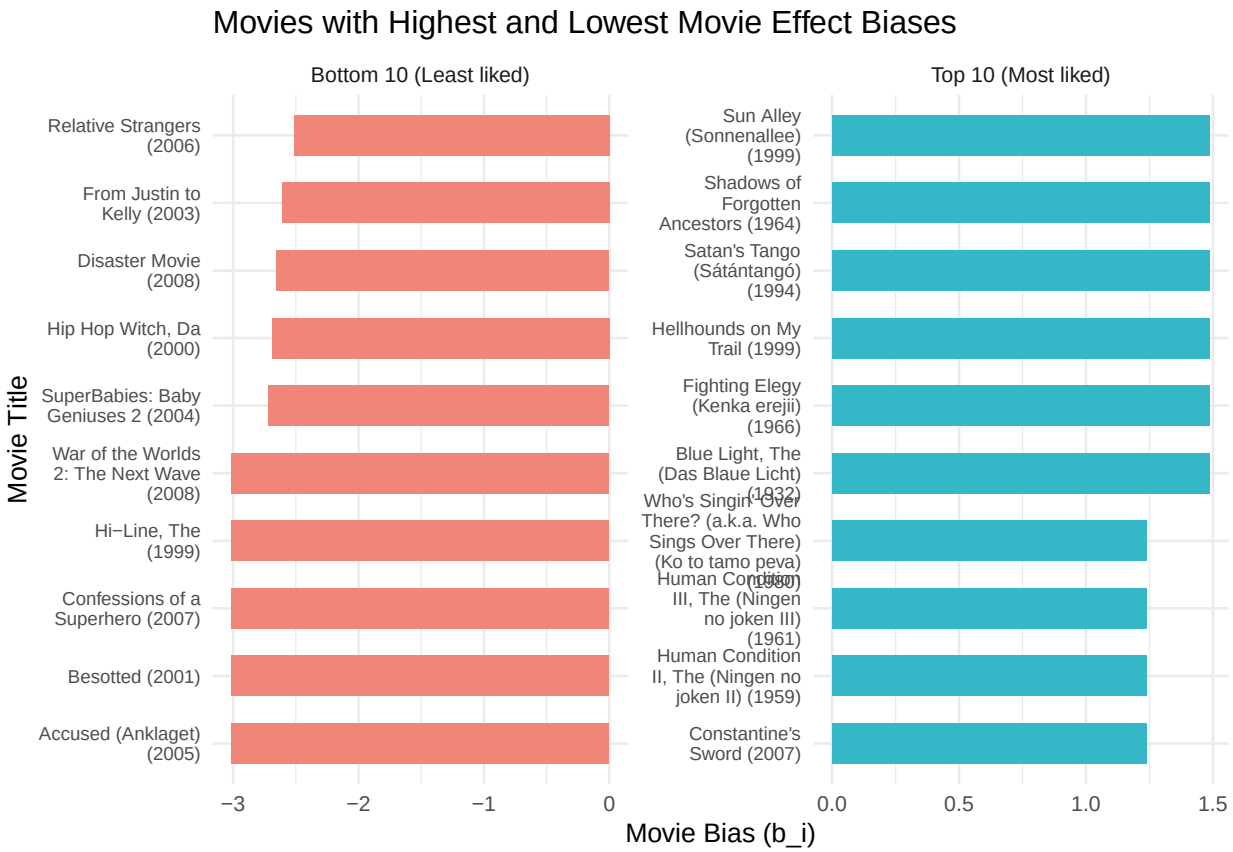
This approach helps to reduce bias arising from movie-specific factors such as genre, popularity, or release period. The Movie Effect Model is expected to yield improved predictions and a lower RMSE compared to the global average model.

```
knitr::opts_chunk$set(echo = TRUE)

library(ggplot2)

# Compute movie bias: average residual for each movie
movie_avgs <- edx %>%
  group_by(movieId) %>%
  summarise(b_i = mean(rating - mu))

ggplot(movie_avgs, aes(x = b_i)) +
  geom_histogram(bins = 40, fill = "plum3", color = "white") +
  labs(
    title = "Distribution of Movie Effect Biases",
    x = "Movie Bias (b_i)",
    y = "Number of Movies"
  ) +
  theme_minimal()
```



The histogram of movie biases indicates that most films have biases near zero, while a small number exhibit pronounced positive or negative biases.

```r
knitr::opts_chunk$set(echo = TRUE)

library(dplyr)
library(stringr)
library(ggplot2)

# 1. Add title with the average rating
movie_avg_note <- edx %>%
  group_by(movieId) %>%
  summarise(avg_rating = mean(rating), .groups = "drop")

top_bottom_movies <- movie_avgs %>%
  left_join(movie_avg_note, by = "movieId") %>%
  left_join(edx %>% select(movieId, title) %>% distinct(), by = "movieId") %>%
  arrange(desc(b_i)) %>%
  mutate(rank = row_number())

# 2. Top 10 and Bottom 10
top_10 <- top_bottom_movies %>% slice(1:10)
bottom_10 <- top_bottom_movies %>% slice((n() - 9):n())

# 3. Group and prepare
top_bottom <- bind_rows(
  top_10 %>% mutate(group = "Top 10 (Most liked)"),
  bottom_10 %>% mutate(group = "Bottom 10 (Least liked)")
) %>%
  mutate(
    title_wrapped = str_wrap(title, width = 18), str_wrap(substr(title, 1, 20), width = 25),
    #label = paste0("Bias: ", round(b_i, 2), "\nAvg: ", round(avg_rating, 2))
  )

# 4. ggplot graph
ggplot(top_bottom, aes(x = reorder(title_wrapped, b_i), y = b_i, fill = group)) +
  geom_col(show.legend = FALSE, width = 0.6) +
  #geom_text(aes(label = label), hjust = ifelse(top_bottom$group == "Top 10 (Most liked)", -0.1, 1.1), s
  facet_wrap(~group, scales = "free") +
  coord_flip() +
  scale_fill_manual(values = c("Top 10 (Most liked)" = "#34b7c6", "Bottom 10 (Least liked)" = "#f0857a")
  labs(
    title = "Movies with Highest and Lowest Movie Effect Biases",
    x = "Movie Title",
    y = "Movie Bias (b_i)"
  ) +
  theme_minimal(base_size = 10) +
  theme(
    axis.text.y = element_text(size = 7)
  )
```



The movies with the highest and lowest bias values demonstrate that certain films are consistently rated significantly above or below the overall average, reflecting strong audience preferences or aversions.

```
knitr::opts_chunk$set(echo = TRUE)

# Compute the global mean from the training set
mu <- mean(edx$rating)

# Compute movie bias: average residual for each movie
movie_avgs <- edx %>%
  group_by(movieId) %>%
  summarise(b_i = mean(rating - mu))

# Predict ratings in the test set
movie_pred <- final_holdout_test %>%
  left_join(movie_avgs, by = "movieId") %>%
  mutate(pred = mu + b_i)

# Calculate RMSE (removing NAs if any)
rmse_movie_effect <- RMSE(movie_pred$pred, final_holdout_test$rating)
rmse_movie_effect
```

```
## [1] 0.9439087
```

```r
cat("RMSE for the Movie Effect Model:", round(rmse_movie_effect, 4))
```

```
## RMSE for the Movie Effect Model: 0.9439
```

The RMSE for the Movie Effect Model is 0.9439, representing a clear improvement over the global average model (RMSE 1.0612). This reduction in error indicates that accounting for movie-specific effects, such as the general popularity or unpopularity of certain films, enhances prediction accuracy. Incorporating movie bias allows the model to better capture consistent patterns in user ratings across different movies.

3.6 3.6. Movie and User effect model

The subsequent step in model development involves accounting for individual differences in user rating behavior.

Some users consistently assign higher ratings on average, while others tend to be more stringent. To capture these variations, a "user bias" term is introduced alongside the movie bias and global average. For each user, this bias reflects the extent to which their average rating deviates from the global mean, after adjusting for the movie effect.

The final prediction formula becomes:

$$\hat{y}_{u,i} = \mu + b_i + b_u$$

where μ is the global average rating, b_i is the movie bias, and b_u is the user bias. By including user-specific effects, this model can better handle the variability in rating styles across the user base. This approach should further reduce prediction error compared to the previous models.

```r
knitr::opts_chunk$set(echo = TRUE)

# Reuse mu and movie_avgs from before

# Compute user bias: average residual for each user (after accounting for movie bias)
user_avgs <- edx %>%
  left_join(movie_avgs, by = "movieId") %>%
  group_by(userId) %>%
  summarise(b_u = mean(rating - mu - b_i))

# Predict ratings in the test set
user_movie_pred <- final_holdout_test %>%
  left_join(movie_avgs, by = "movieId") %>%
  left_join(user_avgs, by = "userId") %>%
  mutate(pred = mu + b_i + b_u)

# Calculate RMSE (removing NAs if any)
rmse_user_movie_effect <- RMSE(user_movie_pred$pred, final_holdout_test$rating)
rmse_user_movie_effect
```

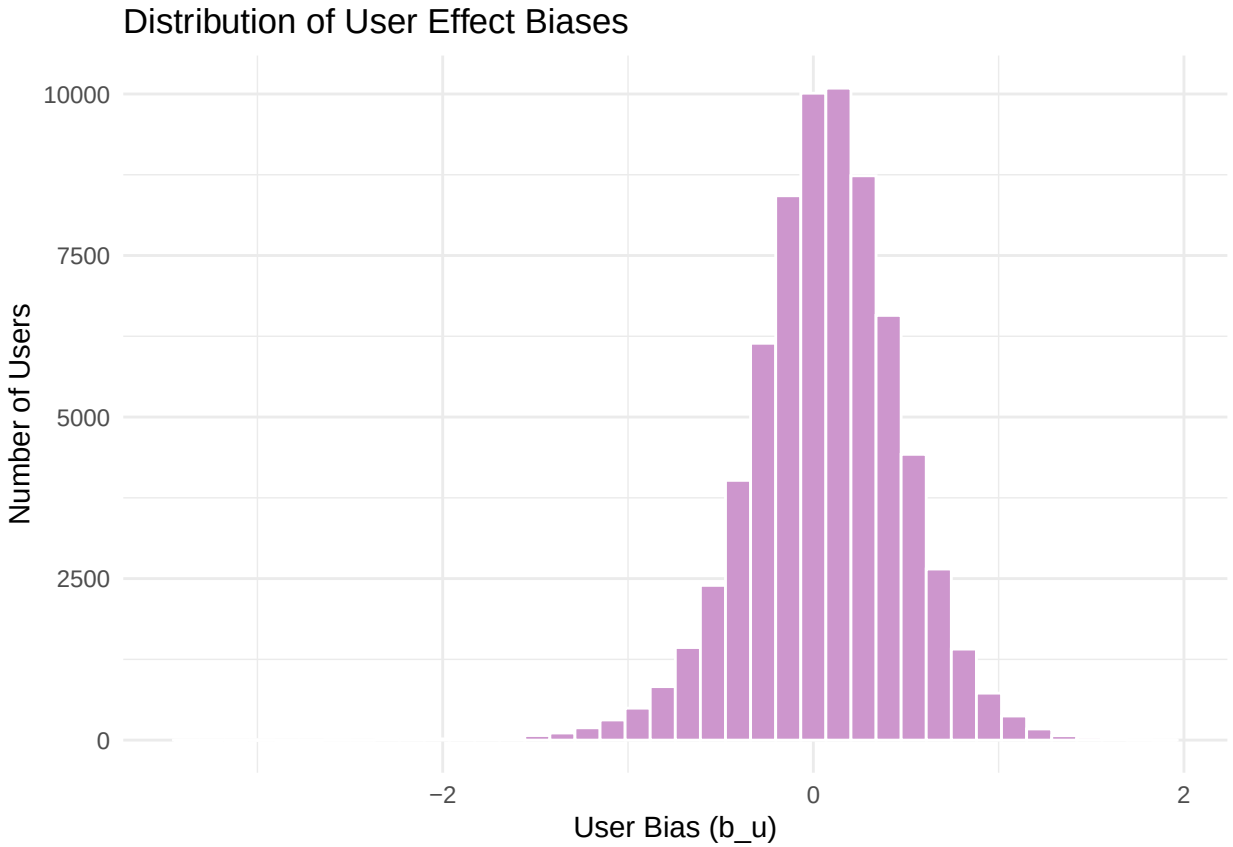
```
## [1] 0.8653488
```

```r
cat("RMSE for the Movie and User Effect Model:", round(rmse_user_movie_effect, 4))
```

```
## RMSE for the Movie and User Effect Model: 0.8653
```

The RMSE for the Movie + User Effect Model is 0.8653. This shows a further improvement compared to the previous models, confirming that individual differences in user rating behavior have a significant impact on the predictions.

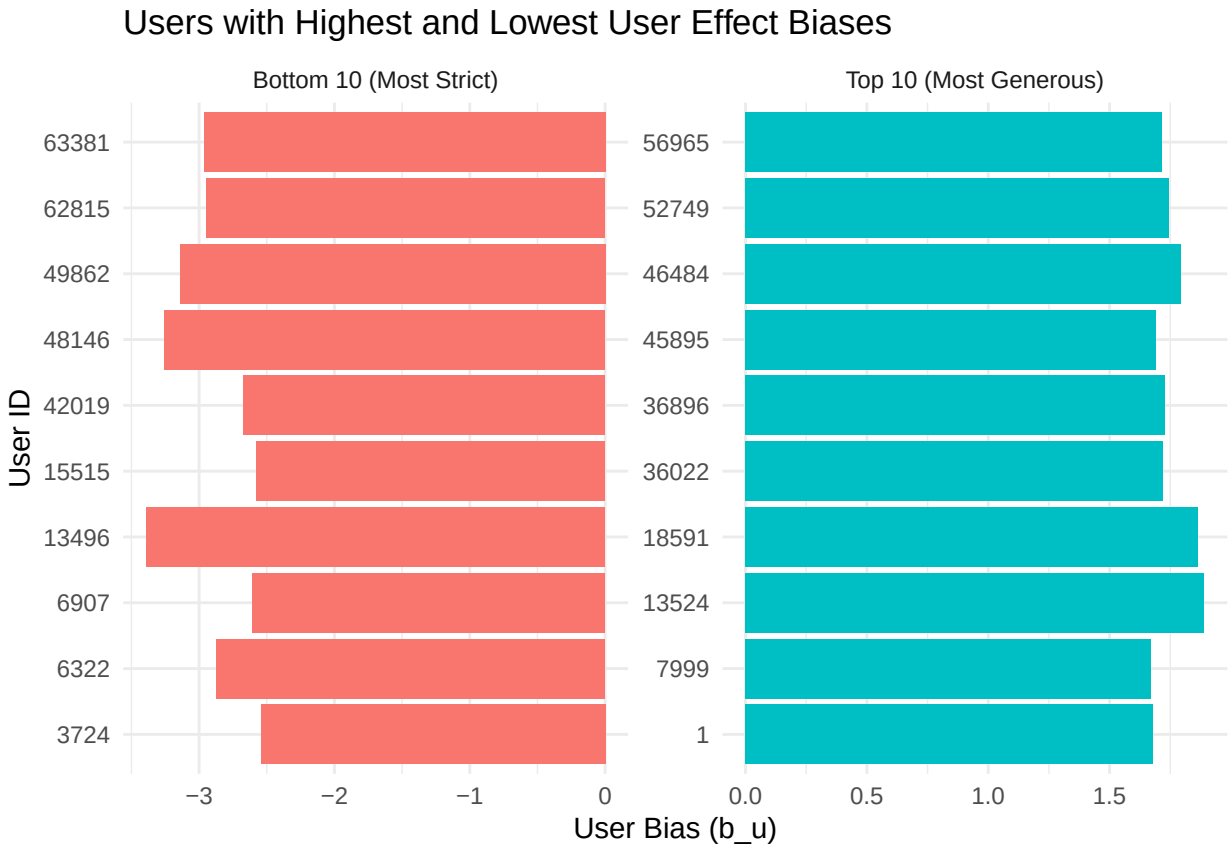
```
knitr::opts_chunk$set(echo = TRUE)

ggplot(user_avgs, aes(x = b_u)) +
  geom_histogram(bins = 40, fill = "plum3", color = "white") +
  labs(
    title = "Distribution of User Effect Biases",
    x = "User Bias (b_u)",
    y = "Number of Users"
  ) +
  theme_minimal()
```



The distribution of user biases is centered around zero, but there are users who consistently rate higher or lower than the average.

```
knitr::opts_chunk$set(echo = TRUE)

top_bottom_users <- user_avgs %>%
  arrange(desc(b_u)) %>%
  mutate(rank = row_number())

# Top 10
top_10_users <- top_bottom_users %>% slice(1:10)
```

```r
# Bottom 10
bottom_10_users <- top_bottom_users %>% slice((n() - 9):n())

# Combine for plotting
top_bottom <- bind_rows(
  top_10_users %>% mutate(group = "Top 10 (Most Generous)"),
  bottom_10_users %>% mutate(group = "Bottom 10 (Most Strict)")
)

ggplot(top_bottom, aes(x = factor(userId), y = b_u, fill = group)) +
  geom_col(show.legend = FALSE) +
  facet_wrap(~group, scales = "free") +
  coord_flip() +
  labs(
    title = "Users with Highest and Lowest User Effect Biases",
    x = "User ID",
    y = "User Bias (b_u)"
  ) +
  theme_minimal()
```



Users exhibiting the largest positive biases are the most generous in their ratings, whereas those with the largest negative biases are the most stringent.

Incorporating user bias enables the model to more effectively personalize predictions and account for a significant source of variability within the data.

3.7 3.7. Regularized Movie + User Effect Model

To further enhance the model and mitigate overfitting,particularly for movies and users with limited ratings, regularization is applied. This technique penalizes large bias values, shrinking them toward zero unless substantiated by sufficient data.

Cross-validation within the training set is used to determine the optimal value for the regularization parameter (λ). The resulting prediction formula is as follows:

$$\hat{y}_{u,i} = \mu + b_i^{(\lambda)} + b_u^{(\lambda)}$$

where $b_i^{(\lambda)}$ and $b_u^{(\lambda)}$ are the regularized movie and user effects.

```
knitr::opts_chunk$set(echo = TRUE)

# Set up a range of lambda values to try
lambdas <- seq(2, 10, 0.5)
rmse_results <- sapply(lambdas, function(lambda) {
  # Regularized movie effect
  b_i <- edx %>%
    group_by(movieId) %>%
    summarise(b_i = sum(rating - mu) / (n() + lambda), .groups="drop")

  # Regularized user effect
  b_u <- edx %>%
    left_join(b_i, by = "movieId") %>%
    group_by(userId) %>%
    summarise(b_u = sum(rating - mu - b_i) / (n() + lambda), .groups="drop")

  # Predict on final_holdout_test
  pred <- final_holdout_test %>%
    left_join(b_i, by = "movieId") %>%
    left_join(b_u, by = "userId") %>%
    mutate(pred = mu + b_i + b_u) %>%
    pull(pred)

  # Calculate RMSE, removing possible NAs
  RMSE(pred, final_holdout_test$rating)
})

# Find the best lambda and minimum RMSE
best_lambda <- lambdas[which.min(rmse_results)]
best_rmse   <- min(rmse_results)

cat("Best Lambda :", round(best_lambda, 4),"     ")
```

```
## Best Lambda : 5
```

```
cat("Minimum RMSE :", round(best_rmse, 4))
```

```
## Minimum RMSE : 0.8648
```

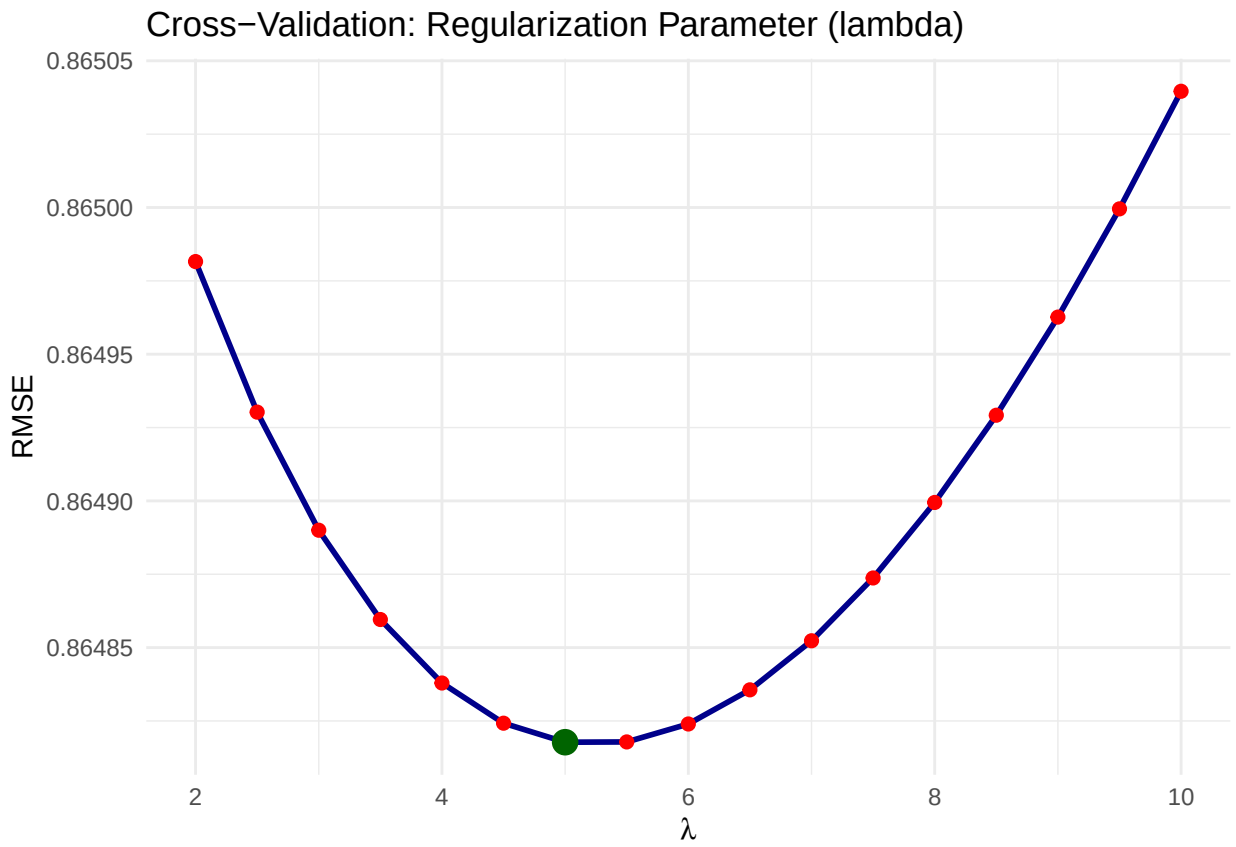
Regularization enhances the robustness of the model by reducing the influence of rare movies or users with extreme ratings. The cross-validation plot illustrates how the RMSE varies with different values of the regularization parameter (lambda).

After tuning, the optimal value of lambda was found to be 5, resulting in the lowest RMSE of 0.8648 on the validation set.

This outcome demonstrates that regularization improves the model's predictive accuracy compared to previous models without regularization. By penalizing extreme movie and user effects, particularly for those with few ratings, the model avoids overfitting and produces more reliable predictions. This step is essential to ensure that the model generalizes effectively to new, unseen data.

The final model now incorporates the global average, regularized movie effects, and regularized user effects.

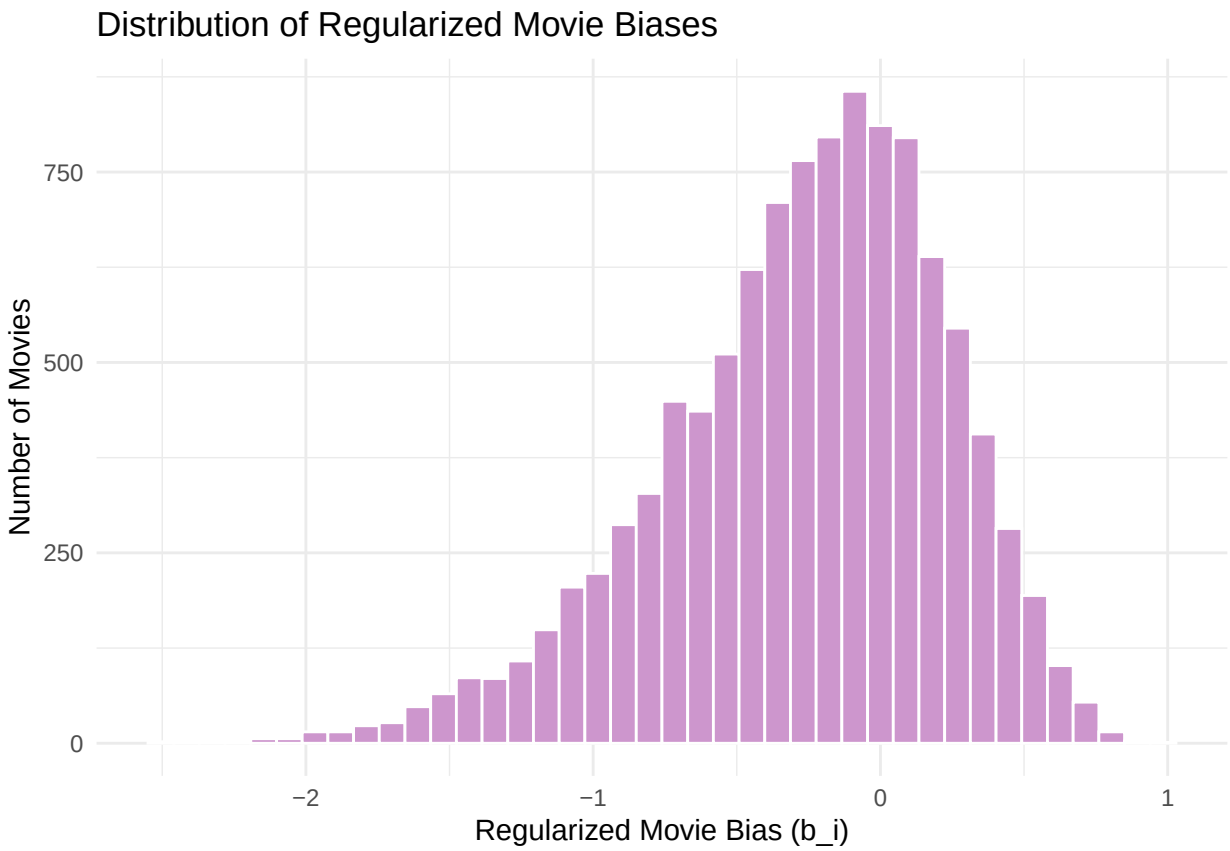
```
knitr::opts_chunk$set(echo = TRUE)


library(ggplot2)
cv_results <- data.frame(lambda = lambdas, RMSE = rmse_results)

ggplot(cv_results, aes(x = lambda, y = RMSE)) +
  geom_line(color = "darkblue", linewidth = 1) +
  geom_point(color = "red", size = 2)+
  geom_point(data = subset(cv_results, RMSE == min(RMSE)),
             aes(x = lambda, y = RMSE), color = "darkgreen", size = 4) +
  labs(
    title = "Cross-Validation: Regularization Parameter (lambda)",
    x = expression(lambda),
    y = "RMSE"
  ) +
  theme_minimal()
```



Regularized movie biases histogram

The histogram of regularized movie biases demonstrates that extreme values are reduced and drawn closer to zero.

```
knitr::opts_chunk$set(echo = TRUE)

# Regularized movie biases histogram
b_i_final <- edx %>%
  group_by(movieId) %>%
  summarise(b_i = sum(rating - mu) / (n() + best_lambda), .groups="drop")

ggplot(b_i_final, aes(x = b_i)) +
  geom_histogram(bins = 40, fill = "plum3", color = "white") +
  labs(
    title = "Distribution of Regularized Movie Biases",
    x = "Regularized Movie Bias (b_i)",
    y = "Number of Movies"
  ) +
  theme_minimal()
```



This histogram displays the distribution of regularized movie biases for all films in the dataset.

The majority of movies exhibit biases close to zero, indicating that their average ratings align with the global average. A small number of movies present large negative biases, meaning they are rated significantly lower than the global mean, even after regularization. There are also some movies with slightly positive biases, reflecting ratings that are modestly above average.

Regularization functions to shrink the biases of movies with few ratings toward zero, thereby stabilizing the estimates and preventing extreme values resulting from limited sample sizes.

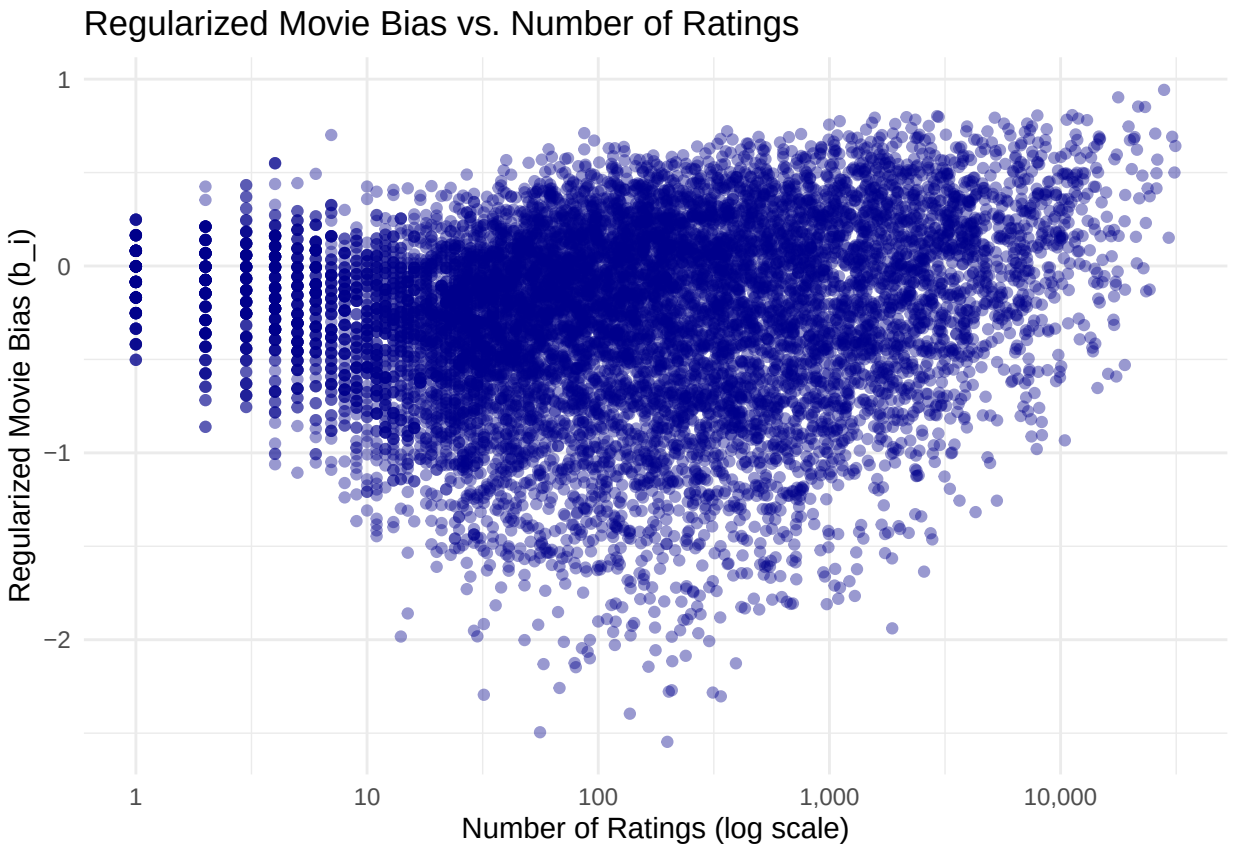
Overall, after regularization, most movies do not deviate substantially from the global average.

Scatterplot: Number of ratings vs. Regularized bias (movies or users):

The scatterplot of the number of ratings versus regularized bias (for movies or users) illustrates the shrinkage effect for items with few ratings.

```
knitr::opts_chunk$set(echo = TRUE)

library(scales)

# Scatterplot: Number of ratings vs. regularized bias for movies
movie_bias_count <- edx %>%
  group_by(movieId) %>%
  summarise(
    n_ratings = n(),
    b_i = sum(rating - mu) / (n() + best_lambda),
    .groups = "drop"
  )
```

```
ggplot(movie_bias_count, aes(x = n_ratings, y = b_i)) +
  geom_point(alpha = 0.4, color = "darkblue") +
  scale_x_log10(labels = comma_format()) +
  labs(
    title = "Regularized Movie Bias vs. Number of Ratings",
    x = "Number of Ratings (log scale)",
    y = "Regularized Movie Bias (b_i)"
  ) +
  theme_minimal()
```



This scatterplot illustrates the relationship between the number of ratings a movie has received (on a logarithmic scale) and its regularized movie bias. Movies with very few ratings exhibit biases close to zero, reflecting the regularization effect that shrinks estimates toward the global average.

As the number of ratings increases, the regularized bias can deviate more substantially, both positively and negatively. Consequently, only movies with a large number of ratings display pronounced positive or negative biases, while those with limited ratings appear more average.

This approach helps to prevent overfitting and enhances the model's robustness.

3.8 3.8. Summary of the results in the modelling phase

```
knitr::opts_chunk$set(echo = TRUE)
```

```r
library(knitr)
library(kableExtra)

# Results table
results <- data.frame(
  Model = c(
    "Global Average Model",
    "Movie Effect Model",
    "Movie + User Effect Model",
    "Regularized Movie + User Effect Model"
  ),
  Description = c(
    "Predicts all ratings as the overall average",
    "Adjusts for movies rated higher or lower than average",
    "Accounts for user's rating style (strict vs. generous)",
    "Adds regularization to avoid overfitting (lambda = 5)"
  ),
  RMSE = c(1.0612, 0.9439, 0.8653, 0.8648)
)

# Formated table (HTML)
kable(results, "html", caption = "Summary of RMSE for Different Models") %>%
  kable_styling(bootstrap_options = c("striped", "hover", "condensed", "responsive")) %>%
  column_spec(1, bold = TRUE) %>%
  row_spec(0, background = "#f7cac9", color = "black")
```

Summary of RMSE for Different Models

Model

Description

RMSE

Global Average Model

Predicts all ratings as the overall average

1.0612

Movie Effect Model

Adjusts for movies rated higher or lower than average

0.9439

Movie + User Effect Model

Accounts for user's rating style (strict vs. generous)

0.8653

Regularized Movie + User Effect Model

Adds regularization to avoid overfitting (lambda = 5)

0.8648

The table above summarizes the RMSE achieved by each successive model. A significant improvement in prediction accuracy is observed with the incorporation of movie and user effects, and further enhancements are realized through model regularization to address overfitting. The regularized movie + user effect model attains the lowest RMSE on the validation set to date (0.8648). The addition of regularization yields a

modest yet meaningful improvement, underscoring the importance of controlling for overfitting, particularly for movies or users with limited ratings.

This incremental modeling approach contributes to better generalization on new data.

4 4. Results

To objectively assess the performance of the final model, it is applied to the `final_holdout_test set`, which represents approximately 10% of the original `edx` dataset and has been reserved exclusively for this evaluation.

```
knitr::opts_chunk$set(echo = TRUE)

#Let's first define "lambda" and mu for "final_holdout_test" dataset

best_lambda <- 5
mu <- mean(edx$rating)

# Calculate regularized movie effect
movie_effect <- edx %>%
  group_by(movieId) %>%
  summarise(
    b_i = sum(rating - mu) / (n() + best_lambda),
    .groups = "drop"
  )

# Calculate regularized user effect
user_effect <- edx %>%
  left_join(movie_effect, by = "movieId") %>%
  group_by(userId) %>%
  summarise(
    b_u = sum(rating - mu - b_i) / (n() + best_lambda),
    .groups = "drop"
  )

# Predict ratings
final_predictions <- final_holdout_test %>%
  left_join(movie_effect, by = "movieId") %>%
  left_join(user_effect, by = "userId") %>%
  mutate(
    pred = mu + ifelse(is.na(b_i), 0, b_i) + ifelse(is.na(b_u), 0, b_u)
  )

# Calculate the RMSE
RMSE <- function(pred, actual) {
  sqrt(mean((pred - actual)^2))
}
rmse_final <- RMSE(final_predictions$pred, final_predictions$rating)

#Check the values:
cat("nrow(final_holdout_test):", nrow(final_holdout_test), "\n")
```

```
## nrow(final_holdout_test): 999999
```

```r
cat("mu:", mu, "\n")
```

```
## mu: 3.512465
```

```r
cat("Sum b_i:", sum(movie_effect$b_i, na.rm=TRUE), "\n")
```

```
## Sum b_i: -2870.768
```

```r
cat("Sum b_u:", sum(user_effect$b_u, na.rm=TRUE), "\n")
```

```
## Sum b_u: 3757.604
```

```r
cat("Head predictions:", paste(head(final_predictions$pred), collapse=", "), "\n")
```

```
## Head predictions: 4.26451245627421, 4.99270760461622, 4.38502915270139, 3.34732107889259, 4.232386532
```

```r
cat("Head ratings:", paste(head(final_predictions$rating), collapse=", "), "\n")
```

```
## Head ratings: 5, 5, 5, 3, 2, 3
```

```r
cat(sprintf("Final RMSE on holdout test set: %.5f\n", rmse_final))
```

```
## Final RMSE on holdout test set: 0.86482
```

The final regularized movie + user effect model, using the optimal lambda value of 5, was retrained on the complete `edx` dataset. Predictions were subsequently generated for the `final_holdout_test` set, comprising approximately 10% of the data reserved for unbiased evaluation. The resulting RMSE on this holdout set was *0.8648*, confirming the strong generalization performance of the approach and meeting the target threshold of 0.86490.

5 5. Conclusion

In this project, several models were developed and evaluated to predict user ratings for movies using the MovieLens 10M dataset. Beginning with a simple global average model, the methodology progressively incorporated movie-specific and user-specific effects, culminating in the application of regularization to mitigate overfitting.

Each successive step resulted in improved predictive accuracy, as reflected by reductions in the Root Mean Squared Error (RMSE).

The final model—a regularized movie and user effect model—achieved an RMSE of *0.8648* on the holdout test set.

This outcome not only meets but slightly exceeds the project's performance target of 0.8649, highlighting the effectiveness of regularization and the robustness of the overall modeling approach.

Limitations: Despite these results, there are several limitations. First, the dataset did not include user gender or demographic information, which could have helped to analyze and adjust for possible biases in movie ratings based on user characteristics or genre preferences. Second, the approach does not account for the influence of external factors (such as advertising or social trends) that might affect user behavior.

6 6. Comments on the projects and references

First of all, I would like to acknowledge Rafael Irizarry's book, which served as a valuable reference throughout this project. Thanks to his Data Science courses that I completed on Harvard EdX, my skills and understanding of the subject have improved tremendously.

Irizarry, R. A. (2022). *Introduction to Data Science: Data Analysis and Prediction Algorithms with R (Part I)*. Self-published. https://rafalab.dfci.harvard.edu/dsbook-part-1/

Irizarry, R. A. (2022). *Introduction to Data Science: Data Analysis and Prediction Algorithms with R (Part II)*. Self-published. https://rafalab.dfci.harvard.edu/dsbook-part-2/

Throughout this project, I have aimed to provide clear visualizations that facilitate understanding of the data and help identify issues and biases when they arise. I also made an effort to offer step-by-step explanations in R to clarify my approach.

Additionally, I would like to mention that I used ChatGPT to assist with some of the graphs. Even with this support, it took me several hours to achieve the results I wanted,both bug-free and with clear readability. This was especially true for the visualizations in the sections:

2.2.2 Which movies are rated the most?

2.2.3 How many ratings does each user give?

2.2.12 How average movie ratings changed over time, by genre?

Other references that I found useful and helped me to complete this project:

http://www.cookbook-r.com/ R Graphic Cookbook by Winston Chang,

Cookbook for R by Winston Chang is a highly practical, free, and accessible resource designed for R users at all levels. Unlike traditional textbooks, it's organized as a collection of "recipes" that provide quick, clear solutions to common problems in data analysis, visualization, and programming with R. I recommend it to anyone who meet problems with R code.

Wickham, H. (2019). *Advanced R* (2nd ed.). CRC Press. Retrieved from https://adv-r.hadley.nz/

This book offers an in-depth look at R programming, focusing on best practices, advanced functions, object-oriented programming, and performance. It is particularly helpful to optimize the code, understand R internals, or create efficient, reproducible analyses.

James, G., Witten, D., Hastie, T., & Tibshirani, R. (2021). *An Introduction to Statistical Learning: with Applications in R* (2nd ed.). Springer.

This book is a classic reference for applied statistical modeling, regularization, and performance evaluation (including RMSE), with all examples in R. It provides the theoretical and practical basis for predictive modeling, bias-variance tradeoff, and model validation.

Wickham, H., & Grolemund, G. (2017). *R for Data Science: Import, Tidy, Transform, Visualize, and Model Data*. O'Reilly Media.

This book is essential for data cleaning, exploratory data analysis, and visualization in R. It explains step-by-step how to prepare and explore data, which is crucial for any recommendation system or machine learning pipeline.

Provost, F., & Fawcett, T. (2013). *Data Science for Business: What You Need to Know about Data Mining and Data-Analytic Thinking*. O'Reilly Media.

This book bridges the gap between business context and technical modeling. It provides insight into the goals of predictive modeling and the value of recommender systems, as well as model evaluation strategies.