Stats 15 Final Project - TMDB Film Rating Analysis

Rohan Shah, Israel Mandujano, Felicia Deshon, Nayan Petrime

2025-06-08

Section 1 - Background Information

1.1 - Motivating Questions and Scope of Analysis

Movies are one of the most popular pastimes in the world. Whether you drive down to the cinema to watch the latest release, or use Netflix from the comfort of your own home, chances are you have a feeling or opinion about the film that you've watched. TMDB is a website that stores movie information, and allows people to rate any movie that they've watched. As a team of movie-loving people, we wanted to ask the question: **what affects the rating of a movie?**

1.2 - Background on TMDB

Overall Info: TMDB (otherwise known as The Movie Database) is a movie database that contains detailed information about films in 50+ languages, as well as provides users an opportunity to rate the quality of films that they choose. Despite their similar name to IMDB, TMDB doesn't use data from the former site, although ratings can be (and often are) similar. The reason we used TMDB is because the site is specifically designed to be great for data collection. A key feature of TMDB is its API which allows people to easily take their stored movie data and use it for data analysis, as we are doing currently!

Ratings Details: TMDB allows registered users to rate movies on a scale from 0.5 to 10.0 in 0.5-point increments. These ratings reflect the user's personal opinion and are combined across all users to calculate the *vote_average*. Unlike some platforms, TMDB does not apply weighted averages or critic adjustments, every user's vote carries equal weight.

Information Details: TMDB collects its data through a combination of user contributions and official studio sources. Registered users around the world add and update movie details like cast, crew, genres, and runtime, while studios provide verified data such as trailers, budgets, and box office revenue. All entries are reviewed by moderators or voted on by the community to ensure accuracy.

API Details: TMDB offers a free and simple way for users to access its movie data through an API. After signing up and requesting an API key, users can search for films, get cast and rating info, and download data for analysis. This tool is especially helpful for developers, researchers, or anyone working on movie-related projects.

Dataset Details: The movies in our database were selected based on TMDB's internal popularity score, which ranks films according to real-time user engagement. This metric is not the same as average rating or total votes, but instead incorporates multiple signals of user interest.

How This Is Accomplished: First, a request is made to the TMDB API to retrieve movies, which are then sorted in descending order of a popularity score. This popularity metric is a proprietary measure that includes factors such as page views, searches, frequency of user interactions (rating submissions, watchlist additions), and recency of engagement (recent movies or trending titles tend to be weighted more heavily).

From this ranked list, the top 10,000 movies at the time of collection were selected, without filtering by genre, time period, or language.

For each movie, metadata is then pulled from the API to be compiled into the dataset, including *title, vote average, vote count, release date, original language, genre, runtime*, and more. Factors such as critical acclaim, box office success, historical or cultural importance, and completeness of data were not accounted for in selection.

Due to this approach, the dataset is biased towards more recent and popularly engaged films. Older or niche movies with fewer interactions are underrepresented, and those that are included are likely classics or cult favorites that maintain high ratings despite fewer votes

1.3 - Variable Explanation

Explanatory Variables

- *release_date*: (character) The date when the movie was first released in theaters or on streaming. Format is “YYYY-MM-DD”.
- *genres*: (character) The genre(s) assigned to a movie, such as Action, Comedy, Drama, etc. A movie may belong to multiple genres.
- *vote_count*: (integer) The total number of users who have rated the movie on TMDB. A higher vote count generally suggests more visibility or popularity.
- *budget*: (integer) The estimated production budget for the movie in US dollars. This is often used to measure investment level or compare with - revenue to calculate profitability.
- *revenue*: (double) The global box office earnings for the movie, in US dollars.
- *runtime*: (integer) The length of the movie in minutes.

Response Variable

- *vote_average*: (double) The average rating the movie has received from TMDB users, on a scale from 0 to 10. This serves as the main response variable for predicting perceived movie quality. You may see us refer to this as a movie’s “rating” in this report.

Section 2 - Exploratory Data Analysis

2.1 - Loading and Preliminary Cleaning of Data

Loading our data

First, we load our necessary libraries and our dataset. We also select the columns that we plan to use for our analysis.

```r
library(tidyverse)
library(dplyr)
library(ggplot2)

movie_data <- read.csv("movies_data.csv")

movie_data <- movie_data %>%
  select(title, release_date, genres, vote_average,
         vote_count, budget, revenue, runtime)

glimpse(movie_data)
```

```
## Rows: 10,000
## Columns: 8
## $ title        <chr> "The Pope's Exorcist", "Ant-Man and the Wasp: Quantumania~
## $ release_date <chr> "2023-04-05", "2023-02-15", "2023-04-05", "2023-04-18", "~
## $ genres       <chr> "['Horror', 'Mystery', 'Thriller']", "['Action', 'Adventu~
## $ vote_average <dbl> 7.4, 6.6, 7.5, 7.2, 6.8, 7.7, 8.3, 7.3, 7.3, 7.5, 7.3, 5.~
## $ vote_count   <int> 619, 2294, 1861, 652, 1510, 7853, 683, 1029, 1298, 964, 3~
## $ budget       <int> 18000000, 200000000, 100000000, 0, 125000000, 460000000, ~
## $ revenue      <dbl> 65675816, 464566092, 1121048165, 0, 133437105, 2319331580~
## $ runtime      <int> 103, 125, 92, 120, 130, 192, 150, 123, 116, 134, 111, 106~
```

Votes Cleanup

We decided to drop movies that have less than 100 user votes to ensure rating averages are statistically meaningful and not skewed by small sample sizes. Including these movies with less activity could cause our analysis to be unrepresentative of real voting patterns. After this cleaning, the total number of movies went from 10,000 to 7,776, meaning this step removed 2,224 or ~22% of our data.

```r
movie_clean <- movie_data %>%
  filter(vote_count >= 100)

nrow(movie_clean)
```

```
## [1] 7776
```

Missing Values Cleanup

When we visually inspected our data, we noticed that there were missing values represented as a 0, but didn't see any NA values. To investigate this, we check for any NA values and we calculate the number of times a value of 0 appears for each column in the dataset.

```r
any(is.na(movie_clean))
```

```
## [1] FALSE
```

```r
print(colSums(movie_clean == 0))
```

```
##        title release_date       genres vote_average   vote_count       budget
##            0            0            0            0            0         2809
##      revenue      runtime
##         2504            3
```

We found that the budget and revenue columns contained a substantial amount of zeros—2,809 and 2,504 respectively. Additionally, the runtime column contained 3 zeros. At first, when we removed any column containing a zero value, it cut our data by a huge margin. We were left with 4,435 films, less than half of the 10,000 films we started with. Since the zero values were concentrated in the budget, revenue, and runtime variables, we decided to keep two versions of the data instead of removing a large amount of our dataset for the entire project. For any analysis that involves variables with significant missing values, we use data saved as *movie_small*, but for anything else, we maintain the *movie_clean* dataset, allowing us to use as much data as possible for analysis that does not depend on the columns with missing values.

```
movie_small <- movie_clean[!apply(movie_clean == 0, 1, any), ]

nrow(movie_small)
```

```
## [1] 4435
```

2.2 - Variable-Specific Data Preparation

Release Year

In order to make analysis of release date more convenient and informative, we create a new column named *release_year* which is just the year of each movie.

```
movie_clean %>%
  select(release_date) %>%
  head()
```

```
##   release_date
## 1   2023-04-05
## 2   2023-02-15
## 3   2023-04-05
## 4   2023-04-18
## 5   2023-03-15
## 6   2022-12-14
```

```
movie_clean$release_date <- ymd(movie_clean$release_date)
movie_clean$release_year <- year(movie_clean$release_date)

movie_small$release_date <- ymd(movie_small$release_date)
movie_small$release_year <- year(movie_small$release_date)

movie_clean %>%
  select(release_year) %>%
  head()
```

```
##   release_year
## 1         2023
## 2         2023
## 3         2023
## 4         2023
## 5         2023
## 6         2022
```

Genres

Within our dataset, each movie is associated with a list of one or more genres stored as a single string. To make this information easier to work with while preserving the multi-genre diversity of the data, we create logical indicator variables for each genre. First, we clean the string genres by converting them to lowercase and removing brackets and quotation marks. Next, we check all unique genres values so we know how many indicator variables to create.

```r
# Convert genre strings to lowercase and remove brackets and quotes
movie_clean <- movie_clean %>%
  mutate(genres = str_to_lower(genres),
         genres = str_remove_all(genres, "\\[|\\]|'|\""))

# Check all unique genre values
all_genres <- movie_clean %>%
  separate_rows(genres, sep = ",\\s*") %>%
  distinct(genres) %>%
  arrange(genres) %>%
  print()
```

```
## # A tibble: 19 x 1
##    genres
##    <chr>
##  1 action
##  2 adventure
##  3 animation
##  4 comedy
##  5 crime
##  6 documentary
##  7 drama
##  8 family
##  9 fantasy
## 10 history
## 11 horror
## 12 music
## 13 mystery
## 14 romance
## 15 science fiction
## 16 thriller
## 17 tv movie
## 18 war
## 19 western
```

Now, we generate a list of logical indicator variables for genres. Each new column corresponds to a specific genre and represents whether a genre is present in a given movies' genre list or not. Finally, we reshape the data from a wide format in a new dataset, *movie_genres*, with one column per genre to a long format where each row represents a single movie-genre pairing. We then filter this long format data to include only genres that are present for each movie. This final structure allows us to view and analyze the full set of genres associated with each movie.

```r
# Create genre indicator columns
movie_clean <- movie_clean %>%
  mutate(
    is_action       = str_detect(genres, "action"),
    is_adventure    = str_detect(genres, "adventure"),
    is_animation    = str_detect(genres, "animation"),
    is_comedy       = str_detect(genres, "comedy"),
    is_crime        = str_detect(genres, "crime"),
    is_drama        = str_detect(genres, "drama"),
    is_family       = str_detect(genres, "family"),
    is_fantasy      = str_detect(genres, "fantasy"),
    is_horror       = str_detect(genres, "horror"),
    is_romance      = str_detect(genres, "romance"),
    is_scifi        = str_detect(genres, "science fiction"),
    is_thriller     = str_detect(genres, "thriller"),
    is_mystery      = str_detect(genres, "mystery"),
    is_war          = str_detect(genres, "war"),
    is_documentary  = str_detect(genres, "documentary"),
    is_western      = str_detect(genres, "western"),
    is_music        = str_detect(genres, "music"),
    is_history      = str_detect(genres, "history"),
    is_tv_movie     = str_detect(genres, "tv movie")
  )

movie_small <- movie_small %>%
  mutate(
    is_action       = str_detect(genres, "action"),
    is_adventure    = str_detect(genres, "adventure"),
    is_animation    = str_detect(genres, "animation"),
    is_comedy       = str_detect(genres, "comedy"),
    is_crime        = str_detect(genres, "crime"),
    is_drama        = str_detect(genres, "drama"),
    is_family       = str_detect(genres, "family"),
    is_fantasy      = str_detect(genres, "fantasy"),
    is_horror       = str_detect(genres, "horror"),
    is_romance      = str_detect(genres, "romance"),
    is_scifi        = str_detect(genres, "science fiction"),
    is_thriller     = str_detect(genres, "thriller"),
    is_mystery      = str_detect(genres, "mystery"),
    is_war          = str_detect(genres, "war"),
    is_documentary  = str_detect(genres, "documentary"),
    is_western      = str_detect(genres, "western"),
    is_music        = str_detect(genres, "music"),
    is_history      = str_detect(genres, "history"),
    is_tv_movie     = str_detect(genres, "tv movie")
  )

# Reshape wide to long so that each genre becomes a column
movie_genres <- movie_clean %>%
  pivot_longer(
    cols = starts_with("is_"),
    names_to = "genre",
    values_to = "has_genre"
  ) %>%
  filter(has_genre == TRUE) %>%
  mutate(genre = str_remove(genre, "is_"))

# Display first 20 entries in new data table
movie_genres %>%
  select(title, genres, genre, has_genre) %>%
  head(20)
```

```
## # A tibble: 20 x 4
##    title                            genres                           genre has_genre
##    <chr>                             <chr>                         <chr> <lgl>
##  1 The Pope’s Exorcist               horror, mystery, thriller     horr~ TRUE
##  2 The Pope’s Exorcist               horror, mystery, thriller     thri~ TRUE
##  3 The Pope’s Exorcist               horror, mystery, thriller     myst~ TRUE
##  4 Ant-Man and the Wasp: Quantumania action, adventure, science~ acti~ TRUE
##  5 Ant-Man and the Wasp: Quantumania action, adventure, science~ adve~ TRUE
##  6 Ant-Man and the Wasp: Quantumania action, adventure, science~ scifi TRUE
##  7 The Super Mario Bros. Movie       animation, adventure, fami~ adve~ TRUE
##  8 The Super Mario Bros. Movie       animation, adventure, fami~ anim~ TRUE
##  9 The Super Mario Bros. Movie       animation, adventure, fami~ come~ TRUE
## 10 The Super Mario Bros. Movie       animation, adventure, fami~ fami~ TRUE
## 11 The Super Mario Bros. Movie       animation, adventure, fami~ fant~ TRUE
## 12 Ghosted                           action, comedy, romance     acti~ TRUE
## 13 Ghosted                           action, comedy, romance     come~ TRUE
## 14 Ghosted                           action, comedy, romance     roma~ TRUE
## 15 Shazam! Fury of the Gods          action, comedy, fantasy, a~ acti~ TRUE
## 16 Shazam! Fury of the Gods          action, comedy, fantasy, a~ adve~ TRUE
## 17 Shazam! Fury of the Gods          action, comedy, fantasy, a~ come~ TRUE
## 18 Shazam! Fury of the Gods          action, comedy, fantasy, a~ fant~ TRUE
## 19 Avatar: The Way of Water          science fiction, adventure~ acti~ TRUE
## 20 Avatar: The Way of Water          science fiction, adventure~ adve~ TRUE
```

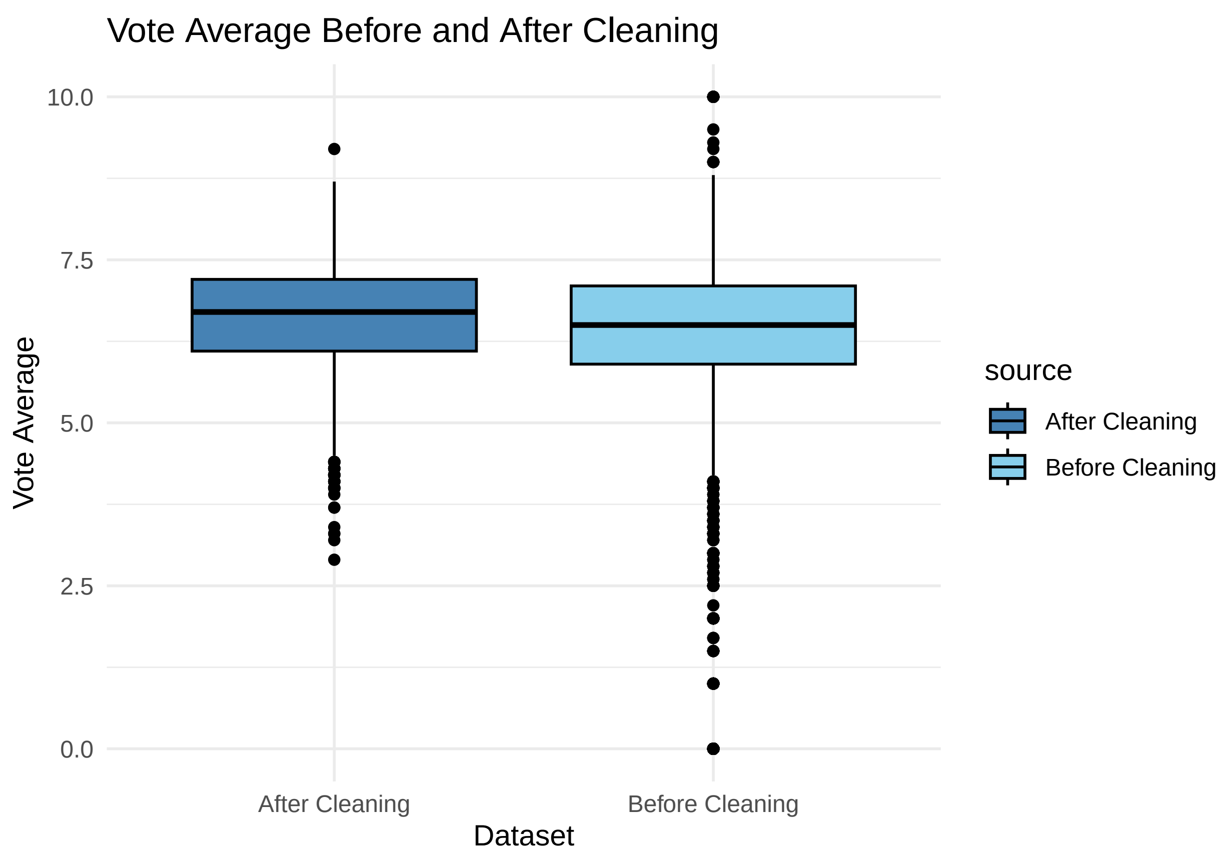
2.3 - Univariate Distributions

Now that we’ve completed the initial data cleaning, we explore the univariate distributions of individual variables in our dataset. This step helps us understand the characteristics of variables—such as shape, range, center, and frequency of values—before moving on to analyzing relationships between variables to answer our fundamental research question. As we inspect each variable, we also identify potential outliers or anomalies that could impact further analysis.

Vote Average

First, we plot side-by-side boxplots of the distributions of average user ratings across movies both before and after cleaning. We do this to examine the impact of filtering out movies with fewer than 100 votes.

```r
combined_votes <- bind_rows(
  movie_data %>%
    select(vote_average) %>%
    mutate(source = "Before Cleaning"),
  movie_clean %>%
    select(vote_average) %>%
    mutate(source = "After Cleaning")
)

ggplot(combined_votes, aes(x = source, y = vote_average, fill = source)) +
  geom_boxplot(color = "black") +
  labs(
    title = "Vote Average Before and After Cleaning",
    x = "Dataset",
    y = "Vote Average"
  ) +
  scale_fill_manual(values = c("Before Cleaning" = "skyblue",
                               "After Cleaning" = "steelblue")) +
  theme_minimal()
```



This visualization shows us that our filtering step removed many high and low end outliers from the data. This demonstrates how a small amount of votes can severely skew ratings data. By excluding these, we likely improved the quality of our analysis by minimizing the influence of rating distortions from small voter counts. To quickly summarize the distribution of *vote_average* in our cleaned dataset and investigate outliers, we create a five-number summary.

```r
summary(movie_clean$vote_average)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
##   2.900   6.100   6.700   6.641   7.200   9.200
```

Something interesting to note is that the maximum vote average is only 9.2; anything higher was removed in our cleaning step. To take a closer look at outliers, we create a data table listing all the movies with very high or very low ratings. Based on our five-number summary, we know that Q1 is 6.1 and Q3 is 7.2, so we can use these values to filter and find outliers. We display the head of both the top and bottom of the resulting filtered set to highlight some of the extreme examples on both ends of the rating spectrum.

```r
lower_bound <- 6.1 - 1.5 * (7.2-6.1)
upper_bound <- 7.2 + 1.5 * (7.2-6.1)

movie_clean %>%
  filter(vote_average < lower_bound | vote_average > upper_bound) %>%
  select(title, vote_average, vote_count) %>%
  arrange(desc(vote_average)) %>%
  head()
```

```
##                                     title vote_average vote_count
## 1 BTS: Permission to Dance on Stage - LA          9.2        141
## 2                          Fantastic Four          4.4       5525
## 3                             Slender Man          4.4       1772
## 4                                    Vice          4.4        497
## 5                            The Avengers          4.4        617
## 6                   2-Headed Shark Attack          4.4        238
```

```
movie_clean %>%
  filter(vote_average < lower_bound | vote_average > upper_bound) %>%
  select(title, vote_average, vote_count) %>%
  arrange(vote_average) %>%
  head()
```

```
##                                     title vote_average vote_count
## 1                    Dragonball Evolution          2.9       1851
## 2                       Battlefield Earth          3.2        712
## 3                       Alone in the Dark          3.3        522
## 4                          Disaster Movie          3.3        922
## 5            The Star Wars Holiday Special          3.4        403
## 6 The Human Centipede 3 (Final Sequence)          3.7        669
```

Inspecting outlier movies gives us insight into real examples within the data. Through this, we learn that there is only one high end outlier in our dataset—a concert film. Since people who watch concert films for musical artists are typically already interested in the artist, it makes sense why ratings would be higher for this kind of movie than most other films in the dataset. There are many more low end outliers; we discover there are a total of 57 low end outliers by counting how many films fall below the lower threshold.

```
movie_clean %>% filter(vote_average < lower_bound) %>% count()
```

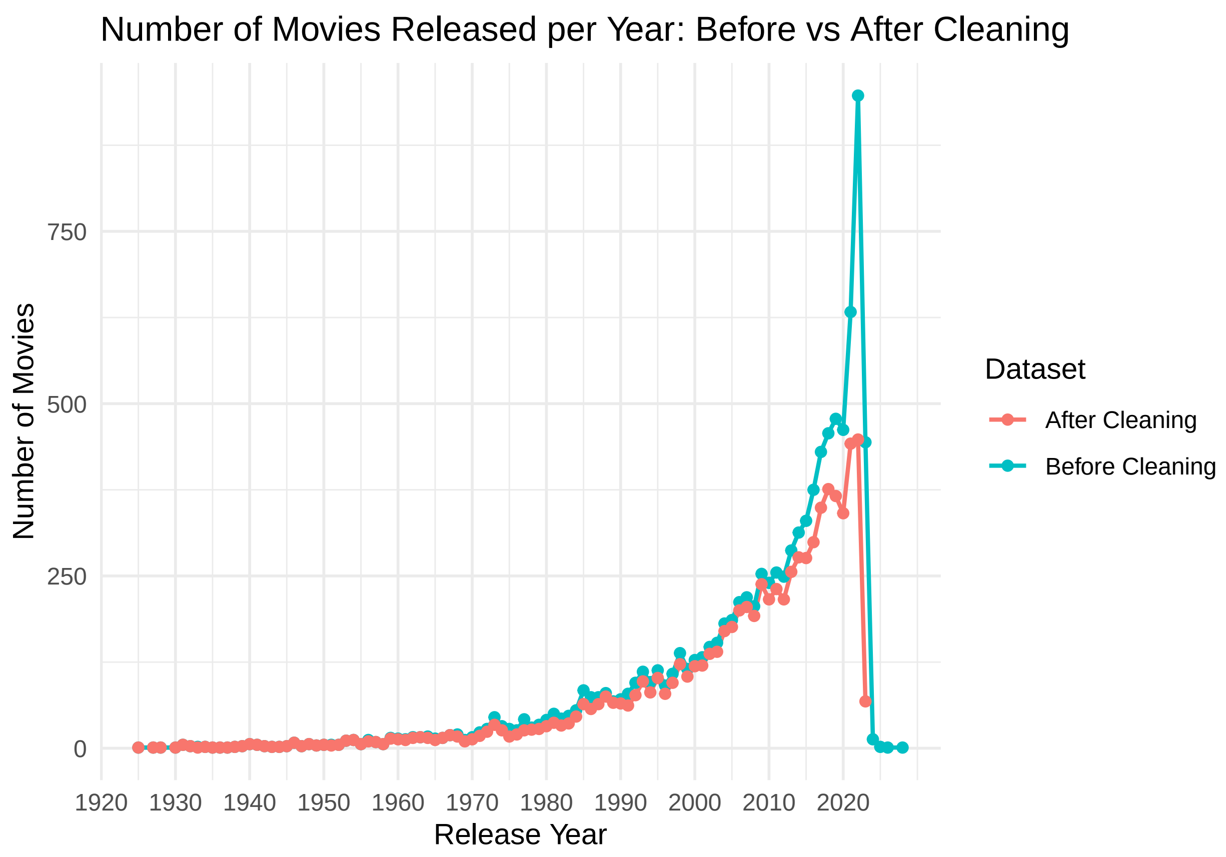
```
##    n
## 1 57
```

Release Date

We also inspect the release years of movies we removed from our dataset for having too little votes (<100) to understand the impact of our filtering.

```
movie_data$release_date <- ymd(movie_data$release_date)
movie_data$release_year <- year(movie_data$release_date)

before_cleaning <- movie_data %>%
  filter(!is.na(release_year)) %>%
  count(release_year) %>%
  mutate(source = "Before Cleaning")

after_cleaning <- movie_clean %>%
  filter(!is.na(release_year)) %>%
  count(release_year) %>%
  mutate(source = "After Cleaning")

combined_years <- bind_rows(before_cleaning, after_cleaning)
```

```
ggplot(combined_years, aes(x = release_year, y = n, color = source)) +
  geom_line(linewidth = 0.75) +
  geom_point() +
  labs(
    title = "Number of Movies Released per Year: Before vs After Cleaning",
    x = "Release Year",
    y = "Number of Movies",
    color = "Dataset"
  ) +
  scale_x_continuous(breaks = seq(min(1900), max(2025), by = 10)) +
  theme_minimal()
```



The plot shows us that our filtering step primarily excluded films released in the last decade. This likely reflects how newer movies have had less time to accumulate votes, resulting in lower vote counts.

Budget & Revenue

We inspect the largest and smallest values for both budget and revenue in our *movie_small* dataset. This helps us identify any possible outliers, unusual values, or errors in entry that could skew our analysis. Examining these extremes allows us to assess data quality and decide if any values require cleaning or further investigation.

```
# Display smallest budgets
movie_small %>%
  arrange(budget) %>%
  select(title, budget, revenue) %>%
  head()
```

```
##                              title budget revenue
## 1                Sex and Death 101      5       1
## 2 Chestnut: Hero of Central Park      6      10
## 3      The Notorious Bettie Page     90 1410778
## 4                           Primer   7000  545436
## 5                   Pink Flamingos  12000 6000000
## 6    The Cabinet of Dr. Caligari  18000    8811
```

```
# Display largest budgets
movie_small %>%
  arrange(desc(budget)) %>%
  select(title, budget, revenue) %>%
  head()
```

```
##                                         title    budget    revenue
## 1                          Operation Red Sea 579330426  579220560
## 2                     Avatar: The Way of Water 460000000 2319331580
## 3 Pirates of the Caribbean: On Stranger Tides 379000000 1045713802
## 4                      Avengers: Age of Ultron 365000000 1405403694
## 5                            Avengers: Endgame 356000000 2794731755
## 6                       Avengers: Infinity War 300000000 2046239637
```

```
# Display smallest revenues
movie_small %>%
  arrange(revenue) %>%
  select(title, budget, revenue) %>%
  head()
```

```
##                              title   budget revenue
## 1                Sex and Death 101        5       1
## 2 Chestnut: Hero of Central Park        6      10
## 3                    Cross of Iron  6000000     201
## 4                      The Fanatic 18000000    3153
## 5         Kickboxer: Retaliation 13000000    4537
## 6                  The Good Doctor  6000000    5206
```

```
# Display largest revenues
movie_small %>%
  arrange(desc(revenue)) %>%
  select(title, budget, revenue) %>%
  head()
```

```
##                           title    budget    revenue
## 1                        Avatar 237000000 2923706026
## 2             Avengers: Endgame 356000000 2794731755
## 3     Avatar: The Way of Water 460000000 2319331580
## 4                       Titanic 200000000 2264162353
## 5 Star Wars: The Force Awakens 245000000 2068223624
## 6       Avengers: Infinity War 300000000 2046239637
```

Movies that stood out to us in these entries were ‘Sex and Death 101’ and ‘Chestnut: Hero of Central Park’ for their abnormally small budget and revenue, ‘The Notorious Bettie Page’ for its small budget, ‘Cross of

Iron' for its small revenue, and 'Operation Red Sea' for its unexpected maximum budget. Upon further research, we found that most of these extreme examples had inaccurate budget and revenue information. For example, we found that 'Sex and Death 101' and 'The Notorious Bettie Page' did not have publicly available budget information. Additionally, the actual budget for 'Operation Red Sea' was reported to be approximately 70 million USD. In our dataset, it is the maximum budget value of 579330426.

These discrepancies suggest possible data entry errors, which could explain why they are such anomalies among other values. We also looked up the revenue of 'Operation Red Sea' and found that it made approximately 579.2 million USD at the box office. Interestingly, the budget value recorded in our dataset is only around 100,000 off from this value. This suggests that the budget may have been mistakenly entered as a near-duplicate of the revenue.

We decided to remove these clear errors from the dataset, but it's important to acknowledge that other data entry errors may exist in the dataset but remain unnoticed and unremoved simply because they are less noticeable. We filtered out these high discrepancies from our *movie_small* dataset for future analysis involving budget and revenue.

```r
movie_small <- movie_small %>%
  filter(!title %in% c("Operation Red Sea", "Sex and Death 101",
                     "Chestnut: Hero of Central Park",
                     "Cross of Iron", "The Notorious Bettie Page"))
```

2.4 - Vote Average Versus Other Variables & Statistical Tests

Now that we've taken a look at the distributional shape of variables in the dataset, we can begin to investigate which features are most associated with higher or lower ratings.

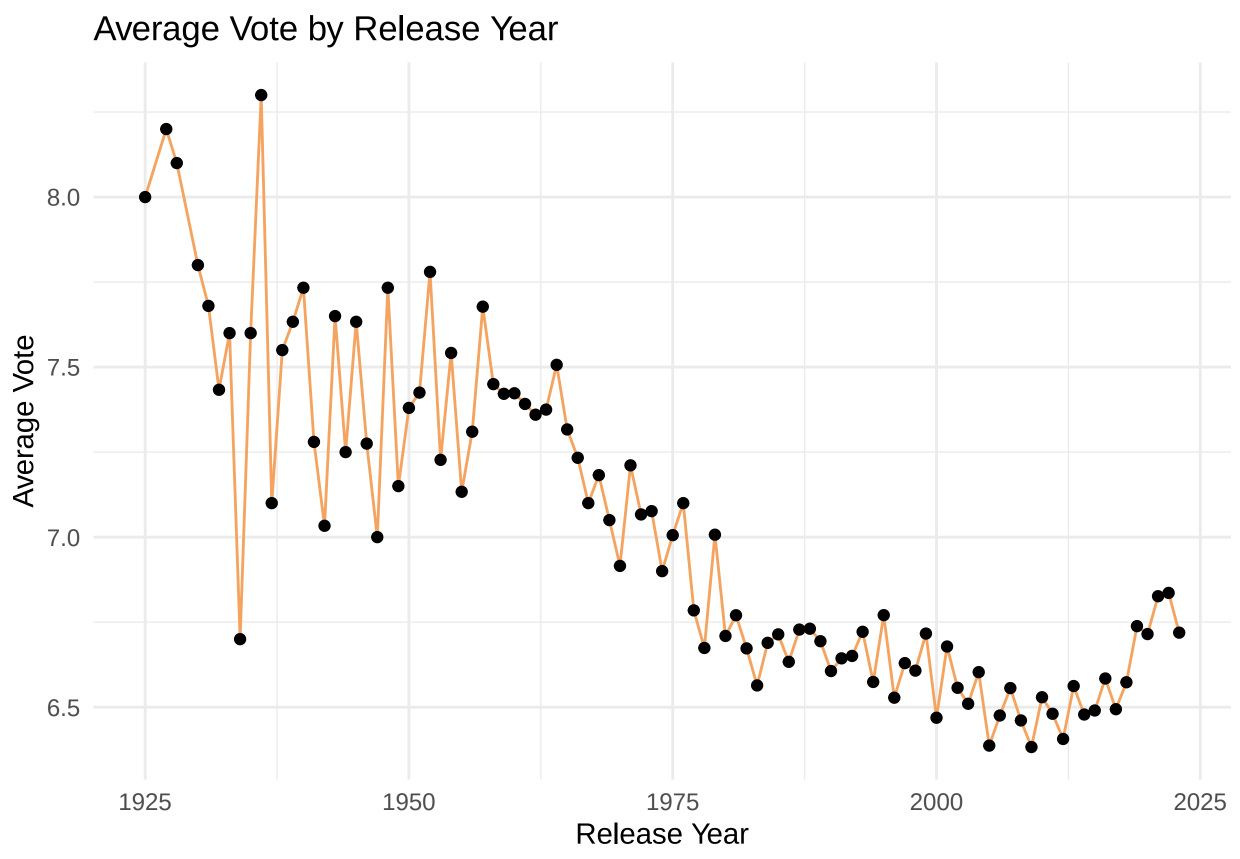
Throughout this section, we will include statistical tests to evaluate whether the relationships we observe are statistically significant. The following code is focused on preparing the data for those analyses.

```r
movie_small <- movie_small %>%
  mutate(
    profit = revenue - budget,
    roi = profit / budget,
    budget_bin = cut(budget, breaks = c(0, 1e6, 1e7, 5e7, 1e8, 2e8, 1e9)),
    revenue_bin = cut(revenue, breaks = c(0, 1e6, 1e7, 5e7, 1e8, 2e8, 1e9)),
    profit_bin = cut(profit, breaks = c(-Inf, 0, 1e6, 1e7, 5e7, 1e8, 1e9)),
    roi_bin = cut(roi, breaks = c(-Inf, 0, 1, 2, 5, 10, Inf))
  )

movie_clean <- movie_clean %>%
  mutate(
    runtime_bin = cut(runtime, breaks = c(0, 60, 90, 120, 150, Inf)),
    vote_bin = cut(vote_count, breaks = c(0, 100, 1000, 5000, 10000, 50000, Inf)),
    decade = floor(release_year / 10) * 10
  )

movie_small <- movie_small %>%
    mutate(decade = floor(release_year / 10) * 10)

anova_variance_explained <- function(model, factor_name) {
  anova_table <- summary(model)[[1]]
  ss_group <- anova_table[factor_name, "Sum Sq"]
  ss_resid <- anova_table["Residuals", "Sum Sq"]
  prop_explained <- ss_group / (ss_group + ss_resid)
  return(round(prop_explained, 4))
}
```

Release Year vs. Average Rating

To begin exploring how average ratings relate to other variables in the dataset, we first examine how *vote_average* varies over time. In this plot, we group movies by *release_year* and calculate the mean *vote_average* for each year.

```r
movie_clean %>%
  group_by(release_year) %>%
  summarize(avg_vote = mean(vote_average, na.rm = TRUE)) %>%
  ggplot(aes(x = release_year, y = avg_vote)) +
  geom_line(color = "sandybrown") +
  geom_point() +
  labs(title = "Average Vote by Release Year",
       x = "Release Year",
       y = "Average Vote") +
  theme_minimal()
```



We find that films released earlier tend to have higher average ratings compared to more recent films. This pattern likely reflects the way our dataset was constructed—since it contains the 10,000 most popular movies according to a proprietary TMDB metric influenced by recent traction and search activity, newer movies are more likely to appear in the dataset regardless of quality.

Is this Statistically Significant?

Since release year is a categorical variable, we use an ANOVA test to determine whether there is a statistically significant relationship between a movie's release year and its average rating. We also conduct a separate test using release decade.

```
aov_year  <- aov(vote_average ~ release_year, data = movie_clean)
aov_decade  <- aov(vote_average ~ decade, data = movie_clean)

print(summary(aov_decade))
```

```
##                Df Sum Sq Mean Sq F value Pr(>F)
## decade          1     75   74.51   115.6 <2e-16 ***
## Residuals    7774   5011    0.64
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
print(summary(aov_year))
```

```
##                 Df Sum Sq Mean Sq F value Pr(>F)
## release_year     1     88   87.61   136.3 <2e-16 ***
## Residuals     7774   4998    0.64
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
cat("Year:      ", anova_variance_explained(aov_year, "release_year"),       "\n")
```

```
## Year:       0.0172
```

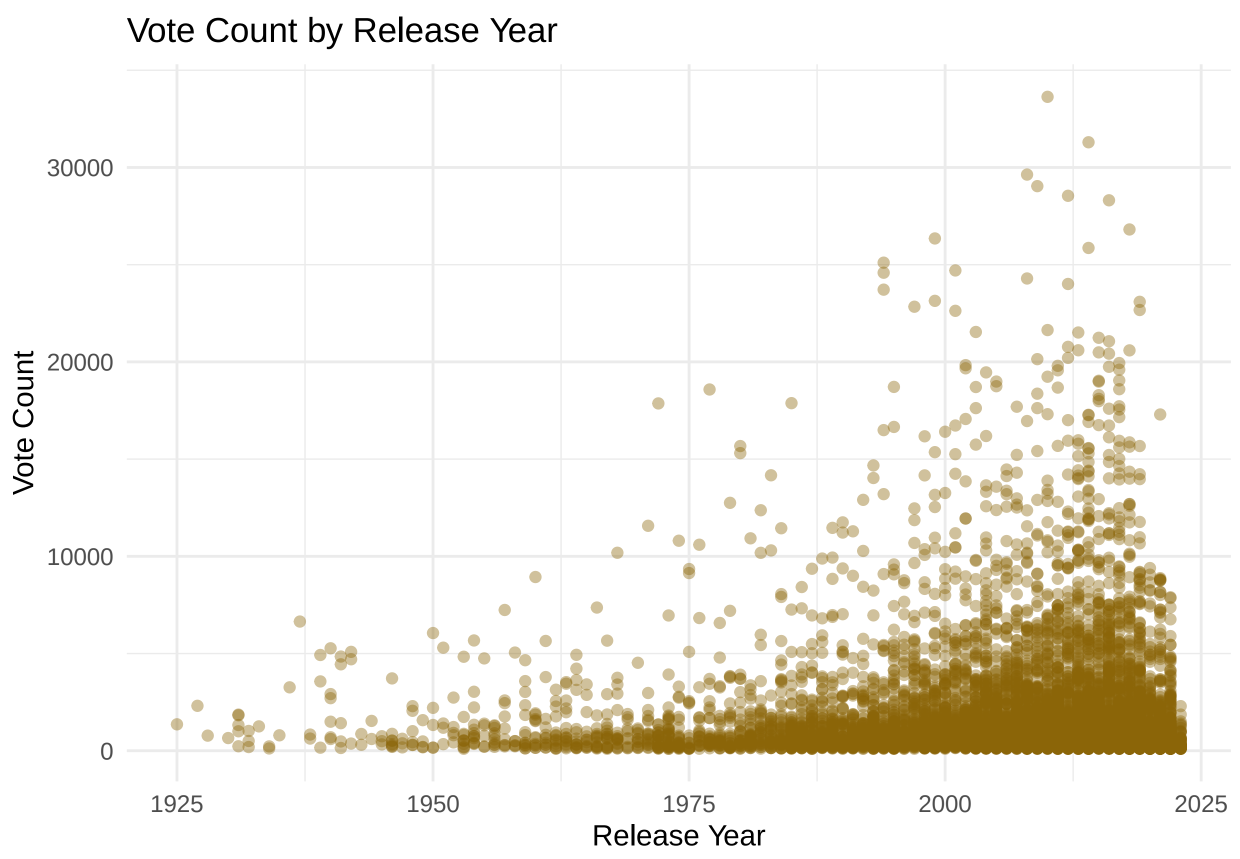
```
cat("Decade:      ", anova_variance_explained(aov_decade, "decade"),       "\n")
```

```
## Decade:       0.0147
```

For both *release_year* and *release_decade*, the p-values were well below the 0.05 threshold, indicating that each has a statistically significant relationship with average vote rating. However, to see how much they impacted the *vote_average*, we calculated their sum of squares (Sum Sq) contributions to the overall variance. In both cases, the explained variance was under the 0.02 mark, suggesting that while the results are statistically significant, release timing contributes minimally to differences in average ratings.

To better understand the trends observed in the previous plot, we next examine how *vote_count* varies by release year. We chose to include vote count here to illustrate the impact of recency bias in TMDB data.

```
movie_clean %>%
  ggplot(aes(x = release_year, y = vote_count)) +
  geom_point(alpha = 0.4, color = "darkgoldenrod4") +
  labs(title = "Vote Count by Release Year",
       x = "Release Year",
       y = "Vote Count") +
  theme_minimal()
```

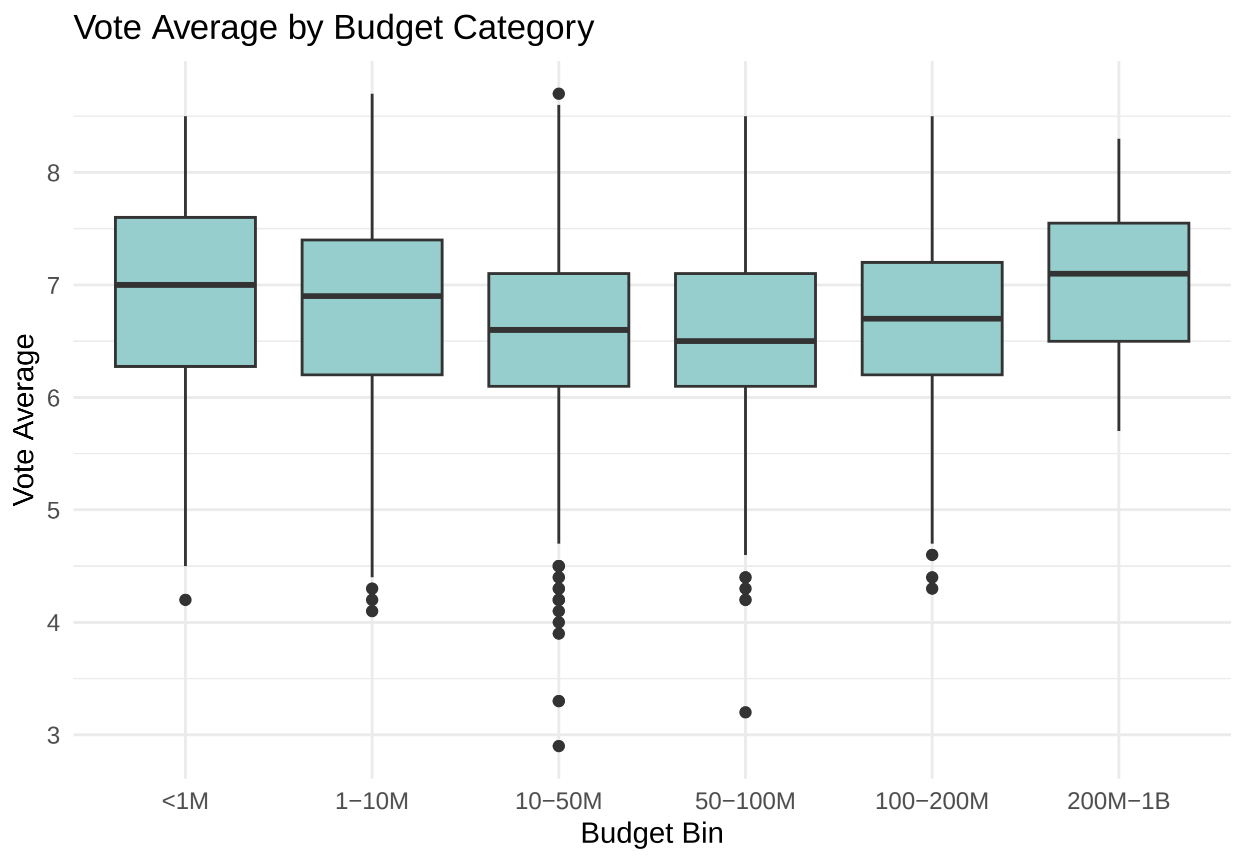


This graph shows that newer movies tend to have much higher vote counts, revealing a clear selection bias in the dataset. As we explored in our Background Information section, TMDB's popularity-based filtering means the sampling favors recent user engagement. As a result, older films that make it to the top 10,000 are typically critically acclaimed or cult classics, while newer movies make the list more easily because they are trending, even if their ratings are lower. This skews the older films to appear to have much higher ratings in this sample, even though this might be disproportionately high. It is also important to note that many of the movies that were removed from our dataset when we filtered out those with less than 100 votes were released in the last decade, which further supports the conjecture that new movies can enter the dataset much more easily than old ones due to site traffic, even though they may be much less established. Examining vote count here helps contextualize why release year trends in vote average should be interpreted with caution based on the nature of the dataset.

Budget vs Average Rating

To explore whether a movie's budget is associated with its average rating, we group movies into six *budget* ranges and visualize their *vote_average* using boxplots. The budget bins range from low-budget films to blockbuster-level budgets. This allows us to compare the distribution of ratings across different grouped *budget* sizes.

```
movie_small %>%
  mutate(budget_bin = cut(budget,
                          breaks = c(0, 1e6, 1e7, 5e7, 1e8, 2e8, 1e9),
                          labels = c("<1M", "1-10M", "10-50M", "50-100M",
                                     "100-200M", "200M-1B"))) %>%
  ggplot(aes(x = budget_bin, y = vote_average)) +
  geom_boxplot(fill = "paleturquoise3") +
  labs(title = "Vote Average by Budget Category",
       x = "Budget Bin",
       y = "Vote Average") +
  theme_minimal()
```

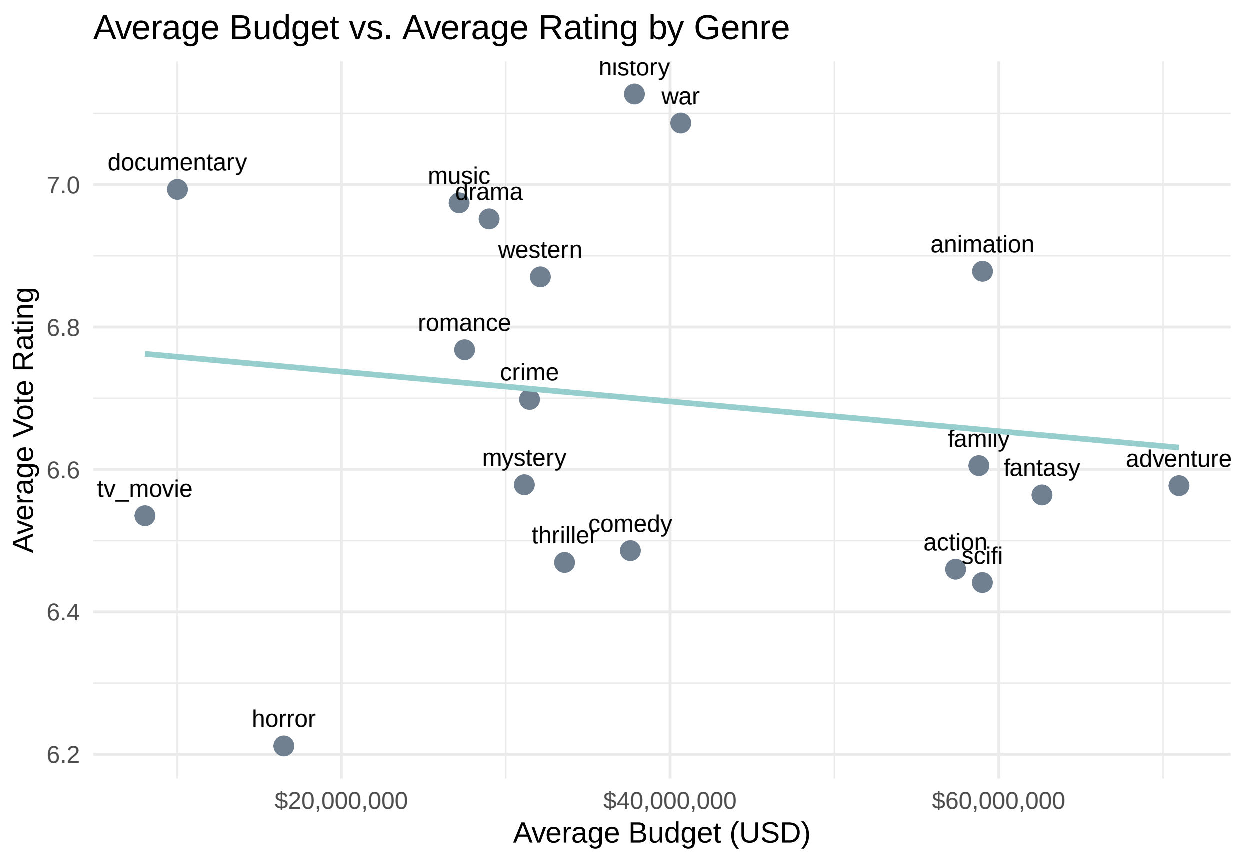


Interestingly, the median vote averages across all the budget categories are relatively flat, which suggests that larger budgets do not necessarily translate into higher ratings. Since the boxplots appear inconclusive regarding a strong relationship, we investigate this further in our Statistical Tests and Modeling section to determine whether *budget* levels have a statistically significant impact on ratings.

To further investigate the relationship between movie budget and ratings, we examine this relationship across movie genres. We calculate the average *budget* and average rating within each *genre*. This helps us determine whether genres tend to spend more and whether that increases or decreases average ratings. Each point on the scatter plot represents a *genre*, with the x-axis being mean *budget* and the y-axis being mean rating. A linear trend line helps highlight any overall trend between *budget* and ratings.

```
# create the genre_budget_rating dataset
genre_budget_rating <- movie_genres %>%
  filter(!is.na(budget), budget > 0, !is.na(vote_average)) %>%
  group_by(genre) %>%
  summarise(
    avg_budget = mean(budget, na.rm = TRUE),
    avg_rating = mean(vote_average, na.rm = TRUE),
    .groups = "drop"
  )

# Plot
ggplot(genre_budget_rating, aes(x = avg_budget, y = avg_rating)) +
  geom_point(color = "slategray", size = 3) +
  geom_text(aes(label = genre), vjust = -1.1, size = 3) +
  labs(
    title = "Average Budget vs. Average Rating by Genre",
    x = "Average Budget (USD)",
    y = "Average Vote Rating"
  ) +
  theme_minimal() +
  scale_x_continuous(labels = scales::dollar_format()) +
  geom_smooth(method = "lm", se = FALSE, color = "paleturquoise3")
```

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



The linear regression line slopes down slightly, indicating that average vote ratings tend to decrease as average budgets increase. From the visualization, we see that genres like Adventure and Fantasy have the highest average budgets, yet their ratings are not significantly higher than other genres. Conversely, History and War films have high ratings despite more modest budgets.

Is this Statistically Significant?

Since the *budget* is both continuous and binned, we run linear regression and ANOVA for this variable.

```
#Linear Regression:
print(summary(lm(vote_average ~ budget, data = movie_small)))
```

```
##
```

```
## Call:
## lm(formula = vote_average ~ budget, data = movie_small)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -3.7531 -0.5444  0.0435  0.5485  2.0461
##
## Coefficients:
##                Estimate Std. Error t value Pr(>|t|)
## (Intercept)  6.658e+00  1.567e-02 424.793   <2e-16 ***
## budget      -1.597e-10  2.502e-10  -0.638    0.523
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.7885 on 4428 degrees of freedom
## Multiple R-squared:  9.2e-05,    Adjusted R-squared:  -0.0001338
## F-statistic: 0.4074 on 1 and 4428 DF,  p-value: 0.5233
```

```
#ANOVA
aov_budget  <- aov(vote_average ~ budget_bin, data = movie_small)
print(summary(aov_budget))
```

```
##                Df Sum Sq Mean Sq F value   Pr(>F)
## budget_bin      5   46.7   9.346   15.28 6.49e-15 ***
## Residuals    4424 2706.7   0.612
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

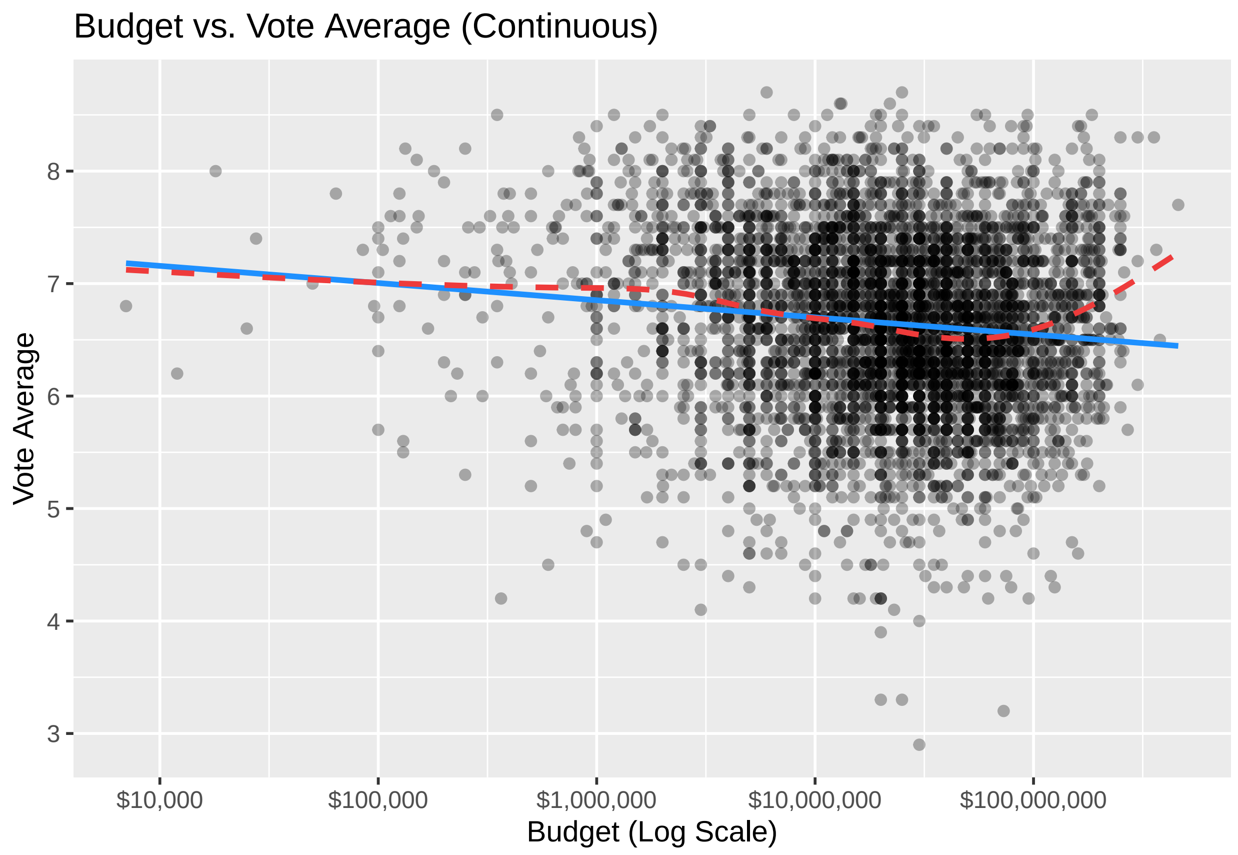
```
cat("Budget Bin:  ", anova_variance_explained(aov_budget, "budget_bin"),  "\n")
```

```
## Budget Bin:   0.017
```

This is the first instance where our statistical tests yield conflicting results. The linear regression test suggests that *budget* is not statistically significant, whereas the ANOVA test indicates that it is. This discrepancy typically means that the relationship between *budget* and average rating is not linear, and therefore the linear regression is not able to capture the pattern effectively. We can visualize the data with both a linear (lm) and a non-linear (loess) smoothing line to see if they differ.

```
ggplot(movie_small, aes(x = budget, y = vote_average)) +
  geom_point(alpha = 0.3) +
  geom_smooth(method = "lm", se = FALSE, color = "dodgerblue") +         # linear trend
  geom_smooth(se = FALSE, color = "brown2", linetype = "dashed") +    # loess (curved) trend
  scale_x_log10(labels = scales::dollar_format()) +  # log scale if needed
  labs(title = "Budget vs. Vote Average (Continuous)",
       x = "Budget (Log Scale)",
       y = "Vote Average")
```

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



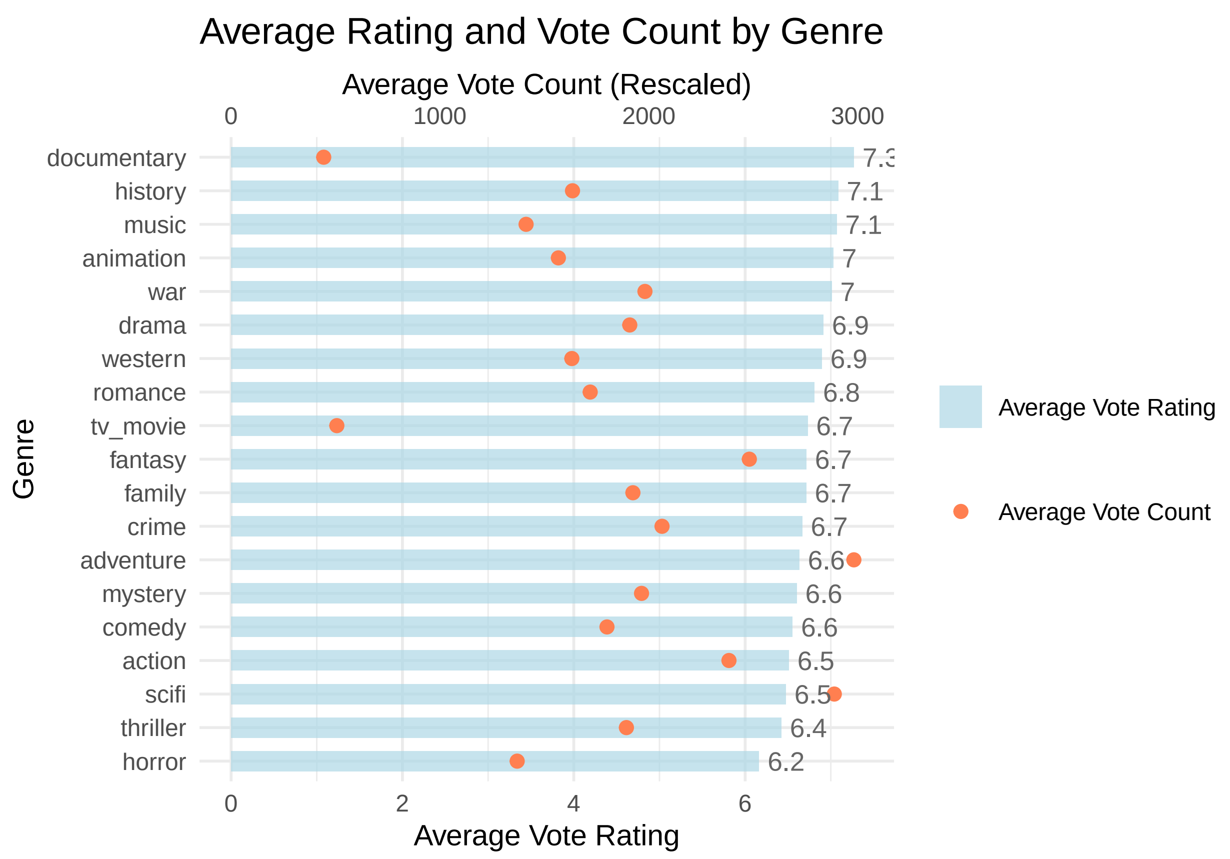
We initially expected the loess and linear regression lines to differ a lot more, since the tests came back with different conclusions. However, the loess line largely aligns with the lm line, up until the higher end of the budget range—which may be contributing to the statistical difference. Even in the ANOVA test, only about 1.7% of the variance in vote average is explained by budget, indicating that it is not a strong predictor of movie *vote_average*.

Genre vs. Average Rating

To better understand how movie ratings vary across different genres, we calculated the average vote ratings for each genre in the dataset in order to see which genres tend to receive higher or lower audience scores. Additionally, we also include average vote counts for each genre. By plotting both metrics together, we can better contextualize the average ratings relative to voter engagement.

```r
# Create genre_stats with average vote rating and average vote count per genre
genre_stats <- movie_genres %>%
  group_by(genre) %>%
  summarize(
    avg_rating = mean(vote_average, na.rm = TRUE),
    avg_vote_count = mean(vote_count, na.rm = TRUE),
    .groups = "drop"
  )

# Rescale vote count to roughly match rating scale
max_vote <- max(genre_stats$avg_vote_count, na.rm = TRUE)
max_rating <- max(genre_stats$avg_rating, na.rm = TRUE)
scale_factor <- max_vote / max_rating
```

```r
# Plot average vote rating and vote count across genres (with points instead of smooth line)
ggplot(genre_stats, aes(x = reorder(genre, avg_rating))) +
  geom_col(aes(y = avg_rating, fill = "Average Vote Rating"), alpha = 0.7, width = 0.6) +
  geom_point(aes(y = avg_vote_count / scale_factor, color = "Average Vote Count"), size = 2) +
  geom_text(aes(y = avg_rating + 0.1, label = round(avg_rating, 1)),
            hjust = 0, size = 3.5, color = "gray40") +
  scale_y_continuous(
    name = "Average Vote Rating",
    sec.axis = sec_axis(~ . * scale_factor, name = "Average Vote Count (Rescaled)")
  ) +
  scale_fill_manual(values = c("Average Vote Rating" = "lightblue")) +
  scale_color_manual(values = c("Average Vote Count" = "coral")) +
  coord_flip() +
  labs(
    title = "Average Rating and Vote Count by Genre",
    x = "Genre",
    fill = NULL,
    color = NULL
  ) +
  theme_minimal() +
  theme(
    plot.title = element_text(size = 14),
    axis.title = element_text(size = 11)
  )
```



This plot reveals how genres like Documentary, History, and Music receive the highest average ratings, yet they tend to have relatively lower vote counts—suggesting that these genres attract smaller, yet more niche audiences who may vote more positively than the general public. In contrast, genres such as Action, Science Fiction, and Thriller receive a larger amount of votes but have lower average ratings. This pattern reflects their appeal to a wider audience and higher production frequency, which can lead to more diverse audience opinions and thus higher variability in average vote ratings.

Next, we look at whether the volume of movies released within a *genre* each year influences vote averages. We wanted to investigate whether a higher quantity of movies released annually per *genre* correlates with an increase or decrease in average ratings. By plotting these metrics over time and across genres, we can see if market saturation leads to higher or lower audience reception.

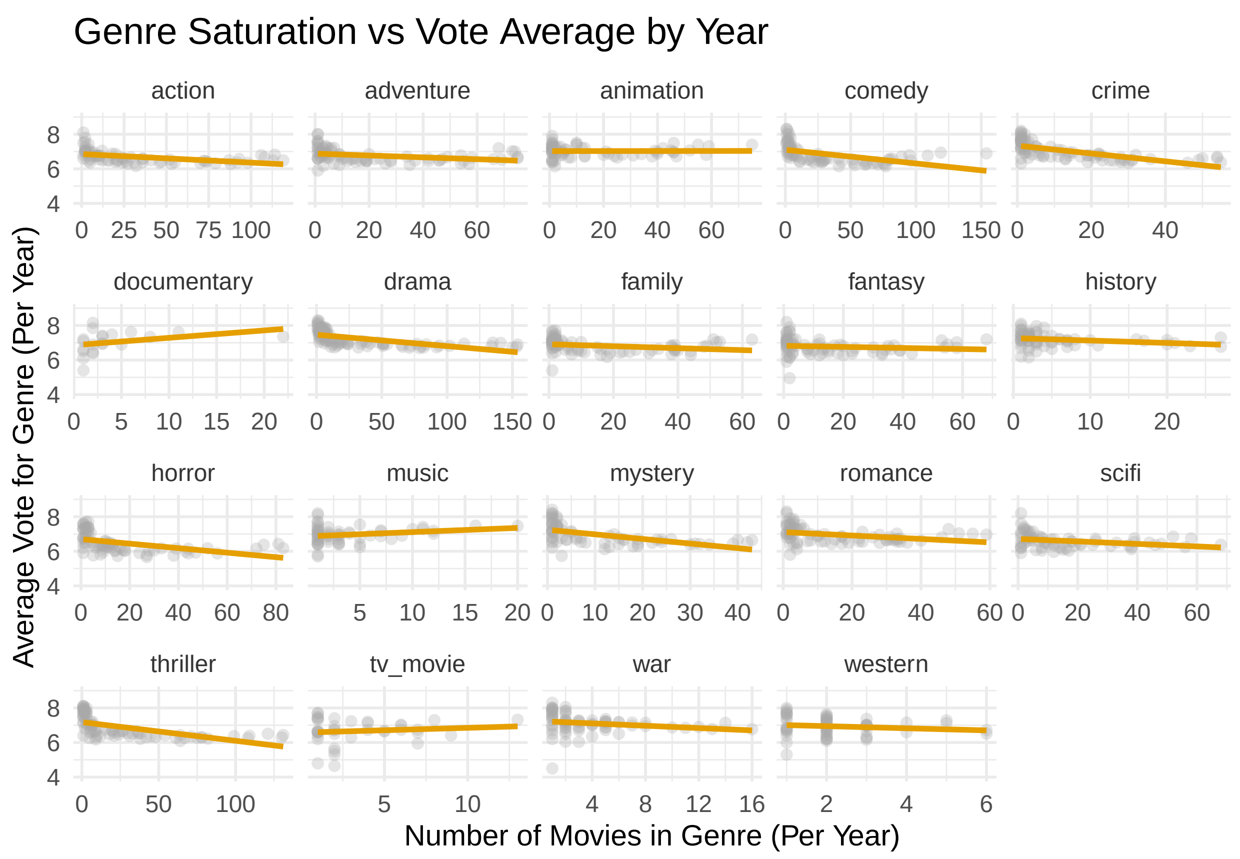
```r
movie_genres %>%
  group_by(release_year, genre) %>%
  summarize(
    n_movies = n(),
    avg_vote = mean(vote_average, na.rm = TRUE),
    .groups = "drop"
  ) %>%
  ggplot(aes(x = n_movies, y = avg_vote)) +
  geom_point(alpha = 0.3, color = "darkgray") +
  geom_smooth(method = "lm", se = FALSE, color = "#E69F00", size = 1) +
  scale_y_continuous(breaks = seq(4, 9, by = 2), limits = c(4, 9)) +
  facet_wrap(~ genre, scales = "free_x") +
  labs(
    title = "Genre Saturation vs Vote Average by Year",
    x = "Number of Movies in Genre (Per Year)",
    y = "Average Vote for Genre (Per Year)"
  ) +
  theme_minimal() +
  theme(
    plot.title = element_text(size = 14),
    strip.text = element_text(color = "#333333"),
    axis.title = element_text(size = 11)
  )
```

```
## 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.
```

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

```
## Warning: Removed 1 row containing non-finite outside the scale range
## (`stat_smooth()`).
```

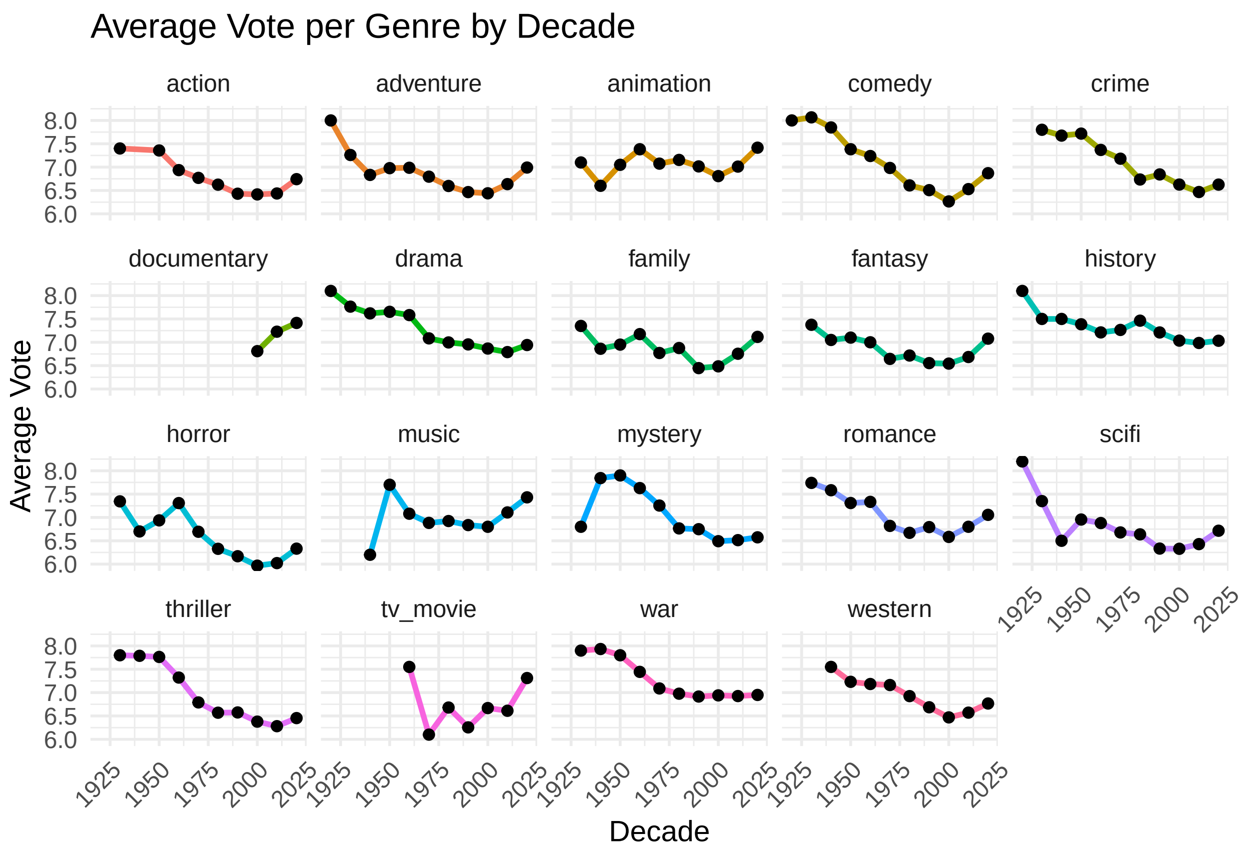
```
## Warning: Removed 1 row containing missing values or values outside the scale range
## (`geom_point()`).
```



The visualization reveals that most genres show minimal relationship between saturation and average ratings, shown by the relatively flat orange trend lines across the majority of genres. However, Documentary stands out as a notable exception, both increasing production volume over time and showing a slight upward trend in ratings. Conversely, Thriller and Comedy display a noticeable downward slope, suggesting that as more movies are produced in these genres per year, the average rating tends to decrease slightly. These patterns suggest that certain genres may be more vulnerable to oversaturation effects than others. Overall, there is a general lack of correlation between production frequency and vote average over most genres.

To explore how genre popularity may have changed over time, we visualize the average vote rating for each genre by decade to investigate broader trends for genres over the years. Grouping by *decade* allows us to better identify any sustained shifts in audience preferences.

```
movie_genres <- movie_genres %>%
  mutate(decade = floor(release_year / 10) * 10)

movie_genres %>%
  group_by(decade, genre) %>%
  summarize(avg_vote = mean(vote_average, na.rm = TRUE), .groups = "drop") %>%
  ggplot(aes(x = decade, y = avg_vote, group = genre)) +
  geom_line(aes(color = genre), size = 1) +
  geom_point(color = "black", size = 1.5) +
  facet_wrap(~ genre) +
  labs(title = "Average Vote per Genre by Decade",
       x = "Decade",
       y = "Average Vote") +
  theme_minimal() +
  theme(
    legend.position = "none",
    axis.text.x = element_text(angle = 45, hjust = 1)
  )
```

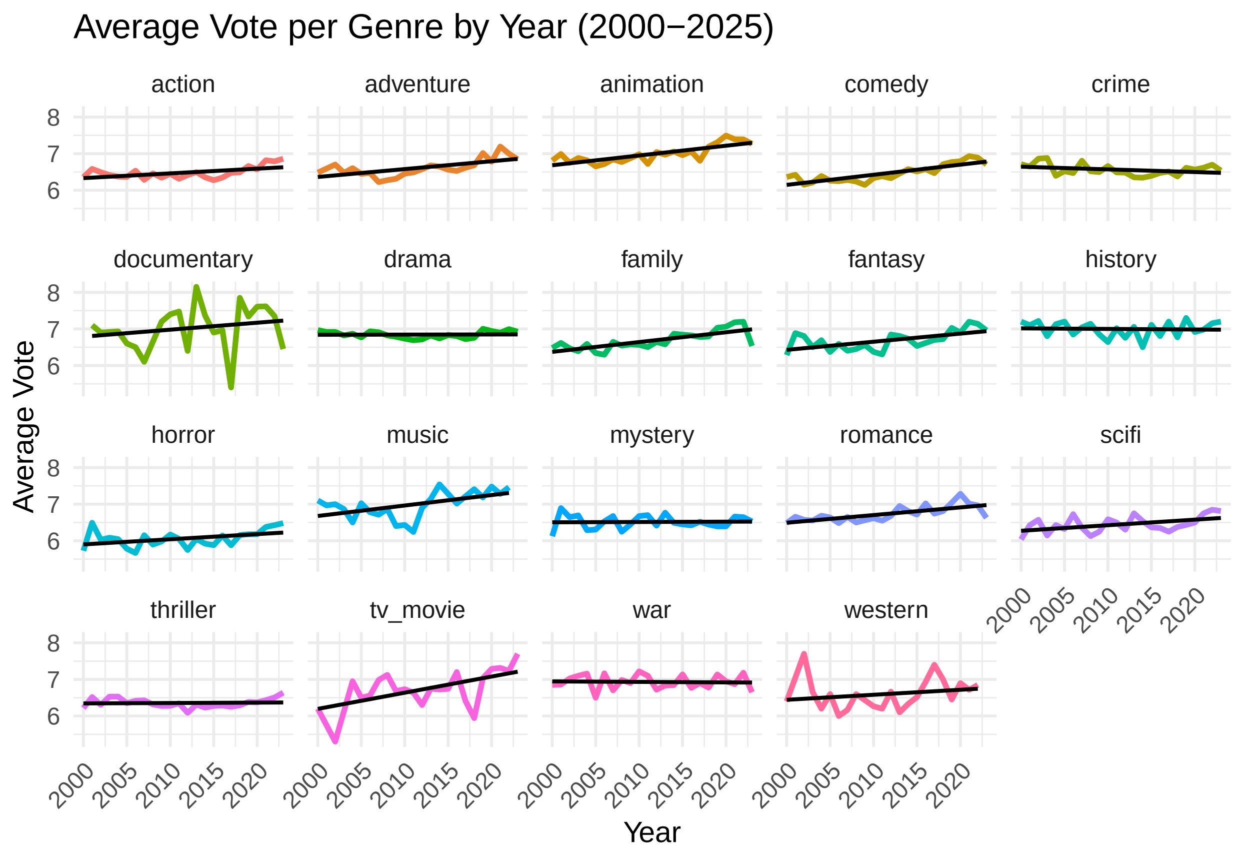


This visualization reveals a pattern of declining ratings across genres over the decades, which aligns with our previous finding that films in the dataset that were released earlier tend to have higher average ratings compared to more recent films. This widespread decline supports our earlier hypothesis about dataset bias, where older films that make it to the top 10,000 are typically critically acclaimed classics or cult favorites, while newer movies are included more easily due to TMDB's popularity-based filtering that favors recent user engagement. The Documentary genre does stand out for its clear upward trajectory from the 1950s on, but this dissimilar pattern could also be attributed to the smaller amount of data within that genre.

To gain a more detailed and recent perspective on how movie ratings have changed over time, we focus specifically on the years 2000 to 2025. We create a plot to track the average vote rating by *genre* per *year*, offering a closer look at modern trends. We add a linear regression line for each genre's data to highlight changes in audience for each *genre*.

```
movie_genres %>%
  filter(release_year >= 2000 & release_year <= 2025) %>%
  group_by(release_year, genre) %>%
  summarize(avg_vote = mean(vote_average, na.rm = TRUE), .groups = "drop") %>%
  ggplot(aes(x = release_year, y = avg_vote, group = genre)) +
  geom_line(aes(color = genre), size = 1) +
  geom_smooth(method = "lm", se = FALSE, color = "black", size = 0.7) +
  facet_wrap(~ genre) +
  labs(
    title = "Average Vote per Genre by Year (2000-2025)",
    x = "Year",
    y = "Average Vote"
  ) +
  theme_minimal() +
  theme(
    legend.position = "none",
    axis.text.x = element_text(angle = 45, hjust = 1)
  )
```

```
## ‘geom_smooth()‘ using formula = ’y ~ x’
```



This visualization generally reveals consistency in how audiences rate movies of different genres over the past 25 years, with ratings showing very minimal upward or downward trends. Some genres show more volatility, including Documentary, TV Movies, and Westerns, showcasing greater fluctuation in average ratings from year to year, while the rest of the genres maintain relatively stable ratings. The general pattern in the plots suggests that audience preferences have remained consistent across most genres in the last two decades.

Is this Statistically Significant

To determine whether the relationship between *genre* and *vote_average* is statistically significant, we use an ANOVA test, since genre is a categorical variable.

```
aov_genre <- aov(vote_average ~ genre, data = movie_genres)
print(summary(aov_genre))
```

```
##                  Df Sum Sq Mean Sq F value Pr(>F)
## genre          18   1084   60.24   101.6 <2e-16 ***
## Residuals   21248  12604    0.59
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
cat("Genre:       ", anova_variance_explained(aov_genre, "genre"),         "\n")
```

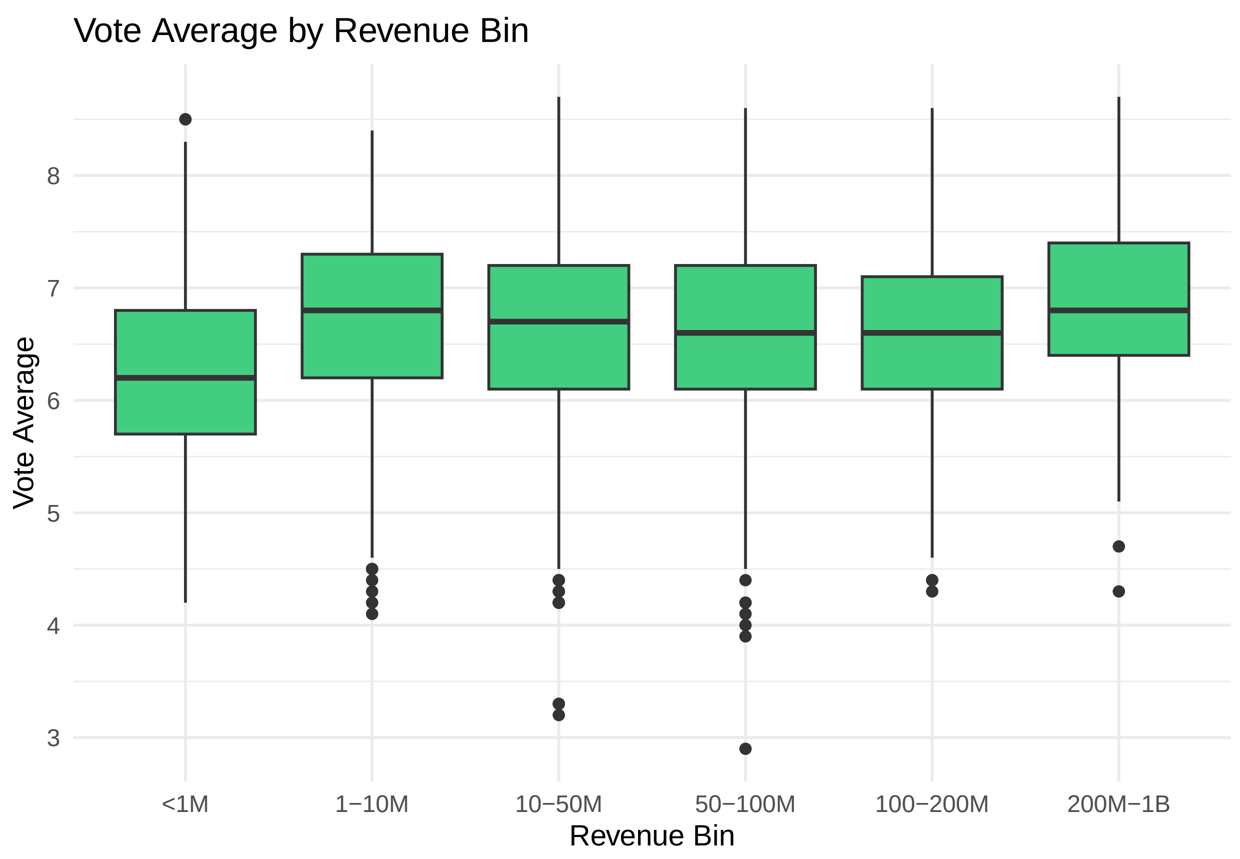
```
## Genre:        0.0792
```

From this, we can see that the p-value is well below 0.05, indicating a statistically significant relationship between *genre* and *vote_average*. Additionally, the proportion of variance explained is nearly 8%, which makes for a solidly meaningful and predictive relationship. While not a large effect, it can be considered moderate in practical terms.

Revenue vs Average Rating

To explore how a movie's financial success might relate to its audience reception, we binned movies by their total *revenue* and compared average vote ratings across these categories.

```
movie_small %>%
  mutate(revenue_bin = cut(revenue,
                           breaks = c(0, 1e6, 1e7, 5e7, 1e8, 2e8, 1e9),
                           labels = c("<1M", "1-10M", "10-50M", "50-100M",
                                      "100-200M", "200M-1B"))) %>%
  filter(!is.na(revenue_bin)) %>%
  ggplot(aes(x = revenue_bin, y = vote_average)) +
  geom_boxplot(fill = "seagreen3") +
  labs(title = "Vote Average by Revenue Bin",
       x = "Revenue Bin",
       y = "Vote Average") +
  theme_minimal()
```



This visualization presents the relationship between *revenue* and *vote_average* for movies in our dataset. We find that median ratings are relatively stable across grouped revenue categories. The films with revenue under $1M show the widest distribution of ratings and the lowest median. However, films in the $10-50M, $50-100M, and $100-200M ranges all maintain similar median ratings. This suggests that once a certain threshold is reached, additional revenue does not necessarily translate to better audience ratings. The highest group $200M-$1B does show a slight uptick in median rating, along with a tighter distribution and fewer low-rated outliers, suggesting that blockbuster-level productions tend to deliver more consistent ratings.

Is this Statistically Significant?

Like *budget*, *revenue* is a continuous variable that we've binned, so we apply both linear regression and ANOVA to test the relationship.

```
print(summary(lm(vote_average ~ revenue, data = movie_small)))
```

```
##
## Call:
## lm(formula = vote_average ~ revenue, data = movie_small)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -3.7037 -0.5024  0.0162  0.5535  2.1159
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) 6.565e+00  1.374e-02  477.68   <2e-16 ***
## revenue     6.573e-10  5.559e-11   11.82   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.7764 on 4428 degrees of freedom
## Multiple R-squared:  0.0306, Adjusted R-squared:  0.03039
## F-statistic: 139.8 on 1 and 4428 DF,  p-value: < 2.2e-16
```

```r
aov_revenue <- aov(vote_average ~ revenue_bin, data = movie_small)
print(summary(aov_revenue))
```

```
##               Df Sum Sq Mean Sq F value Pr(>F)
## revenue_bin    5   69.1  13.822   22.87 <2e-16 ***
## Residuals   4376 2645.0   0.604
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
## 48 observations deleted due to missingness
```

```r
cat("Revenue Bin: ", anova_variance_explained(aov_revenue, "revenue_bin"), "\n")
```

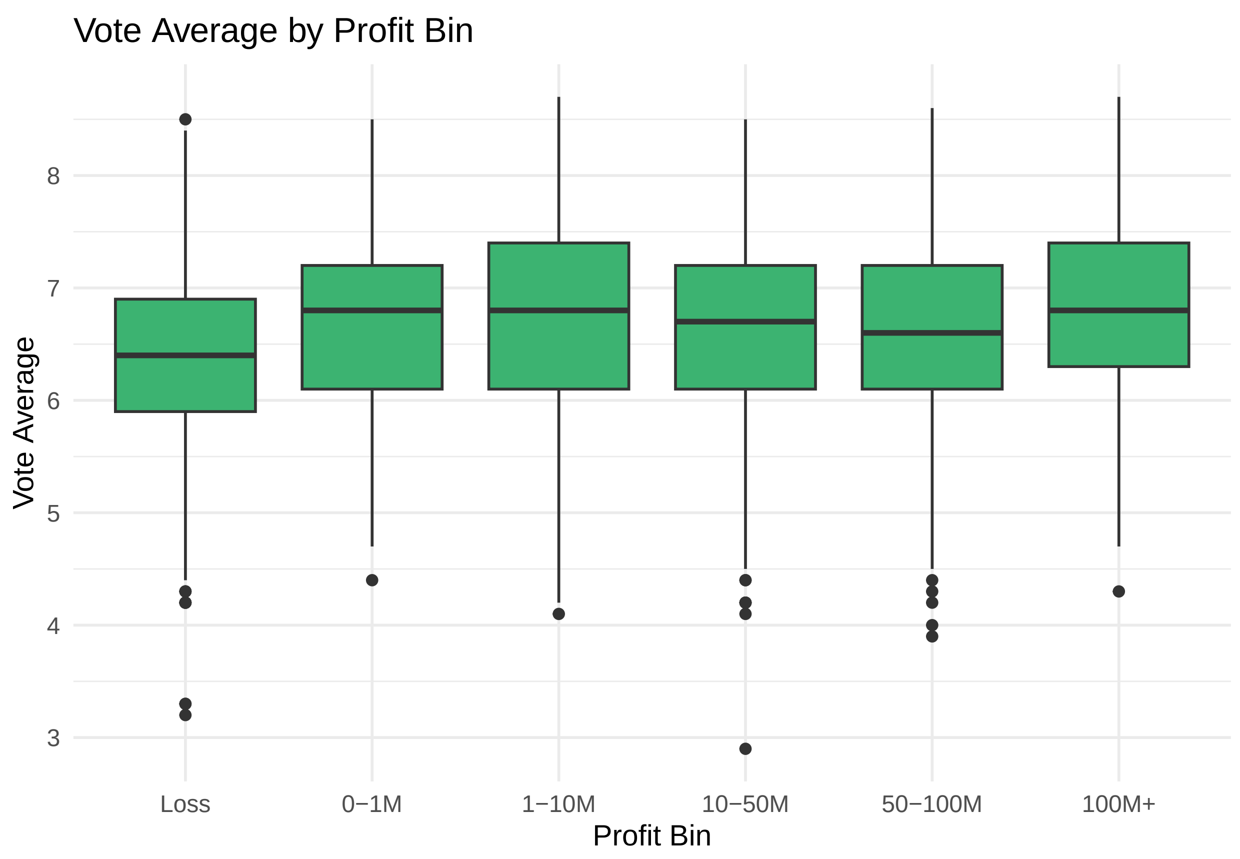
```
## Revenue Bin:  0.0255
```

Both linear regression and ANOVA indicate a statistically significant relationship between *revenue* and *vote_average*. However, the linear regression's R^2 value of 0.03 suggests that *revenue* is a weak predictor of *vote_average*. Similarly, ANOVA shows that binned revenue accounts for only about 2% of the variance in vote average, indicating it has a minimal practical impact.

Profit vs. Average Rating

To analyze how a movie's financial success relates to audience ratings, we calculated profit by subtracting *budget* from *revenue*. We then grouped movies into *profit* bins to analyze how average vote ratings vary across different *profit* ranges to help us understand whether more profitable movies tend to receive higher audience scores.

```r
movie_small <- movie_small %>%
  mutate(profit = revenue - budget)

movie_small %>%
  mutate(profit_bin = cut(profit,
                          breaks = c(-Inf, 0, 1e6, 1e7, 5e7, 1e8, 1e9),
                          labels = c("Loss", "0-1M", "1-10M", "10-50M",
                                     "50-100M", "100M+"))) %>%
  filter(!is.na(profit_bin)) %>%
  ggplot(aes(x = profit_bin, y = vote_average)) +
  geom_boxplot(fill = "mediumseagreen") +
  labs(title = "Vote Average by Profit Bin",
       x = "Profit Bin",
       y = "Vote Average") +
  theme_minimal()
```



We originally expected this visualization to show a stronger correlation between *profit* and audience ratings, but the pattern challenged our predictions. Intuitively, films that lose money show the lowest median rating, but once films achieve profitability, films across all categories maintain relatively consistent median ratings. This plateau suggests that the threshold between financial failure and success is more critical for audience perception than the magnitude of that success. Even the most profitable films in the $100M+ group don't demonstrate notably higher ratings compared to modestly profitable films. This data suggests that crossing the profitability threshold is more important for audience reception than achieving significantly high returns.

Is this Statistically Significant

Since *profit* is a continuous numerical variable that we've also binned for visualization, we used both linear regression and ANOVA to test the strength and significance of its relationship with *vote_average*.

```r
print(summary(lm(vote_average ~ profit, data = movie_small)))
```

```
##
## Call:
## lm(formula = vote_average ~ profit, data = movie_small)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -3.6951 -0.5058  0.0135  0.5483  2.1276
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) 6.569e+00  1.296e-02  506.72   <2e-16 ***
## profit      9.143e-10  6.474e-11   14.12   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.7714 on 4428 degrees of freedom
## Multiple R-squared:  0.0431, Adjusted R-squared:  0.04288
## F-statistic: 199.4 on 1 and 4428 DF,  p-value: < 2.2e-16
```

```
aov_profit  <- aov(vote_average ~ profit_bin, data = movie_small)
print(summary(aov_profit))
```

```
##                Df Sum Sq Mean Sq F value Pr(>F)
## profit_bin      5  115.7  23.143   38.99 <2e-16 ***
## Residuals    4398 2610.7   0.594
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
## 26 observations deleted due to missingness
```

```
cat("Profit Bin:  ", anova_variance_explained(aov_profit, "profit_bin"),  "\n")
```

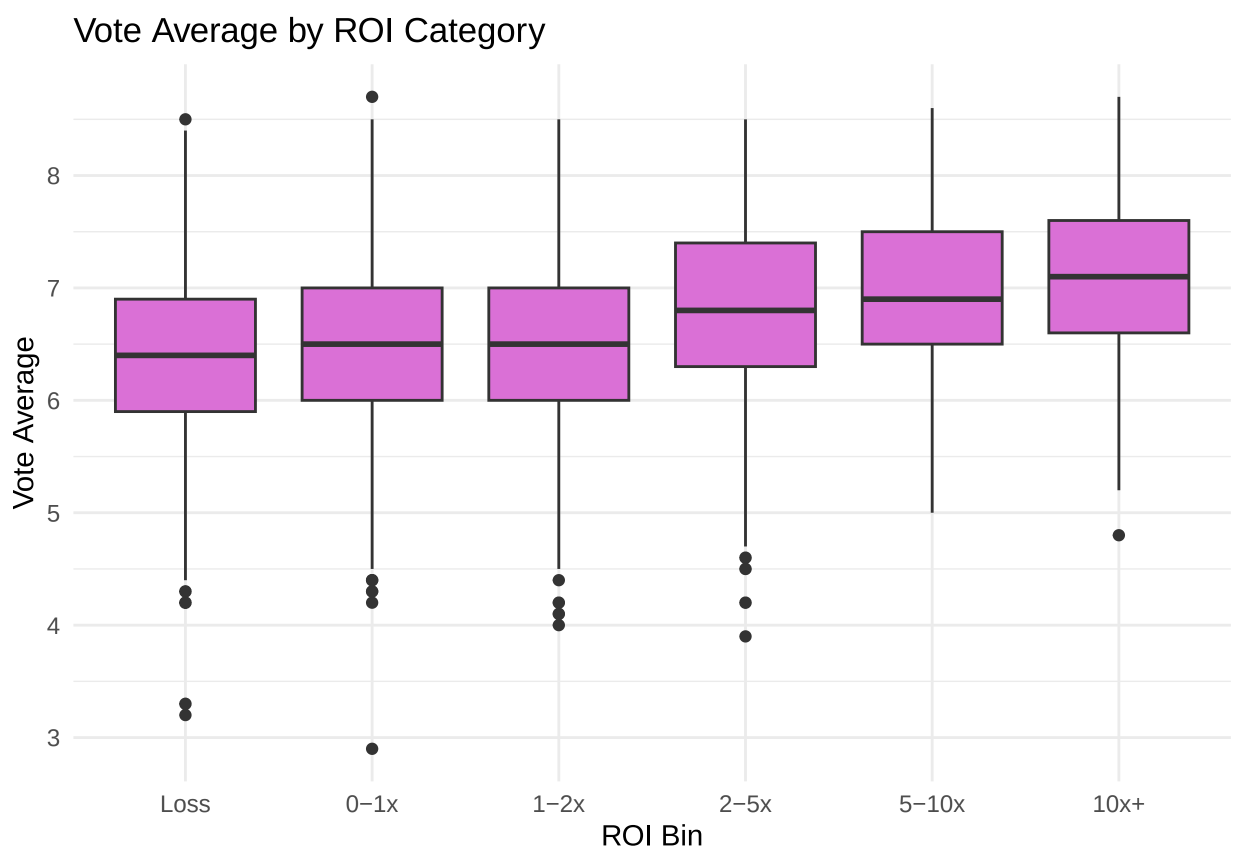
```
## Profit Bin:   0.0424
```

Both the ANOVA and linear regression results indicate that *profit* is statistically significant, however, the R^2 value from the linear regression is just 0.043, suggesting that profit is not a strong predictor of vote average. Similarly, the ANOVA shows that profit accounts for only 4.2% of the variance in vote average, indicating that while the relationship exists, its effect size is small.

ROI vs Average Rating

Building on our previous analysis of *revenue* and *profit* in relation to average ratings, we now calculate return on investment (ROI) as a measure of a film's financial success. While *revenue* and *profit* provide insight into the overall earnings of a movie, they don't take into account the *budget* of the movie that produced that success. ROI is calculated as profit relative to budget, providing a standardized way to evaluate how effectively a film turns investment into a return. This is useful for comparing a large dataset of movies with vastly varying budgets. We group movies into ROI bins ranging from financial losses to returns greater than 10x the initial budget and examine how average ratings vary across these categories using boxplots.

```
movie_small <- movie_small %>%
  mutate(roi = (revenue - budget) / budget)

movie_small %>%
  filter(budget > 1e6) %>%
  mutate(roi = (revenue - budget) / budget,
         roi_bin = cut(roi, breaks = c(-Inf, 0, 1, 2, 5, 10, Inf),
                       labels = c("Loss", "0-1x", "1-2x", "2-5x", "5-10x",
                                  "10x+"))) %>%
  ggplot(aes(x = roi_bin, y = vote_average)) +
  geom_boxplot(fill = "orchid") +
  labs(title = "Vote Average by ROI Category",
       x = "ROI Bin",
       y = "Vote Average") +
  theme_minimal()
```



This visualization shows us that ROI may be a more meaningful indicator of audience voting behavior than absolute earnings. As expected, films that lose money have the lowest median ratings. However, unlike the plateau effect seen with revenue and profit, we observe a clear upward trend in ratings as ROI increases. The highest ROI group of 10x+ also shows a tighter distribution, suggesting greater consistency in audience satisfaction.

Is this Statistically Significant

Since *roi* is a continuous numerical variable and was also binned for visualization, we used both linear regression and ANOVA to test the statistical significance of its relationship with vote average.

```r
#Linear regression
print(summary(lm(vote_average ~ roi, data = movie_small)))
```

```
##
## Call:
## lm(formula = vote_average ~ roi, data = movie_small)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -3.7455 -0.5435  0.0475  0.5546  2.0557
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) 6.6441101  0.0120106 553.189  < 2e-16 ***
## roi         0.0014626  0.0004164   3.513 0.000448 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.7875 on 4428 degrees of freedom
## Multiple R-squared:  0.002779,   Adjusted R-squared:  0.002554
## F-statistic: 12.34 on 1 and 4428 DF,  p-value: 0.000448
```

```
#ANOVA
aov_roi <- aov(vote_average ~ roi_bin, data = movie_small)
print(summary(aov_roi))
```

```
##                Df Sum Sq Mean Sq F value Pr(>F)
## roi_bin         5    241   48.20   84.87 <2e-16 ***
## Residuals    4424   2512    0.57
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
cat("ROI Bin:     ", anova_variance_explained(aov_roi, "roi_bin"),        "\n")
```

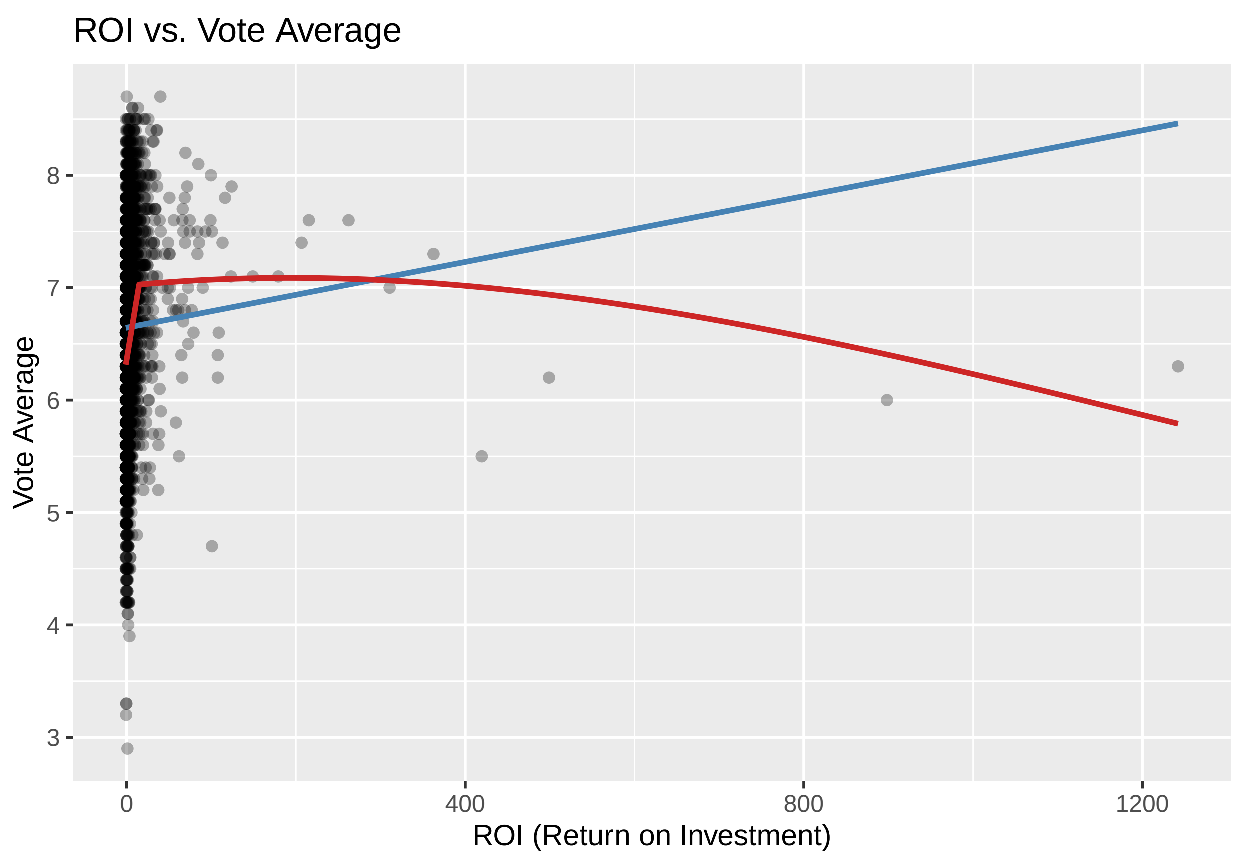
```
## ROI Bin:      0.0875
```

ROI is statistically significant in both the linear regression and ANOVA tests. However, the binned version of ROI appears to be a much more effective predictor of *vote_average*. We can see that the R^2 value for linear regression is just 0.003, which indicates basically no predictive power. In contrast, the ANOVA test shows that ROI is responsible for 8.75% of the variance in vote average, a modest but meaningful proportion. This suggests that categorizing ROI reveals patterns that the continuous metric does not.

To better understand why binned ROI performs more effectively, we plotted the relationship between ROI and vote average using both linear and loess smoothing methods without binning ROI.

```
ggplot(movie_small, aes(x = roi, y = vote_average)) +
  geom_point(alpha = 0.3) +
  geom_smooth(method = "lm", se = FALSE, color = "steelblue") +
  geom_smooth(se = FALSE, color = "firebrick3") +
  labs(title = "ROI vs. Vote Average",
       x = "ROI (Return on Investment)",
       y = "Vote Average")
```

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



As shown, the loess line (in red) reveals a highly curved relationship, while the linear regression line (in blue) fails to capture this complexity. This illustrates why the linear model yields a low R^2, as ROI's relationship with vote average is clearly nonlinear. Binning ROI helps to highlight this pattern more clearly, which explains the stronger results observed in the ANOVA test.

We also noticed that a few movies in the dataset have extremely high ROIs. We thought this was interesting and wanted to see which movies they were. Here, we identify the nine films with an ROI greater than 200!

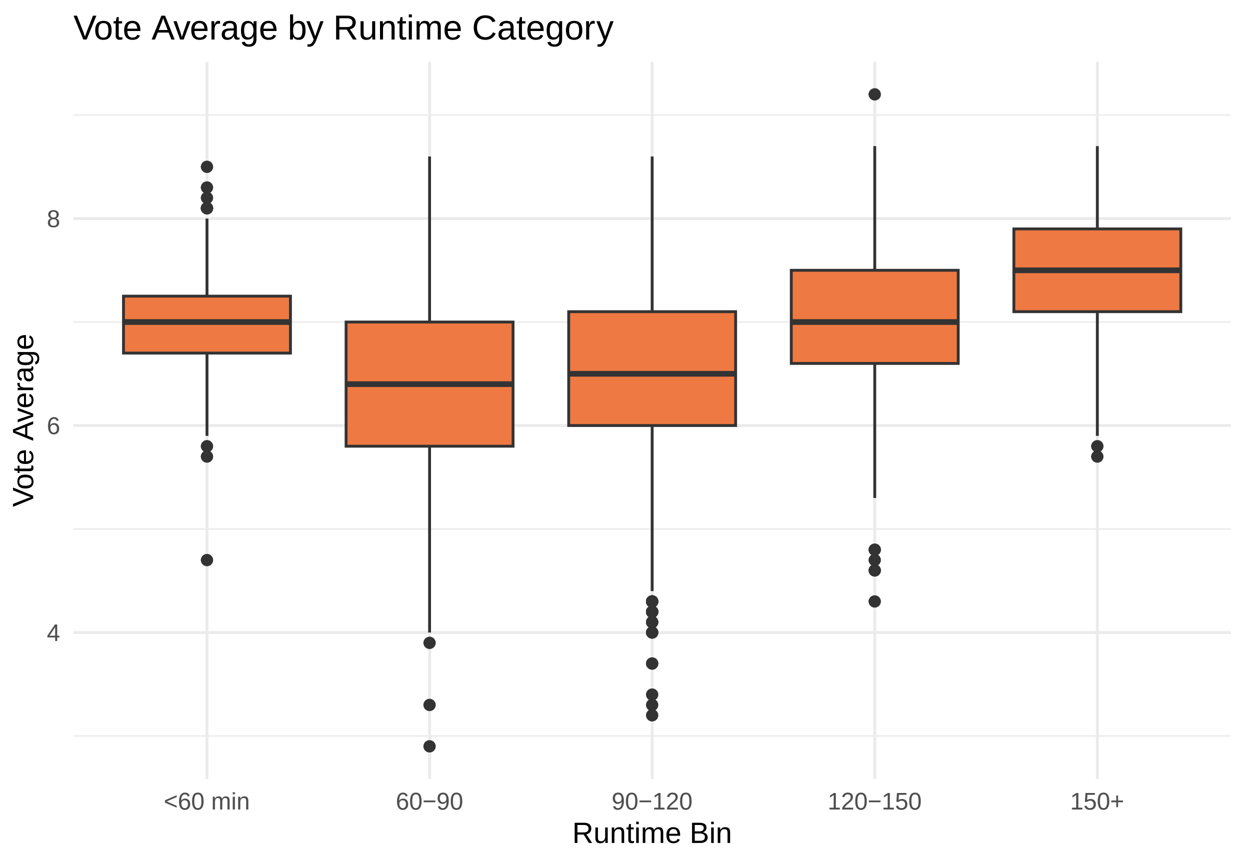
```r
movie_small %>%
  select(title, roi, budget, revenue, profit) %>%
  filter(roi > 200) %>%
  arrange(desc(roi))
```

```
##                           title       roi budget   revenue    profit
## 1      The Blair Witch Project 1242.1955 200000 248639099 248439099
## 2          Paranormal Activity  898.3293 215000 193355800 193140800
## 3               Pink Flamingos  499.0000  12000   6000000   5988000
## 4                   Open Water  419.5227 130000  54667954  54537954
## 5 The Texas Chain Saw Massacre  362.5294  85000  30900000  30815000
## 6                        Bambi  310.7100 858000 267447150 266589150
## 7     Night of the Living Dead  262.1579 114000  30000000  29886000
## 8                    Halloween  215.2277 325000  70274000  69949000
## 9          The Way of the Dragon  206.6923 130000  27000000  26870000
```

Runtime vs Average Rating

Next, we explore the relationship between a film's *runtime* and its average audience rating. We group *runtime* into categories and investigate whether certain lengths correlate with higher or lower ratings.

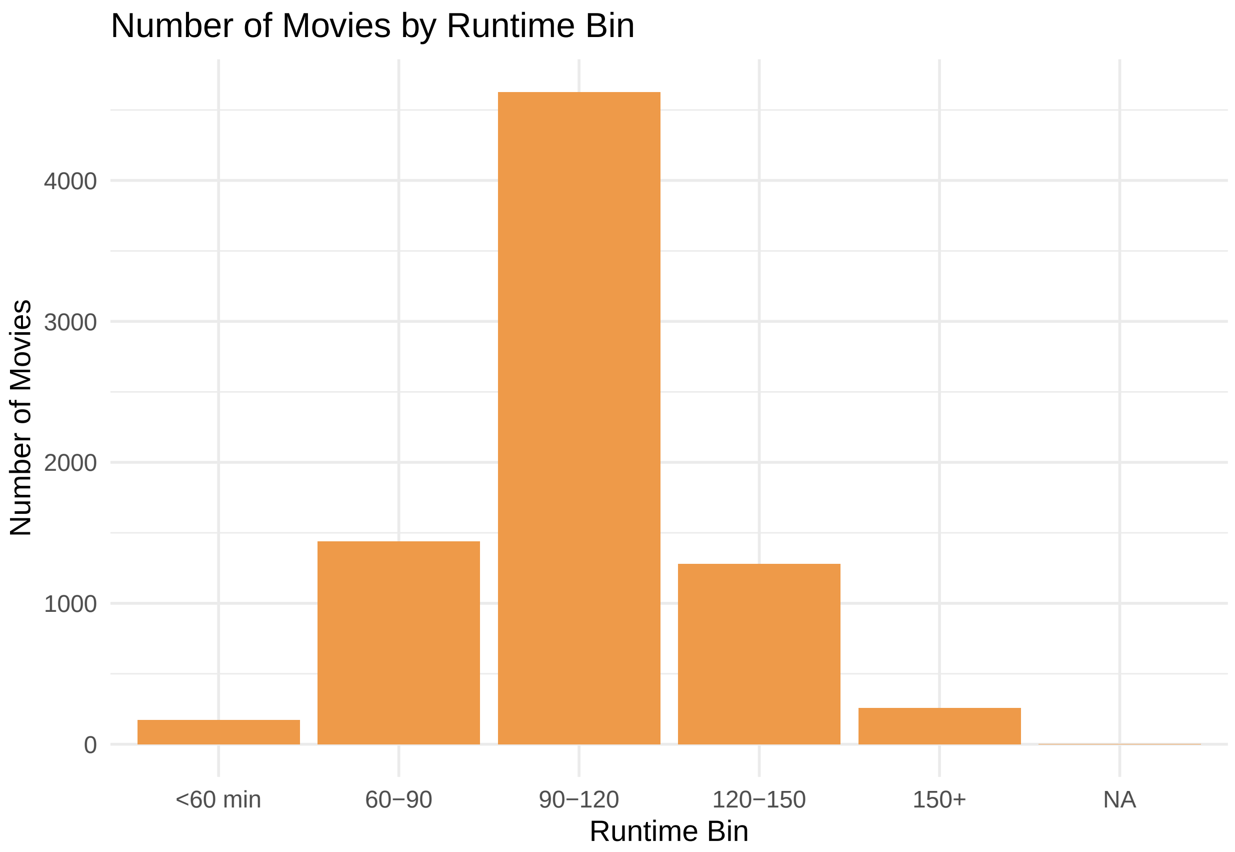
```r
movie_clean %>%
  mutate(runtime_bin = cut(runtime,
                           breaks = c(0, 60, 90, 120, 150, Inf),
                           labels = c("<60 min", "60-90", "90-120", "120-150",
                                      "150+"))) %>%
  filter(!is.na(runtime_bin)) %>%
  ggplot(aes(x = runtime_bin, y = vote_average)) +
  geom_boxplot(fill = "sienna2") +
  labs(title = "Vote Average by Runtime Category",
       x = "Runtime Bin",
       y = "Vote Average") +
  theme_minimal()
```



The relationship between film runtime and audience rating shows a clear upward trend in average ratings with longer films, with the longest films achieving the highest ratings. There is a notable dip in the median for films in the 60-90 minute range. However, the pattern overall suggests that audiences rate films with longer runtimes higher.

To expand on how our *runtime* relates to audience ratings, we also wanted to uncover the distribution of movies across different *runtime* categories. We create a visualization to show the count of films within each *runtime* bin, highlighting which lengths are most common in the dataset.

```r
movie_clean %>%
  mutate(runtime_bin = cut(runtime,
                           breaks = c(0, 60, 90, 120, 150, Inf),
                           labels = c("<60 min", "60-90", "90-120", "120-150",
                                      "150+"))) %>%
  count(runtime_bin) %>%
  ggplot(aes(x = runtime_bin, y = n)) +
  geom_col(fill = "tan2") +
  labs(title = "Number of Movies by Runtime Bin",
       x = "Runtime Bin",
       y = "Number of Movies") +
  theme_minimal()
```



This distribution chart reveals an interesting contrast when compared alongside the previous analysis of the relationship between *runtime* and rating average. While film production follows a normal distribution shape centered around the 90-120 minute mark, vote ratings actually increase with larger runtimes that represent a smaller portion of the total movie count. This data shows that the movie industry overwhelmingly produces movies in the 90-120 minute range, which could reflect standard theater experience or practical constraints such as theater scheduling and audience attention spans. However, this is disconnected from the *vote_average* rating versus *runtime* distribution—the 150+ minute films achieve the highest audience ratings but represent the smallest production category, while the dominant 90-120 minute category shows more modest ratings despite being the industry standard. It's possible that filmmakers that extend runtimes only do so when they have compelling movies that justify the longer length and result in higher audience satisfaction. The abundance of films in the 90-120 minute mark may include those that are filling a conventional format for the industry and thus may sacrifice artistry for efficiency.

Is this Statistically Significant?

Runtime is a continuous numerical variable, so we use linear regression to analyze it. Since we also binned *runtime* into categories, we apply ANOVA to test for significance in its categorical form as well.

```
print(summary(lm(vote_average ~ runtime, data = movie_clean)))
```

```
##
## Call:
## lm(formula = vote_average ~ runtime, data = movie_clean)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -3.5676 -0.5058  0.0225  0.5531  2.4714
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) 5.7358552  0.0405636  141.40   <2e-16 ***
## runtime     0.0086089  0.0003763   22.88   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.7829 on 7774 degrees of freedom
## Multiple R-squared:  0.06308,    Adjusted R-squared:  0.06296
## F-statistic: 523.4 on 1 and 7774 DF,  p-value: < 2.2e-16
```

```
aov_runtime <- aov(vote_average ~ runtime_bin, data = movie_clean)
print(summary(aov_runtime))
```

```
##               Df Sum Sq Mean Sq F value Pr(>F)
## runtime_bin    4    506  126.49   214.6 <2e-16 ***
## Residuals   7768   4579    0.59
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
## 3 observations deleted due to missingness
```

```
cat("Runtime Bin: ", anova_variance_explained(aov_runtime, "runtime_bin"), "\n")
```

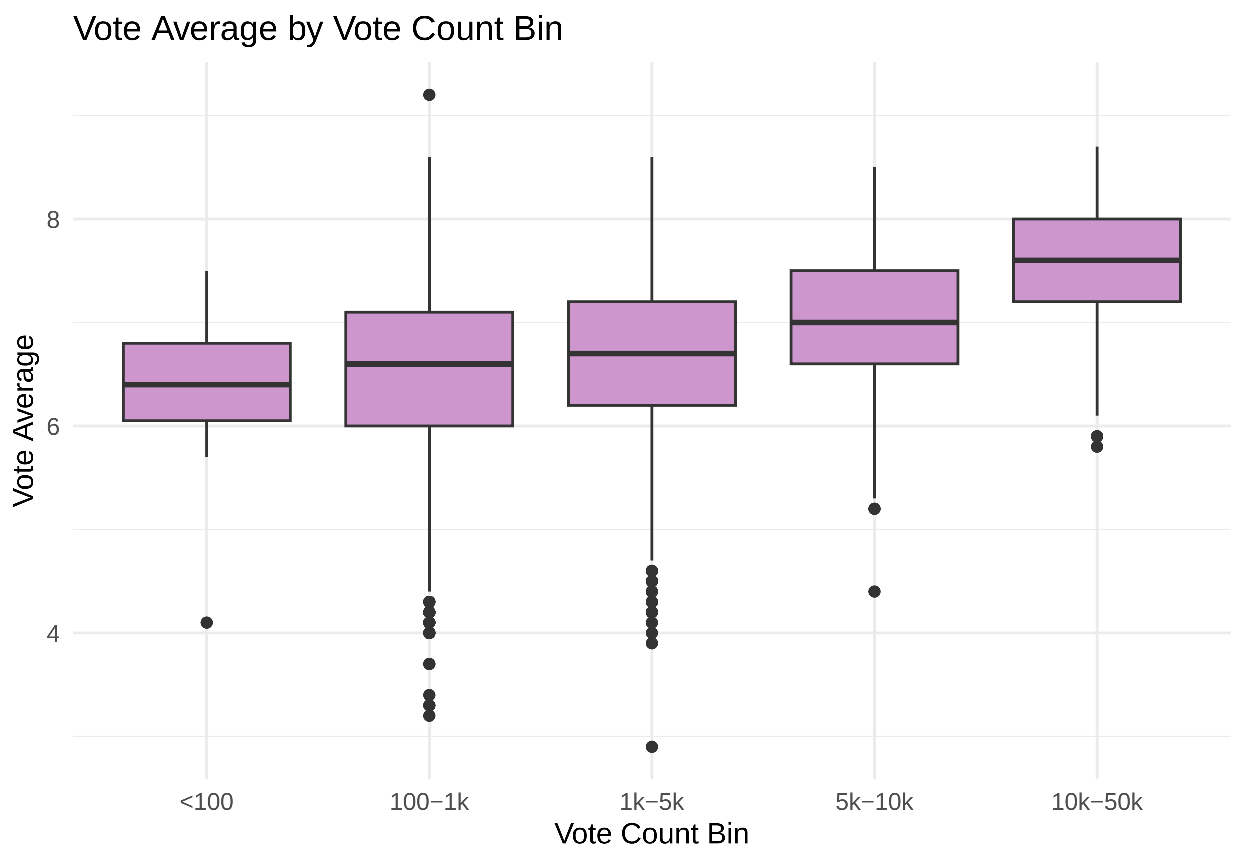
```
## Runtime Bin:  0.0995
```

Both linear regression and ANOVA indicate that *runtime* is statistically significant. However, the magnitude of its impact on *vote_average* differs between the two tests. The R^2 from linear regression is 0.06, meaning that *runtime* is a weak predictor of *vote_average* in its continuous form. In contrast, when *runtime* is binned and analyzed with ANOVA, it accounts for nearly 10% of the variance. This demonstrates that binned *runtime* is a much stronger and more meaningful predictor of *vote_average*.

Vote Count vs Average Rating

Finally, we explore the relationship between the number of votes a movie receives and its average audience rating. Since *vote_count* can reflect a movie's popularity, it's valuable to examine how audience engagement with films relates to patterns in average rating. We grouped movies into bins based on *vote_count* ranges, from fewer than 100 votes to over 50,000 votes, to identify patterns in average ratings across different levels of vote count.

```r
movie_clean %>%
  mutate(vote_bin = cut(vote_count,
                        breaks = c(0, 100, 1000, 5000, 10000, 50000, Inf),
                        labels = c("<100", "100-1k", "1k-5k", "5k-10k", "10k-50k", "50k+"))) %>%
  ggplot(aes(x = vote_bin, y = vote_average)) +
  geom_boxplot(fill = "plum3") +
  labs(title = "Vote Average by Vote Count Bin",
       x = "Vote Count Bin",
       y = "Vote Average") +
  theme_minimal()
```



We find a clear positive relationship between *vote_count* and *vote_average*, suggesting that content with more votes tends to receive higher ratings. There is also decreasing variability in ratings as vote count increases, suggesting that popular movies receive more consistent ratings.

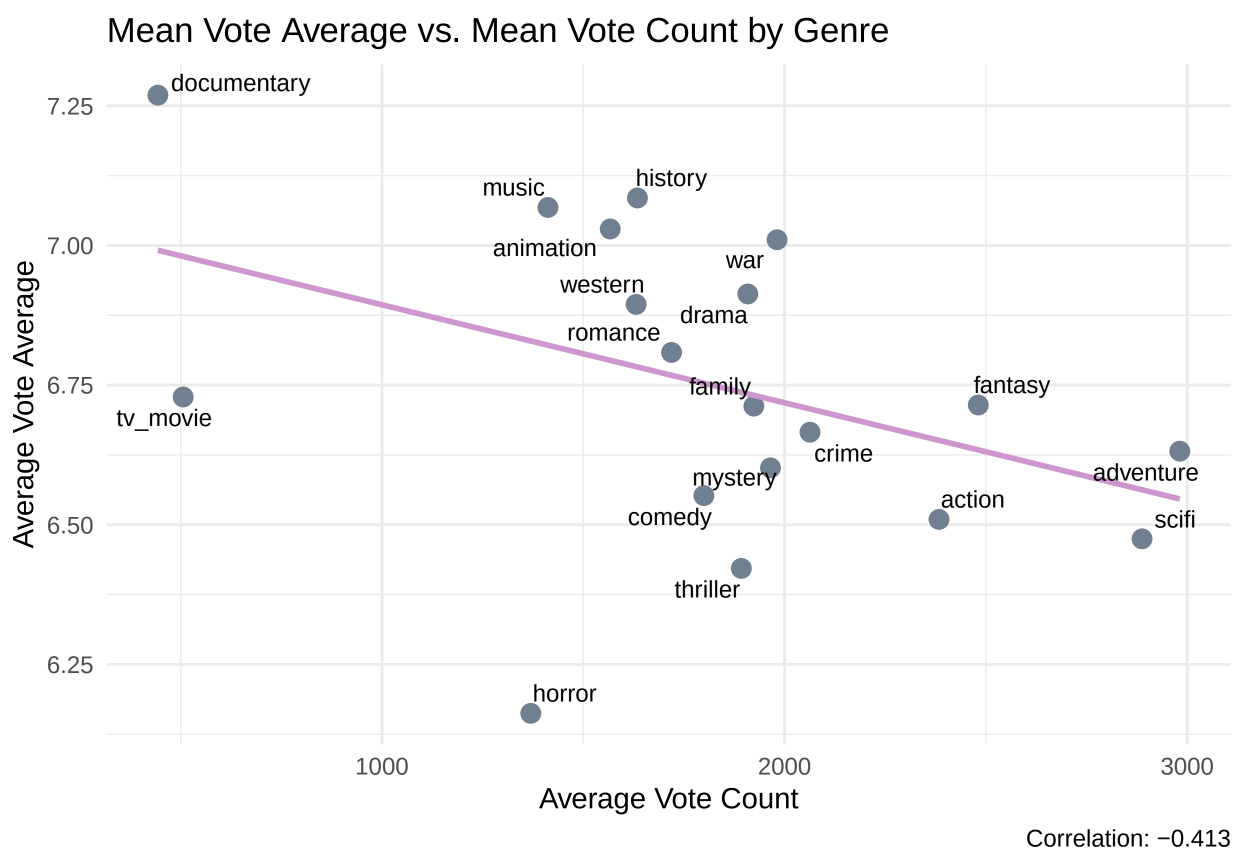
To further investigate the relationship between *vote_count* and *vote_average*, we shift our analysis from individual items to genre-level patterns. By aggregating the data at the *genre* level, we can see whether the relationship between vote quantity and ratings differs for different genre categories. We include a correlation value and regression line to quantify the strength and direction of the relationship between these metrics.

```r
# Calculate average vote count and average vote average per genre
genre_summary <- movie_genres %>%
  group_by(genre) %>%
  summarize(
    avg_vote_count = mean(vote_count, na.rm = TRUE),
    avg_vote = mean(vote_average, na.rm = TRUE)
  )

# Calculate correlation
correlation <- cor(genre_summary$avg_vote_count, genre_summary$avg_vote, use = "complete.obs")

print(paste("Correlation between mean vote count and mean vote average across genres:",
            round(correlation, 3)))
```

```
## [1] "Correlation between mean vote count and mean vote average across genres: -0.413"
```

```r
# Plot avg_vote_count vs avg_vote
library(ggrepel)  # for better labels

ggplot(genre_summary, aes(x = avg_vote_count, y = avg_vote)) +
  geom_point(color = "slategray", size = 3) +
  geom_smooth(method = "lm", se = FALSE, color = "plum3") +
  geom_text_repel(aes(label = genre), size = 3) +
  labs(
    title = "Mean Vote Average vs. Mean Vote Count by Genre",
    x = "Average Vote Count",
    y = "Average Vote Average",
    caption = paste("Correlation:", round(correlation, 3))
  ) +
  theme_minimal()
```

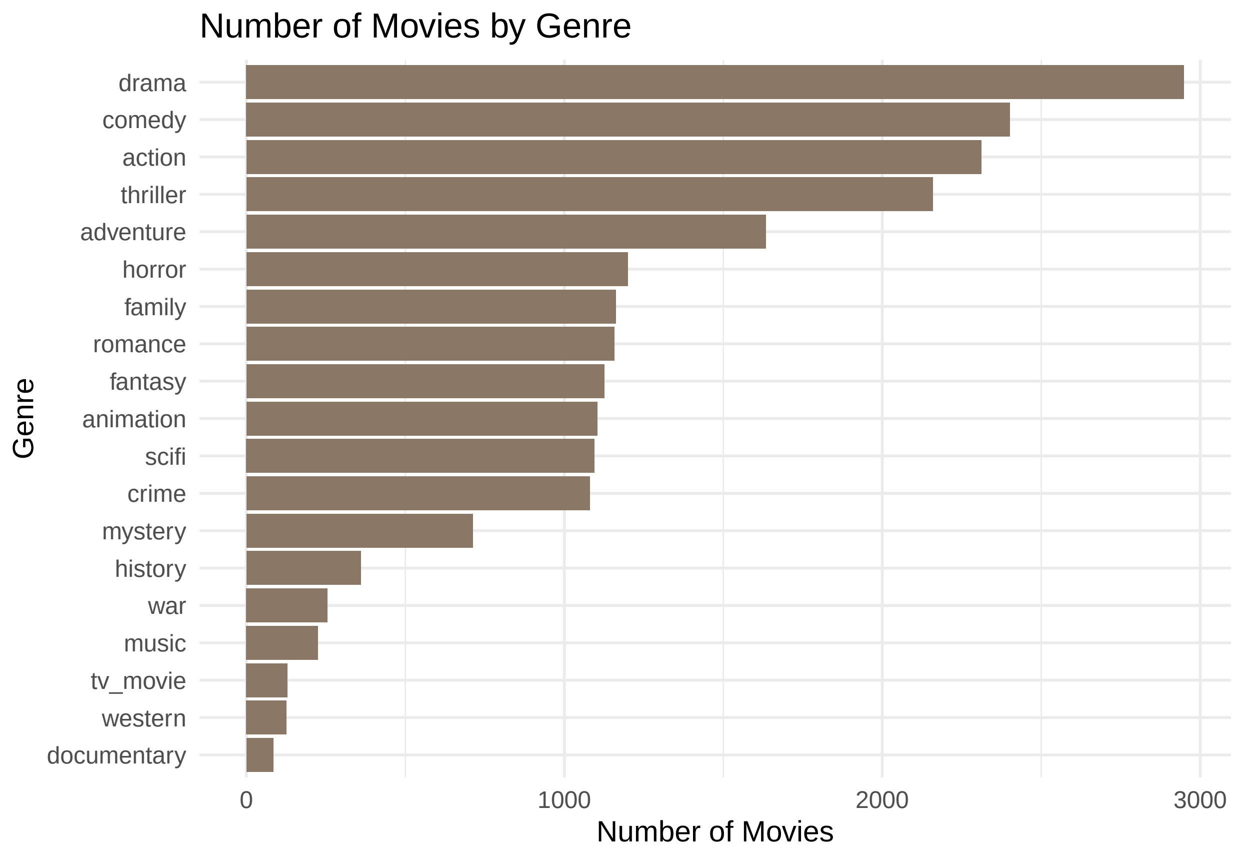
```
## `geom_smooth()` using formula = 'y ~ x'
```



In this graph which introduces *genre* distinctions within the relationship between *vote_count* and *vote_average*, we see an interesting contrast between the overall pattern observed in the previous visualization. When analyzing all movies without regard for *genre*, we learned that movies with more votes tend to have higher ratings. However, these new findings show us a moderate negative correlation (-0.413) between average vote count and average vote rating on the genre level.

To contextualize this further, we also graph the number of movies per genre within the dataset.

```
movie_genres %>%
  count(genre, sort = TRUE) %>%
  ggplot(aes(x = reorder(genre, n), y = n)) +
  geom_col(fill = "peachpuff4") +
  coord_flip() +
  labs(title = "Number of Movies by Genre",
       x = "Genre",
       y = "Number of Movies") +
  theme_minimal()
```



This distribution of movies reveals a divide between the most popular categories and more niche genres when it comes to the relationship between *vote_count* and *vote_average*. Genres with a smaller amount of movies like Documentary, History, and Music receive higher average ratings despite vote counts. Conversely, more popular genres like Drama, Action, and Adventure receive many more average votes but lower average ratings. This could be due to the fact that these more popular genres attract a wider range of quality levels even as they accumulate more total votes. Thus, genre popularity does not equate to quality. These genres are often mass-produced, leading to more vote activity but a lower average rating overall.

This genre-level pattern represents a fascinating reversal of the pattern observed for all movies earlier. At the individual film level, more votes typically signifies higher *vote_average*, but when we aggregate to the *genre* level, this relationship flips. The mass production of popular genres could reduce the quality of films and bring down the average for that particular genre.

Is this Statistically Significant?

Vote_count is a continuous numerical variable, which makes it well-suited for a linear regression analysis. Since we also binned *vote_count*, we additionally ran an ANOVA test on the binned data to compare results.

```
#Linear Regression
print(summary(lm(vote_average ~ vote_count, data = movie_clean)))
```

```
##
## Call:
## lm(formula = vote_average ~ vote_count, data = movie_clean)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -3.7338 -0.5126  0.0176  0.5545  2.6903
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) 6.499e+00  1.038e-02  625.85   <2e-16 ***
## vote_count  7.260e-05  2.818e-06   25.76   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.7763 on 7774 degrees of freedom
## Multiple R-squared:  0.07866,    Adjusted R-squared:  0.07854
## F-statistic: 663.7 on 1 and 7774 DF,  p-value: < 2.2e-16
```

```r
#ANOVA
aov_vote <- aov(vote_average ~ vote_bin, data = movie_clean)
print(summary(aov_vote))
```

```
##               Df Sum Sq Mean Sq F value Pr(>F)
## vote_bin       4    367   91.65   150.9 <2e-16 ***
## Residuals   7771   4719    0.61
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```r
cat("Vote Bin:    ", anova_variance_explained(aov_vote, "vote_bin"),       "\n")
```

```
## Vote Bin:     0.0721
```

In the linear regression analysis, *vote_count* was found to be statistically significant, but the R^2 value is quite low at approximately 0.08, indicating it has limited predictive power for vote average. The ANOVA test also shows statistical significance, supporting the findings from the regression. The proportion of variance explained by vote count bins is about 7%, suggesting a moderate effect. This implies that while raw *vote_count* isn't highly predictive on its own, binning the variable helps reveal a clearer relationship with *vote_average*.

Section 3 - Statistical Modeling & Conclusion

3.1 - Machine Learning Model

Since many of our variables showed statistical significance, we decided to explore whether a simple machine learning model could predict vote average. We use linear regression with *runtime*, *vote_count*, *budget*, *revenue*, *profit*, *ROI*, *decade*, and *genre* as predictor variables to assess the model's ability to predict the *vote_average*.

```r
set.seed(123)
sample_indices <- sample(nrow(movie_small), 0.8 * nrow(movie_small))
train <- movie_small[sample_indices, ]
test <- movie_small[-sample_indices, ]

ml_model <- lm(vote_average ~ runtime + vote_count + budget + revenue +
                      profit + roi + decade +
                      is_action + is_adventure + is_animation + is_comedy +
                      is_crime + is_drama + is_family + is_fantasy +
                      is_horror + is_romance + is_scifi + is_thriller +
                      is_mystery + is_war + is_documentary + is_western +
                      is_music + is_history + is_tv_movie,
                      data = train)

predictions <- predict(ml_model, newdata = test)
rmse <- sqrt(mean((predictions - test$vote_average)^2))
cat("Linear Regression RMSE:", round(rmse, 4), "\n")
```

```
## Linear Regression RMSE: 0.6419
```

The RMSE (Root Mean Squared Error) of our model was 0.6419, meaning that the model's predictions of a movie's rating were, on average, off by about 0.64 points. To evaluate how meaningful this level of error is, we compare it to the interquartile ranges of vote averages in both the *movie_small* and *movie_clean* datasets, providing context for how much variation typically exists in audience ratings.

```r
summary(movie_small$vote_average)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   2.900   6.100   6.700   6.651   7.200   8.700
```

```r
summary(movie_clean$vote_average)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   2.900   6.100   6.700   6.641   7.200   9.200
```

Both datasets share the same first and third quartile values for vote average—6.7 and 7.2, respectively—resulting in an interquartile range of just 0.5 points. Given that our model's RMSE is 0.6419, this means the model's average error exceeds the typical range of vote averages, suggesting that it does not predict ratings very accurately with the given variables. However we wanted to check if our model was better than a random guess and so we looked at what the RSME would be for an average guess:

```r
baseline_pred <- mean(train$vote_average)
baseline_rmse <- sqrt(mean((baseline_pred - test$vote_average)^2))
cat("Baseline RMSE (mean guess):", round(baseline_rmse, 4), "\n")
```

```
## Baseline RMSE (mean guess): 0.7895
```

The RSME here was noticeably higher at ~.79 meaning that our model does help predict vote averages even if it's not yet at an accuracy that can be applied to anything in the real world.

To better understand our model's behavior, we also examine the summary of the regression coefficients to see how each variable contributed to the predictions.

```r
coef(ml_model)
```

```
##        (Intercept)            runtime         vote_count             budget 
##       1.626892e+01       1.325309e-02       1.010580e-04      -5.033872e-09 
##            revenue             profit                roi             decade 
##      -2.867902e-11                 NA       1.836151e-04      -5.591276e-03 
##      is_actionTRUE   is_adventureTRUE   is_animationTRUE      is_comedyTRUE 
##                 NA                 NA                 NA                 NA 
##       is_crimeTRUE       is_dramaTRUE      is_familyTRUE     is_fantasyTRUE 
##                 NA                 NA                 NA                 NA 
##      is_horrorTRUE     is_romanceTRUE       is_scifiTRUE    is_thrillerTRUE 
##                 NA                 NA                 NA                 NA 
##     is_mysteryTRUE         is_warTRUE is_documentaryTRUE     is_westernTRUE 
##                 NA                 NA                 NA                 NA 
##       is_musicTRUE     is_historyTRUE    is_tv_movieTRUE 
##                 NA                 NA                 NA 
```

```r
summary(ml_model)$coefficients
```

```
##                  Estimate   Std. Error     t value      Pr(>|t|)
## (Intercept)  1.626892e+01 1.509928e+00  10.7746366  1.170863e-26
## runtime      1.325309e-02 5.472054e-04  24.2195821 4.978402e-120
## vote_count   1.010580e-04 4.139294e-06  24.4143079 8.555278e-122
## budget      -5.033872e-09 3.433152e-10 -14.6625385  2.665915e-47
## revenue     -2.867902e-11 9.075634e-11  -0.3160002  7.520210e-01
## roi          1.836151e-04 3.569686e-04   0.5143732  6.070232e-01
## decade      -5.591276e-03 7.519669e-04  -7.4355346  1.299709e-13
```

As seen in the model summary, *genre* did not contribute to the predictions—each genre-related variable returned an NA coefficient. Among the remaining variables, the summary provides helpful insight into their contributions. Based on the t-values, *runtime*, *vote_count*, *budget*, and *decade* had a statistically significant impact on the model, while *revenue* and *roi* did not appear to contribute meaningfully in addition to the others.

Looking at the coefficients, *runtime*, *vote_count*, and *roi* had positive values, suggesting that as these variables increase, the predicted *vote_average* also tends to increase—indicating they may be associated with higher audience ratings. In contrast, *budget*, *revenue*, and *decade* had negative coefficients, meaning higher values in these variables were associated with lower predicted ratings.

Although the model's predictive accuracy was limited, with a much higher than ideal RMSE, this analysis still provides interesting insight into which variables were most aligned with audience ratings and in what direction they influenced the predictions.

3.2 Conclusion

Through our exploration and modeling, we identified several noteworthy relationships between film attributes and their average audience ratings. While many of the variables we tested were statistically significant, their individual impact on *vote_average* was generally moderate to weak. For instance, *runtime* and *vote_count* showed a consistent positive relationship with *vote_average*, but the effect sizes were small. *Genre, roi*, and *budget* also demonstrated some influence when binned or grouped, but as standalone variables, they struggled to explain much of the variation in ratings. Our linear regression model, which included multiple variables, achieved an RMSE of approximately 0.64. While this indicates some predictive ability, it also suggests there is a limit to how well these features alone can predict a movie's rating.

3.3 Reflection & Future Directions

To improve this analysis in the future, we would consider incorporating a larger and more diverse dataset—such as the TMDB 1 Million Movies dataset from Kaggle—to mitigate selection bias toward more popular or mainstream films. Access to a more powerful computing environment would also allow us to process larger datasets more efficiently and apply computationally intensive modeling techniques. Additionally, exploring more advanced machine learning models could help capture nonlinear relationships and interactions that simple linear models may miss.

Ultimately, our findings suggest that while measurable variables like *vote_count* and *runtime* contribute to audience ratings in meaningful ways, film reception likely depends on a combination of nuanced and less quantifiable factors that extend beyond what we could capture with this dataset.