STSCI 4100: IMDb Analysis

Tony Oh (do256), William Rhee (wr86)
May 16, 2025

TABLE OF CONTENTS

[1] INTRODUCTION:

IMDb documents reviews of released movies and films in non-commercial datasets — these datasets contain vast amounts of metadata regarding viewer ratings, genres, release years, and more. In this project, we want to understand if these factors dictate any trends in film popularity and genre success.

In this project, we will be using four datasets from IMDb:

1. **title.basics.tsv**: A dataset providing all the basic details about a broad range of films from 1892 to 2026.
2. **title.crew.tsv**: A dataset providing the ID's specifically of the director and writers for the movies from Dataset 1.
3. **title.principals.tsv**: A dataset providing details about other crew (editors, producers, etc.) for the movies from Dataset 1.
4. **title.ratings.tsv**: A dataset that contains the rating data from IMDb's users for the movies from Dataset 1,

Our analysis focuses on a **wide range of films** — from shorts, TV series, movies, and even video content — in the period **1892 to 2026**. We will be answering **four questions** to explore trends in ratings, creative roles, and genres across the whole IMDb dataset.

1. **Ratings**: Does the number of votes per film affect the ratings?
2. **Creative role (writer)**: Does having a writer affect the ratings?
3. **Actor roles**: Does having certain actors have an affect on ratings?
4. **Genres**: Which genres get the highest ratings?

---

[2] DATA EXPLORATION:

There are four points in this section:

A. Importing the Datasets.
B. A Quick Glance.
C. Specific Breakdowns.
D. Combining the Datasets.

[A] Importing the datasets.

Firstly, to reproduce this project, clone our **GitHub repository**. Then, open this file (`analysis.Rmd`) and set your working directory to this project's root directory.

We provide two methods in importing and saving the datasets.

1. Download the exact RDS files we worked on. This method is the most accurate with our analysis. To access these files, download the **four RDS files**. When downloaded, place the four RDS files into the folder `./rds` of the project's root directory.

- **IMPORTANT NOTE**: Note that this set of datasets is a result of subsetting the rows of each dataset in `import_save.Rmd` to match the `tconst` values of `ratings` since `ratings` is the limiting dataset in number of rows. That is, we deemed `ratings` to hold information about films that is necessary for all of our analysis. Thus, we removed rows in `basics`, `crew`, and `principals` that represented films that were not represented in `ratings`. A majority of films in `basics`, `crew`, and `principals` that **did not have ratings** were removed according to the films available in `ratings`. More detail about how this choice was executed can be found in `import_save.Rmd`.

2. Download the updated IMDb dataset from the IMDb source. This method may not align with the results we found in our analysis. To use this method of retrieving the datasets, follow the instructions detailed in `import_save.Rmd`.

Once you have the RDS files set up in the `./rds` directory, read in the RDS files. Instructions on this process can be found in `analysis.Rmd`.

[B] A Quick Glance.

As a quick reference, here are the datasets we will be looking at:

Dataset	Observations	Variables
`basics`	1,568,336	9
`crew`	1,568,333	3
`principals`	21,693,507	6
`ratings`	1,568,336	3

[C] Specific Breakdowns.

[C1] The basics dataset

The `basics` dataset and the takeaways from our analysis on it:

- **tconst**: The unique ID for a specific film. There are no missing values. There are exactly 1,568,336 ID's (i.e. individual films), one for every row.
- **titleType**: The kind of film — it can be a short, movie, tvShort, and more. There are no missing values. There are 10 unique kinds of films in this dataset, ranging from shorts to video games.
- **primaryTitle**: The popular title of the film. There are no missing values. There are 1,158,575 primaryTitles in the dataset ($<$ 1,568,336).
- **originalTitle**: The original title of the film. There are no missing values. There are 1,176,721 originalTitles in the dataset ($<$ 1,568,336 and $>$ 1,158,575).
- **isAdult**: The indicator variable — 1 if it's an adult film, 0 otherwise. There are no missing values. Approximately 1.56% of all the films in the dataset are adult films.
- **startYear**: The film's release year. There are missing values! Ignoring missing values, the earliest release date was 1874, while the latest release date was 2025.
- **endYear**: The film's ending year, IF the film was a TV show (NA otherwise). There are missing values! Ignoring missing values, the earliest end date was 1932, while the latest end date was 2030.
- **runtimeMinutes**: The film's runtime (in minutes). There are missing values! Ignoring missing values, the shortest film is 0 minutes long, while the longest film is 3,692,080 minutes (roughly 61,534 hours) long. The average length of films was 58.39153 minutes.
- **genres**: The film's list of genres. There are missing values!

[C2] The crew dataset

The `crew` dataset and the takeaways from our analysis on it:

- **tconst**: There are no missing values. There are exactly 1,568,333 ID's (i.e. individual films), one for every row. Note that this value is minorly less than the number of values in `basics` and `ratings`.
- **directors**: The film's list of directors. There are some missing values. This means not every movie in our data will have a specified director.
- **writers**: The film's list of writers. There are some missing values. This means not every movie in our data will have a specified writer.

[C3] The principals dataset

The `principals` dataset and the takeaways from our analysis on it:

- **tconst**: There are no missing values. However, the number of unique film ID's does not equal the number of rows — this is because upon close inspection, we can see that there are duplicated ID's across several rows. This is different from the other three datasets, where there is one unique film ID for each row.
- **ordering**: The order in credits. There are no missing values. The ordering from most to least important ranges from 1 to 75.
- **nconst**: The unique ID for a specific person such as an actor, director, and more. There are no missing values.
- **category**: The role of a specific person such as an actor, director, and more. There are no missing values. There are 13 unique categories in the dataset, from director to casting director.
- **job**: The job title such as a producer, editor, and more. There are missing values — this is

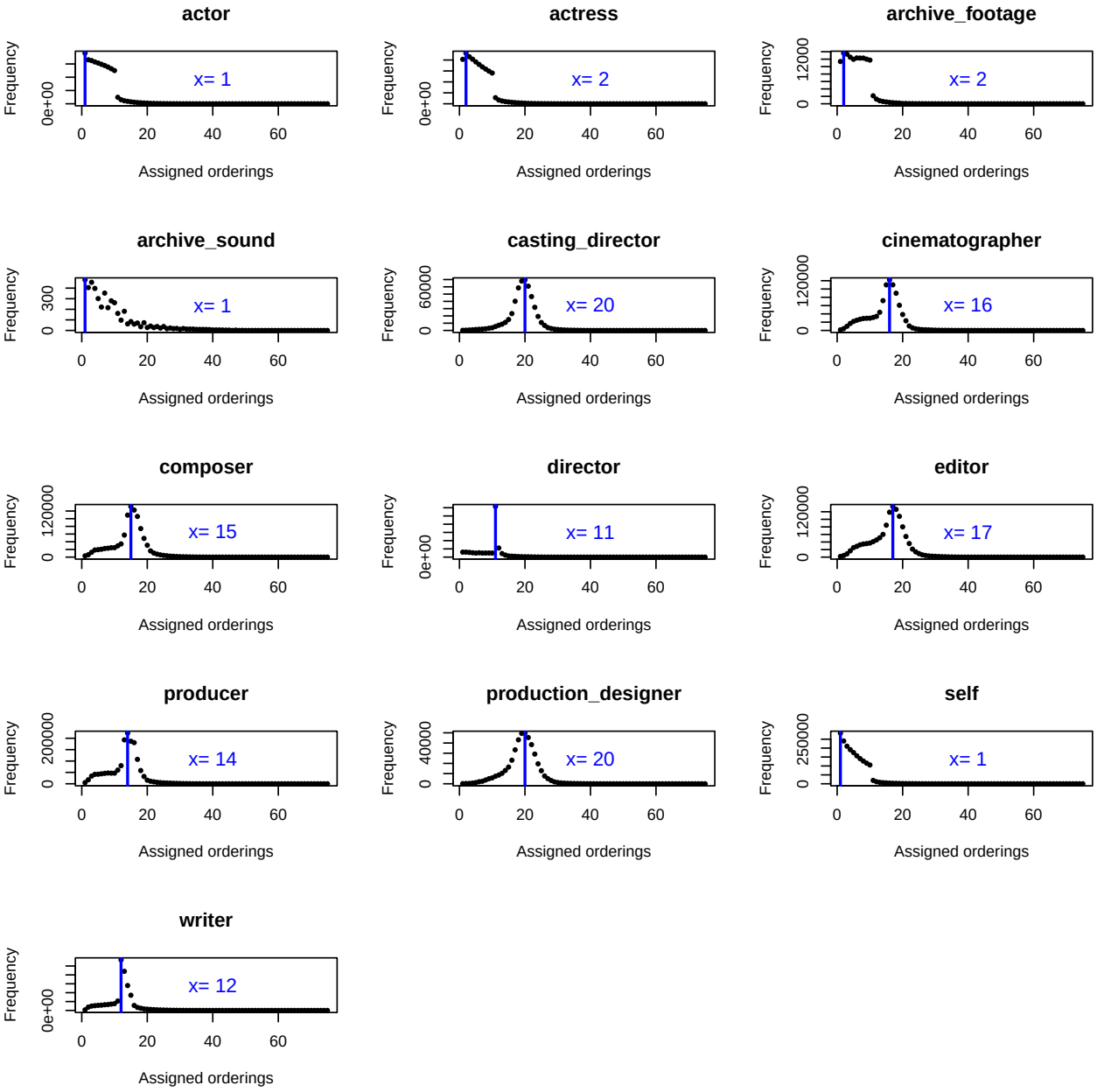
because of the duplicate values from the first column. There are 30,167 unique values within this column.

- **characters**: The character that a specific person has played as. There are missing values — this is because of the duplicate values from the first column. There are 2,507,211 unique values within this column.

A side-note

We know that there are 13 unique categories within the **category** column. We also know that the ordering ranges from 1 to 75, where 1 is most important and 75 is least important.

So as a SIDE-NOTE, let's make a DataFrame (and plots) that count how many times each job **category** appears in a specific **ordering**.



As we can see, the code matches what we'd expect if we eyeball what the code is doing to `my.table`.

[C4] The ratings dataset

The 'ratings" dataset and the takeaways from our analysis on it:

- **tconst**: There are no missing values. There are exactly 1,568,336 ID's (i.e. individual films), one for every row.
- **averageRating**: The average rating, which was given to reviewers on a 1-10 scale. There are no missing values. The worst review ever given was 1/10, while the best were 10/10. The average rating across movies from the 1890's all the way to present day is roughly 6.95.
- **numVotes**: The number of reviewers. There are no missing values. Of all 1,568,336 films, 37,066 films received the lowest number of reviews (only 5 reviews). Of all 1,568,336 films, only one film (in the 84,878th row) has the highest 3,042,484 reviews.

D. Combining the Datasets.

Let's combine the four datasets:

1. The basics dataset.
2. The crew dataset.
3. The principals dataset.
4. The ratings dataset.

```
# COMBINING DATASETS:
# [1] Combining basics, crew, and ratings into one data table.
full_dt <- basics[crew, on = "tconst"]
full_dt <- full_dt[ratings, on = "tconst"]

# [2] Checking the dimensions (rows, col).
dim(basics)      # This is correct - it should be (1,568,336 rows, 9 columns).
dim(crew)        # This is correct - it should be (1,568,333 rows, 3 columns).
dim(ratings)     # This is correct - it should be (1,568,336 rows, 3 columns).
dim(full_dt)  # This is interesting - we expected (1,568,333 rows, 13 columns).
# This is because crew is a limiting dataset as it has less rows than others.
```

As we can see:

1. We are able to merge `basics` and `ratings` based on the first column. This is because all of their values within the first column match.
2. However, our final dataset contains 1,568,336 rows. It has the right number of columns, but it seems that we are **not** being bottlenecked by the number of rows in crews. This may be a problem when we try to analyze the crew data but some are missing from `full_dt`.

Let's run diagnostics to see why this is happening.

```
# DIAGNOSTICS: [PART 1]
# [1] Check matching of basics and ratings
# Check: Are all tconst values in ratings represented in basics?
all(ratings$tconst %in% basics$tconst) # TRUE. That is good.
# Check: Do all tconst values in basics have ratings data?
all(basics$tconst %in% ratings$tconst) # TRUE. That is good.
```

```
# [2] Check matching of basics and crew
# Check: Are all tconst values in crew represented in basics?
all(crew$tconst %in% basics$tconst) # TRUE. That is good.
# Check: Do all tconst values in basics have crew data?
all(basics$tconst %in% crew$tconst) # FALSE. That may be an issue.
# [3] Check matching of basics and crew
# Check: Are all tconst values in principals represented in basics?
all(principals$tconst %in% basics$tconst) # TRUE. That is good.
# Check: Do all tconst values in basics have principals data?
all(basics$tconst %in% principals$tconst) # FALSE. That may be an issue.
```

Here, we found out that both `crew` and `principals` lack `tconst` values compared to `basics`. That means we may have to trim all datasets to match that of `crew` when doing analysis that requires all datasets.

However, we have checked that `basics` and `ratings` match in its `tconst` values. This is due to the fact that we initially subsetted all datasets based on `ratings`. `basics` had `tconst` values for every film and was simply subsetted by `ratings`.

Since we have found a mismatch between `crew`, `principals` and the two other datasets, we will subset those datasets once again to provide consistency throughout our data.

```
# DIAGNOSTICS: [PART 2]
# [1] Finding bottlenecking dataset
length(unique(basics$tconst))     # 1568336. This is expected.
length(unique(crew$tconst))       # 1568333. This could be a bottleneck.
length(unique(principals$tconst)) # 1542357. This seems like the real bottleneck.
length(unique(ratings$tconst))    # 1568336. This is expected.
```

After subsetting the rows of each dataset to match that of the unique `tconst` values in principals:

```
# [3] Check matching
length(unique(basics$tconst))     # 1542357
length(unique(crew$tconst))       # 1542357
length(unique(principals$tconst)) # 1542357
length(unique(ratings$tconst))    # 1542357
```

Now, all the datasets have the same set of unique `tconst` values.

However, by the nature of the `principals` dataset, we can't store its multiple rows for each `tconst` value into one row in our final data table.

```
# DIAGNOSTICS: [PART 3]
# [1] Check whether the datasets can be merged
length(basics$tconst)     # 1542357. This is expected.
length(crew$tconst)       # 1542357. This is expected.
length(principals$tconst) # 21693507. This needs to be fixed.
length(ratings$tconst)    # 1542357. This is expected.
```

Clearly, the principals dataset needs to be cleaned before we can use it with the rest of our datasets. Since each `tconst` value can hold information of multiple rows, we combine those four rows into a

single row by modifying the other columns into array form that holds in order the original values. More details on this can be found in the actual provided code.

```
##    ordering    nconst        category                     job characters
##       <int>    <char>          <char>                  <char>     <char>
## 1:        1 nm1588970            self                    <NA>   ["Self"]
## 2:        2 nm0005690        director                    <NA>       <NA>
## 3:        3 nm0005690        producer                producer       <NA>
## 4:        4 nm0374658 cinematographer director of photography       <NA>
```

```
## [1] "[\"Self\"]"
```

Now that the principals dataset is cleaned, we can safely merge all our datasets into one final dataset `full_dt`.

```
# DIAGNOSTICS: [PART 4]
full_dt <- basics[crew, on = "tconst"]
full_dt <- full_dt[principals, on = "tconst"]
full_dt <- full_dt[ratings, on = "tconst"]
nrow(full_dt) # 1,542,357. This is good.
```

---

[3] DATA ANALYSIS:

Now that we have gone through the datasets and obtained a final data table `full_dt` to work with, we'll tackle **five questions** starting with the most intuitive one first.

QUESTION 1:

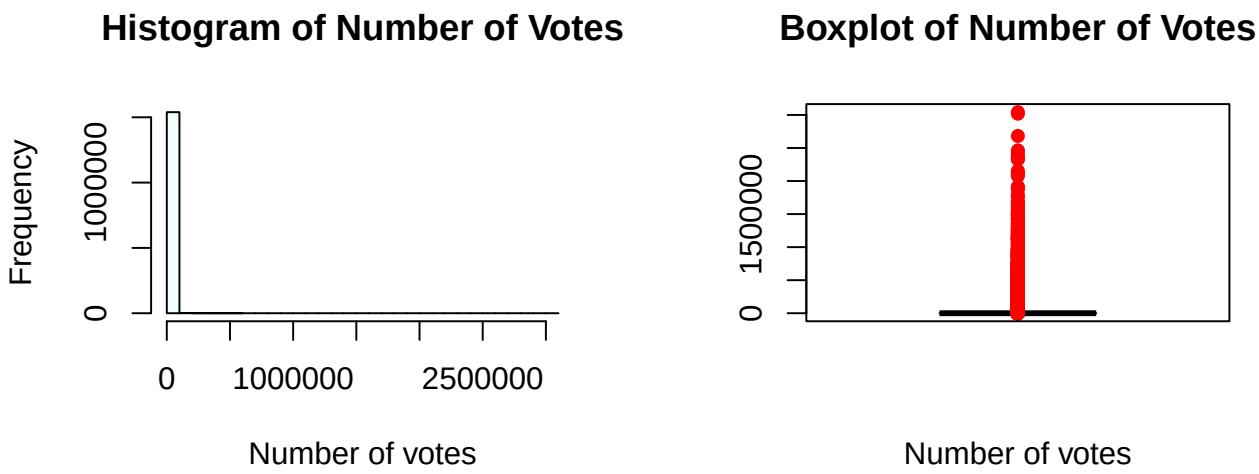
Our first question is: **does the number of votes affect the ratings?**

In the real world, many people want to know how good a movie is before watching it. To do so, they go online to check the movie's reviews. But here's the thing everyone suspects:

- Films that have very few votes (i.e. unpopular films) may not have the most reliable ratings, as their reviews don't contain much information.
- Films that have lots of votes (i.e. popular films) may suggest that the movie has larger positive reception or audience appeal.

Specifically, we'll investigate whether or not films that have more votes tend to have higher or lower ratings (i.e. whether popularity is correlated with perceived quality).

This will provide a good opening introduction, and hopefully provide some intuition, to our analysis. Let us first look at the distribution of `numVotes`.

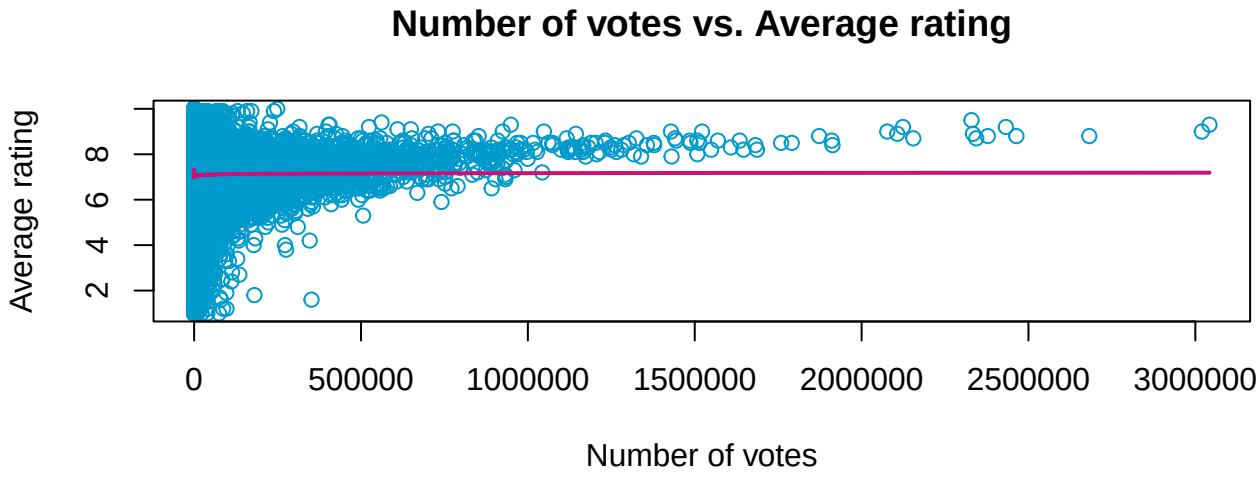



```
##     Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##        5      12      26    1040     102 3042484
```

Outliers are colored red, and we see that the box plot deems most of the visible points as outliers. The information so far suggests a heavy right skew.

As we can see, the number of votes in our dataset is extremely right-skewed, where:

- Most movies have very few votes.
- Few movies have lots of votes.
- So far, the data doesn't seem to follow a strictly linear pattern. The data seems to follow some sort of nonlinear pattern, however. Let us fit a local regression model — a non-parametric statistical method — to visualize the trajectory when we logarithmize `numVotes` (x).



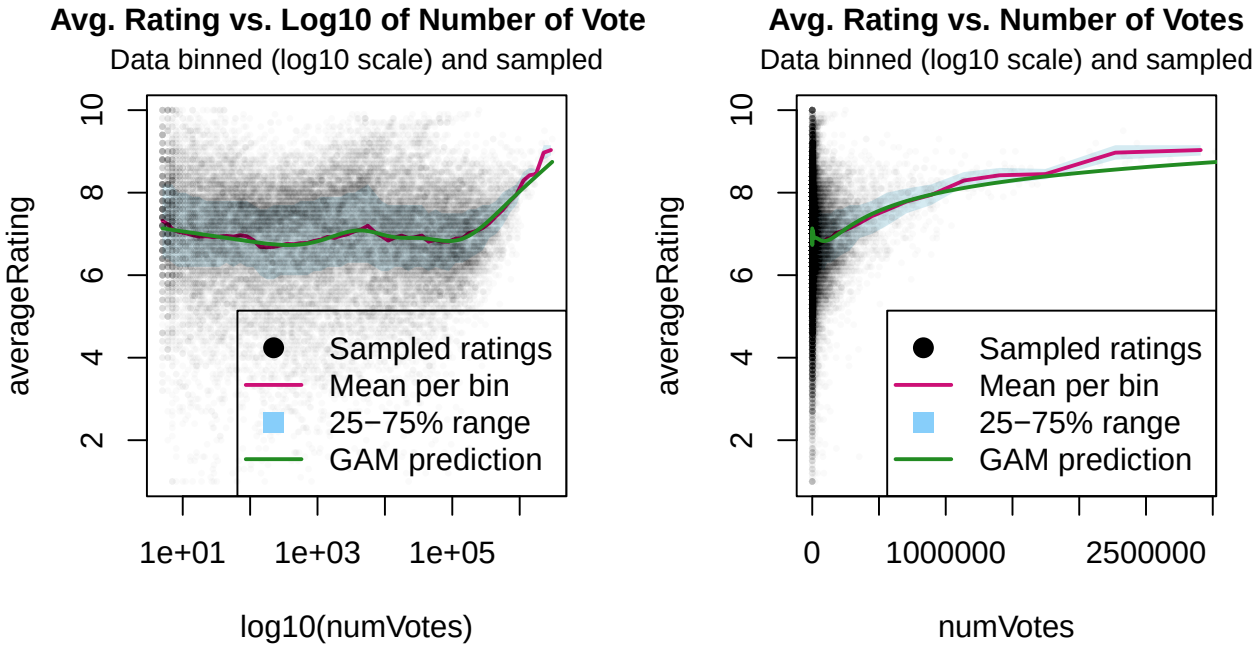
The local regression model doesn't seem to give us a helpful fitted line on the trend of `averageRating` for each film. Let us try plotting it again but also bin the logarithm of the x-axis and sample a number of values rather than relying on the multitude of values heavily focused on the left side of the graph.

By binning and sampling, we mean we will split the x-axis (log10 of `numVotes`) into bins of a certain width (0.1 in our case) and sample a number of films (500 in our case) within each bin for our analysis.

Then, we fit a Generalized Additive Model (GAM) to capture the non-linearity between `numVotes`

and `averageRating` — in our case, modeling how averageRating changes with the logarithmic number of votes `log10(numVotes)`. We hope that this approach gives us a more stable and interpretable fit than LOWESS when dealing with our data.

Let us plot all of this twice both with the x-axis as the logarithmic number of votes and with the x-axis as the non-logarithmic number of votes.



The left plot shows strong evidence that as the number of votes increases, a film's `averageRating` fluctuates until it has a steep increase for its rating if the film gets an extremely high number of votes.

In the second plot, we again see a similar trend where a majority of the films with less than 500,000 votes fluctuate heavily in `averageRating`, but as the number of votes increases steeply, the average rating values also gradually stabilize and increase. Note that this plot follows the same GAM as before on the logarithmic `numVotes` variable.

Let us look at the summary of our GAM:

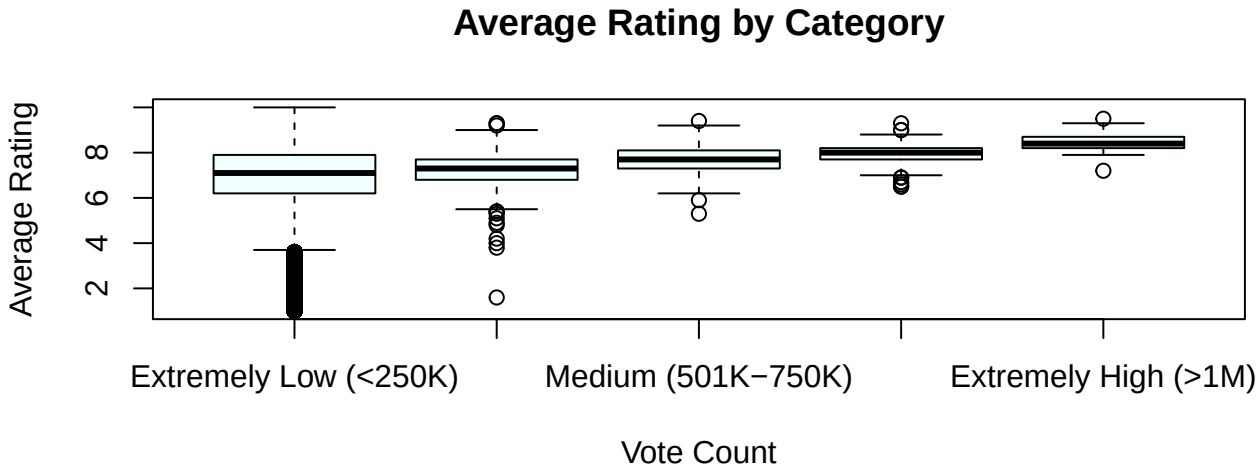
```
## Adjusted R-squared:  0.007398515 ,  Deviance explained:  0.7404043 %
```

```
## Smooth term significance:
```

```
##                edf   Ref.df        F p-value
## s(x_log) 8.590076 8.923249 1288.894       0
```

The GAM holds a p-value less than 2e-16. Seeing our plots, we see a stable horizontal line for its prediction line until higher values of `numVotes`. Thus, we have good evidence from the extremely low p-value that this trend is true. Additionally, it seems that the average rating for a film with not too extremely large number of votes is about 6.95017.

Lastly, here is a boxplot in support of our analysis.



The apperance of this box plot follows the same idea that higher values of vote counts indicate higher average ratings, only if extremely high.

Here's what we can take away from this analysis:

1. The plotted graph of average ratings against the number of votes exhibits some nonlinear pattern.
   - Specifically, films that have less votes tend to have a wider spread of average ratings, roughly between 2 to 8.
   - Specifically, films that have more votes tend to have a smaller spread of average ratings.
2. As the number of votes increases, the average ratings stays stable at around 7-8 before increasing as a film gets a much larger number of votes.
3. Movies with extremely high votes (at least a million) have narrow/tight distributions with a much higher average rating than the others. This suggests that the number of votes a film receives can act as an indicator of the film's resulting ratings, only if the number of votes (`numVotes`) is extremely high.

QUESTION 2:

The next question we want to answer is: **does having a writer affect the ratings?**

Let's take a step back to understand why we're even asking this question. Upon a quick glance at the dataset, we can see that its column **writers** is filled with a mixture of filled and missing values. In other words, not every film has a specified writer.

As a result, it naturally follows to ask how this would affect the film's ratings. We first created a new column that represented in binary whether the film has a writer or not.

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  0.0000  1.0000  1.0000  0.7758  1.0000  1.0000
```

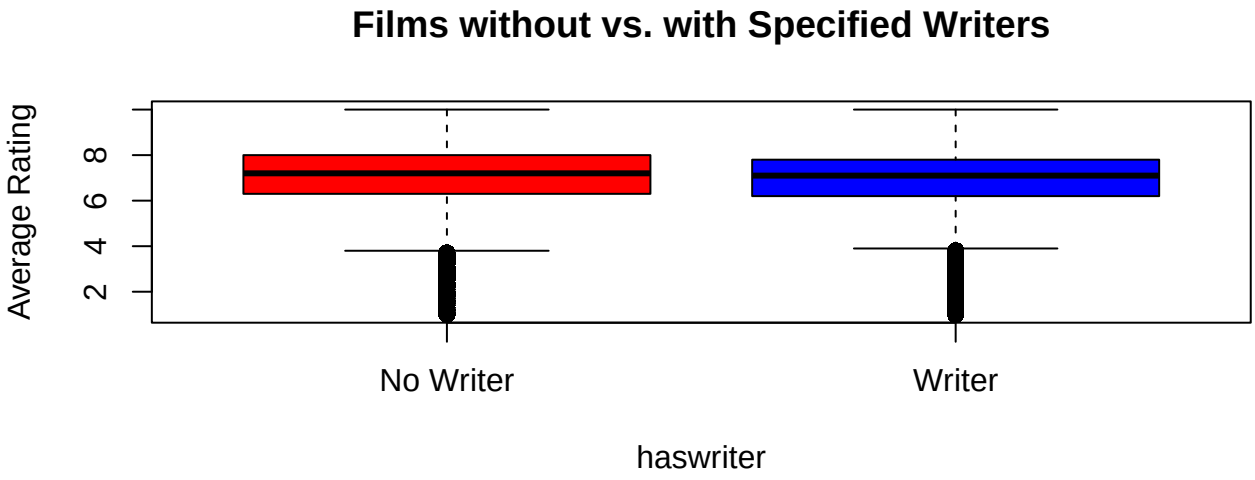
```
## [1] 0.7757672
```

Here, the min and max are obviously 0 and 1. The mean of the column `haswriter` is 0.7758. This means that roughly 77.6% of our films in the dataset have a specified writer.

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    1.00    6.20    7.10    6.95    7.90   10.00
```

```
## [1] 6.950173
```

The worst films have an average rating of 1/10. The best films have an average rating of 10/10. The **average** average rating, as mentioned before, is 6.95/10.




```
## Mean (No Writer): 7.024177 ,  Mean (Writer):    6.928782
```

```
## p-value:          <2e-16 ,  95% CI:          ( 0.09002716 , 0.1007628 )
```

Here's our takeaway from our analysis:

1. In our comparison of films that have specified and unspecified writers, we see that their boxplots are identical. Their distributions are relatively the same with identical means around 7.0.
2. There visually doesn't seem to be much of a difference between their means.
3. Just to be safe, we conducted a two-sample t-test. This is where things got interesting:
    - The p-value is less than 2.2e-16, which is significant.
    - Technically, this means we reject the null and conclude there is enough evidence to suggest there is a statistically significant difference between these two means.
    - But this obviously does not align with our findings from the dataset.
    - In addition, the t-test's results shows us that the mean for "No Writer" is 7.024177 while the mean for "Writer" is 6.928782. Those means are incredibly close and identical in the **practical** sense. We simply do not care whether a film has a rating that is 0.095395/10 higher than another.
4. To get around this caveat, we asked ourselves two questions:
    - How could a film not have writers? They do; it's just that some films in the dataset have no specified writers because of data incompleteness, especially for really old or lesser-known films where the writers may have not been well documented.
    - Do all audiences necessarily care about who the writers are for an upcoming film? Unless it's a well-known writer who is mentioned in the trailers, probably not! They likely pay more attention to the director(s) and actors.
    - This is an example where the context of the situation need to be considered, not just a statistical test's dry facts.

All in all, having a writer or not in the given dataset does not seem to affect the ratings of a film.

QUESTION 3:

Our third question is: **"Does having certain actors have an affect on ratings? Or is this just an effect of hiring good actors for good movies?"**

To answer this question, we decided to use the following columns from the full dataset `full_dt`:

- `category`: Each entry specifies roles for each film, such as "actor", "director", etc. Note that unfortunately, not every entry has specified actors ("actor", "actress").
- `tconst`: Each entry specifies a specific movie.
- `averageRating`: Each entry gives the average rating of movies with actor information.

To begin, we **filtered out** the full dataset `full_dt` to only have entries where the **category** column does have specified actors ("actor", "actress").

```
## [1] 1226885      17
```

Here is what we know about our filtered dataset `actors.dataset`:

1. It contains 1,226,885 observations and 17 variables.
2. In other words, out of all given 1,542,357 films, only 1,226,885 of them contain actor information while the rest didn't.
3. A film can have **more than one** actor specified within the **category** column.

Note: Although the filtered dataset does contain movies that have actor information, it unfortunately does not **not always** specify what role the actor played. Consequently, we can't always tell if some particular actor played a big role or not.

That being said, let's conduct our analysis for question 3. Our analysis is broken down into **four observations**, as listed below.

Observation 1. Extreme Ratings

After cleaning the dataset, we define a DataFrame that displays a list of actors who starred in films ordered from highest to least average weighted ratings. This DataFrame is shown below:

```
##        nconst weighted.average.rating total.votes number.of.films
##        <char>                   <num>       <int>           <int>
## 1: nm11000153                      10          23               1
## 2:  nm2436198                      10          11               1
## 3:  nm1449677                      10          11               1
## 4:  nm1451968                      10          11               1
## 5:  nm1451969                      10          11               1
## 6:  nm1450118                      10          11               1
```

As we can see, actors who starred in films with the top ratings (like 10/10) were actors who, **at least according to our dataset**, starred in only a **single film**.

Clearly, this doesn't tell us very much.

- Why? Because an actor who only starred in only one film that got a 10/10 review will obviously have a weighted average film rating of 10/10. That clearly doesn't say much in our analysis.

- If an actor who say starred in 1000 films still ended up getting a top weighted rating of 10/10, then that would mean significantly more.

Observation 2. Cautionaries

To get our feet even further wet, let's check out two completely arbitrary and specifically chosen actors: Bobby Connelly (1909-1922) and Jerold T. Hevener (1873-1947). Their IMDb ID's are "nm0175050" and "nm0381936," respectively.

Here are their values from the DataFrame created in observation 1:

```
##       nconst weighted.average.rating total.votes number.of.films
##       <char>                   <num>       <int>           <int>
## 1: nm0381936                7.045506         178              10
## 2: nm0175050                6.464286         910              13
```

Now let's explain what these results are saying. Bobby Connelly was a famous child-actor who unfortunately passed away young. **According to the dataset**, he starred in 13 films, which seems reasonable given his short career.

However, there seems to be some mistake in the dataset. **According to the dataset**, Jerold T. Hevener was an actor who starred in 10 films. In actuality, he starred in around 36 films.

This result tells us that we should approach these numbers with caution — the dataset isn't guaranteed to give us back an accurate number of films some actor starred in.

Observation 3. Converging Ratings

In the previous section, we investigated two very completely arbitrary actors.

Now, let's investigate a super specific actor — the one who has starred in the **most** number of films in our dataset.

Filtering for the maximum number of films within our dataset, we get the following result:

```
##       nconst weighted.average.rating total.votes number.of.films
##       <char>                   <num>       <int>           <int>
## 1: nm0048389                7.964585    12753444            8230
```

Here, the actor who starred in the most number number of films is **Dee Bradley Baker** (ID: "nm0048389"). He is an extremely well-known voice actor who voices in many films and TV cartoons ever since the early 2000's.

According to this dataset, all the 8,230 films he voiced in have an average weighted rating of 7.964585 out of 10 based on 12,753,444 votes. That is a relatively high rating with a large sample voting size.

But here's the natural: does this one actor's situation imply that if you're an actor who starred in **many** films that received **large voting sizes**, then your films will have an overall higher weighted average rating?

To investigate this catch, we decided to **deliberately** investigate other extremely famous actors who we know also starred in many movies. Specifically, we investigated the following actors (with their IDMb ID's obtained from a quick **Google search**):

1. Leonardo DiCaprio; ID: "nm0000138"
2. Morgan Freeman; ID: "nm0000151"
3. Tom Hanks; ID: "nm0000158"
4. Robert Downey Jr; ID: "nm0000375"
5. Scarlett Johansson; ID: "nm0424060"

```
##       nconst weighted.average.rating total.votes number.of.films
##       <char>                   <num>       <int>           <int>
## 1: nm0000138                7.998845    18761495              82
## 2: nm0000151                7.914311    19659977             240
## 3: nm0000158                7.775600    19522793             210
## 4: nm0000375                7.713361    22081110             186
## 5: nm0424060                7.619085    25317458              94
```

```
## The overall mean of the entire dataset's average rating is 6.928019.
```

As we can see, these chosen famous actors have weighted average ratings between 7.6 and 8, based off of **millions** of votes and **hundreds** of films.

Their weighted average ratings are **slightly higher** than the dataset's global average, which is around 6.928019 out of 10. This suggests that these high-profile actors do tend to appear in **well-received, widely watched** films.

Now here's the thing: as humans, our intuition generally expects high-profile actors to have extremely high ratings such as 9/10 or 10/10. But as we can see, these famous actors' ratings are not necessarily extremely high in the dataset.

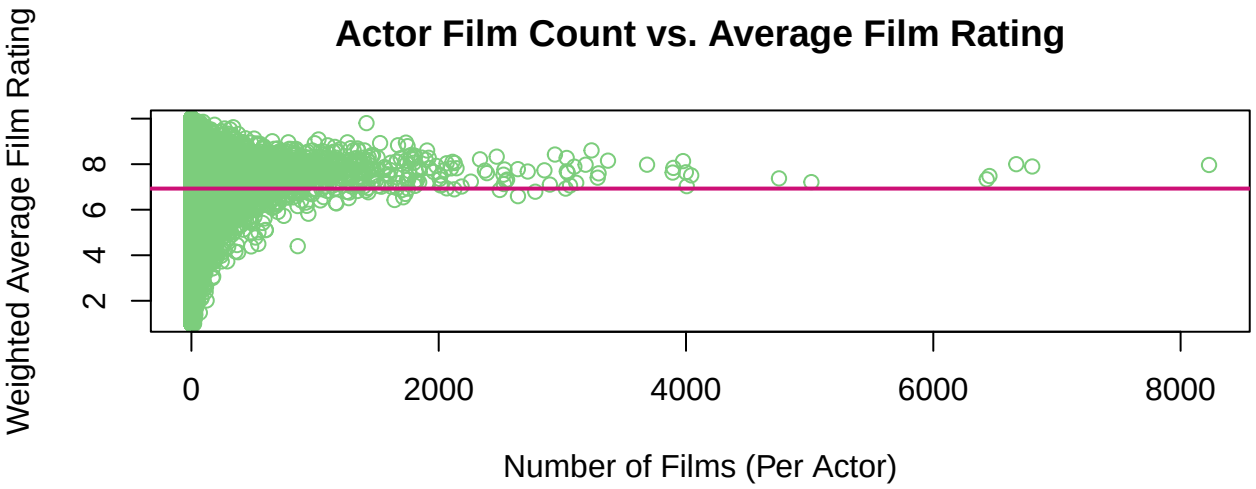
In fact, we've already seen actors who had extreme ratings of 10/10. These are specifically actors who were featured in **a smaller number of films** and received **fewer votes**.

This difference between actors who have moderate and extremely high ratings is a classic example of the **Law of Large Numbers**: as the number of films increases, average ratings tend to converge towards the mean.

Colloquially speaking:

- Suppose you're an actor that was just in **one** film reviewed by say only **10 people**. Suppose that one film got a perfect 10/10 rating. Then your weighted average rating will be a 10/10.
- Suppose you're an actor that was just in **one** film reviewed by say only **10 people**. Suppose that one film got a bad 1/10 rating. Then your weighted average rating will be a 1/10.
- But suppose you're an actor that was featured in **many films** reviewed by **millions of people**. Then your resulting weighted average rating will likely not be a perfect 10/10 — it's going to be **MUCH LOWER** and **CLOSER** to the entire dataset's mean rating.

To make sure our point is there, here is a supporting plot that visualizes the Law of Large Numbers:



Here is what we should takeaway from this plot:

- **Left side**: Actors featured in extremely small amounts of films tend to have extremely high weighted ratings. But as we said earlier, these actors could be actors who actually featured in more films than the dataset says, OR small actors whose few films have done well.
- **Right side**: Actors featured in many films had much lower ratings than what's shown on the left side of the plot. Their ratings tend to be roughly around the weighted average ratings of the entire dataset (roughly around 6).

Observation 4. Large Ratings and Large Voting Size

This part of the analysis does **not** prove that specific actors **cause** their films to have high ratings. Instead, this part of the analysis will hint at how frequently top movies have top actors.

Note: **Just** for this observation, we will (for now) focus on full-length **movies**, excluding TV show episodes.

We created two tables.

Here is the first table:

```
##                                     primaryTitle averageRating numVotes
##                                           <char>         <num>    <int>
## 1:                       The Shawshank Redemption           9.3  3042484
## 2:                                  The Godfather           9.2  2123901
## 3:                                The Dark Knight           9.0  3019519
## 4: The Lord of the Rings: The Return of the King           9.0  2076994
## 5:                                Schindler's List           9.0  1521787
```

This is a table that lists the top 10 movies that BOTH have the highest rating and highest number of votes.

Here is the second table:

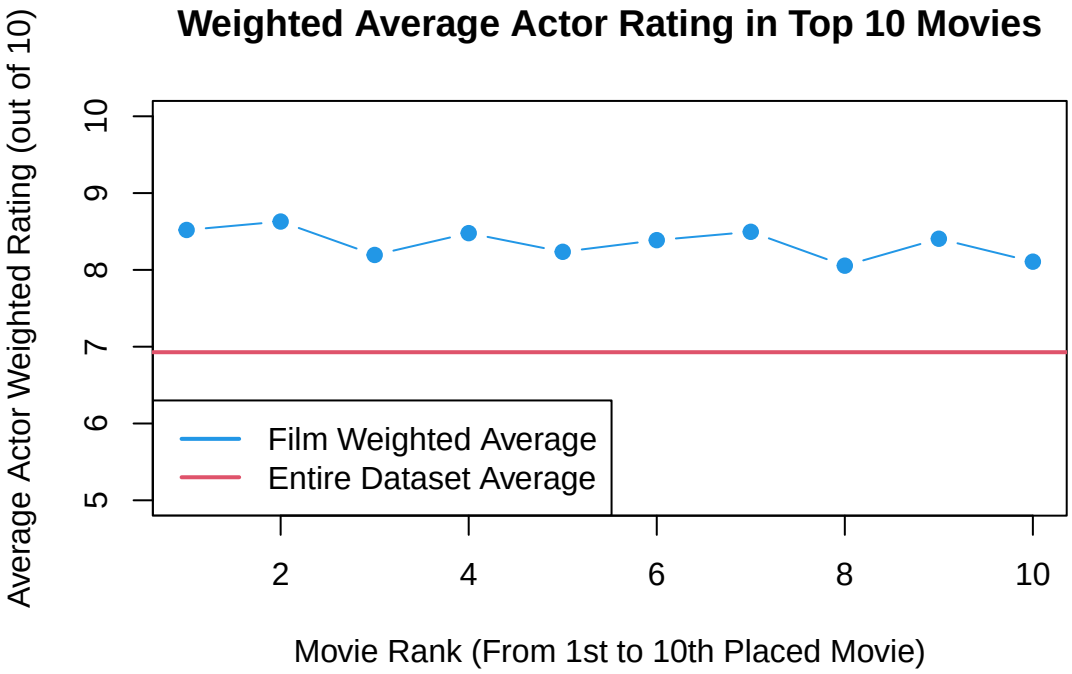
```
##          avg.weighted.rating avg.number.of.films top.actor.rating
## Movie 1             8.519878               338.1         9.127888
## Movie 2             8.629481                95.9         9.194541
## Movie 3             8.194301               107.2         8.895936
```

```
## Movie 4          8.478570          149.6          8.995000
## Movie 5          8.235334           83.6          8.999648
## Movie 6          8.387191           69.6          8.928679
## Movie 7          8.495751          208.0          8.997379
## Movie 8          8.054948          413.7          8.900000
## Movie 9          8.405614          152.7          8.946819
## Movie 10         8.106939          110.1          8.628751
```

This is a summarizing table that shows the average weighted rating and the average number of films for **all** the actors within **each** movie. It also shows the highest actor rating received among all the actors within **each** movie.

What should we care about in regards to this summarizing table? We can use this table to compare each of the top 10 movie's weighted average rating to the entire dataset's weighted average rating.

Here is a plot that visually accomplishes exactly that. It shows how top movies tend to feature top-rated actors.



As we can see, the top 10 **highest-rated**, **popular** movies (excluding episodes) often feature actors with **above-average weighted ratings**. This is indicated by the blue plots consistently placed above the red line.

This plot suggests that great movies do attract talented or well-received actors, but it does NOT guarantee that those actors bring high ratings on their own.

Conclusion

In conclusion, our analysis shows that while well-known actors tend to star in **well-received** films with **strong** ratings, this does not necessarily mean that their presence **causes** those ratings to be high.

In other words:

- We do not find clear evidence that certain actors cause higher movie ratings.
- However, we do find **correlation**: top-rated movies tend to include actors who also have strong average film ratings.

Specifically, we found out the following:

- **Observation 1**: The **extremely high** average ratings, such as a perfect 10/10, mainly come from actors who appeared in only one or two highly-rated, niche films.
- **Observations 2-3**: The weighted average ratings for **major actors** — Leonardo DiCaprio, Morgan Freeman, etc. — fall between 7.6 and 8.0, which is solidly above average but not necessarily that extremely high.
- **Observations 2-3**: This distinction is a result of the **Law of Large Numbers**, where actors with many films tend to regress toward the dataset's overall average. On the other hand, actors with just a handful of films with smaller vote counts can only appear to have perfect ratings.
- **Observation 4**: Excluding episodes, the **top 10 highest-rated popular movies** often feature actors with above-average weighted ratings.

At the end of the day, the evidence supports the idea that great actors are more likely to have been selected into great films, rather than the idea that actors are **causing** higher ratings. In other words, good movies pick good actors to star (and not always the other way around).

In actuality, movie quality is likely a result of many factors working in combination with each other **beyond** only the actors, number of votes, and average rating. The director, production value, script, and other factors could be at play.

If we want to look more into the **causes** of highly-rated and highly-popular movies, we'll need to look more into **causal inference**, which is another huge topic worth worth looking into in **future investigations**.

QUESTION 4:

Lastly, we'll finally ask: **which genres get the highest ratings?**

For this analysis, we will **assume** that we're working with **pure genres**.

- We have defined a vector **pure.genres** that contains all 27 unique, pure genres in our dataset.
- A **pure** genre is the opposite of a combined genre. For instance, our vector's entries say words such as "Action" or "Comedy" rather than something such as "Action Comedy."

It's important to note that our vector **pure.genres** does not include the words "Short" and NA:

- Long story short: NA is a missing value, while "Short" is not a genre. (It's a format.)
- In addition: when you quickly look at the column "genres" within our dataset, you will briefly see plenty of rows that contain a genre that says "Short" (and nothing more).
- This is clearly not helpful, as it tells us nothing about what the genre of the film was.

```
## [1] 0.01744343
```

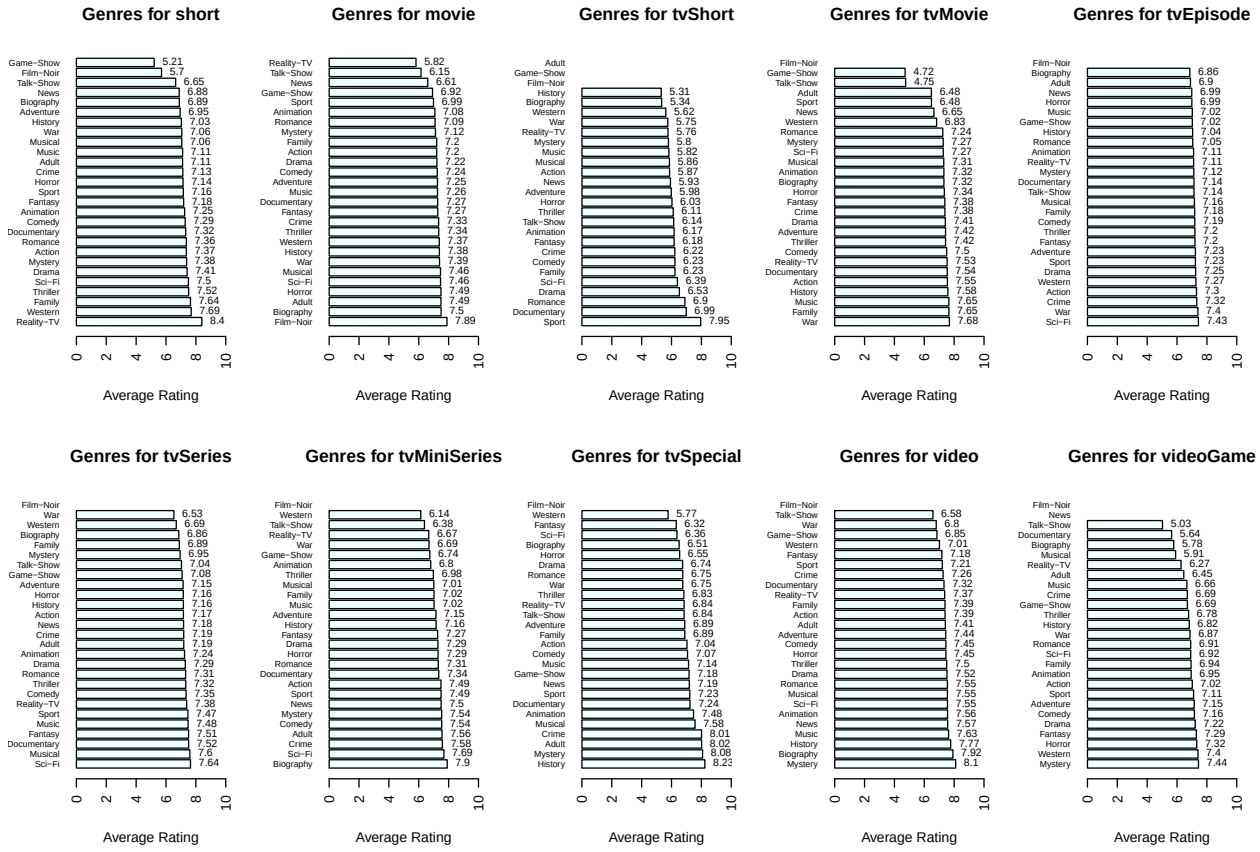
```
## [1] 0.01391636
```

```
## [1] 0.0313598
```

So what can we do about this caveat?

Using computations computed in R, we found that there are 26,904 rows that just say "Short" and 21,464 rows that contain missing values. They make up 1.74% and 1.39% of the dataset's total rows, respectively. In total, they make up around 3% of the dataset's total rows.

- That is **really, really small**.
- So, we made the decision to **ignore** films whose corresponding genre says just "Short" or NA.



Here is what we should take away from the plot.

1. Each plot displays the least to most ranked genres within each kind of film — shorts, movies, video games, etc.
2. All these plots have genres with weighted average ratings roughly between 5 to 7.
3. This range of average ratings is clearly a lot narrower than we probably intuitively expected to see. Ratings can go from 1-10, but these go from 5-7.

Back then, we simply calculated a regular **average** using the mean() function, which incorrectly acts as if all films have received the same number of votes.

- Consequently, we got (some admittedly) bizarre results.
- For instance, three of our box plots used to suggest that adult genres have been the most highly rated genres for tvMovies, tvSeries, and tvMiniSeries.

So instead, we computed a **weighted average** rating for each genre.

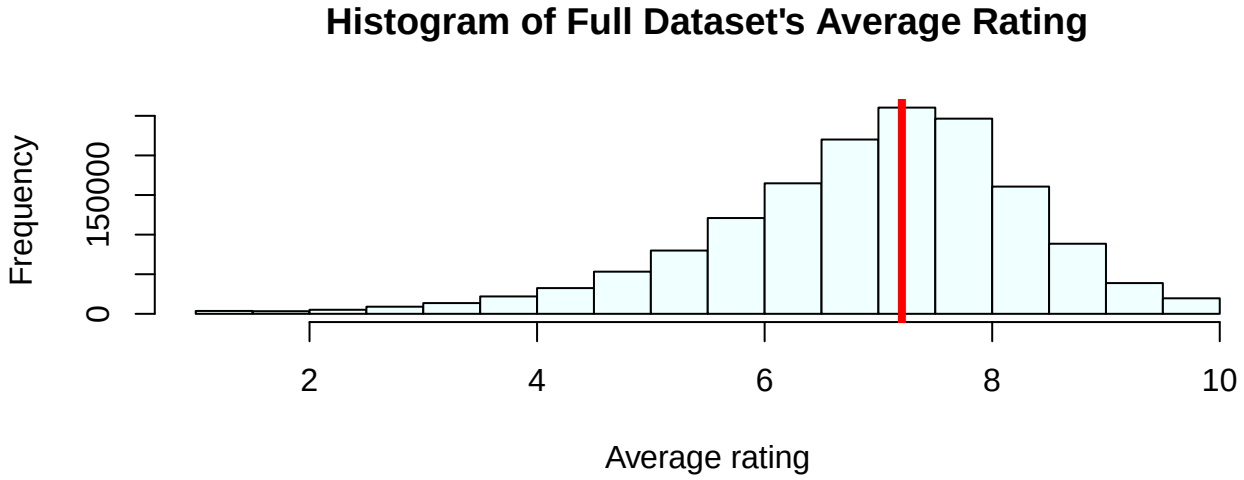
- This is crucial because upon looking at the column **numVotes**, we can see that not every film has had the same number of votes. In other words, not every film has had the same number of reviewers.

- Also, not all box plots have a plotted bar for every genre. For instance, the lower right box plot tells us that there weren't any videogames under the "News" genre (which makes sense). In these kinds of cases, where some genres have zero matched films, the code made sure to automatically exclude those genres from the weighted average calculation.
- The number of votes clearly influences how each genre's computed average rating will turn out.
- By computing a weighted average, we are giving less influence to small genres that have skewed/inflated ratings and more influence to larger genres.

Now, what could explain the narrow range of a rating of 5-7?

To start, let's check out the distribution of the averageRating column from our dataset.

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    1.00    6.20    7.10    6.95    7.90   10.00
```



As we can see:

- The distribution of the **averageRating** column from our dataset is already indeed narrow, with a center around 7.206151 (i.e. the weighted average of the entire dataset).
- The distribution visually appears to be roughly left-skewed.
- The distribution has a relatively small interquartile range (IQR), where the first quartile 6.2 and third quartile 7.9 are not that far apart.

There are some possible reasons why the distribution of the full dataset's average rating is clustered around the 5-7 range, even though ratings range from 1-10.

1. **Selection bias**: People generally review films that they're really interested in or chose to watch voluntarily. They may also review films that they personally find really controversial. Overall, people are likely to give reviews for films that stick out to them for any reason. This could affect the histogram and land its final focus around 5-7.
2. **IMDb anti-skew measures**: IMDb is said to employ anti-skew measures when collecting their survey data — they employ certain weights to ensure that meaningful reviews are counted more while meaningless, jokingly, deliberately, and excessively negative or positive reviews are counted less. This could affect the histogram and land its final focus around 5-7.

Some final notes to conclude this section with:

- One factor that **could** lead to a change in results is the fact that we had to omit films whose specified genres either said "Short" or "genre." If we knew what the genres of those films actually were, then their contribution could change the outcome of our computed weighted averages.
- In addition, the average ratings don't necessarily determine what's more popular — it's **also** important to consider other factors such as **total box office revenue**.

SUMMARY

Across our analysis of the IMDb dataset, we identified several trends between film ratings, creators, and genres:

1. **Popularity and Ratings**:
   Films with more votes tend to have higher average ratings — but only when vote counts are extremely high. Popularity may reflect wider appeal, but not necessarily better quality.
2. **Writer Presence**:
   Having a specified writer has a statistically significant but practically negligible effect on ratings, which highlights the importance of interpreting outputs with context beyond just p-values.
3. **Actor Roles**:
   Famous actors seem to be associated with better-rated films but by correlation, not causation. Actors that are featured in fewer films often show extreme average ratings due to the Law of Large Numbers.
4. **Genres**:
   Average ratings across each genre are narrow, mostly around 7 again. This could be due to selection effects and IMDb's internal weighting system. Comedy and drama tend to dominate.

Overall, our analysis explored the multiple factors that may affect a film's ratings. However, no single feature offers reasonable and strong evidence to predict a film's rating. We focused on exploring trends and ways of good analysis, especially since our dataset was so large and caused most statistical evidence to be falsely convincing.

We have an additional question that we explored available in the file `analysis_q5.pdf`. There, we go more in depth into how the experience of individuals in creative roles affect a film's rating.

CONTRIBUTIONS:

This collaborative project was done by Tony Oh and William Rhee. The analysis workflow was developed jointly, with both members actively contributing to implementation and refinement of the final report.

- **Tony Oh** led the exploration and analysis for Questions 1, 2, and 5.
- **William Rhee** led the exploration and analysis for Questions 3, 4, and 5.