STAT GU5243: Project1

Minseul, Pablo, Qiujun, Feriran

2026-02-20

Introduction and Dataset Description:

This project investigates whether *perceived film quality* (proxied by **IMDb ratings**) is associated with *Netflix popularity* as measured by how widely and how long a film appears in Netflix's weekly "Top 10" lists across countries. In other words, we ask: do higher-rated movies also become broader and longer-lasting Top 10 hits on Netflix?

To study this, we combine two public datasets that capture different sides of the story: Netflix Top 10 lists reflect platform popularity/visibility, while IMDb ratings provide an audience-based signal of perceived quality.

To do this, we: 1. **Clean and aggregate** a weekly, country-level Netflix Top 10 dataset into a **one-row-per-film** table with global popularity metrics. 2. **Clean and preprocess** IMDb title and rating tables, restricting to movies and expanding genres. 3. **Merge** the two datasets using a standardized, title-based key (`movie_key`) so we can jointly analyze popularity and ratings. 4. Conduct exploratory analysis to understand (i) how global "spread" relates to "persistence/duration" on Netflix and (ii) whether IMDb ratings help explain Netflix popularity.

Key Constructs (Global Popularity Metrics)

Because Netflix data is originally recorded at the **country–week–title** level, we aggregate it into a **one-row-per-film** table and summarize each film's global performance using the following constructs:

- **Global spread (breadth):** number of distinct countries where the film appears in the Top 10 at least once
- **Global persistence (depth):** maximum number of weeks the film survives in any single country's Top 10
- **Global duration (time span):** weeks between the first and last observed appearance globally
- **Best rank:** best (lowest) weekly rank achieved globally *(rank 1 = best)*

These metrics allow us to describe different "types" of Netflix success (e.g., global-but-short-lived vs. local-but-long-lasting).

Data Source 1: Netflix Weekly Top 10 (Films) — Kaggle

What it is:
A weekly snapshot of Netflix's Top 10 rankings by country. We restrict the data to the **Films** category.

Unit of observation (raw):
country × week × film title
Each row corresponds to a film appearing in a given country's Top 10 list for a specific week, along with its rank.

Key variables used:

- `country_name`, `country_iso2`: country identifiers
- `week`: week-ending date for the Top 10 snapshot
- `weekly_rank`: rank from 1 to 10 *(1 = best)*
- `show_title`: film title as listed on Netflix
- `cumulative_weeks_in_top_10`: running total weeks the title has appeared in that country's Top 10 up to that week

Why this dataset is useful:
It provides a consistent and comparable measure of *relative popularity among the most visible titles* across many countries and time periods.

Limitations:
It only includes titles that enter the **Top 10**, and it does not provide direct view counts (e.g., hours watched). Therefore, our measures capture **Top 10 visibility and persistence**, not overall demand among all Netflix content.

Data Source 2: IMDb Official Datasets (title.basics + title.ratings)

What it is:
Two official IMDb tables:

- `title.basics`: title metadata (type, year, runtime, genres, etc.)
- `title.ratings`: audience ratings and vote counts

We restrict the dataset to **movies**.

Unit of observation:
An IMDb title (movie), identified by `tconst`.

Key variables used:

From `title.basics`:

- `primaryTitle`: movie title
- `startYear`: release year
- `runtimeMinutes`: runtime
- `genres`: genre labels (later expanded for analysis)
- `titleType`: used to filter to `movie` only

From `title.ratings`:

- `averageRating`: average IMDb rating *(our quality proxy)*
- `numVotes`: number of user votes *(important for rating reliability)*

Why this dataset is useful:
IMDb ratings provide an audience-based quality signal. In this report, we treat **IMDb `averageRating` as a proxy for perceived film quality**, while `numVotes` gives context for how stable/reliable a rating might be.

Limitations:
Ratings reflect IMDb users (selection bias) and are not Netflix-specific. Even high-rated movies may not become Netflix Top 10 hits due to platform promotion, timing, language availability, or regional tastes.

The remainder of this section describes the two source datasets in more detail and explains how we transformed them into analysis-ready tables.

Dataset 1: Netflix "Top 10" TV Shows and Films

Data Aquisition, Cleaning, Checking

Link to Kaggle: https://www.kaggle.com/datasets/dhruvildave/netflix-top-10-tv-shows-and-films

Load libraries, set paths for initial data, and create an output folder (if non-existant)

```
library(tidyverse)
```

```
## -- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
## v dplyr     1.1.4     v readr     2.1.5
## v forcats   1.0.1     v stringr   1.5.2
## v ggplot2   4.0.0     v tibble    3.3.0
## v lubridate 1.9.4     v tidyr     1.3.1
## v purrr     1.1.0
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
## i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become error
```

```
library(lubridate)
library(readr)
library(stringr)
library(janitor)
```

```
##
## Attaching package: 'janitor'
##
## The following objects are masked from 'package:stats':
##
##     chisq.test, fisher.test
```

```
library(glue)

raw_path <- "netflix_top_10_raw.csv"
if (!file.exists(raw_path)) raw_path <- "/mnt/data/netflix_top_10_raw.csv"
```

```r
out_dir <- "data/processed"
if (!dir.exists(out_dir)) dir.create(out_dir, recursive = TRUE)
```

Reading the raw data as strings first: This forces every column to have a string in the data, so that we can avoid parsing mistakes! clean_names() standardizes columns to be lower_case.

```r
df_raw <- read_csv(
  raw_path,
  col_types = cols(
    country_name = col_character(),
    country_iso2 = col_character(),
    week = col_character(),
    category = col_character(),
    weekly_rank = col_character(),
    show_title = col_character(),
    season_title = col_character(),
    cumulative_weeks_in_top_10 = col_character()
  ),
  show_col_types = FALSE
) |>
  clean_names()
```

Standardizing string formatting: str_squish() allows us to collapse all internal spaces into 1. This is to prevent issues that would arise from “USA” and ” USA ” being treated as different values during aggregation. Then, we use na_if(.x, “ “) to turn empty strings into”NA”. This step is important so that we do not mistakenly interpret empty strings as valid entries.

```r
# standardize string formatting
#    - collapse multiple spaces
#    - empty strings as NA
df_std <- df_raw %>%
  mutate(across(where(is.character), ~ str_squish(.x))) %>%
  mutate(across(where(is.character), ~ na_if(.x, "")))
```

Since this project’s scope is only limited to films, we must filter the data using the “category” column. However, before applying the filter, just in case the “category” column has all “films”, “FILMS”, “Films”, we standardize all of them to “Films” to prevent case-sensitivity issues and ensure accurate filtering.

```r
df_films_raw <- df_std %>%
  mutate(category = str_to_title(category)) %>%
  filter(category == "Films")
```

df_films_raw contains only Netflix Top 10 films for multiple weeks. Now, we must clean the data so it becomes analytical data instead of just text.

First, we convert the country_code(country_iso2) to uppercase letters, in the case that the raw dataset contains “us” instead of “US”. This is to prevent problems when we later group the data by country codes. Next, we convert weeks (week_date) to a date object instead of keeping it as a string for future time analysis operations.

Then, we convert rank(weekly_rank_int) and cumulative week data(cum_weeks_int) to integers so that we can ensure numerical comparisons and validation checks during aggregation. Lastly, we do a diagnostic check to see if any values failed to convert.

```r
# week -> Date, ranks/weeks -> integer, ISO2 -> uppercase.
df_films_typed <- df_films_raw %>%
  mutate(
    country_iso2 = str_to_upper(country_iso2),
    week_date = ymd(week),

    weekly_rank_int = parse_integer(weekly_rank),
    cum_weeks_int   = parse_integer(cumulative_weeks_in_top_10)
  )
type_issues <- list(
  week_parse_fail = df_films_typed %>% filter(is.na(week_date) & !is.na(week)) %>% nrow(),
  weekly_rank_parse_fail = df_films_typed %>% filter(is.na(weekly_rank_int) & !is.na(weekly_rank)) %>% n
  cum_weeks_parse_fail = df_films_typed %>% filter(is.na(cum_weeks_int) & !is.na(cumulative_weeks_in_top
)
print(type_issues)
```

```
## $week_parse_fail
## [1] 0
##
## $weekly_rank_parse_fail
## [1] 0
##
## $cum_weeks_parse_fail
## [1] 0
```

Because the "season_title" column is not applicable for movies, we change it to "Movie". Then, we drop rows missing critical fields, like date, rank, and title. To make the data more readable, we concluded that instead of the week column being noted as the starting date, we should add 2 more columns: 1. week_num: a sequential index of unique weeks (most earliest week gets "1") 2. week_label: human-readable label ("week 1")

```r
df_films_add_week_num <- df_films_typed %>%
  mutate(
    season_title = if_else(is.na(season_title), "Movie", season_title)
  ) %>%
  filter(
    !is.na(week_date),
    !is.na(weekly_rank_int),
    !is.na(show_title)
  ) %>%
  mutate(
    week_num   = dense_rank(week_date),
    week_label = paste0("week ", week_num)
  )
```

This is a data integrity step. First, we detect duplicate rows, and then remove those duplicates. This step ensures data integrity by detecting and removing duplicate rows. Duplicate rows may pollute the data and infalte counts or distort aggregation results.

```r
#"natural key" and keep the first row per key.
key_cols <- c(
  "country_iso2", "week_date", "category", "weekly_rank_int",
  "show_title", "season_title", "cum_weeks_int"
)

dup_table <- df_films_add_week_num %>%
  count(across(all_of(key_cols)), name = "n") %>%
  filter(n > 1)

message(glue("Duplicate key groups found: {nrow(dup_table)}"))
```

```
## Duplicate key groups found: 0
```

```
# If duplicates exist, keep the first one and move on.
df_films_nodup <- df_films_add_week_num %>%
  distinct(across(all_of(key_cols)), .keep_all = TRUE)
```

This is a data integrity check to ensure the rank variable conforms to the structure of a Top 10 dataset. Here, we create a table of rows where the rank failed parsing, or is less than 1, or is greater than 10. This is important because a rank must be through 1-10, as the data is a top 10 dataset.

```
bad_weekly_rank <- df_films_nodup %>%
  filter(is.na(weekly_rank_int) | weekly_rank_int < 1 | weekly_rank_int > 10)

if (nrow(bad_weekly_rank) > 0) {
  warning(glue("Found {nrow(bad_weekly_rank)} rows with weekly_rank outside 1-10. Inspect `bad_weekly_ra
} else {
  message("weekly_rank check PASSED: all values are within 1-10.")
}
```

```
## weekly_rank check PASSED: all values are within 1-10.
```

This step represents the final step of cleaning the dataset before merges. Using transmute(), we rename and select the relevant columns so it is easier for us to use it.

```
df_clean <- df_films_nodup %>%
  transmute(
    country_name,
    country_iso2,
    week = week_num,            # numeric week index
    week_date,                  # keep actual date for reference
    category,
    weekly_rank = weekly_rank_int,
    show_title,
    season_title,
    cumulative_weeks_in_top_10 = cum_weeks_int
  )
```

After the above operations, we have ultimately changed the raw data into data we can actualize analyze. We save this newly organized dataset as “netflix_films_clean.csv”, and create a cleaning_summary list that shows information about the dataset. This is to ensure that the data is clean and is what we intended for it to be.

```r
#save
write_csv(df_clean, file.path(out_dir, "netflix_films_clean.csv"))

# Quick summary
cleaning_summary <- list(
  n_raw = nrow(df_raw),
  n_films_before_clean = nrow(df_films_raw),
  n_films_after_clean = nrow(df_clean),
  missing_season_title_after = sum(is.na(df_clean$season_title)),
  weekly_rank_range = range(df_clean$weekly_rank, na.rm = TRUE)
)
print(cleaning_summary)
```

```
## $n_raw
## [1] 112300
##
## $n_films_before_clean
## [1] 56150
##
## $n_films_after_clean
## [1] 56150
##
## $missing_season_title_after
## [1] 0
##
## $weekly_rank_range
## [1]  1 10
```

```r
#preview
# Print a few rows + the week_number → date mapping  to confirm everything lines up.
out_path <- file.path(out_dir, "netflix_films_clean.csv")

cat("Working directory:", getwd(), "\n")
```

```
## Working directory: /Users/minseulkim/Documents/GitHub/AppliedDSProject-1
```

```r
cat("Cleaned dataset saved to:", normalizePath(out_path, winslash = "/"), "\n\n")
```

```
## Cleaned dataset saved to: /Users/minseulkim/Documents/GitHub/AppliedDSProject-1/data/processed/netfl
```

```r
cat("First 10 rows of df_clean (sorted):\n")
```

```
## First 10 rows of df_clean (sorted):
```

```r
print(df_clean %>% arrange(week_date, country_iso2, weekly_rank) %>% head(10))
```

```
## # A tibble: 10 x 9
##    country_name    country_iso2  week week_date  category weekly_rank show_title
##    <chr>           <chr>        <int> <date>     <chr>          <int> <chr>
##  1 United Arab Em~ AE               1 2021-07-04 Films              1 Haseen Di~
##  2 United Arab Em~ AE               1 2021-07-04 Films              2 The Hobbi~
##  3 United Arab Em~ AE               1 2021-07-04 Films              3 Fatherhood
##  4 United Arab Em~ AE               1 2021-07-04 Films              4 Me Before~
##  5 United Arab Em~ AE               1 2021-07-04 Films              5 The Hobbi~
##  6 United Arab Em~ AE               1 2021-07-04 Films              6 Mommy Iss~
##  7 United Arab Em~ AE               1 2021-07-04 Films              7 Wish Drag~
##  8 United Arab Em~ AE               1 2021-07-04 Films              8 Run All N~
##  9 United Arab Em~ AE               1 2021-07-04 Films              9 The Hobbi~
## 10 United Arab Em~ AE               1 2021-07-04 Films             10 Monster-i~
## # i 2 more variables: season_title <chr>, cumulative_weeks_in_top_10 <int>
```

```
cat("\nWeek number mapping (first 10):\n")
```

```
##
## Week number mapping (first 10):
```

```
print(df_clean %>% distinct(week, week_date) %>% arrange(week_date) %>% head(10))
```

```
## # A tibble: 10 x 2
##     week week_date
##    <int> <date>
##  1     1 2021-07-04
##  2     2 2021-07-11
##  3     3 2021-07-18
##  4     4 2021-07-25
##  5     5 2021-08-01
##  6     6 2021-08-08
##  7     7 2021-08-15
##  8     8 2021-08-22
##  9     9 2021-08-29
## 10    10 2021-09-05
```

```
cat("\nColumn types (glimpse):\n")
```

```
##
## Column types (glimpse):
```

```
dplyr::glimpse(df_clean)
```

```
## Rows: 56,150
## Columns: 9
## $ country_name               <chr> "Argentina", "Argentina", "Argentina", "Arg~
## $ country_iso2               <chr> "AR", "AR", "AR", "AR", "AR", "AR", "AR", "~
## $ week                       <int> 60, 60, 60, 60, 60, 60, 60, 60, 60, 60, 59,~
## $ week_date                  <date> 2022-08-21, 2022-08-21, 2022-08-21, 2022-0~
## $ category                   <chr> "Films", "Films", "Films", "Films", "Films"~
## $ weekly_rank                <int> 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 1, 2, 3, 4, ~
## $ show_title                 <chr> "Look Both Ways", "Day Shift", "Bank Robber~
## $ season_title               <chr> "Movie", "Movie", "Movie", "Movie", "Movie"~
## $ cumulative_weeks_in_top_10 <int> 1, 2, 2, 1, 1, 4, 1, 2, 2, 1, 1, 3, 1, 1, 1~
```

Data Preprocessing + Feature Engineering

In this step, the main goal is to take the cleaned weekly top 10 film dataset and turn it into one row per movie. This means grouping the weekly and country-level data for each film and summarizing it into overall popularity metrics across countries. We combined country-week data rows for each movie into popularity metrics such as:

- **global_persistence** = The longest number of weeks the film survived in any single country = how long the movie hangs around somewhere –> depth
- **global_spread** = number of unique countries each film appears in –> breadth

Inputs (from netflix_films_clean.csv)

We only use these columns to create the final preprocessed data: `- show_title - country_name - weekly_rank - cumulative_weeks_in_top_10` (`cuWk_top_10`) `- week` (numeric week index: 1, 2, 3, …) —

```r
# Packages
library(dplyr)
library(stringr)
library(readr)
library(ggplot2)
library(knitr)
```

The first step is to load the data:

```r
in_path <- "data/processed/netflix_films_clean.csv"
if (!file.exists(in_path)) in_path <- "/mnt/data/netflix_films_clean.csv"

out_dir <- "data/processed"
if (!dir.exists(out_dir)) dir.create(out_dir, recursive = TRUE)

top10 <- read_csv(in_path, show_col_types = FALSE)

cat("Loaded:", in_path, "\n")
```

```
## Loaded: data/processed/netflix_films_clean.csv
```

```r
cat("Rows x Cols:", nrow(top10), "x", ncol(top10), "\n\n")
```

```
## Rows x Cols: 56150 x 9
```

```r
# Quick peek at the column names
print(names(top10))
```

```
## [1] "country_name"                "country_iso2"
## [3] "week"                        "week_date"
## [5] "category"                    "weekly_rank"
## [7] "show_title"                  "season_title"
## [9] "cumulative_weeks_in_top_10"
```

Although we filtered the data to films earlier, we do it again to ensure accuracy. Then, we use transmute() to drop unnecessary columns, create & rename columns, standardize formatting, and convert column types. The crucial step here is that we create 2 new columns: - `movie_key`: standardized lowercase title used for grouping (to ensure consistent grouping) - `week_number`: `week` as an integer

However, before doing this, to make the movie_key be more accurately aligned with any other dataset, we will make a cleaning rule (will apply this to IMDb as well) make_key(). This is important because grouping in R is case and whitespace-sensitive, meaning that even the smallest formatting differences can be the defining factor for treating pieces of data differently. Furthermore, because we later merge this dataset with the IMDb dataset, we will need to apply this same cleaning rule to the IMDb dataset later. Standardizing the title format will allow the matching process to be as accurate as possible.

```r
make_key <- function(x) {
  x |>
    str_to_lower() |>
    str_replace_all("[^a-z0-9 ]", "") |>
    str_squish()
}
```

```r
films <- top10 %>%
  filter(category == "Films" | is.na(category)) %>%  # in case category is missing
  transmute(
    movie_title  = str_squish(show_title),                 # readable title
    movie_key    = make_key(show_title),   # stable grouping key

    country_name = str_squish(country_name),

    week_number  = as.integer(week),
    weekly_rank  = as.integer(weekly_rank),

    cuWk_top_10  = as.integer(cumulative_weeks_in_top_10)
  )

cat("films rows/cols:", nrow(films), "x", ncol(films), "\n")
```

```
## films rows/cols: 56150 x 6
```

```r
cat("unique movies:", n_distinct(films$movie_key), "\n")
```

```
## unique movies: 2640
```

```r
cat("unique countries:", n_distinct(films$country_name), "\n\n")
```

```
## unique countries: 94
```

```r
kable(head(films, 10))
```

movie_title	movie_key	country_name	week_number	weekly_rank	cuWk_top_10
Look Both Ways	look both ways	Argentina	60	1	1
Day Shift	day shift	Argentina	60	2	2

movie_title	movie_key	country_name	week_number	weekly_rank	cuWk_top_10
Bank Robbers: The Last Great Heist	bank robbers the last great heist	Argentina	60	3	2
The Next 365 Days	the next 365 days	Argentina	60	4	1
The Angry Birds Movie 2	the angry birds movie 2	Argentina	60	5	1
Purple Hearts	purple hearts	Argentina	60	6	4
River Runs Red	river runs red	Argentina	60	7	1
1917	1917	Argentina	60	8	2
Code Name: Emperor	code name emperor	Argentina	60	9	2
Royalteen	royalteen	Argentina	60	10	1

After this step, we have simplified/standardized the dataset so it only contains the variables required for building popularity metrics. This prepares the data for country & movie-level aggregation, and later, merging with the second dataset.

However, still, the dataset is at the movie x country x week level. This means each movie appears multiple times for each country and week it was in the Netflix Top 10 list. To aggregate the data to the movie x country level, we use group_by to group the data by movie_key, movie_title, and country_name. Simply put, from **one row per week**, we collapse the data to **one row per (movie, country)** .

Using summarize(), we also create 3 new statistics: - `persistence_in_country`: maximum value of cumulative weeks in that specific country
- `best_rank_in_country`: best rank achieved in that country
- `first_week_in_country`, `last_week_in_country`: first and last observed week the movie appeared in the Top 10 list.

```r
movie_country <- films %>%
  group_by(movie_key, movie_title, country_name) %>%
  summarise(
    persistence_in_country = max(cuWk_top_10, na.rm = TRUE),
    best_rank_in_country   = min(weekly_rank, na.rm = TRUE),
    first_week_in_country  = min(week_number, na.rm = TRUE),
    last_week_in_country   = max(week_number, na.rm = TRUE),
    .groups = "drop"
  )

cat("movie_country rows/cols:", nrow(movie_country), "x", ncol(movie_country), "\n\n")
```

```
## movie_country rows/cols: 28085 x 7
```

```r
kable(head(movie_country, 10))
```

movie_key	movie_title	country_name	persistence_in_country	best_rank_in_country	first_week_in_country	last_week_in_country
	...	Bahrain	3	1	30	32
	...	Belgium	1	10	31	31
	...	Brazil	1	9	31	31
	...	Egypt	5	1	30	34
	...	France	2	3	30	31
	...	Guadeloupe	2	3	30	31
	...	Israel	2	1	30	31
	...	Jordan	5	1	30	34

movie_key	movie_title	country_name	persistence_in_country	best_rank_in_country	first_week_in_country	last_week_in_country
	…	Kuwait	4	1	30	33
	…	Lebanon	8	1	30	37

```
write_csv(movie_country, file.path(out_dir, "movie_country_table.csv"))
cat("\nSaved: data/processed/movie_country_table.csv\n")
```

```
##
## Saved: data/processed/movie_country_table.csv
```

```
head(films)
```

```
## # A tibble: 6 x 6
##   movie_title          movie_key country_name week_number weekly_rank cuWk_top_10
##   <chr>                <chr>     <chr>              <int>       <int>       <int>
## 1 Look Both Ways       look bot~ Argentina             60           1           1
## 2 Day Shift            day shift Argentina             60           2           2
## 3 Bank Robbers: The ~ bank rob~ Argentina             60           3           2
## 4 The Next 365 Days    the next~ Argentina             60           4           1
## 5 The Angry Birds Mo~ the angr~ Argentina             60           5           1
## 6 Purple Hearts        purple h~ Argentina             60           6           4
```

Next, we aggregate once more (using group_by) from movie x country to one row per movie by summarizing across all countries. Then, to “calculate” its popularity, we summarize it with 4 metrics:

- **global_persistence (depth)**: highest week count (persistence_in_country) in any single country
- **global_spread (breadth)**: the number of distinct countries in which the film appeared in the Netflix Top 10 list at least once
- **global_duration (time span)**: weeks between first and last appearance across all countries
- **best_rank**: lowest rank value achieved globally (rank 1 = best)

```
final_netflix_top_10_data <- movie_country %>%
  group_by(movie_key) %>%
  summarise(
    movie_title = first(movie_title),
    global_persistence = max(persistence_in_country, na.rm = TRUE),
    global_spread      = n_distinct(country_name),
    global_duration    = max(last_week_in_country, na.rm = TRUE) -
                         min(first_week_in_country, na.rm = TRUE) + 1,
    best_rank          = min(best_rank_in_country, na.rm = TRUE),
    .groups = "drop"
  ) %>%
  arrange(desc(global_spread), desc(global_persistence), best_rank)

cat("final_netflix_top_10_data rows/cols:", nrow(final_netflix_top_10_data), "x", ncol(final_netflix_top
```

```
## final_netflix_top_10_data rows/cols: 2640 x 6
```

```r
kable(head(final_netflix_top_10_data, 15))
```

movie_key	movie_title	global_persistence	global_spread	global_duration	best_rank
red notice	Red Notice	39	94	41	1
army of thieves	Army of Thieves	21	94	30	1
dont look up	Don't Look Up	19	94	20	1
love hard	Love Hard	9	94	9	1
kate	Kate	8	94	8	1
the tinder swindler	The Tinder Swindler	8	94	8	1
the unforgivable	The Unforgivable	8	94	8	1
blood red sky	Blood Red Sky	6	94	57	1
the guilty	The Guilty	6	94	24	1
sweet girl	Sweet Girl	5	94	5	1
texas chainsaw massacre	Texas Chainsaw Massacre	5	94	5	1
the harder they fall	The Harder They Fall	4	94	4	1
the kissing booth 3	The Kissing Booth 3	4	94	4	1
the royal treatment	The Royal Treatment	4	94	4	1
beckett	Beckett	3	94	3	1

```r
write_csv(final_netflix_top_10_data, file.path(out_dir, "final_netflix_top_10_data.csv"))
cat("\nSaved: data/processed/final_netflix_top_10_data.csv  (FINAL movie-level table)\n")
```

```
##
## Saved: data/processed/final_netflix_top_10_data.csv  (FINAL movie-level table)
```

By this point, final_netflix_top_10_data contains the final movie-level dataset derived from the raw Kaggle dataset. Through repeated cleaning, aggregation, and standardization, the data was transformed from weekly, per-country observations into a summarized one row per film. In addition, we constructed global popularity metrics to capture the films' overall performance across countries. Now, the dataset is ready to be merged with the IMDb dataset. Creating a one-row-per-film dataset was essential to conduct meaningful film-level, global comparisons, since now each observations represents a distinct movie rather than repeated weekly entries for multiple countries.

Dataset 2: IMDB DATA

Data Acquisition + Cleaning

IMDB dataset files are too large for Github uploading. Download IMDb data from: https://datasets.imdbws.com/ Download title_basics.tsv.gz, and title.ratings.tsv.gz

Place the files in project folder before running the code. Files are too large to upload into Github repository.

```r
library(tidyverse)
library(readr)
```

Load the dataset csv files (make sure they are in the project/working directory).:

First, we download the title.basics dataset, aka the file with all the titles. read_tsv() means that we are reading tab-separate values. Since IMDb encodes missing values as "\N", we convert them into proper NA. After setting all columns as text, we go ahead and specify isAdult, startYear, endYear, runtimeMinutes into their proper types. This avoids wrong automatic guessing, which may cause future complications when we analyze it.

```
# Load data
title_basics <- read_tsv(
  "title.basics.tsv.gz",
  na = "\\N",
  col_types = cols(
    .default = col_character(),
    isAdult = col_integer(),
    startYear = col_double(),
    endYear = col_double(),
    runtimeMinutes = col_double()
  ))
```

```
## Warning: One or more parsing issues, call `problems()` on your data frame for details,
## e.g.:
##   dat <- vroom(...)
##   problems(dat)
```

```
problems(title_basics)
```

```
## # A tibble: 1,074 x 5
##        row   col expected  actual            file
##      <int> <int> <chr>     <chr>             <chr>
##  1 1096407     8 9 columns 8 columns         ""
##  2 1096407     8 a double  Reality-TV        ""
##  3 1505038     8 9 columns 8 columns         ""
##  4 1505038     8 a double  Talk-Show         ""
##  5 1891314     8 9 columns 8 columns         ""
##  6 1891314     8 a double  Documentary       ""
##  7 2002045     8 9 columns 8 columns         ""
##  8 2002045     8 a double  Talk-Show         ""
##  9 2154882     8 9 columns 8 columns         ""
## 10 2154882     8 a double  Family,Game-Show ""
## # i 1,064 more rows
```

Looking at the problem table, we can see that some rows are shifted left, meaning that they have only 8 columns instead of the expected 9. Column 8 should have runtimeMinutes, but for a few rows, there are genre values (Reality-TV, Talk-Show, etc.). This is because there are missing tab separaters in the raw data, which causes subsequent value sto shift into the wrong column positions. However, given that there are nearly 10 million rows in the dataset and only 1074 rows have this error, we can ignore this negligible 0.01% and safely treat them as NA values.

We now load the IMDb ratings data. The raw dataset only has 3 columns: tconst, averageRating, and numVotes, so we convert each of these column's datatypes into the types we want.

```
title_ratings <- read_tsv(
  "title.ratings.tsv.gz",
  na = "\\N",
  col_types = cols(
    tconst = col_character(),
    averageRating = col_double(),
    numVotes = col_double()
  ))
```

```
head(title_basics)
```

```
## # A tibble: 6 x 9
##   tconst    titleType primaryTitle       originalTitle isAdult startYear endYear
##   <chr>     <chr>     <chr>              <chr>           <int>     <dbl>   <dbl>
## 1 tt0000001 short     Carmencita         Carmencita          0      1894      NA
## 2 tt0000002 short     Le clown et ses c~ Le clown et ~       0      1892      NA
## 3 tt0000003 short     Poor Pierrot       Pauvre Pierr~       0      1892      NA
## 4 tt0000004 short     Un bon bock        Un bon bock         0      1892      NA
## 5 tt0000005 short     Blacksmith Scene   Blacksmith S~       0      1893      NA
## 6 tt0000006 short     Chinese Opium Den  Chinese Opiu~       0      1894      NA
## # i 2 more variables: runtimeMinutes <dbl>, genres <chr>
```

Data Preprocessing

Now, we must merge the title_basics datset with the title_ratings dataset. We do "by = tconst" as we match the rows where the title ID is the same. We do full_join() because we must keep all the possible data. For example, even though a movie's title exists in the database yet there is no rating for it, we should still keep the data and mark the rating as NA.

```
df <- title_basics |>
  full_join(title_ratings, by = "tconst")
head(df)
```

```
## # A tibble: 6 x 11
##   tconst    titleType primaryTitle       originalTitle isAdult startYear endYear
##   <chr>     <chr>     <chr>              <chr>           <int>     <dbl>   <dbl>
## 1 tt0000001 short     Carmencita         Carmencita          0      1894      NA
## 2 tt0000002 short     Le clown et ses c~ Le clown et ~       0      1892      NA
## 3 tt0000003 short     Poor Pierrot       Pauvre Pierr~       0      1892      NA
## 4 tt0000004 short     Un bon bock        Un bon bock         0      1892      NA
## 5 tt0000005 short     Blacksmith Scene   Blacksmith S~       0      1893      NA
## 6 tt0000006 short     Chinese Opium Den  Chinese Opiu~       0      1894      NA
## # i 4 more variables: runtimeMinutes <dbl>, genres <chr>, averageRating <dbl>,
## #   numVotes <dbl>
```

df is the master IMDb table that contains all title types (movies, TV shows, etc), has all genres in one comma-separated string, and rating info. It also has every frow from both datasets.

Now, we create mdf, a dataset that is useful for future analysis. First, from the raw dataset, we will restrict the data to rows represent movies. Then, we will separate the genre column to multiple rows, so that a single movie becomes multiple rows. We also select the relevant columns to simplify our data. Most importantly, we add a movie_key variable to prepare for the future merge with the Netflix Top 10 dataset. We generate this key using the make_key() function that was previously applied to the Netflix Top 10 dataset as well. This will allow consistent title-based matching across datasets. We do a left_join() with the ratings dataset because we only want to keep the movies that have ratings.

```
make_key <- function(x) {
  x |>
    str_to_lower() |>
    str_replace_all("[^a-z0-9 ]", "") |>
    str_squish()
}
```

```
imdb_movies_final <- title_basics |>
  filter(titleType == "movie", !is.na(genres)) |>
  transmute(
    tconst,
    primaryTitle,
    originalTitle,
    startYear,
    runtimeMinutes,
    genres,
    movie_key = make_key(primaryTitle)   # for later Netflix merge (title-based)
  ) |>
  left_join(
    title_ratings |>
      select(tconst, averageRating, numVotes),
    by = "tconst"                        # IMDb internal join (ID-based)
  ) |>
  separate_rows(genres, sep = ",") |>
  mutate(genres = str_squish(genres))

head(imdb_movies_final)
```

```
## # A tibble: 6 x 9
##   tconst    primaryTitle originalTitle startYear runtimeMinutes genres movie_key
##   <chr>     <chr>        <chr>             <dbl>          <dbl> <chr>  <chr>
## 1 tt0000009 Miss Jerry   Miss Jerry         1894             45 Roman~ miss jer~
## 2 tt0000147 The Corbett~ The Corbett--      1897            100 Docum~ the corb~
## 3 tt0000147 The Corbett~ The Corbett--      1897            100 News   the corb~
## 4 tt0000147 The Corbett~ The Corbett--      1897            100 Sport  the corb~
## 5 tt0000574 The Story o~ The Story of~      1906             70 Action the stor~
## 6 tt0000574 The Story o~ The Story of~      1906             70 Adven~ the stor~
## # i 2 more variables: averageRating <dbl>, numVotes <dbl>
```

imdb_movies_final is the clean, movie-level, IMDb-derived dataset with genre expansion and IMDb ratings attached when available.

Merging the Data:

Here, we are merging the Netflix data with the IMDB data using movie_key. Since Netflix does not have IMDb's unique identifier for movies (tconst), we had to rely on a standardized function for the titles: make_key(). First, we restrict IMDb dataset to movies that appear in the Netflix dataset using a semi_join(). Next, using left_join(), we add the popularity metrics from the Netflix dataset.

```r
netflix_imdb_final <- imdb_movies_final |>
  semi_join(final_netflix_top_10_data, by = "movie_key") |>
  left_join(final_netflix_top_10_data, by = "movie_key")

# Diagnostics: how many Netflix movies matched IMDb?
merge_summary <- list(
  n_netflix_movies = n_distinct(final_netflix_top_10_data$movie_key),
  n_imdb_movies_in_merged = n_distinct(netflix_imdb_final$movie_key),
  match_rate = n_distinct(netflix_imdb_final$movie_key) / n_distinct(final_netflix_top_10_data$movie_key
)
print(merge_summary)
```

```
## $n_netflix_movies
## [1] 2640
##
## $n_imdb_movies_in_merged
## [1] 2308
##
## $match_rate
## [1] 0.8742424
```

```r
head(netflix_imdb_final)
```

```
## # A tibble: 6 x 14
##   tconst    primaryTitle originalTitle startYear runtimeMinutes genres movie_key
##   <chr>     <chr>        <chr>             <dbl>          <dbl> <chr>  <chr>
## 1 tt0003337 Robin Hood   Robin Hood         1913             NA Adven~ robin ho~
## 2 tt0003606 Aftermath    Aftermath          1914             NA Drama  aftermath
## 3 tt0004403 The Nightin~ The Nighting~      1914             NA Crime  the nigh~
## 4 tt0004403 The Nightin~ The Nighting~      1914             NA Drama  the nigh~
## 5 tt0004696 The Three M~ The Three Mu~      1914             NA Drama  the thre~
## 6 tt0004761 Vendetta     Vendetta           1914             NA Drama  vendetta
## # i 7 more variables: averageRating <dbl>, numVotes <dbl>, movie_title <chr>,
## #   global_persistence <int>, global_spread <int>, global_duration <dbl>,
## #   best_rank <int>
```

Overall, we have created a combined dataset with both global Netflix popularity metrics and IMDB ratings for each film.

Looking at the merge summary, we observe that there were 2,640 distinct Netflix films (after preprocessing and aggregation), and we successfully matched 2,308 of them with the IMDb dataset and appended ratings to it. This corresponds to a 87.4% match rate, a relatively successful number considering the fact that the match was not based on a shared unique ID, but only a standardized title string. The missing 13% person may be because of title variations, such as subtitles or part numbers going inside the title. Moreover, some Netflix movies have Netflix-specific naming.

Exploratory Data Analysis (EDA):

Only Using Netflix Top 10 Dataset (Final)

Are globally distributed films more persistent?

In other words, do films that appear in more countries' Top 10 lists tend to last longer in the Top 10?

We will compute the Pearson correlation coefficient between global spread of popularity and each time-based metric: - spread vs persistence
- spread vs duration **A Pearson correlation measures linear relationship strength between two numeric variables. It is between -1 and +1.

```
spread_persist_cor  <- cor(final_netflix_top_10_data$global_spread, final_netflix_top_10_data$global_pe
spread_duration_cor <- cor(final_netflix_top_10_data$global_spread, final_netflix_top_10_data$global_du

cat("Correlation(global_spread, global_persistence):", round(spread_persist_cor, 4), "\n")
```

```
## Correlation(global_spread, global_persistence): 0.4098
```

```
cat("Correlation(global_spread, global_duration):   ", round(spread_duration_cor, 4), "\n")
```

```
## Correlation(global_spread, global_duration):    0.1917
```

Answering the question “Are globally distributed films more persistent,” we can say that since the correlation coefficient for spread vs persistence was 0.4092, a moderately positive value, films that appear in more countries’ top 10 lists do tend to last longer in at least one country. However, because it is not a 0.9, how much geographically the film has gained popularity accounts to some of its persistence, but not all of it. There are clear exceptions, such as local long-term hits or global flash films.

The correlation coefficient for spread vs duration of 0.1916, a weakly positive value. This suggests that films that appear in more countries’ top 10 lists tend to remain active over a longer global time span, but this relationship is weak and there are a lot of deviations.

Overall, these coefficients suggest that geographic breadth of popularity is more strongly associated with persistence in popularity than how long it is popular. Simply put, being popular in many countries help a film succeed somewhere strongly, but it does not necessarily extend its overall lifespan of popularity.

As a reminder, global_persistence is the maximum amount of weeks in one country, while global_duration is the time span between film’s first and last appearance in the top 10 list anywhere in the world. Although in many cases these values may be similar, they can differ when a film appears in different countries at very different times.

That being said, there may be many explanations to this difference in correlation coefficients. One possible explanation is that many Netflix films are released across many countries at the same time, implying that the traction the movie receives happens at similar times around many different countries. As a result, wider distribution may increase the likelihood of strong performance in at least one market, but it does not mean the overall time span of popularity will increase as well. If we had focused on theatrical releases, we may see a stronger relationship between spread and duration, as the time difference between international releases and domestic releases are large.

Distributions”

We examine the distribution of the three popularity metrics for each movie so we can understand their overall range, central tendency, and variability. Using the summary() function, we find the minimum, first quartile, median, mean, etc for each metric. It allows us to understand: - how long films typically persist in the

```
cat("Summary: global_persistence\n"); print(summary(final_netflix_top_10_data$global_persistence))
```

```
## Summary: global_persistence
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
##   1.000   1.000   2.000   2.403   3.000  39.000
```

```
cat("\nSummary: global_spread\n");      print(summary(final_netflix_top_10_data$global_spread))
```

```
## 
## Summary: global_spread
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
##    1.00    1.00    3.00   10.64   10.00   94.00
```

```
cat("\nSummary: global_duration\n");    print(summary(final_netflix_top_10_data$global_duration))
```

```
## 
## Summary: global_duration
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
##    1.00    1.00    2.00   10.99   15.00   60.00
```

From the summarized global_persistence data, we can see that half of all films last 2 weeks or less in their strongest country. Since the mean (2.4) is larger than the median (2) and the max-value is 39, a value far from the mean, we see that this distribution is right-skewed. Overall, most films have short lifespans, but there are a few films that last a very long time.

From the summarized global_spread data, we can see that the mean (10.62) is significantly larger than the median (3), implying a strong right skew. Therefore, this implies that most films' popularities are local to a few countries, but there are a few films that achieve extreme wide international popularity, pulling the mean upward.

From the summarized global_duration data, we can again see that the mean(10.97) is significantly larger than the median (2), showing another strong right skew. Again, there are a few outliers.

Overall, the distribution of popularity metrics reveal right-skewness. This helps explain the correlation results. Because most films have low levels of spread, persistence, and duration, the majority of the dataset values lie in a tight region of low numbers–yet just a few have very large values. Thus, the correlation essentially stems from these few extreme values, and a moderate, not a strong, correlation is produced.

Identifying anomalies (two types)

A) "Local long-term hits" Films that stayed 10+ weeks but only appeared in 1–2 countries.

B) "Global flash" Films that hit 30+ countries, thus widely reached, but only lasted 1 week.

```
anomaly_local_long_term_hit <- final_netflix_top_10_data %>%
  filter(global_persistence >= 10, global_spread <= 2) %>%
  arrange(desc(global_persistence), best_rank)

anomaly_global_flash <- final_netflix_top_10_data %>%
  filter(global_spread >= 30, global_persistence == 1) %>%
  arrange(desc(global_spread), best_rank)

cat("Anomaly A (Local long-term hits) count:", nrow(anomaly_local_long_term_hit), "\n")
```

```
## Anomaly A (Local long-term hits) count: 7
```

```r
cat("Anomaly B (global flash) count:", nrow(anomaly_global_flash), "\n\n")
```

```
## Anomaly B (global flash) count: 6
```

```r
cat("Local long term hit preview:\n")
```

```
## Local long term hit preview:
```

```r
kable(head(anomaly_local_long_term_hit, 10))
```

movie_key	movie_title	global_persistence	global_spread	global_duration	best_rank
omo ghetto the saga	Omo Ghetto: the Saga	19	2	34	1
bet on friendship	Bet on Friendship	14	2	27	1
upin ipin the lone gibbon kris	Upin & Ipin: The Lone Gibbon Kris	14	1	41	6
night in paradise	Night in Paradise	13	1	42	5
asakusa kid	Asakusa Kid	11	1	12	1
brother in love 2	Brother in Love 2	10	2	13	1
sinkhole	Sinkhole	10	1	12	1

```r
cat("\nGlobal flash preview:\n")
```

```
##
## Global flash preview:
```

```r
kable(head(anomaly_global_flash, 10))
```

movie_key	movie_title	global_persistence	global_spread	global_duration	best_rank
the next 365 days	The Next 365 Days	1	92	1	1
look both ways	Look Both Ways	1	89	1	1
royalteen	Royalteen	1	66	1	1
dont kill me	Don’t Kill Me	1	51	1	2
blasted	Blasted	1	41	1	3
emma	Emma.	1	31	30	3

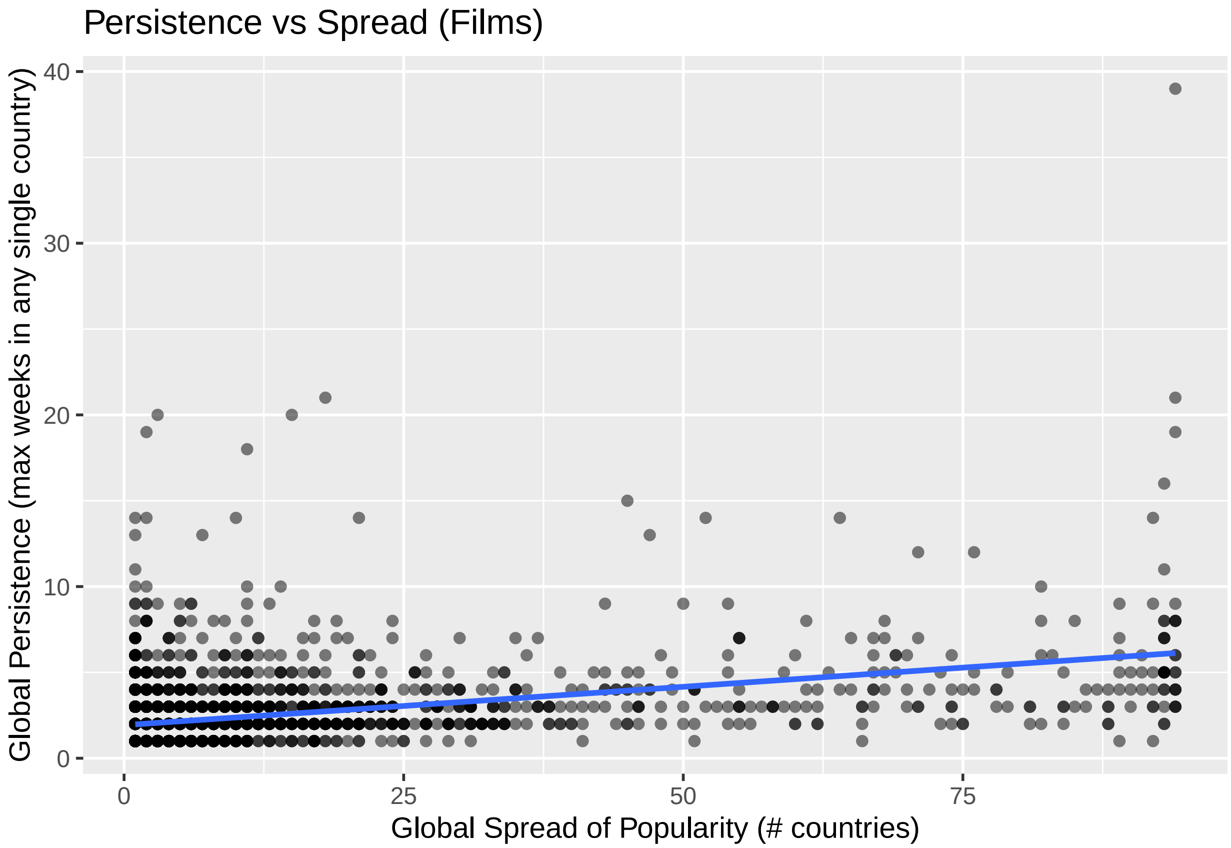
```r
write_csv(anomaly_local_long_term_hit, file.path(out_dir, "anomaly_local_long_term_hit.csv"))
write_csv(anomaly_global_flash, file.path(out_dir, "anomaly_global_flash.csv"))

cat("\nSaved:\n- data/processed/anomaly_local_long_term_hit.csv\n- data/processed/anomaly_global_flash.
```

```
##
## Saved:
## - data/processed/anomaly_local_long_term_hit.csv
## - data/processed/anomaly_global_flash.csv
```

Scatterplot: persistence vs spread

```
p <- ggplot(final_netflix_top_10_data, aes(x = global_spread, y = global_persistence)) +
  geom_point(alpha = 0.5) +
  geom_smooth(method = "lm", se = FALSE) +
  labs(
    x = "Global Spread of Popularity (# countries)",
    y = "Global Persistence (max weeks in any single country)",
    title = "Persistence vs Spread (Films)"
  )
p
```

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



```
ggsave(
  filename = file.path(out_dir, "persistence_vs_spread.png"),
  plot = p,
  width = 7, height = 5, dpi = 200
)
```

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

```r
cat("Saved plot: data/processed/persistence_vs_spread.png\n")
```

```
## Saved plot: data/processed/persistence_vs_spread.png
```

Looking at this scatterplot, we can see that the regression line's slope is positive. This means that as global spread of popularity increases, the global persistence tends to increase as well. However, if we take a closer look at the individual points, on the left side, we see a dense black region, implying that most films are low-spread, and low-persistent. Then, as spread increases, the dispersion of global persistence increases as well (some are short-lived, yet some are high). The few outliers (~90 movies and ~39 weeks) drive the positive trend. This suggests that if a film enters the Top 10 in more countires, it has more opportunities to perform strongly in at least one of them, but it does not guarantee continuing popularity.

Final output recap for Dataset #1

Output Datasets/Images (written to `data/processed/`)

- `movie_country_table.csv` (one row per movie × country)
- `anomaly_local_cult.csv` (10+ weeks but only 1–2 countries) –> Based off on the final modeling dataset, this is the subset of movies with high persistence but low spread of popularity. In simple terms, these are local long-term hits.
- `anomaly_global_flash.csv` (30+ countries but only 1 week) –> Based off on final modeling dataset, this is the subset of movies with high spread of popularity but low persistence. In simple terms, these are global flash hits.
- `persistence_vs_spread.png` (scatterplot) –> Scatterplot showing global persistence vs global spread.

The final,movie-level dataset, derived from Dataset #1 (Top 10 Netflix Dataset) saved to: - `data/processed/final_netflix_` Columns of this final dataset: - `movie_title` - `global_persistence` - `global_spread` - `global_duration` - `best_rank`

```r
cat("Final table preview (top 10):\n")
```

```
## Final table preview (top 10):
```

```r
kable(head(final_netflix_top_10_data, 10))
```

movie_key	movie_title	global_persistence	global_spread	global_duration	best_rank
red notice	Red Notice	39	94	41	1
army of thieves	Army of Thieves	21	94	30	1
dont look up	Don't Look Up	19	94	20	1
love hard	Love Hard	9	94	9	1
kate	Kate	8	94	8	1
the tinder swindler	The Tinder Swindler	8	94	8	1
the unforgivable	The Unforgivable	8	94	8	1
blood red sky	Blood Red Sky	6	94	57	1
the guilty	The Guilty	6	94	24	1
sweet girl	Sweet Girl	5	94	5	1

Using IMDb Dataset

Which movie was the most highly-rated?

We find the top-rated film. However, we only consider movies with a significant amount of votes (1000 votes) to prevent obscure films with a very few high ratings appearing at the top.
Then, we save this new data into our project directory.

```
top_rated <- df |>
  filter(numVotes >= 1000, !is.na(averageRating)) |>
  arrange(desc(averageRating)) |>
  head(20)

top_rated
```

```
## # A tibble: 20 x 11
##    tconst     titleType primaryTitle     originalTitle isAdult startYear endYear
##    <chr>      <chr>     <chr>            <chr>            <int>     <dbl>   <dbl>
##  1 tt2301451  tvEpisode Ozymandias       Ozymandias           0      2013      NA
##  2 tt39196717 movie     Ranapati Shivra~ Ranapati Shi~        0      2026      NA
##  3 tt0701989  tvEpisode Everyone's Wait~ Everyone's W~        0      2005      NA
##  4 tt11028174 tvEpisode The View from H~ The View fro~        0      2020      NA
##  5 tt11294360 tvEpisode End of the Prol~ End of the P~        0      2019      NA
##  6 tt1204265  tvEpisode Sozin's Comet, ~ Sozin's Come~        0      2008      NA
##  7 tt12187040 tvEpisode Plan and Execut~ Plan and Exe~        0      2022      NA
##  8 tt1310121  tvEpisode Re;              Re;                  0      2008      NA
##  9 tt1683088  tvEpisode Face Off         Face Off             0      2011      NA
## 10 tt2178784  tvEpisode The Rains of Ca~ The Rains of~        0      2013      NA
## 11 tt2301455  tvEpisode Felina           Felina               0      2013      NA
## 12 tt35841519 movie     Madham           Madham               0      2026      NA
## 13 tt3748420  tvEpisode Anger × and × L~ Anger × and ~        0      2014      NA
## 14 tt38504411 tvEpisode I'll Believe in~ I'll Believe~        0      2025      NA
## 15 tt4283088  tvEpisode Battle of the B~ Battle of th~        0      2016      NA
## 16 tt4283094  tvEpisode The Winds of Wi~ The Winds of~        0      2016      NA
## 17 tt6066926  tvEpisode Dream: To See I~ Dream: To Se~        0      1997      NA
## 18 tt6381882  tvEpisode A Regular Epic ~ A Regular Ep~        0      2017      NA
## 19 tt8084176  tvEpisode 407 Proxy Authe~ 407 Proxy Au~        0      2019      NA
## 20 tt9313978  tvEpisode Victory and Dea~ Victory and ~        0      2020      NA
## # i 4 more variables: runtimeMinutes <dbl>, genres <chr>, averageRating <dbl>,
## #   numVotes <dbl>
```

```
write_csv(top_rated, file.path(out_dir, "top_rated_movies.csv"))

cat("Saved: data/processed/top_rated_movies.csv\n")
```

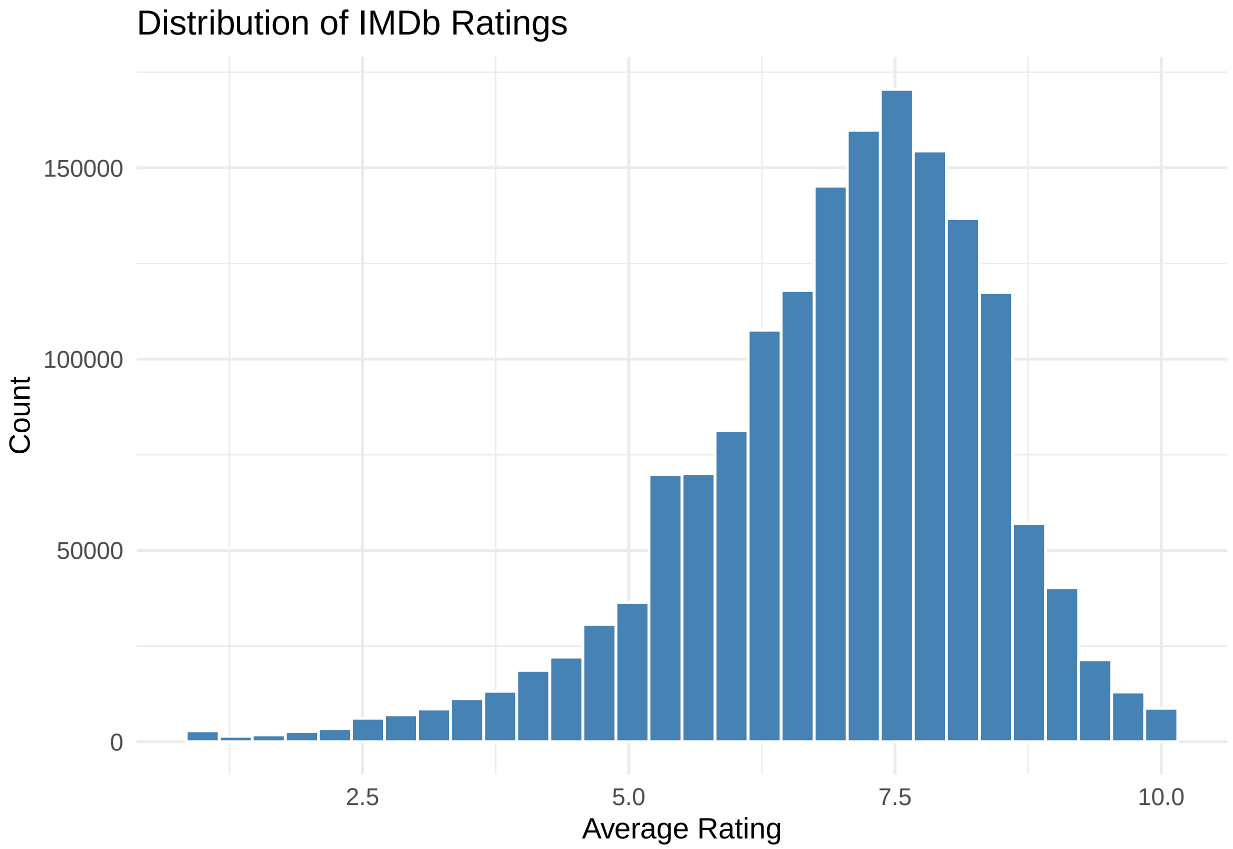
```
## Saved: data/processed/top_rated_movies.csv
```

Looking at the results, we can see that Ozymandias was the top-rated, with a significant number of 286687 votes.

Distribution of IMDb Ratings

We saved this distribution plot into our project directory as an image for future reference.

```
rating_distribution_plot <- ggplot(df |> filter(!is.na(averageRating)),
       aes(x = averageRating)) +
  geom_histogram(bins = 30, fill = "steelblue", color = "white") +
  theme_minimal() +
  labs(title = "Distribution of IMDb Ratings",
       x = "Average Rating",
       y = "Count")

rating_distribution_plot
```



```
ggsave(
  filename = file.path(out_dir, "imdb_rating_distribution.png"),
  plot = rating_distribution_plot,
  width = 7, height = 5, dpi = 200
)

cat("Saved plot: data/processed/imdb_rating_distribution.png\n")
```

```
## Saved plot: data/processed/imdb_rating_distribution.png
```

The distribution is unimodal and is slightly left-skewed as it has a longer tail toward lower ratings. Most movies are between the ratings of 6.0 and 8.0, meaning that IMDb ratings are general concentrated in that number. Morever, it is hard for the ratings to be low (<3) and high (>9). The slight left-skewness implies that is harder to get an extremely higher rating than an extremely lower one. This sort of distribution is common for crowd-sourced rating systems as the large size of data reduces the impact of extreme outliers.

Which genres have the highest average IMDb rating?

We could just look at the basic average rating by genre. However, some genres may have very few movies and artificially high averages, or there might be a few number of votes dominating the high-rating. Thus, we willl use number of votes to weight the ratings.

First, we remove movies with missing ratings/vote counts (both are needed to compute a vote-weighted average). Then, after grouping the dataset by genre (genres were separated into separate rows previously), for each genre, we compute vote-weighted average rating (weighted_rating), the number of movies in that genre, and the total number of votes. This allows us to consider sample size instead of just the rating.

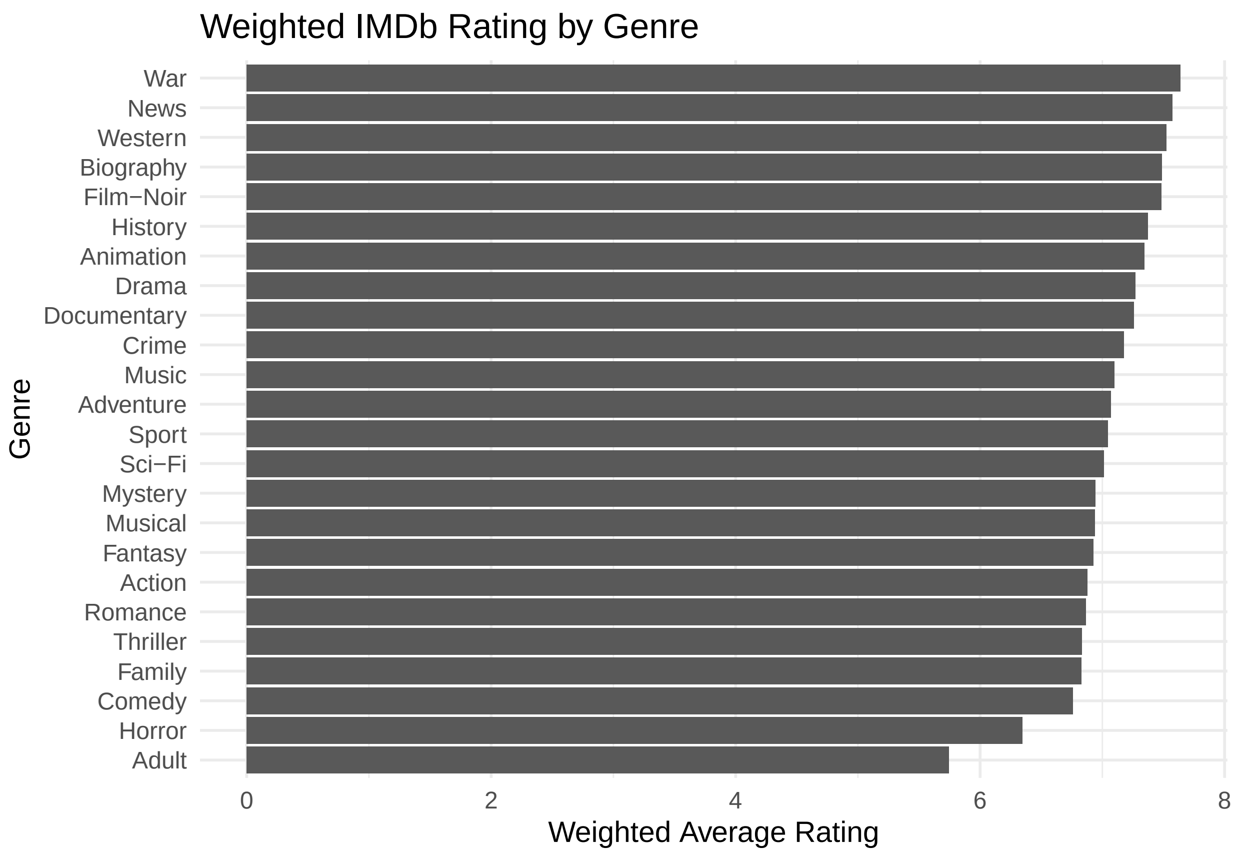
Finally, we sort genres from highest to lowest weighted average rating.

```r
genre_weighted <- imdb_movies_final |>
  filter(!is.na(averageRating), !is.na(numVotes)) |>
  group_by(genres) |>
  summarise(
    weighted_rating = weighted.mean(averageRating, numVotes, na.rm = TRUE),
    n_movies = n(),
    total_votes = sum(numVotes, na.rm = TRUE),
    .groups = "drop"
  ) |>
  filter(n_movies >= 200) |>
  arrange(desc(weighted_rating))

head(genre_weighted)
```

```
## # A tibble: 6 x 4
##   genres    weighted_rating n_movies total_votes
##   <chr>               <dbl>    <int>       <dbl>
## 1 War                  7.64     6716    28936294
## 2 News                 7.57      682      249930
## 3 Western              7.52     5061    12384089
## 4 Biography            7.49    10762    81289783
## 5 Film-Noir            7.48      877     4123876
## 6 History              7.37     9573    35511767
```

Then, to visualize this, we generate a bar graph displaying the vote-weighted rating for each genre. We made sure to reorder the genre so that it is shown from lowest to highest.
To make it easier to understand, we flipped the axes so that the genres are shown vertically. We saved this plot into our project directory as an image for future reference.

```r
genre_weighted_rating_plot <- genre_weighted |>
  ggplot(aes(x = reorder(genres, weighted_rating),
             y = weighted_rating)) +
  geom_col() +
  coord_flip() +
  labs(
    title = "Weighted IMDb Rating by Genre",
    x = "Genre",
    y = "Weighted Average Rating"
  ) +
  theme_minimal()

genre_weighted_rating_plot
```



```r
ggsave(
  filename = file.path(out_dir, "weighted_imdb_rating_by_genre.png"),
  plot = genre_weighted_rating_plot,
  width = 8, height = 6, dpi = 200
)

cat("Saved plot: data/processed/weighted_imdb_rating_by_genre.png\n")
```

```
## Saved plot: data/processed/weighted_imdb_rating_by_genre.png
```

Looking at the results, movies related to war has the highest rating!

Using Merged Final Dataset:

Does IMDb rating correspond to Netflix popularity?

First, we will check linear correlation between the ratings and the three metrics of popularity we defined when we were working with the Netflix dataset.

```r
cor_results <- netflix_imdb_final |>
  summarise(
    cor_rating_spread = cor(averageRating, global_spread, use = "complete.obs"),
    cor_rating_persistence = cor(averageRating, global_persistence, use = "complete.obs"),
    cor_rating_duration = cor(averageRating, global_duration, use = "complete.obs")
  )

print(cor_results)
```

```
## # A tibble: 1 x 3
##   cor_rating_spread cor_rating_persistence cor_rating_duration
##               <dbl>                  <dbl>               <dbl>
## 1           -0.0239                 0.0615              0.0865
```

The correlations between rating and the metrics of popularity are -0.02 (spread), 0.06 (persistence), and 0.09 (duration). The correlations with persistence and duration are weakly positive but negligible in magnitude. Overall, the popularity metrics show a weak linear relationship with IMDb ratings, and this implies that IMDb rating alone is not strongly associated nor definitely predictive of Netflix popularity. To take a step further, we can inference that Netflix popularity may be influenced by marketing, regional preferences, language accessibility, and many more other factors than IMDb ratings.

It is important to note that correlation does not imply causation. While we are examining whether IMDb ratings are related to Netflix popularity, the direction of influence cannot be determined. However, because the correlations are extremely small, it is very unlikely that there is any meaningful causal effect on either side.

We now check the scatterplots, as correlation only measures linear relationships and there may be other relationships.

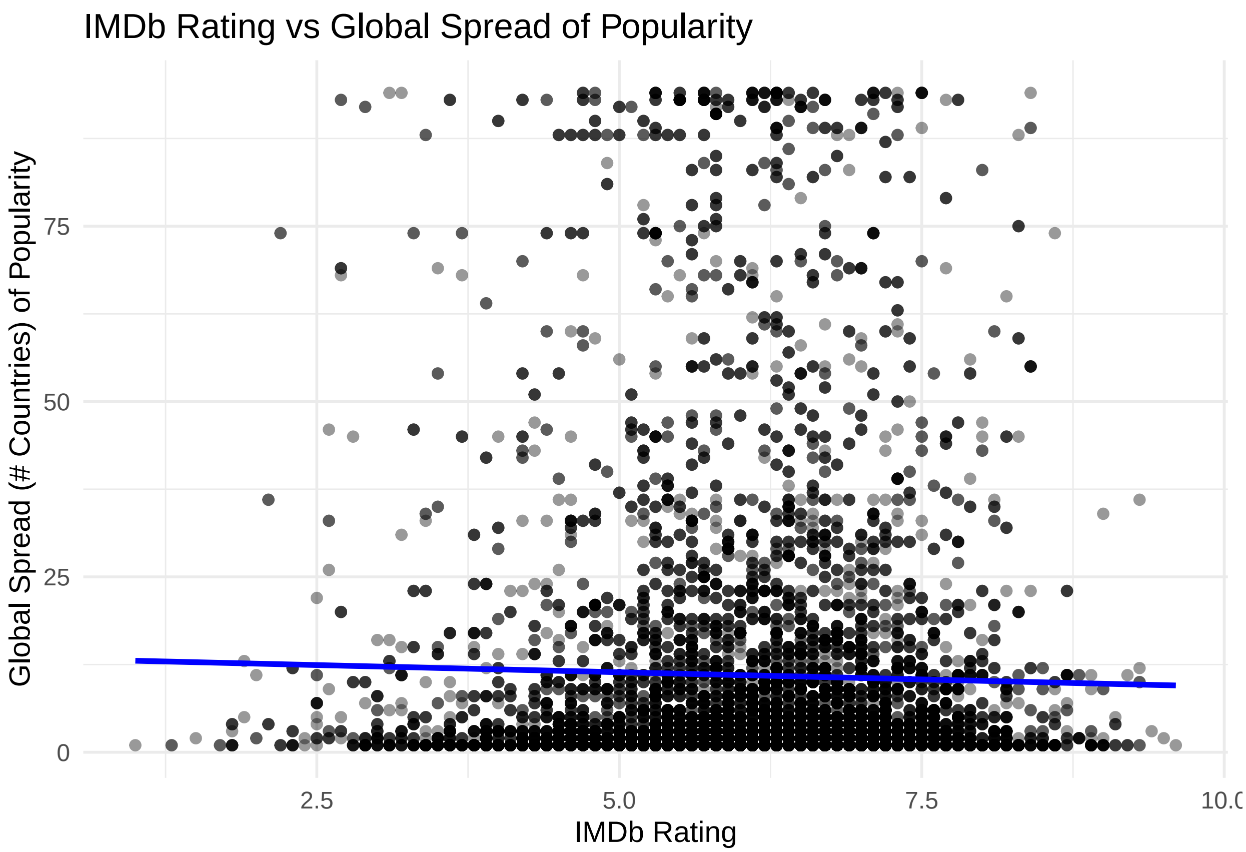
First, the scatterplot with global_spread.

```
p1 <- netflix_imdb_final |>
  ggplot(aes(x = averageRating, y = global_spread)) +
  geom_point(alpha = 0.4) +
  geom_smooth(method = "lm", color = "blue", se = FALSE) +
  theme_minimal() +
  labs(
    title = "IMDb Rating vs Global Spread of Popularity",
    x = "IMDb Rating",
    y = "Global Spread (# Countries) of Popularity"
  )

p1
```

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

```
## Warning: Removed 2816 rows containing non-finite outside the scale range
## (`stat_smooth()`).
```

```
## Warning: Removed 2816 rows containing missing values or values outside the scale range
## (`geom_point()`).
```



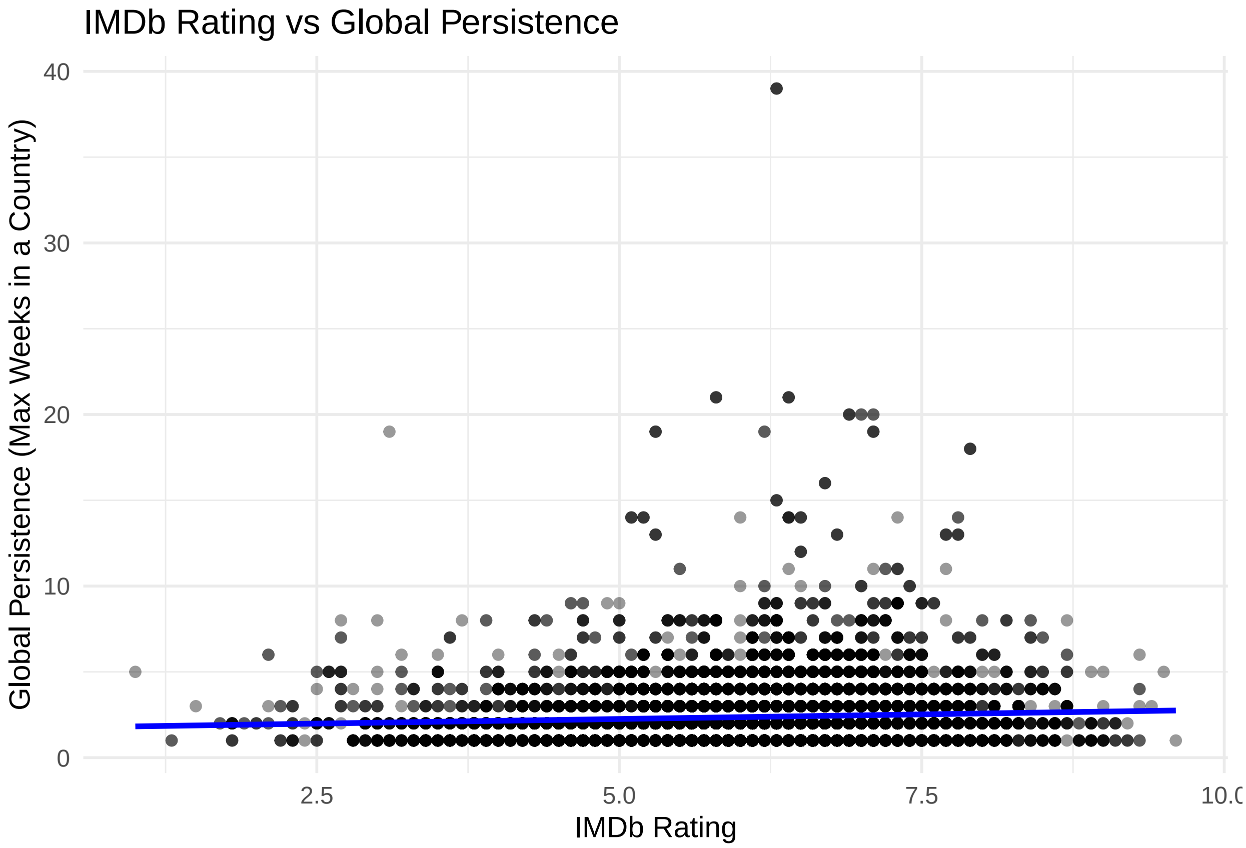
The points look very widely scattered, and unlike the common assumption, a higher rating does not correspond to greater popularity. There is no clear relationship, nor a strong slope. One insight that can be found is that movies with a higher global spread often exist across mid-range ratings. This implies that being highly rated is not a requirement for global spread of popularity. This is consistent with the near-zero correlation coefficient of 0.02.

Second, the scatterplot with global_persistence:

```r
p2 <- netflix_imdb_final |>
  ggplot(aes(x = averageRating, y = global_persistence)) +
  geom_point(alpha = 0.4) +
  geom_smooth(method = "lm", color = "blue", se = FALSE) +
  theme_minimal() +
  labs(
    title = "IMDb Rating vs Global Persistence",
    x = "IMDb Rating",
    y = "Global Persistence (Max Weeks in a Country)"
  )

p2
```

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

```
## Warning: Removed 2816 rows containing non-finite outside the scale range
## (`stat_smooth()`).
## Warning: Removed 2816 rows containing missing values or values outside the scale range
## (`geom_point()`).
```



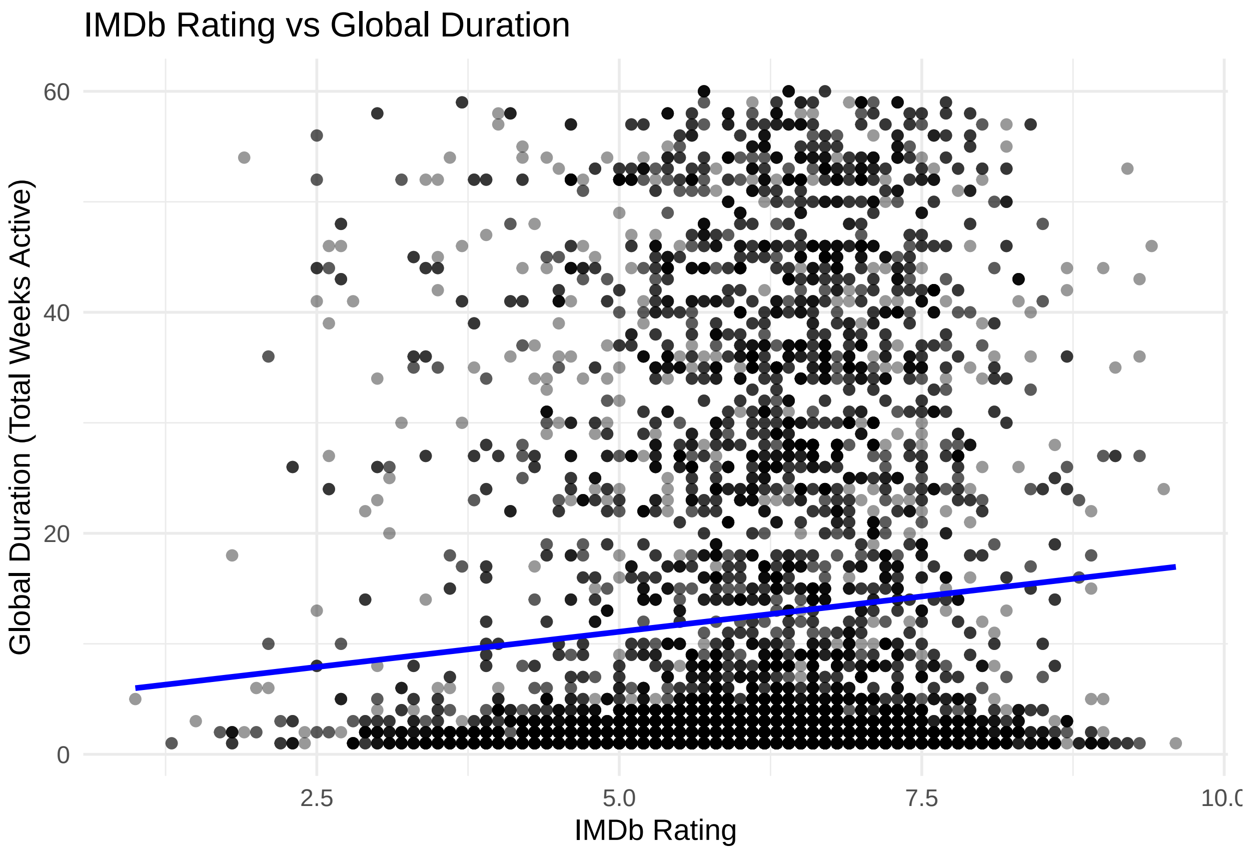
For the scatterplot of IMDb rating vs. global persistence, the regression line is almost flat, supporting the correlation coefficient of 0.06. Most points are heavily clustered between persistence values of 1 - 10 weeks. Yet, there are a few outliers with a global persistence of 15-40 weeks, but they appear within moderate IMDb ratings. This suggests that most films do not remain in the Top 10 for more than 2 months, regardless of how high of a rating they have received.

Third, the scatterplot woth global_duration.

```r
p3 <- netflix_imdb_final |>
  ggplot(aes(x = averageRating, y = global_duration)) +
  geom_point(alpha = 0.4) +
  geom_smooth(method = "lm", color = "blue", se = FALSE) +
  theme_minimal() +
  labs(
    title = "IMDb Rating vs Global Duration",
    x = "IMDb Rating",
    y = "Global Duration (Total Weeks Active)"
  )

p3
```

```
## `geom_smooth()` using formula = 'y ~ x'
## Warning: Removed 2816 rows containing non-finite outside the scale range
## (`stat_smooth()`).
```

```
## Warning: Removed 2816 rows containing missing values or values outside the scale range
## (`geom_point()`).
```



Compared to spread and persistence, the global duration metric exhibits a slightly stronger positive relationship with IMDb ratings. This regression line shows a slight upward slope, aligning with its correlation coefficient of r = 0.086. However, if we look at r^2, it is 0.0074. This value means that only 0.7% of the variation in global duration can be statistically associated with differences in IMDb rating. In other words, more than 99% of variation is due to other factors.
Looking at the plot, while higher-rated films slightly tend to have longer durations on average, the possible range of duration values is still extremely large across all rating levels. Throughout the entire IMDb rating spectrum, both short and long durations appear, implying that rating does not significantly show how long a film is popular.

Finally, we will save these plots as images into our project directory for future reference.

```r
ggsave(file.path(out_dir, "rating_vs_spread.png"), p1, width = 7, height = 5, dpi = 200)
```

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

```
## Warning: Removed 2816 rows containing non-finite outside the scale range
## (`stat_smooth()`).
## Warning: Removed 2816 rows containing missing values or values outside the scale range
## (`geom_point()`).
```

```r
ggsave(file.path(out_dir, "rating_vs_persistence.png"), p2, width = 7, height = 5, dpi = 200)
```

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

```
## Warning: Removed 2816 rows containing non-finite outside the scale range
## (`stat_smooth()`).
## Removed 2816 rows containing missing values or values outside the scale range
## (`geom_point()`).
```

```r
ggsave(file.path(out_dir, "rating_vs_duration.png"), p2, width = 7, height = 5, dpi = 200)
```

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

```
## Warning: Removed 2816 rows containing non-finite outside the scale range
## (`stat_smooth()`).
## Removed 2816 rows containing missing values or values outside the scale range
## (`geom_point()`).
```

```r
cat("Saved rating vs popularity plots.\n")
```

```
## Saved rating vs popularity plots.
```

All in all, consistent with the correlation coefficient results, the analysis indicates that IMDb rating is not strongly associated with Netflix popularity. The scatterplots reinforce this statement by showing a wide dispersion in values of popularity metrics across ALL IMDb rating values. Whether the movie is highly-rated or not, any popularity metric is possible, suggesting that other factors such as language, release time, and marketing, play a large role in Netflix popularity.

Summary of Key Findings

1) Breadth and depth are related, but not deterministically

Our Netflix-derived popularity metrics show that **global spread (breadth)** and **global persistence (depth)** are related, but the relationship is **probabilistic rather than deterministic**.

- The Pearson correlation between **global spread** and **global persistence** is **0.4092** (moderately positive).
  This suggests that films reaching more countries tend to be more persistent in at least one market.
- In contrast, the correlation between **global spread** and **global duration** is **0.1916** (weakly positive). This implies that broad geographic reach increases the chance of strong performance somewhere, but does **not necessarily extend the overall global lifespan** of Top 10 popularity.

Interpretation: Netflix success can come from different mechanisms. Some films spread widely due to cross-market appeal, while others stay strong through deep local resonance. Wide reach helps, but it does not guarantee long-lasting global attention.

2) Netflix popularity is highly right-skewed

Across the Netflix-derived popularity metrics (**spread**, **persistence**, and **duration**), the distributions are **highly right-skewed**:

- Most films have **low spread / low persistence / short duration**.
- A small number of films achieve **extreme values** and dominate the upper tail.

This pattern is consistent with **blockbuster-like dynamics**, where a few titles capture a disproportionate share of attention while most remain local and short-lived. One reason is that Top 10 visibility can create a feedback loop: once a film becomes visible, platform exposure and word-of-mouth may help it stay visible longer than typical films.

---

3) Two anomaly patterns reveal distinct "types" of Netflix success

To make the above patterns more interpretable, we identified two notable anomaly groups:

- **Local long-term hits:** films with **high persistence (10 weeks)** but **low spread (2 countries)**. These titles likely reflect strong local cultural resonance, language preference, or region-specific marketing. They may not travel widely, but they can stay competitive for a long time in one or two markets.

- **Global flash hits:** films with **high spread (30 countries)** but **persistence of 1 week**.
  This pattern is consistent with broad but short-lived attention—such as simultaneous release spikes or strong initial platform promotion—followed by fast rotation out of the Top 10 as new titles enter.

Together, these anomaly groups reinforce that Netflix popularity is **multi-dimensional**: a film can be high on breadth but low on depth (or the reverse).

---

4) IMDb rating is a poor predictor of Netflix popularity (in our merged sample)

In the merged Netflix–IMDb dataset, IMDb ratings show **very weak** relationships with Netflix popularity metrics:

- Correlation between IMDb rating and **global spread**: **-0.02**

- Correlation between IMDb rating and **global persistence**: **0.06**

- Correlation between IMDb rating and **global duration**: **0.09**

Scatterplots further show wide dispersion of popularity across all rating levels. This implies that other drivers—such as **release timing**, **platform promotion**, **language availability**, **franchise effects**, and **regional tastes**—likely dominate Netflix Top 10 performance.

> Note: This does not mean film "quality" is irrelevant. Rather, it suggests IMDb rating alone (as a single summary number) does not capture the main determinants of Top 10 reach and persistence on Netflix.

---

5) Genre differences exist in IMDb ratings, but do not translate cleanly into Netflix popularity

Weighted average ratings by genre indicate that some genres (e.g., **war**) rank highly in IMDb "quality." However, these higher ratings do not consistently map to broader or longer Top 10 presence on Netflix.

A plausible explanation is that IMDb ratings reflect viewer evaluation among a particular user base, while Netflix Top 10 popularity reflects **mass-market engagement and platform distribution**. Some genres may be highly rated but appeal to narrower audiences, whereas other genres may travel more easily across markets due to accessibility, marketing, or broad entertainment value.

Overall takeaway

Overall, our findings suggest that Netflix Top 10 success has multiple "modes": broad reach is moderately linked to strong performance somewhere, but does not necessarily extend global lifespan, and IMDb ratings alone do not explain why some films become widely and persistently popular on Netflix.

Challenges Faced and Future Recommendations:

Challenges faced

1. **Title-based merging without a shared unique identifier.**
   Netflix does not provide IMDb's unique title ID (`tconst`), so the merge relied on standardized text titles (`movie_key`). This creates unavoidable mismatch risk from subtitles, punctuation, alternate spellings, translations, and Netflix-specific naming. Even so, we achieved an **87.4%** match rate (2,308 matched out of 2,640 Netflix films), but the remaining unmatched titles can bias results if mismatches are systematic.

2. **Granularity mismatch and loss of temporal detail.**
   The Netflix data is originally at the **movie × country × week** level, while the merged modeling dataset is **one row per movie**. Aggregation is necessary for film-level comparison, but it also compresses information about *when* and *where* popularity occurs (e.g., staggered country releases).

3. **Data quality issues in large raw sources.**

   - The IMDb `title.basics` file contains a small number of malformed rows (shifted columns due to missing separators).

   - The Netflix dataset can contain duplicates or inconsistent string formatting, requiring careful cleaning and integrity checks.

4. **Interpretation limits of correlation.**
   Correlation assesses linear association and does not imply causation. In addition, Netflix Top 10 appearance is affected by platform-specific factors (recommendation algorithms, catalog availability, promotions) that are not observed in our datasets.

Future recommendations

1. **Add language and region features.**
   A natural next step is to test whether **original language** (and availability of dubbing/subtitles) explains spread or persistence. Language may interact strongly with geographic reach, especially for non-English films.

2. **Use stronger entity resolution for matching.**
   Improve matching with fuzzy joins (edit distance), alias tables, or external mapping resources (e.g., TMDb IDs) to reduce title-variant mismatches and increase the robustness of the merged dataset.

3. **Incorporate additional predictors beyond IMDb rating.**
   Enrich the model with variables such as release year, runtime, franchise/series indicators, marketing proxies (social media trends), and country-level Netflix penetration to better explain popularity.

4. **Model popularity with appropriate statistical tools.**
   Since popularity metrics are right-skewed, consider log transforms and count models. Beyond simple correlation, a multivariate regression (or tree-based model) could quantify which features matter most while controlling for confounders.

5. **Preserve time dynamics.**
   Rather than collapsing to one row per film, analyze a panel structure (film-country-week) to study diffusion patterns (e.g., how popularity spreads across regions over time) and to separate simultaneous global releases from staggered adoption.

Link to GitHub repository:

https://github.com/mk2023/AppliedDSProject-1

Each Member's Contribution:

Pablo: IMDb Dataset Acquisition, Cleaning, Preprocessing. Qiujun: Kaggle Dataset Acquisition, Cleaning, Preprocessing. Minseul: Combining Dataset #1(Kaggle) & #2 (IMDb) CLeaning & Preprocessing Stages, Merging Dataset, Exploratory Data Analysis, Feature Engineering. Feiran: Introduction, Summary of Key Findings, Challenges Faced and Future Recommendations.