Exploring Netflix Movies Dataset with TidyVerse

Sabina Baraili (original), Sachi Kapoor (extension)

November 21, 2025

Contents

Extended with additional TidyVerse examples by a classmate for DATA 607 (Fall 2025).

1. Introduction

This vignette explores the **Kaggle Netflix titles dataset** using the TidyVerse.
We will:

- Load and clean the Netflix dataset.
- Focus on **movies** (as opposed to TV shows).
- Use `dplyr` to summarize data and create derived variables.
- Use `tidyr` to split multi-valued columns (e.g., multiple countries and genres in one cell).
- Use `ggplot2` to visualize trends in movie releases.
- Use `forcats` to focus on the most common genres and reorder factor levels for clearer plots.

The goal is to demonstrate a realistic data-analysis workflow using TidyVerse functions.

2. Load and Inspect the Netflix Dataset

```r
# The dataset is often named "netflix_titles.csv" on Kaggle.
# This chunk looks for the file either in a "data/" folder or in the current folder.

data_path <- dplyr::case_when(
  file.exists("data/netflix_titles.csv") ~ "data/netflix_titles.csv",
  file.exists("netflix_titles.csv")      ~ "netflix_titles.csv",
  TRUE                                   ~ "data/netflix_titles.csv" # default if you keep it in data/
)

netflix_raw <- readr::read_csv(data_path)

# Peek at the structure of the dataset
glimpse(netflix_raw)
```

```
## Rows: 8,807
## Columns: 12
## $ show_id      <chr> "s1", "s2", "s3", "s4", "s5", "s6", "s7", "s8", "s9", "s1~
## $ type         <chr> "Movie", "TV Show", "TV Show", "TV Show", "TV Show", "TV ~
## $ title        <chr> "Dick Johnson Is Dead", "Blood & Water", "Ganglands", "Ja~
## $ director     <chr> "Kirsten Johnson", NA, "Julien Leclercq", NA, NA, "Mike F~
## $ cast         <chr> NA, "Ama Qamata, Khosi Ngema, Gail Mabalane, Thabang Mola~
## $ country      <chr> "United States", "South Africa", NA, NA, "India", NA, NA,~
## $ date_added   <chr> "September 25, 2021", "September 24, 2021", "September 24~
## $ release_year <dbl> 2020, 2021, 2021, 2021, 2021, 2021, 2021, 1993, 2021, 202~
## $ rating       <chr> "PG-13", "TV-MA", "TV-MA", "TV-MA", "TV-MA", "TV-MA", "PG~
## $ duration     <chr> "90 min", "2 Seasons", "1 Season", "1 Season", "2 Seasons~
## $ listed_in    <chr> "Documentaries", "International TV Shows, TV Dramas, TV M~
## $ description  <chr> "As her father nears the end of his life, filmmaker Kirst~
```

Commentary

- `readr::read_csv()` reads the CSV into a tibble (`netflix_raw`).
- `glimpse()` shows us column names and data types (for example: `type`, `title`, `director`, `cast`, `country`, `date_added`, `release_year`, `rating`, `duration`, `listed_in`, etc.).

3. Focus on Movies and Select Relevant Columns

The dataset contains both **movies** and **TV shows**. For this vignette, we'll focus only on movies to keep the analysis more consistent.

```r
netflix_movies <- netflix_raw %>%
  # Keep only rows where type is "Movie"
  filter(type == "Movie") %>%
  # Select a subset of columns that are useful for our analysis
  select(
    title,
    country,
    date_added,
    release_year,
    rating,
    duration,
    listed_in
  )
```

```r
# Show the first few rows of our movies-only dataset
head(netflix_movies)
```

```
## # A tibble: 6 x 7
##   title                country date_added release_year rating duration listed_in
##   <chr>                <chr>   <chr>             <dbl> <chr>  <chr>    <chr>
## 1 Dick Johnson Is Dead United~ September~         2020 PG-13  90 min   Document~
## 2 My Little Pony: A N~ <NA>    September~         2021 PG     91 min   Children~
## 3 Sankofa              United~ September~         1993 TV-MA  125 min  Dramas, ~
## 4 The Starling         United~ September~         2021 PG-13  104 min  Comedies~
## 5 Je Suis Karl         German~ September~         2021 TV-MA  127 min  Dramas, ~
## 6 Confessions of an I~ <NA>    September~         2021 TV-PG  91 min   Children~
```

Commentary

- `filter(type == "Movie")` removes TV shows, so we only analyze movies.
- `select()` keeps a lean set of columns to reduce clutter and memory usage.
- Having a smaller set of variables makes later transformations easier to read and interpret.

4. Clean Text Columns and Handle Missing Values

Some columns (like `country` and `listed_in`) are character strings that may contain multiple comma-separated values.
Before splitting them apart, we'll do a bit of cleaning.

```r
netflix_movies_clean <- netflix_movies %>%
  mutate(
    # Replace missing or empty country values with an explicit label
    country = if_else(is.na(country) | country == "", "Unknown", country),

    # Trim whitespace and normalize spacing in "listed_in" (genres)
    listed_in = stringr::str_squish(listed_in),

    # Make sure date_added is in Date format where available
    date_added = lubridate::mdy(date_added)
  )

# Quick summary of missing values in key columns
netflix_movies_clean %>%
  summarise(
    n_movies       = n(),
    missing_country = sum(country == "Unknown"),
    missing_genre   = sum(is.na(listed_in) | listed_in == "")
  )
```

```
## # A tibble: 1 x 3
##   n_movies missing_country missing_genre
##      <int>           <int>         <int>
## 1     6131             440             0
```

Commentary

- `if_else()` is used to replace missing countries with `"Unknown"` rather than leaving them as `NA`.
- `stringr::str_squish()` removes extra internal spaces in the `listed_in` (genre) field.
- `lubridate::mdy()` converts the text dates (e.g., `"September 9, 2019"`) into proper `Date` objects for possible time-based filtering.

5. Use `tidyr::separate_rows()` for Countries and Genres

Many Netflix movies belong to **multiple countries** and **multiple genres**, stored as comma-separated lists in a single cell.
`tidyr::separate_rows()` helps us "un-nest" these into one row per country or genre.

```
netflix_movies_long <- netflix_movies_clean %>%
  # Split multiple countries into separate rows
  separate_rows(country, sep = ",[[:space:]]*") %>%
  # Split multiple genres into separate rows
  separate_rows(listed_in, sep = ",[[:space:]]*") %>%
  # Standardize the new columns by trimming whitespace again
  mutate(
    country  = stringr::str_squish(country),
    genre    = stringr::str_squish(listed_in)
  ) %>%
  # Keep a neat set of columns
  select(title, country, genre, release_year, rating, duration, date_added)

# Inspect the transformed data
head(netflix_movies_long)
```

```
## # A tibble: 6 x 7
##   title                    country genre release_year rating duration date_added
##   <chr>                    <chr>   <chr>        <dbl> <chr>  <chr>    <date>
## 1 Dick Johnson Is Dead     United~ Docu~         2020 PG-13  90 min   2021-09-25
## 2 My Little Pony: A New G~ Unknown Chil~         2021 PG     91 min   2021-09-24
## 3 Sankofa                  United~ Dram~         1993 TV-MA  125 min  2021-09-24
## 4 Sankofa                  United~ Inde~         1993 TV-MA  125 min  2021-09-24
## 5 Sankofa                  United~ Inte~         1993 TV-MA  125 min  2021-09-24
## 6 Sankofa                  Ghana   Dram~         1993 TV-MA  125 min  2021-09-24
```

Commentary

- After `separate_rows()`, a movie that originally had `"India, United States"` in `country` now appears on **two rows**, one for each country.
- Similarly, a movie with `"Comedies, Dramas"` in `listed_in` now appears on two rows with `genre = "Comedies"` and `genre = "Dramas"`.
- This structure is more suitable for counting and grouping by country or genre.

6. How Many Movies Does Netflix Release Each Year?

First, we'll count how many movies in the dataset were released each year.

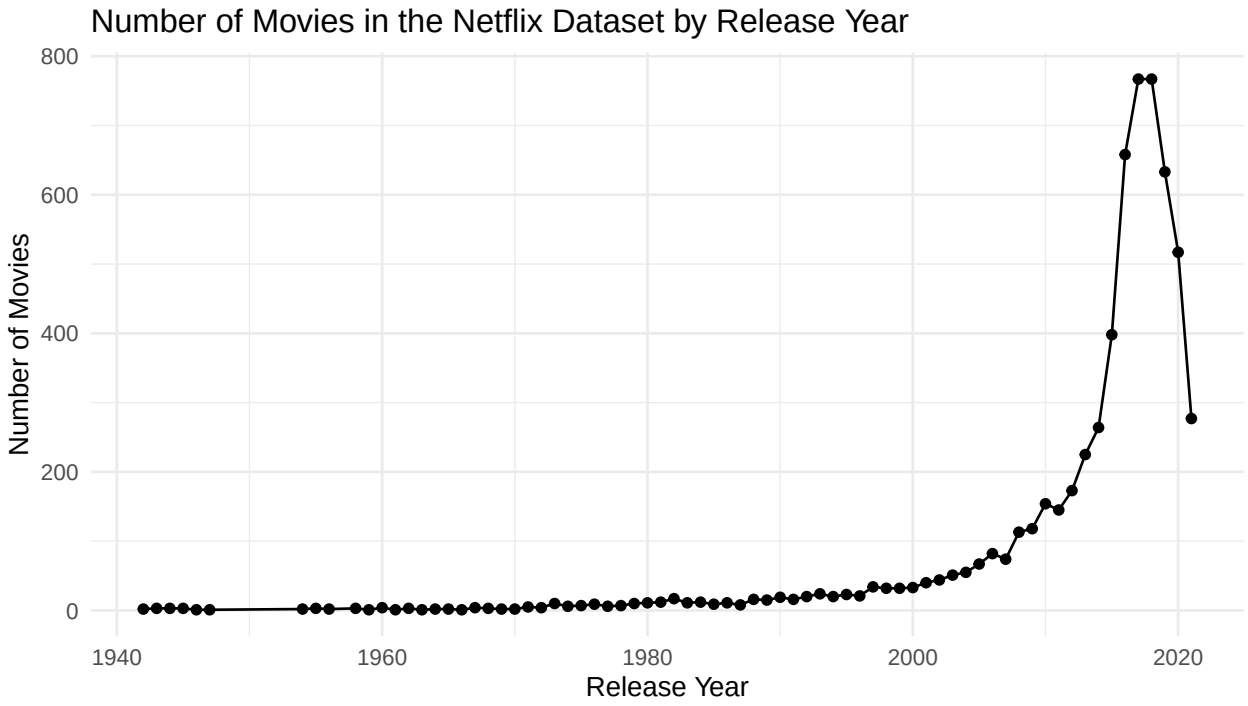
```
movies_per_year <- netflix_movies_long %>%
  distinct(title, release_year) %>%  # avoid double-counting movies that appear in multiple countries/g
  count(release_year, name = "n_movies") %>%
  arrange(release_year)

head(movies_per_year)
```

```
## # A tibble: 6 x 2
##   release_year n_movies
##          <dbl>    <int>
## 1         1942        2
## 2         1943        3
```

```
## 3         1944        3
## 4         1945        3
## 5         1946        1
## 6         1947        1
```

Now we'll visualize this trend using a line chart.

```
movies_per_year %>%
  ggplot(aes(x = release_year, y = n_movies)) +
  geom_line() +
  geom_point() +
  labs(
    title = "Number of Movies in the Netflix Dataset by Release Year",
    x = "Release Year",
    y = "Number of Movies"
  ) +
  theme_minimal()
```



Commentary

- `distinct(title, release_year)` ensures that each movie is counted once per year, even if it belongs to multiple countries or genres.
- The line chart shows how the number of movies in the dataset changes over time.
  Depending on the data, we might see increases in more recent years as Netflix expanded its catalog.

7. What Are the Most Common Genres?

We will now focus on **genres**, using `forcats::fct_lump_n()` to keep only the most common ones and lump everything else into `"Other genres"`.

```
genre_counts <- netflix_movies_long %>%
  count(genre, sort = TRUE)
```

```r
head(genre_counts, 10)
```

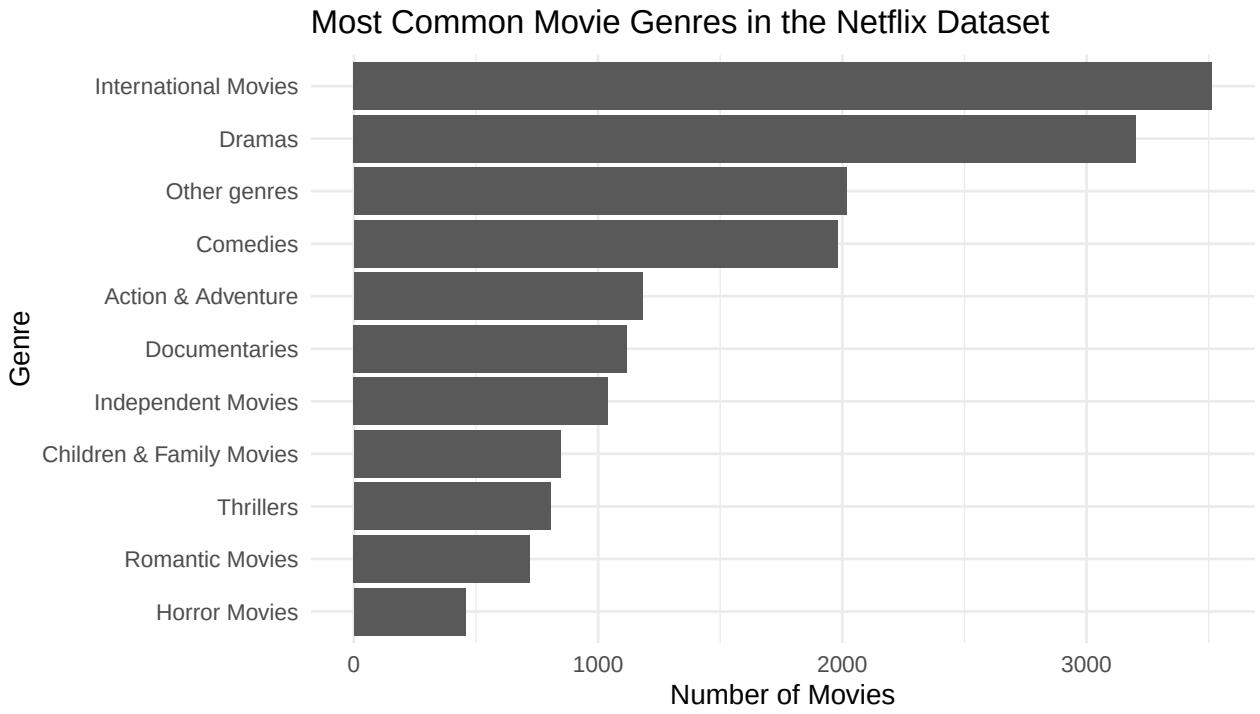
```
## # A tibble: 10 x 2
##    genre                        n
##    <chr>                    <int>
##  1 International Movies      3513
##  2 Dramas                    3202
##  3 Comedies                  1981
##  4 Action & Adventure        1182
##  5 Documentaries             1118
##  6 Independent Movies        1040
##  7 Children & Family Movies   845
##  8 Thrillers                  806
##  9 Romantic Movies            722
## 10 Horror Movies              458
```

```r
# Keep the top 10 genres and lump the rest
movies_genres_lumped <- netflix_movies_long %>%
  mutate(
    genre_lumped = fct_lump_n(genre, n = 10, other_level = "Other genres")
  )

# Count movies by the lumped genre
top_genre_summary <- movies_genres_lumped %>%
  count(genre_lumped, name = "n_movies") %>%
  mutate(
    # Reorder factor levels so bars appear in descending order
    genre_lumped = fct_reorder(genre_lumped, n_movies)
  )

top_genre_summary
```

```
## # A tibble: 11 x 2
##    genre_lumped             n_movies
##    <fct>                       <int>
##  1 Action & Adventure           1182
##  2 Children & Family Movies      845
##  3 Comedies                     1981
##  4 Documentaries                1118
##  5 Dramas                       3202
##  6 Horror Movies                 458
##  7 Independent Movies           1040
##  8 International Movies         3513
##  9 Romantic Movies               722
## 10 Thrillers                     806
## 11 Other genres                  2016
```

```r
top_genre_summary %>%
  ggplot(aes(x = genre_lumped, y = n_movies)) +
  geom_col() +
  coord_flip() +
  labs(
    title = "Most Common Movie Genres in the Netflix Dataset",
    x = "Genre",
    y = "Number of Movies"
```

```
  ) +
  theme_minimal()
```



Commentary

- `fct_lump_n(genre, n = 10)` is a convenient way to keep only the **top 10 genres**, combining everything else into `"Other genres"`.
- `fct_reorder()` reorders the bars so that the longest bar (most movies) appears at the top of the plot.
- `coord_flip()` rotates the chart, making genre labels easier to read.

8. Genre Trends Over Time (Faceted Plot)

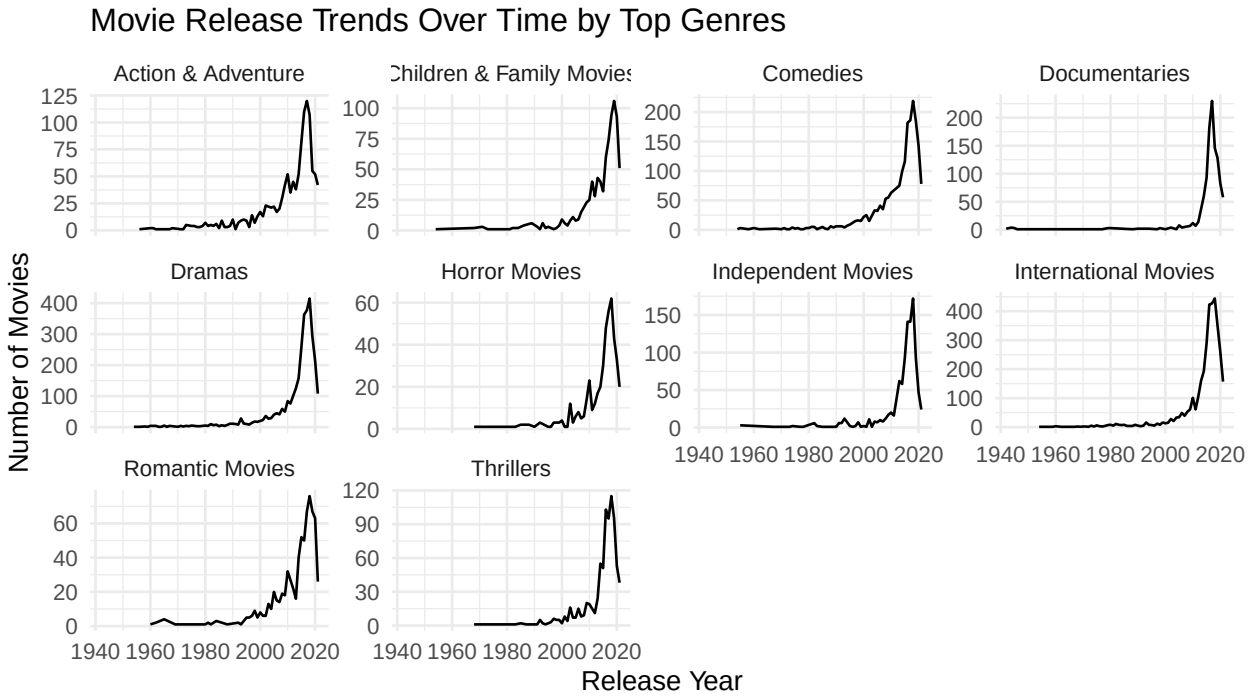
Next, we explore how the **number of movies released per year** varies for the most common genres.

```
genre_year_summary <- movies_genres_lumped %>%
  filter(genre_lumped != "Other genres") %>%   # focus only on the top genres
  group_by(genre_lumped, release_year) %>%
  summarise(
    n_movies = n(),
    .groups = "drop"
  )

head(genre_year_summary)
```

```
## # A tibble: 6 x 3
##   genre_lumped       release_year n_movies
##   <fct>                     <dbl>    <int>
## 1 Action & Adventure         1956        1
## 2 Action & Adventure         1960        2
## 3 Action & Adventure         1961        2
## 4 Action & Adventure         1962        1
## 5 Action & Adventure         1963        1
```

```
## 6 Action & Adventure          1967         1
```

```
genre_year_summary %>%
  ggplot(aes(x = release_year, y = n_movies)) +
  geom_line() +
  facet_wrap(~ genre_lumped, scales = "free_y") +
  labs(
    title = "Movie Release Trends Over Time by Top Genres",
    x = "Release Year",
    y = "Number of Movies"
  ) +
  theme_minimal()
```



Commentary

- We group by both `genre_lumped` and `release_year` to count how many movies per genre per year exist in the dataset.
- `facet_wrap()` creates a small multiple (panel) for each top genre, which makes it easier to compare trends:
  - Some genres may grow over time.
  - Others may appear more stable or sporadic.

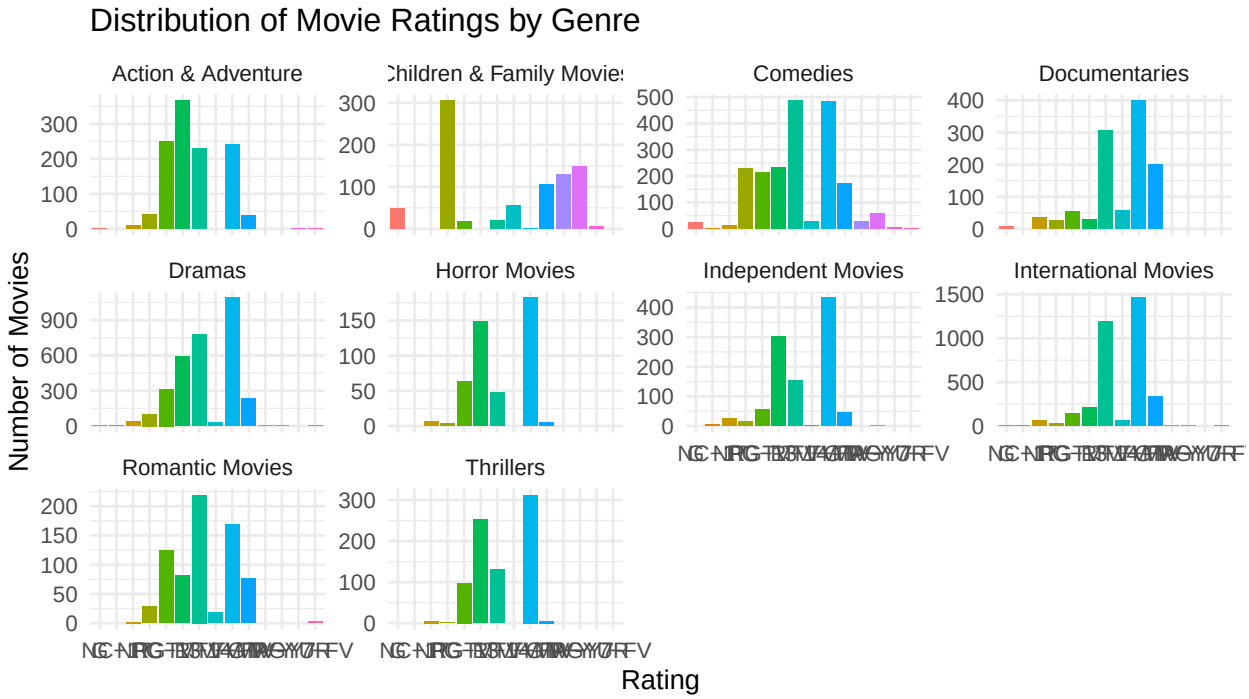
9. Movie Ratings Distribution by Genre

Finally, we can explore how **ratings** (e.g., PG, PG-13, R, TV-MA) are distributed across the most common genres.

```
ratings_by_genre <- movies_genres_lumped %>%
  filter(genre_lumped != "Other genres") %>%
  filter(!is.na(rating)) %>%
  count(genre_lumped, rating)

head(ratings_by_genre)
```

```
## # A tibble: 6 x 3
##   genre_lumped       rating     n
##   <fct>              <chr>  <int>
## 1 Action & Adventure G          1
## 2 Action & Adventure NR        11
## 3 Action & Adventure PG        41
## 4 Action & Adventure PG-13    251
## 5 Action & Adventure R        366
## 6 Action & Adventure TV-14    230
```

```r
ratings_by_genre %>%
  ggplot(aes(x = rating, y = n, fill = rating)) +
  geom_col(show.legend = FALSE) +
  facet_wrap(~ genre_lumped, scales = "free_y") +
  labs(
    title = "Distribution of Movie Ratings by Genre",
    x = "Rating",
    y = "Number of Movies"
  ) +
  theme_minimal()
```



Commentary

- This plot lets us see which genres are more likely to have **mature ratings** (e.g., R, TV-MA) versus more **family-friendly** ratings (e.g., G, PG).
- Filtering out `NA` ratings ensures the counts correspond to known categories.

Summary

This extended vignette demonstrates how to:

- Use **`dplyr`** for data manipulation:
  - `filter()` to keep only movies.

  - `select()` to focus on key columns.
  - `mutate()` to clean and transform variables.
  - `group_by()` and `summarise()` / `count()` to compute summary statistics.
- Use `tidyr` to reshape data with `separate_rows()` so that multi-valued fields like country and genre become one value per row.
- Use `ggplot2` to create:
  - Line charts of movies per year.
  - Bar charts of top genres.
  - Faceted plots of genre-specific trends and rating distributions by genre.
- Use `forcats` to:
  - Lump rare categories with `fct_lump_n()`.
  - Reorder factor levels by frequency using `fct_reorder()` for clearer plots.

Together, these examples show a complete TidyVerse workflow for exploring the Netflix movies dataset.

References

- Kaggle Netflix Dataset: https://www.kaggle.com/datasets/shivamb/netflix-shows
- TidyVerse Documentation: https://www.tidyverse.org/packages/