RWorksheet_#5

Langreo, Liza, Pabriaga, Sison

2024-12-04

```
# Load necessary libraries
library(httr)
library(polite)
library(rvest)
library(dplyr)
```

```
##
## Attaching package: 'dplyr'

## The following objects are masked from 'package:stats':
##
##     filter, lag

## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
library(kableExtra)
```

```
##
## Attaching package: 'kableExtra'

## The following object is masked from 'package:dplyr':
##
##     group_rows
```

```
library(ggplot2)
library(stringr)
```

```
polite::use_manners(save_as = 'polite_scrape.R')
```

```
## v Setting active project to "/cloud/project".

## v Writing 'polite_scrape.R'.

## [ ] Edit 'polite_scrape.R'.
```

```
# IMDb URL for top TV shows
imdb_top_tv_url <- 'https://www.imdb.com/chart/toptv/?ref_=nv_tvv_250'

# Start a polite scraping session with a user agent
scrape_session <- bow(imdb_top_tv_url, user_agent = "Educational")
scrape_session
```

```
## <polite session> https://www.imdb.com/chart/toptv/?ref_=nv_tvv_250
##     User-agent: Educational
##     robots.txt: 35 rules are defined for 3 bots
##    Crawl delay: 5 sec
##   The path is scrapable for this user-agent
```

```r
# Fetch the webpage content
imdb_page_content <- read_html(imdb_top_tv_url)

# Extract TV show titles
show_titles <- imdb_page_content %>%
  html_nodes('h3.ipc-title__text') %>%
  html_text()

# Remove unnecessary header
show_titles <- show_titles[show_titles != "IMDb Charts"]

# Extract ratings
show_ratings <- imdb_page_content %>%
  html_nodes("span.ipc-rating-star--rating") %>%
  html_text()

# Extract vote counts
vote_counts <- imdb_page_content %>%
  html_nodes("span.ipc-rating-star--voteCount") %>%
  html_text()

# Extract episode data
episode_info <- imdb_page_content %>%
  html_nodes('span.sc-300a8231-7.eaXxft.cli-title-metadata-item:nth-of-type(2)') %>%
  html_text()

# Clean and extract episode numbers
episode_counts <- str_extract(episode_info, "\\d+ eps")
episode_counts <- str_remove(episode_counts, " eps")

# Extract release years
year_info <- imdb_page_content %>%
  html_nodes('span.sc-300a8231-7.eaXxft.cli-title-metadata-item') %>%
  html_text()

release_years <- str_extract(year_info, "\\d{4}")
release_years <- release_years[!is.na(release_years)]
release_years <- as.numeric(release_years)
```

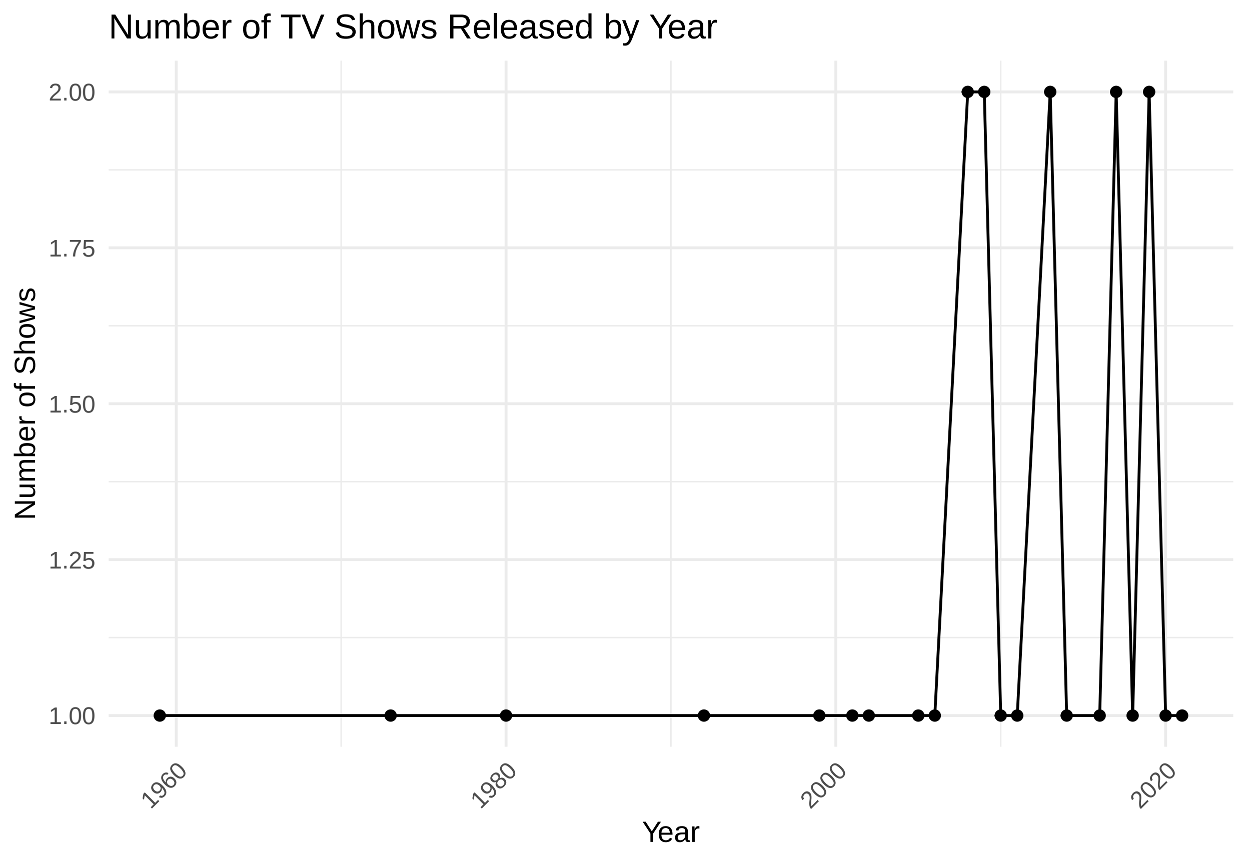
```r
# Function to fetch critic reviews
fetch_critic_reviews <- function(show_url) {
  full_url <- paste0("https://imdb.com", show_url)
  show_page <- read_html(full_url)

  # Retrieve critic reviews
  critic_scores <- show_page %>%
    html_nodes("span.score") %>%
    html_text()

  if (length(critic_scores) > 1) {
    return(critic_scores[2])
  } else {
    return(NA)
  }
}

# Function to fetch popularity ratings
fetch_popularity_scores <- function(show_url) {
  full_url <- paste0("https://imdb.com", show_url)
  show_page <- read_html(full_url)

  popularity_score <- show_page %>%
    html_nodes('[data-testid="hero-rating-bar__popularity__score"]') %>%
    html_text()

  if (length(popularity_score) > 1) {
    return(popularity_score[2])
  } else {
    return(NA)
  }
}
```

```r
# Extract show links
show_urls <- imdb_page_content %>%
  html_nodes("a.ipc-title-link-wrapper") %>%
  html_attr("href")

# Fetch critic reviews for all shows
critic_scores <- sapply(show_urls, fetch_critic_reviews)

# Fetch popularity ratings for all shows
popularity_scores <- sapply(show_urls, fetch_popularity_scores)
```

```r
# Ensure data consistency
max_rows <- max(length(show_titles), length(show_ratings), length(vote_counts), length(episode_counts),

show_titles <- rep(show_titles, length.out = max_rows)
show_ratings <- rep(show_ratings, length.out = max_rows)
vote_counts <- rep(vote_counts, length.out = max_rows)
episode_counts <- rep(episode_counts, length.out = max_rows)
release_years <- rep(release_years, length.out = max_rows)
critic_scores <- rep(critic_scores, length.out = max_rows)
popularity_scores <- rep(popularity_scores, length.out = max_rows)

# Combine into a data frame
imdb_tv_shows <- data.frame(
  Title = show_titles,
  Rating = show_ratings,
  Votes = vote_counts,
  Episodes = episode_counts,
  Year = release_years,
  CriticScore = critic_scores,
  Popularity = popularity_scores,
  stringsAsFactors = FALSE
)

# Save to a CSV file
```

```r
write.csv(imdb_tv_shows, "Top_IMDb_TV_Shows.csv")
```

```r
# Analyze yearly releases
shows_by_year <- imdb_tv_shows %>%
  group_by(Year) %>%
  summarize(ShowCount = n()) %>%
  arrange(Year)

# Plot the number of shows by year
ggplot(shows_by_year, aes(x = Year, y = ShowCount)) +
  geom_line() +
  geom_point() +
  labs(
    title = "Number of TV Shows Released by Year",
    x = "Year",
    y = "Number of Shows"
  ) +
  theme_minimal() +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))
```



```r
# Identify the year with the most shows
peak_year <- shows_by_year %>%
  filter(ShowCount == max(ShowCount))

print(peak_year)
```

```
## # A tibble: 5 x 2
##    Year ShowCount
##    <dbl>      <int>
## 1  2008           2
## 2  2009           2
## 3  2013           2
## 4  2017           2
## 5  2019           2
```