RWorksheet#5_group(Lumauag, Animas, Sanceda)

Matt Andrei Lumauag

2024-11-11

```
# Load necessary libraries
library(httr)      # HTTP requests
```

```
## Warning: package 'httr' was built under R version 4.4.2
```

```
library(polite)    # Polite scraping
```

```
## Warning: package 'polite' was built under R version 4.4.2
```

```
library(rvest)     # Web scraping
```

```
## Warning: package 'rvest' was built under R version 4.4.2
```

```
library(dplyr)     # Data manipulation
```

```
##
## 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) # HTML tables
```

```
## Warning: package 'kableExtra' was built under R version 4.4.2

##
## Attaching package: 'kableExtra'

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

```r
library(ggplot2)   # Data visualization
library(stringr)   # String manipulation
```

```r
# Enable polite scraping and save settings
polite::use_manners(save_as = 'polite_scrape.R')
```

```
## v Setting active project to "D:/RAnalytics".
```

```r
# URL of the IMDB top TV shows page
imdb_url <- 'https://www.imdb.com/chart/toptv/?ref_=nv_tvv_250'

# Start a polite scraping session with a user agent
session <- bow(imdb_url, user_agent = "Educational")
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
# Define the IMDb URL for the top TV shows
imdb_url <- "https://www.imdb.com/chart/toptv/?ref_=nv_tvv_250c"

# Fetch the content of the webpage
webpage_content <- read_html(imdb_url)

# Extract the titles of TV shows
tv_show_titles <- webpage_content %>%
  html_nodes('h3.ipc-title__text') %>%
  html_text()

# Clean the extracted titles by removing the header "IMDb Charts"
tv_show_titles <- tv_show_titles[tv_show_titles != "IMDb Charts"]

# Get the ratings for each show
tv_show_ratings <- webpage_content %>%
  html_nodes("span.ipc-rating-star--rating") %>%
  html_text()

# Extract the vote counts for each show
votes_count <- webpage_content %>%
  html_nodes("span.ipc-rating-star--voteCount") %>%
  html_text()

# Get the number of episodes for each show
episode_data <- webpage_content %>%
  html_nodes('span.sc-300a8231-7.eaXxft.cli-title-metadata-item:nth-of-type(2)') %>%
  html_text()

# Clean up episode data (extract only the episode numbers)
episode_counts <- str_extract(episode_data, "\\d+ eps")
episode_counts <- str_remove(episode_counts, " eps")

# Retrieve the release years for each show
year_data <- webpage_content %>%
  html_nodes('span.sc-300a8231-7.eaXxft.cli-title-metadata-item') %>%
  html_text()

# Extract the release year using regex
release_years <- str_extract(year_data, "\\d{4}")
release_years <- release_years[!is.na(release_years)]  # Remove any NA values
release_years <- as.numeric(release_years)
```

```r
# Function to scrape critic reviews for each show
get_critic_reviews <- function(show_link) {
  full_url <- paste0("https://imdb.com", show_link)
  show_page <- read_html(full_url)

  # Retrieve critic reviews
  critic_scores <- show_page %>%
    html_nodes("span.score") %>%  # Update the selector if necessary
    html_text()

  # Return the second critic score, if available
  if (length(critic_scores) > 1) {
    return(critic_scores[2])  # Return the second score
  } else {
    return(NA)  # Return NA if no review found
  }
}

# Function to fetch popularity ratings for each show
get_popularity_rating <- function(show_link) {
  full_url <- paste0("https://imdb.com", show_link)
  show_page <- read_html(full_url)

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

  # Return the popularity rating if found
  if (length(popularity_score) > 1) {
    return(popularity_score[2])  # The second item should contain the popularity score
  } else {
    return(NA)  # Return NA if no rating is found
  }
}
```

```r
# Extract the links to each TV show's IMDb page
show_links <- webpage_content %>%
  html_nodes("a.ipc-title-link-wrapper") %>%
  html_attr("href")
```

```r
# Loop through each show link to fetch critic reviews
critic_reviews <- sapply(show_links, get_critic_reviews)

# Loop through each show link to fetch popularity ratings
popularity_scores <- sapply(show_links, get_popularity_rating)
```

```r
# Ensure consistency in the length of all data vectors
max_length <- max(length(tv_show_titles), length(tv_show_ratings), length(votes_count), length(episode_

# Repeat data elements to match the maximum length
tv_show_titles <- rep(tv_show_titles, length.out = max_length)
tv_show_ratings <- rep(tv_show_ratings, length.out = max_length)
votes_count <- rep(votes_count, length.out = max_length)
episode_counts <- rep(episode_counts, length.out = max_length)
release_years <- rep(release_years, length.out = max_length)
critic_reviews <- rep(critic_reviews, length.out = max_length)
popularity_scores <- rep(popularity_scores, length.out = max_length)

# Combine all the collected data into a data frame
tv_shows_data <- data.frame(
  Title = tv_show_titles,
  Rating = tv_show_ratings,
  Votes = votes_count,
  EpisodeCount = episode_counts,
  ReleasedYear = release_years,
  CriticReviews = critic_reviews,
  PopularityRating = popularity_scores,
  stringsAsFactors = FALSE
)

# Retrieve the top 50 TV shows from the list
top_50_tv_shows <- tv_shows_data %>%
  slice(1:50)  # Select the first 50 shows based on rank

# Print the top 50 TV shows
print(top_50_tv_shows)
```

```
##                                  Title Rating   Votes EpisodeCount ReleasedYear
## 1                      1. Breaking Bad    9.5  (2.2M)           62         2008
## 2                   2. Planet Earth II    9.5  (162K)            6         2016
## 3                      3. Planet Earth    9.4  (224K)           11         2006
## 4                  4. Band of Brothers    9.4  (546K)           10         2001
## 5                         5. Chernobyl    9.3  (909K)            5         2019
## 6                          6. The Wire    9.3  (391K)           60         2002
## 7        7. Avatar: The Last Airbender    9.3  (391K)           62         2005
## 8                    8. Blue Planet II    9.3   (49K)            7         2017
## 9                      9. The Sopranos    9.2  (500K)           86         1999
## 10     10. Cosmos: A Spacetime Odyssey    9.2  (131K)           13         2014
## 11                          11. Cosmos    9.3   (46K)           13         1980
## 12                      12. Our Planet    9.2   (54K)           12         2019
## 13                 13. Game of Thrones    9.2  (2.4M)           74         2011
## 14                           14. Bluey    9.3   (34K)          194         2018
## 15                15. The World at War    9.2   (31K)           26         1973
## 16 16. Fullmetal Alchemist Brotherhood    9.1  (209K)           68          2009
## 17                   17. Rick and Morty    9.1  (628K)           78          2013
## 18                             18. Life    9.1   (44K)           11          2009
## 19                   19. The Last Dance    9.0  (160K)           10          2020
## 20                20. The Twilight Zone    9.0   (97K)          156          1959
## 21                 21. The Vietnam War    9.1   (29K)           10          2017
## 22                        22. Sherlock    9.1    (1M)           15          2010
## 23                 23. Attack on Titan    9.1  (565K)           98          2013
## 24     24. Batman: The Animated Series    9.0  (123K)           85          1992
## 25                          25. Arcane    9.0  (330K)           18          2021
## 26                      Recently viewed    9.5  (2.2M)           62          2008
##    CriticReviews PopularityRating
## 1            175               20
## 2              6              999
## 3             10            1,778
## 4             34              153
## 5             88              146
## 6             77              104
## 7             57              357
## 8              9            4,122
## 9             93               31
## 10            12            1,571
## 11             8            3,645
## 12            15            2,401
## 13           368               16
## 14             4              373
## 15             5            2,532
## 16            16              474
## 17            94              125
## 18             9            3,057
## 19            28            1,403
## 20            85              337
## 21            13            1,739
## 22           121              160
## 23            64               50
## 24            25              463
## 25            59                1
## 26           175               20
```

```r
# Save the top 50 shows data to a CSV file
write.csv(top_50_tv_shows, "Top_50_tv_shows.csv")

#TV hows ranked from 26 to 50 cannot be scraped due to some reasons
```

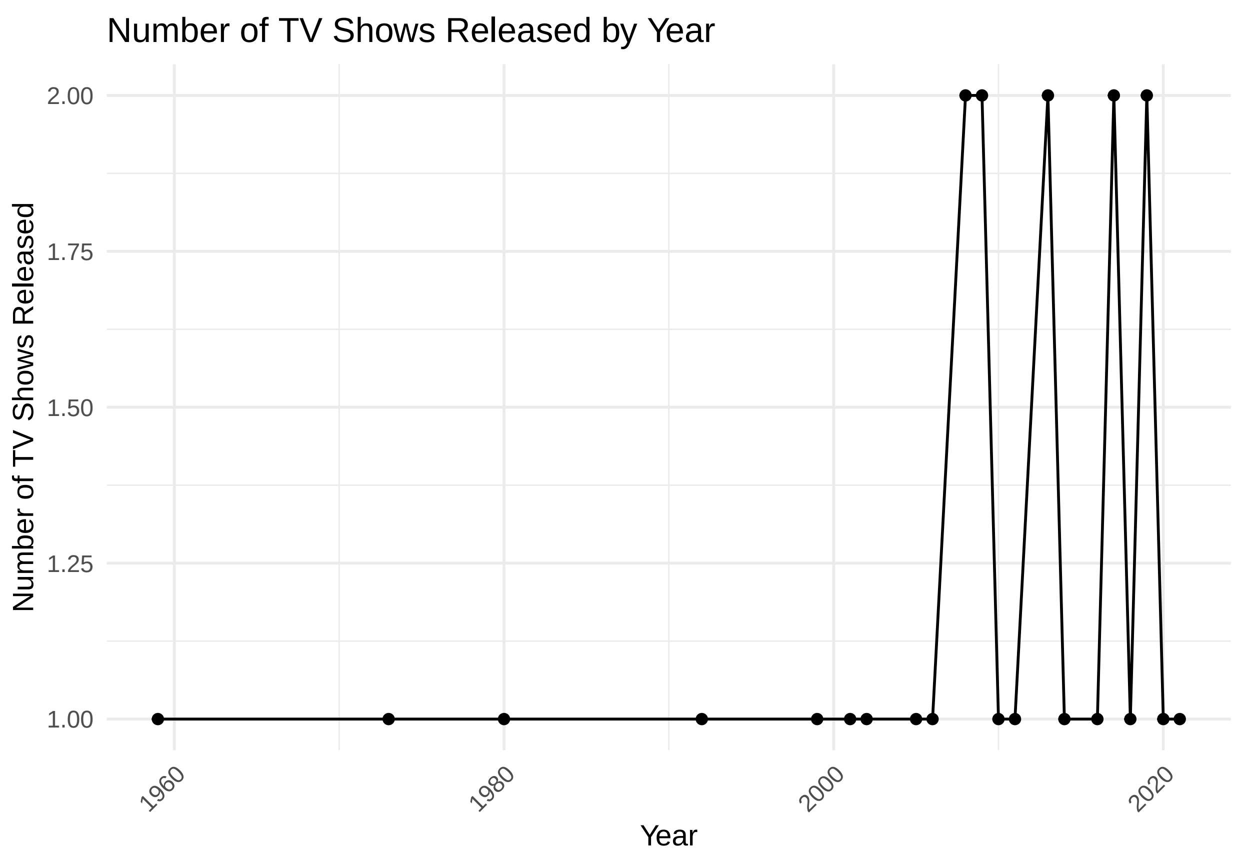
```r
scrape_imdb_reviews <- function(url) {
  # Load the page content
  page <- tryCatch(read_html(url), error = function(e) NULL)
  if (is.null(page)) {
    message("Failed to load page: ", url)
    return(tibble())
  }

  # Extract relevant review data
  reviewers <- page %>%
    html_nodes("a.ipc-link.ipc-link--base") %>%
    html_text() %>%
    .[. != "Permalink"]

  dates <- page %>%
    html_nodes("li.ipc-inline-list__item.review-date") %>%
    html_text()

  ratings <- page %>%
    html_nodes("span.ipc-rating-star--rating") %>%
    html_text() %>%
    as.numeric()

  titles <- page %>%
    html_nodes("h3.ipc-title__text") %>%
    html_text()

  review_texts <- page %>%
    html_nodes("div.ipc-html-content-inner-div") %>%
    html_text()

  # Adjust lengths by padding shorter vectors with NA
  max_length <- max(length(reviewers), length(dates), length(ratings), length(titles), length(review_te

  # Pad vectors with NA if they are shorter than max_length
  reviewers <- c(reviewers, rep(NA, max_length - length(reviewers)))
  dates <- c(dates, rep(NA, max_length - length(dates)))
  ratings <- c(ratings, rep(NA, max_length - length(ratings)))
  titles <- c(titles, rep(NA, max_length - length(titles)))
  review_texts <- c(review_texts, rep(NA, max_length - length(review_texts)))

  # Combine data into a tibble without the helpful votes
  tibble(
    reviewer_name = reviewers,
    review_date = dates,
    rating = ratings,
    review_title = titles,
    review_text = review_texts
  )
}

# List of IMDb links
links <- c(
  "https://www.imdb.com/title/tt7366338/reviews/?ref_=tt_urv_sm",
  "https://www.imdb.com/title/tt0903747/reviews/?ref_=tt_urv_sm",
  "https://www.imdb.com/title/tt5491994/reviews/?ref_=tt_urv_sm",
  "https://www.imdb.com/title/tt0795176/reviews/?ref_=tt_urv_sm",
  "https://www.imdb.com/title/tt0185906/reviews/?ref_=tt_urv_sm"
)

# Initialize an empty tibble to store all reviews
all_reviews <- tibble()
```

```r
# Loop through each link and scrape reviews
for (link in links) {
  reviews <- scrape_imdb_reviews(link)

  # Check if reviews are scraped successfully and limit to 20 reviews per link
  if (nrow(reviews) > 0) {
    reviews <- reviews %>% slice(1:20)  # Limit to the first 20 reviews per link
    all_reviews <- bind_rows(all_reviews, reviews)
  }
}

# View the first 20 reviews after scraping all links
print(all_reviews)
```

```
## # A tibble: 100 x 5
##    reviewer_name     review_date  rating review_title                 review_text
##    <chr>             <chr>         <dbl> <chr>                        <chr>
##  1 curiosityonmars   May 23, 2019     10 They got it right            "I was bor~
##  2 stelmakh          May 10, 2019     10 Goosebumps and tears         "A Belarus~
##  3 natashapekar      May 9, 2019      10 I highly recommend this fi~ "Hi. I'm f~
##  4 m-porpaczi        May 14, 2019     10 No hero wakes up wanting t~ "As my mot~
##  5 Lladerat          May 7, 2019      10 So far looks excellent       "Im ukrain~
##  6 jfirebug          May 20, 2019     10 Incredible                   "My husban~
##  7 thegldt           May 6, 2019      10 Bleak, Unsettling, Hauntin~ "'Chernoby~
##  8 alexander-phoenix May 13, 2019     10 Unbelievable                 "I'm Russi~
##  9 wmeduardowm       May 6, 2019      10 HBO did it again!            "The first~
## 10 Leofwine_draca    Nov 27, 2019     10 Exemplary                    "Quite pos~
## # i 90 more rows
```

```r
# Save to CSV file
write.csv(all_reviews, "IMDBReviews.csv", row.names = FALSE)
```

```r
#Count the number of TV shows released per year
tv_shows_year_count <- tv_shows_data %>%
  group_by(ReleasedYear) %>%
  summarize(num_shows = n()) %>%
  arrange(ReleasedYear)

#Create a time series plot
ggplot(tv_shows_year_count, aes(x = ReleasedYear, y = num_shows)) +
  geom_line() +
  geom_point() +
  labs(
    title = "Number of TV Shows Released by Year",
    x = "Year",
    y = "Number of TV Shows Released"
  ) +
  theme_minimal() +
  theme(
    axis.text.x = element_text(angle = 45, hjust = 1)
  )
```



```r
#Identify the year with the most TV shows released
most_shows_year <- tv_shows_year_count %>%
  filter(num_shows == max(num_shows))

# Print the year with the most releases
print(most_shows_year)
```

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