RWorksheet#5_group(asenjo, elizalde, barrientos)

Asenjo, Elizalde, Barrientos

2024-11-11

```
library(httr)

library(polite)

library(rvest)

library(dplyr)

library(kableExtra)

library(ggplot2)
library(stringr)
library(tinytex)
library(latexpdf)
```

1.

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

url <- 'https://www.imdb.com/chart/toptv/?ref_=nv_tvv_250'


session <- bow(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
```

```
# Load necessary libraries
library(rvest)
library(dplyr)
library(stringr)
library(rmarkdown)

# Define the URL for IMDb Top TV Shows
```

```r
url <- "https://www.imdb.com/chart/toptv/?ref_=nv_tvv_250c"

# Read the webpage content
webpage <- read_html(url)

# Extract Show Titles
show_titles <- webpage %>%
  html_nodes('h3.ipc-title__text') %>%
  html_text()

show_titles <- show_titles[show_titles != "IMDb Charts"]

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

# Extract the number of people who voted
num_votes <- webpage %>%
  html_nodes("span.ipc-rating-star--voteCount") %>%
  html_text()

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

# Clean the episode data (extract only the number of episodes)
episode_counts <- str_extract(episode_info, "\\d+ eps")
episode_counts <- str_remove(episode_counts, " eps")

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

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

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

  # Extract critic reviews
  critic <- show_page %>%
    html_nodes("span.score") %>%  # Adjust this if necessary based on page structure
    html_text()

  # Return the second critic review (if available)
  if (length(critic) > 1) {
```

```
    return(critic[2])  # Take the second item for the critic score
  } else {
    return(NA)  # If no critic review is found
  }
}

# Function to scrape popularity rating for each show
get_popularity_rating <- function(link) {
  complete_link <- paste0("https://imdb.com", link)
  show_page <- read_html(complete_link)

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

  # Return the popularity rating (if available)
  if (length(pop_rating) > 1) {
    return(pop_rating[2])  # The second item should be the popularity rating
  } else {
    return(NA)  # If no popularity rating is found
  }
}

# Extract links to each show's page
links <- webpage %>%
  html_nodes("a.ipc-title-link-wrapper") %>%
  html_attr("href")

# Loop through each show's link and get the critic reviews
critic_reviews <- sapply(links, get_critic_reviews)

# Loop through each show's link and get the popularity rating
popularity_ratings <- sapply(links, get_popularity_rating)

# Ensure all data has the same length before combining
max_length <- max(length(show_titles), length(show_ratings), length(num_votes), length(episode_counts),
show_titles <- rep(show_titles, length.out = max_length)
show_ratings <- rep(show_ratings, length.out = max_length)
num_votes <- rep(num_votes, 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_ratings <- rep(popularity_ratings, length.out = max_length)

# Combine into a data frame
imdb_top_tv_shows <- data.frame(
  Show_Title = show_titles,
  Rating = show_ratings,
  Votes = num_votes,
  Episode_Count = episode_counts,
  Release_Year = release_years,
  Critic_Reviews = critic_reviews,
```

```
  Popularity_Rating = popularity_ratings,
  stringsAsFactors = FALSE
)

top_50_shows <- imdb_top_tv_shows %>%
  slice(1:50)  # Get the top 50 shows

# Print the top 50 shows
print(top_50_shows)

#write.csv(top_50_shows, "Top_50_shows.csv")
```

2.

```
# Load necessary libraries
library(rvest)
library(dplyr)
library(stringr)

# Define a function to scrape IMDb reviews
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 the title of the show using the correct selector
  show_title <- page %>%
    html_nodes("h1[data-testid='hero-title-block__title']") %>%
    html_text(trim = TRUE)

  # Check if the show title was extracted successfully
  if (length(show_title) == 0) {
    message("Failed to extract show title for URL: ", url)
    show_title <- NA  # Set to NA if the title is not found
  }

  # 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()

  helpful_votes <- page %>%
    html_nodes("span.ipc-voting__label__count.ipc-voting__label__count--up") %>%
    html_text() %>%
    as.numeric()

  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(helpful_v

  # 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)))
  helpful_votes <- c(helpful_votes, rep(NA, max_length - length(helpful_votes)))
  review_texts <- c(review_texts, rep(NA, max_length - length(review_texts)))

  # Combine data into a tibble
  tibble(
    show_title = rep(show_title, max_length),  # Add the show title to each review
    reviewer_name = reviewers,
    review_date = dates,
    rating = ratings,
    review_title = titles,
    helpful_votes = helpful_votes,
    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()

# 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
```

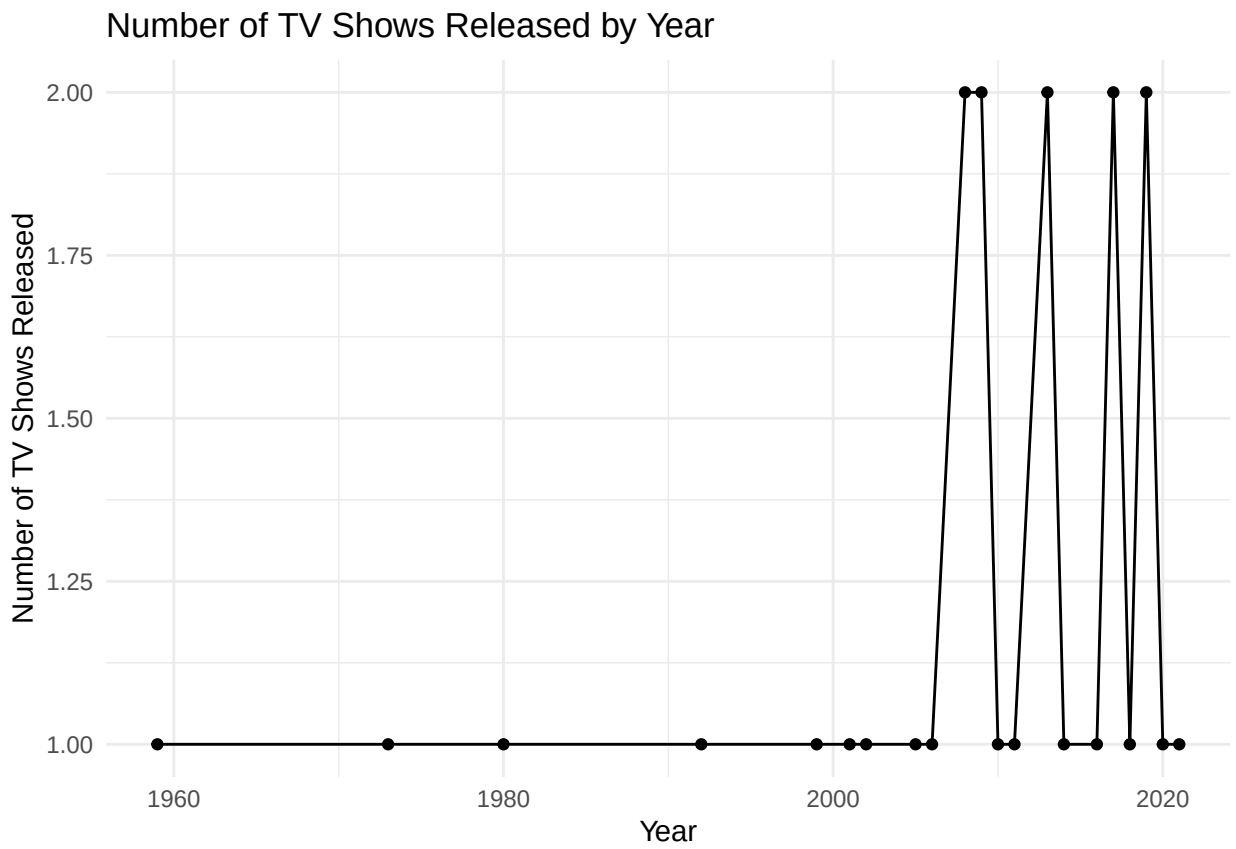
```
  if (nrow(reviews) > 0) {
    reviews <- reviews %>% slice(1:20)  # Limit to the first 20 reviews per link
    all_reviews <- bind_rows(all_reviews, reviews)
  }


}
```

```
## Failed to extract show title for URL: https://www.imdb.com/title/tt7366338/reviews/?ref_=tt_urv_sm

## Failed to extract show title for URL: https://www.imdb.com/title/tt0903747/reviews/?ref_=tt_urv_sm

## Failed to extract show title for URL: https://www.imdb.com/title/tt5491994/reviews/?ref_=tt_urv_sm

## Failed to extract show title for URL: https://www.imdb.com/title/tt0795176/reviews/?ref_=tt_urv_sm

## Failed to extract show title for URL: https://www.imdb.com/title/tt0185906/reviews/?ref_=tt_urv_sm
```

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


#write.csv(all_reviews, "IMDb_reviews.csv")
```

3.

```
# Load necessary libraries
library(ggplot2)
library(dplyr)

# Count the number of TV shows released per year
tv_shows_by_year <- imdb_top_tv_shows %>%
  group_by(Release_Year) %>%
  summarise(Number_of_Shows = n())

# Plot the time series graph
ggplot(tv_shows_by_year, aes(x = Release_Year, y = Number_of_Shows)) +
  geom_line() +  # Add a line plot
  geom_point() +  # Add points at each data point
  labs(title = "Number of TV Shows Released by Year",
       x = "Year",
       y = "Number of TV Shows Released") +
  theme_minimal()
```



4 and 5

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

url <- 'https://www.amazon.com'


session <- bow(url,
               user_agent = "Educational")
session
```

```
## <polite session> https://www.amazon.com
##     User-agent: Educational
##     robots.txt: 138 rules are defined for 5 bots
##    Crawl delay: 5 sec
##   The path is scrapable for this user-agent
```

```
library(rvest)
library(dplyr)
library(stringr)

scrape_amazon_products <- function(base_url, category, num_products = 30) {
  all_data <- data.frame()
  page_number <- 1
```

```r
while (nrow(all_data) < num_products) {
  # Construct the URL for the current page
  url <- paste0(base_url, "&page=", page_number)
  message("Scraping: ", url)

  page <- read_html(url)

  product_titles <- page %>%
    html_nodes("span.a-text-normal") %>%
    html_text(trim = TRUE)

  product_titles <- product_titles[product_titles != "Check each product page for other buying options
  product_titles <- product_titles[product_titles != " Mouse - Season 1"]
  product_titles <- product_titles[product_titles != "2021"]
  product_titles <- product_titles[product_titles != "Sep 25, 2015"]
  product_titles <- product_titles[product_titles != "Jan 1, 2023"]
  product_titles <- product_titles[product_titles != "Mar 1, 2022"]
    product_titles <- product_titles[product_titles != "May 19, 2024"]

  price <- page %>%
    html_nodes('.a-price .a-offscreen') %>%
    html_text(trim = TRUE)

  ratings <- page %>%
    html_nodes('span.a-icon-alt') %>%
    html_text(trim = TRUE) %>%
    str_extract("\\d\\.\\d") %>%
    as.numeric()


  reviews <- page %>%
    html_nodes('.s-link-style .s-underline-text') %>%
    html_text(trim = TRUE)

  descriptions <- page %>%
    html_nodes("span.a-text-normal") %>%
    html_text(trim = TRUE)

  descriptions <- descriptions[descriptions != "Check each product page for other buying options."]
  descriptions <- descriptions[descriptions != "Mouse - Season 1"]
  descriptions <- descriptions[descriptions != "2021"]
  descriptions <- descriptions[descriptions != "Sep 22, 2024"]
  descriptions <- descriptions[descriptions != "Jan 1, 2023"]
  descriptions <- descriptions[descriptions != "May 19, 2024"]
    descriptions <- descriptions[descriptions != "Sep 25, 2015"]


  min_length <- min(length(product_titles), length(price), length(ratings), length(descriptions), leng
  if (min_length == 0) break  # Exit if no products are found on the page

  data <- data.frame(
    ProductTitle = head(product_titles, min_length),
    Price = head(price, min_length),
```

```
      Category = rep(category, min_length),
      Ratings = head(ratings, min_length),
      Reviews = head(reviews, min_length),
      Description = head(descriptions, min_length)
    )


    all_data <- bind_rows(all_data, data)

    page_number <- page_number + 1
  }

  all_data <- head(all_data, num_products)



  return(all_data)
}

motherboard_url <- "https://www.amazon.com/s?k=motherboard&crid=BV9CU0DC1KEH&sprefix=motherboa%2Caps%2C5
monitor_url <- "https://www.amazon.com/s?k=monitor&crid=141H7SDT9TKPQ&sprefix=monitor%2Caps%2C493&ref=nb
mouse_url <- "https://www.amazon.com/s?k=mouse&crid=2MDYM9VSD6KCG&sprefix=mou%2Caps%2C693&ref=nb_sb_noss
keyboard_url <- "https://www.amazon.com/s?k=keyboard&crid=1LFSHKDP0X2J6&sprefix=keyboa%2Caps%2C384&ref=
headset_url <- "https://www.amazon.com/s?k=headset&crid=SHD7Y7OSWLUU&sprefix=headse%2Caps%2C449&ref=nb_

motherboard_products <- scrape_amazon_products(motherboard_url, "Motherboard", 30)
```

```
## Scraping: https://www.amazon.com/s?k=motherboard&crid=BV9CU0DC1KEH&sprefix=motherboa%2Caps%2C509&ref=
```

```
monitor_products <- scrape_amazon_products(monitor_url, "Monitor", 30)
```

```
## Scraping: https://www.amazon.com/s?k=monitor&crid=141H7SDT9TKPQ&sprefix=monitor%2Caps%2C493&ref=nb_sb
```

```
mouse_products <- scrape_amazon_products(mouse_url, "Mouse", 30)
```

```
## Scraping: https://www.amazon.com/s?k=mouse&crid=2MDYM9VSD6KCG&sprefix=mou%2Caps%2C693&ref=nb_sb_noss_
```

```
keyboard_products <- scrape_amazon_products(keyboard_url, "Keyboard", 30)
```

```
## Scraping: https://www.amazon.com/s?k=keyboard&crid=1LFSHKDP0X2J6&sprefix=keyboa%2Caps%2C384&ref=nb_sb
```

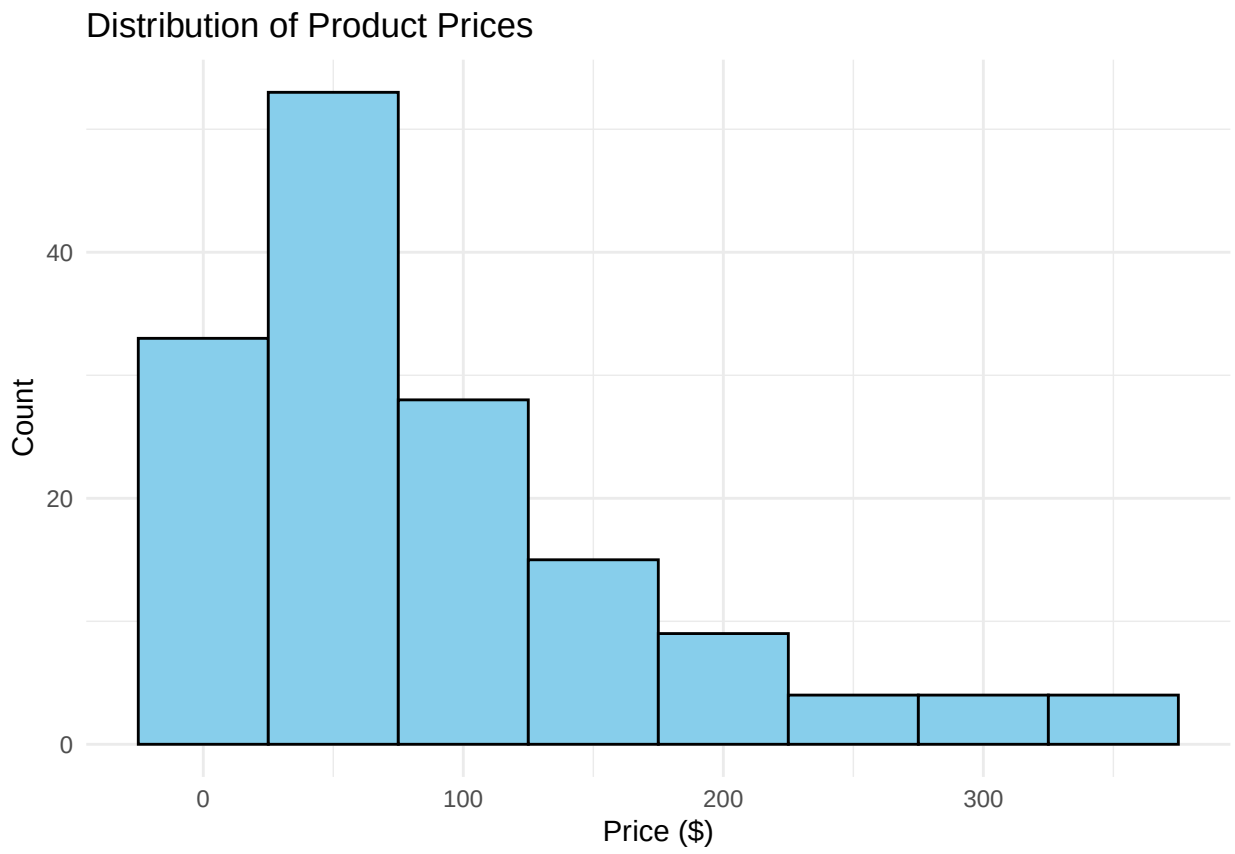
```
headset_products <- scrape_amazon_products(headset_url, "Headset", 30)
```

```
## Scraping: https://www.amazon.com/s?k=headset&crid=SHD7Y7OSWLUU&sprefix=headse%2Caps%2C449&ref=nb_sb_n

## Scraping: https://www.amazon.com/s?k=headset&crid=SHD7Y7OSWLUU&sprefix=headse%2Caps%2C449&ref=nb_sb_n
```

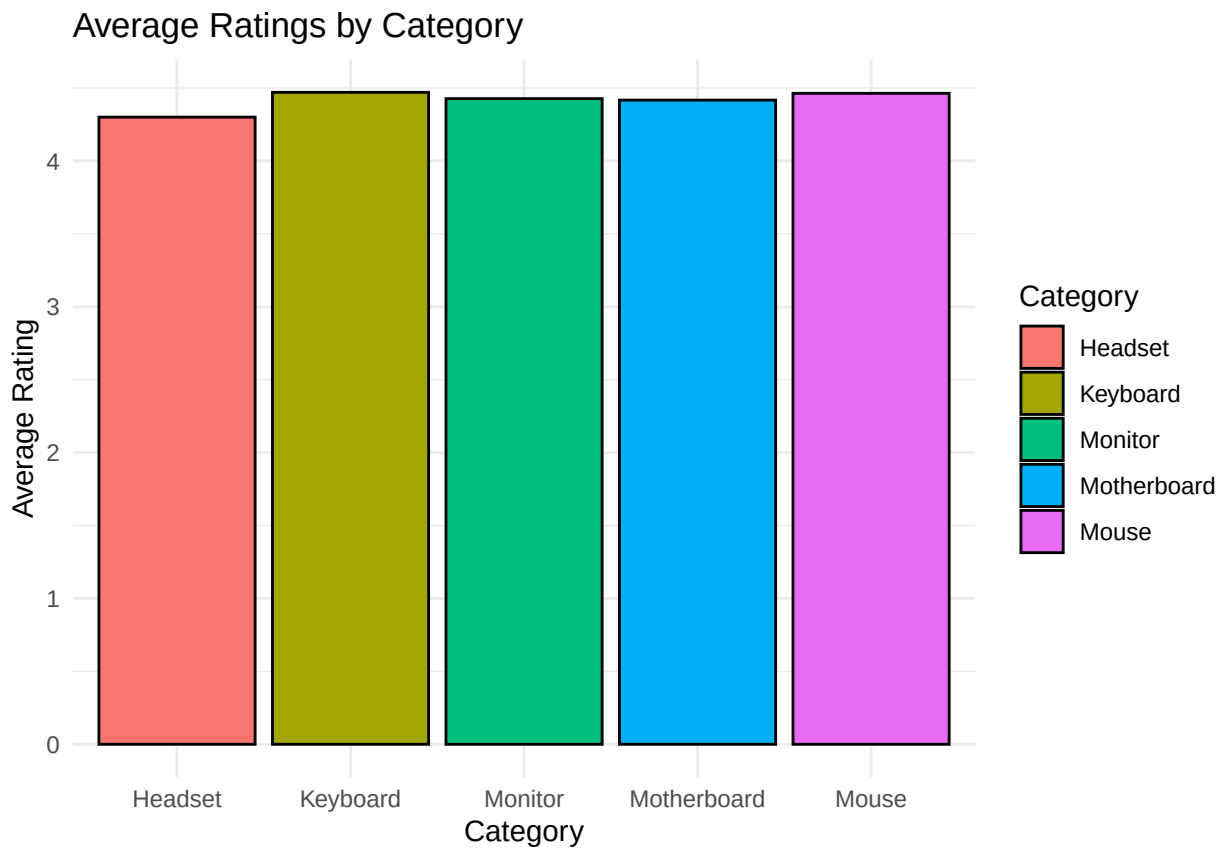
```r
all_products <- bind_rows(motherboard_products, monitor_products, mouse_products, keyboard_products, hea

all_products

#write.csv(all_products, "Amazon_products.csv")
```

6. The data we have extracted is that we scrape 30 products from each 6 categories and we get the product name, price, category, ratings, reviews, and description of every product.
7. The use case for our data that we have extracted is for market research.
8. The graphs provide insights into pricing trends, customer satisfaction, and product popularity across categories, helping identify competitive pricing and top-performing products. They also reveal correlations between ratings and reviews, indicating which products are both highly rated and widely purchased.

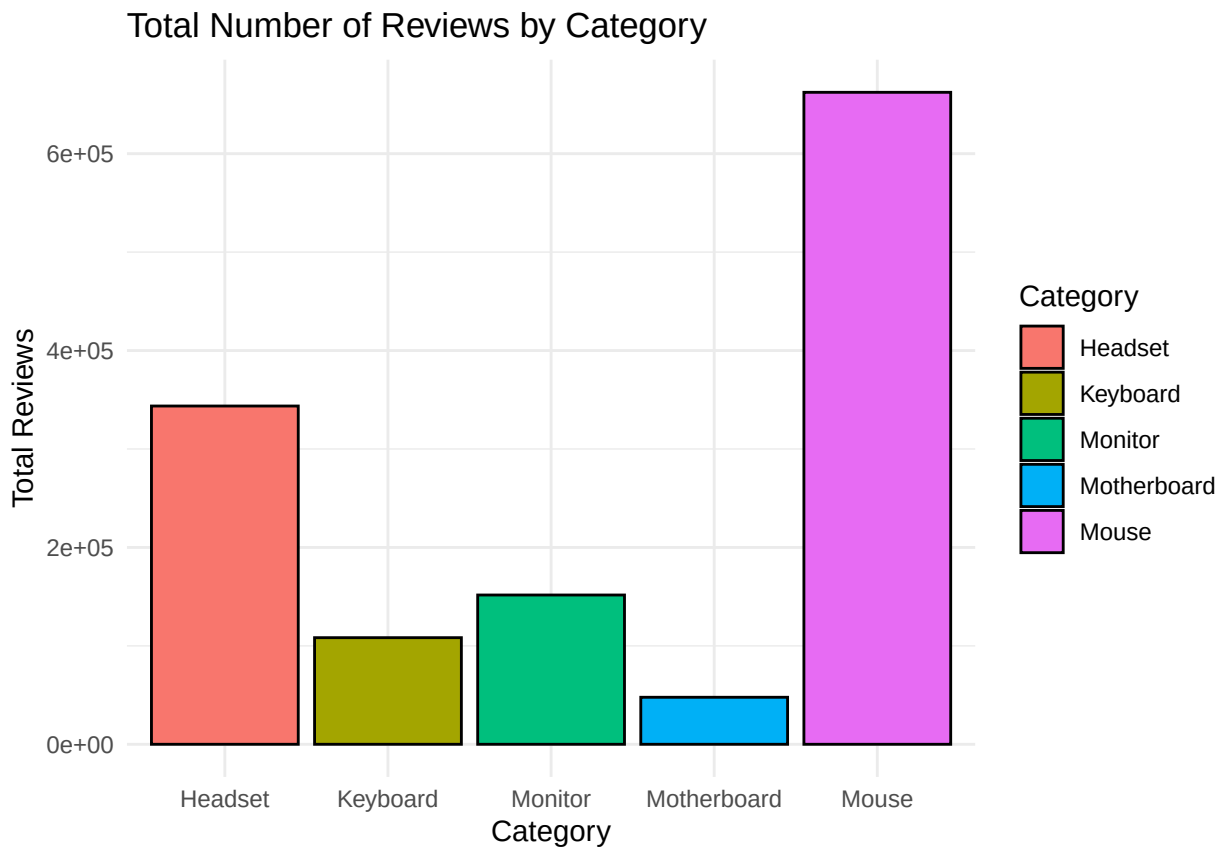
```r
library(ggplot2)
library(dplyr)

# Load data
all_products <- read.csv("Amazon_products.csv")

# Clean up price data (remove $ and convert to numeric)
all_products$Price <- as.numeric(gsub("[$,]", "", all_products$Price))

# Filter out NA or missing data
all_products <- all_products %>%
  filter(!is.na(Price), !is.na(Ratings), !is.na(Reviews))

# Graph 1: Price Distribution
ggplot(all_products, aes(x = Price)) +
  geom_histogram(binwidth = 50, fill = "skyblue", color = "black") +
  labs(title = "Distribution of Product Prices", x = "Price ($)", y = "Count") +
  theme_minimal()
```



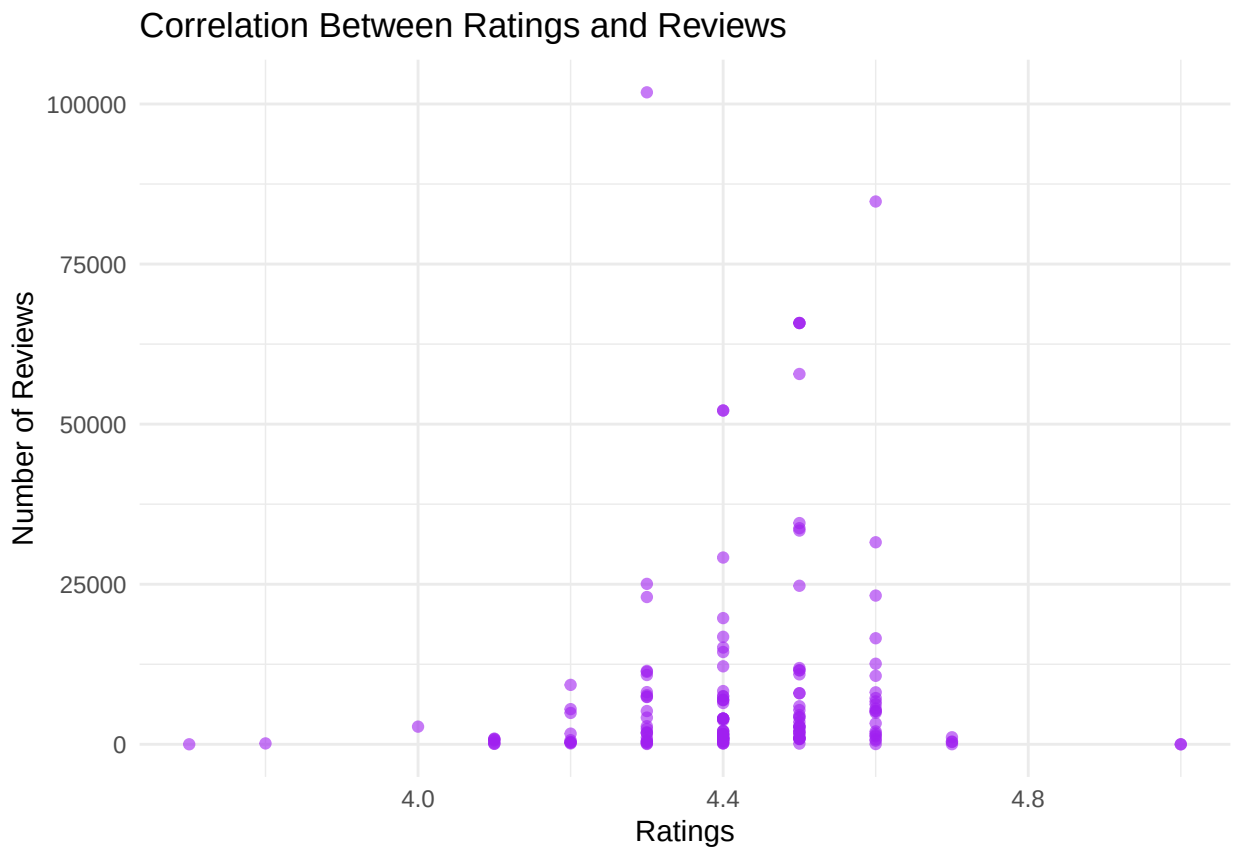

```r
# Graph 2: Average Ratings by Category
avg_ratings <- all_products %>%
  group_by(Category) %>%
  summarize(AverageRating = mean(Ratings, na.rm = TRUE))

ggplot(avg_ratings, aes(x = Category, y = AverageRating, fill = Category)) +
  geom_bar(stat = "identity", color = "black") +
  labs(title = "Average Ratings by Category", x = "Category", y = "Average Rating") +
  theme_minimal()
```




```r
# Graph 3: Number of Reviews by Category
total_reviews <- all_products %>%
  group_by(Category) %>%
  summarize(TotalReviews = sum(as.numeric(gsub("[^0-9]", "", Reviews)), na.rm = TRUE))

ggplot(total_reviews, aes(x = Category, y = TotalReviews, fill = Category)) +
  geom_bar(stat = "identity", color = "black") +
  labs(title = "Total Number of Reviews by Category", x = "Category", y = "Total Reviews") +
  theme_minimal()
```

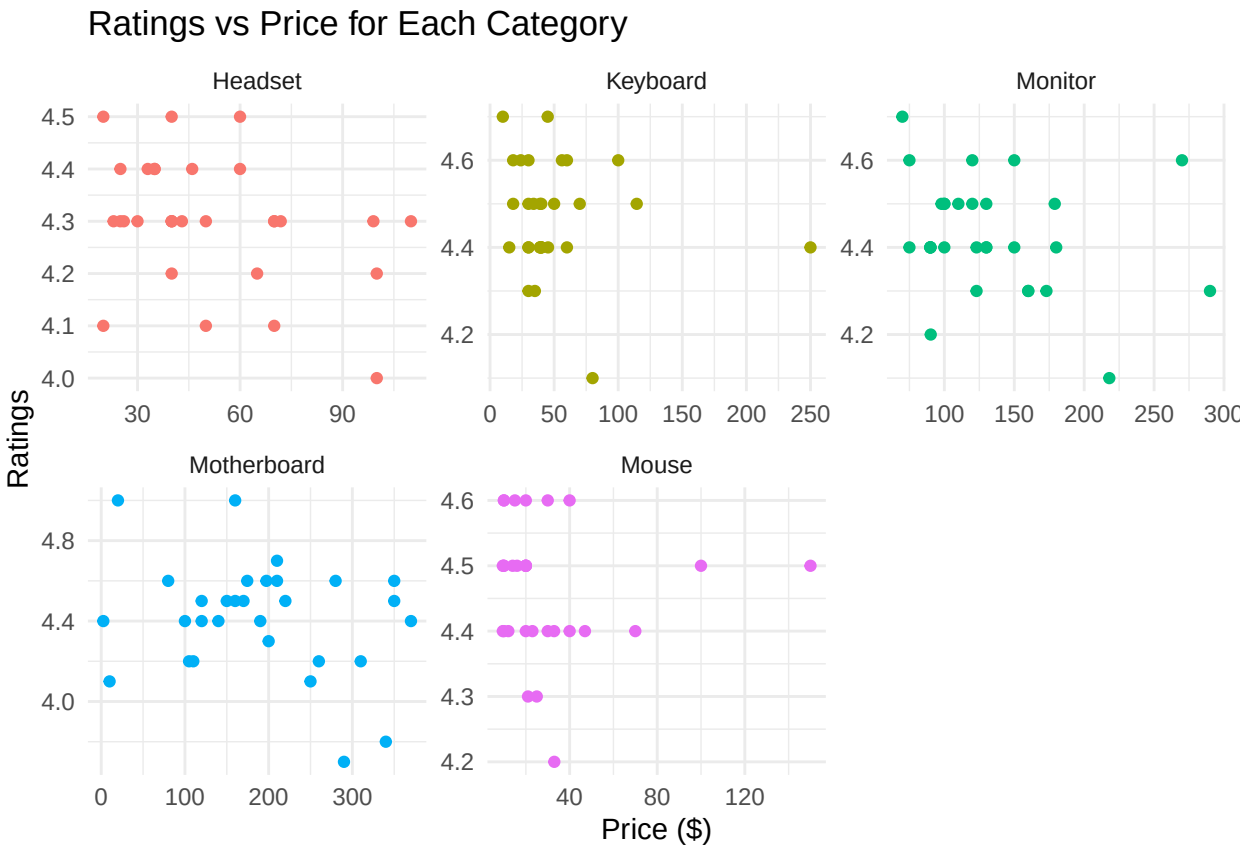



```r
# Graph 4: Correlation Between Ratings and Reviews
ggplot(all_products, aes(x = Ratings, y = as.numeric(gsub("[^0-9]", "", Reviews)))) +
  geom_point(alpha = 0.6, color = "purple") +
  labs(title = "Correlation Between Ratings and Reviews", x = "Ratings", y = "Number of Reviews") +
  theme_minimal()
```



9.

```
library(ggplot2)

ggplot(all_products, aes(x = Price, y = Ratings, color = Category)) +
  geom_point() +
  facet_wrap(~ Category, scales = "free") +
  labs(title = "Ratings vs Price for Each Category",
       x = "Price ($)",
       y = "Ratings") +
  theme_minimal() +
  theme(legend.position = "none")
```



10. Ranking by ratings sorts products from the highest to the lowest-rated, showcasing the top customer-reviewed items. Ranking by price can be done in ascending order (cheapest to most expensive) or descending order (most expensive to cheapest), helping users select products based on their budget.

```
all_products <- read.csv("Amazon_products.csv")


str(all_products)
```

```
## 'data.frame':    150 obs. of  7 variables:
##  $ X           : int  1 2 3 4 5 6 7 8 9 10 ...
##  $ ProductTitle: chr  "1. ASUS TUF Gaming X870-PLUS WiFi AMD AM5 X870 ATX Motherboard, 16+2+1, 80A SI
##  $ Price       : chr  "$289.99" "$309.99" "$349.99" "$369.99" ...
##  $ Category    : chr  "Motherboard" "Motherboard" "Motherboard" "Motherboard" ...
##  $ Ratings     : num  3.7 4.2 4.6 4.4 4.6 4.4 4.6 4.1 4.2 4.5 ...
##  $ Reviews     : chr  "6" "245" "10,695" "1,111" ...
##  $ Description : chr  "ASUS TUF Gaming X870-PLUS WiFi AMD AM5 X870 ATX Motherboard, 16+2+1, 80A SPS I
```

```
# Load the CSV file containing the products
all_products <- read.csv("Amazon_products.csv")

# Ensure the Ratings column is numeric for sorting
all_products$Ratings <- as.numeric(all_products$Ratings)
```

```r
# Check the structure to make sure Ratings are correctly formatted
str(all_products)
```

```
## 'data.frame':    150 obs. of  7 variables:
##  $ X           : int  1 2 3 4 5 6 7 8 9 10 ...
##  $ ProductTitle: chr  "1. ASUS TUF Gaming X870-PLUS WiFi AMD AM5 X870 ATX Motherboard, 16+2+1, 80A SI
##  $ Price       : chr  "$289.99" "$309.99" "$349.99" "$369.99" ...
##  $ Category    : chr  "Motherboard" "Motherboard" "Motherboard" "Motherboard" ...
##  $ Ratings     : num  3.7 4.2 4.6 4.4 4.6 4.4 4.6 4.1 4.2 4.5 ...
##  $ Reviews     : chr  "6" "245" "10,695" "1,111" ...
##  $ Description : chr  "ASUS TUF Gaming X870-PLUS WiFi AMD AM5 X870 ATX Motherboard, 16+2+1, 80A SPS I
```

```r
library(dplyr)

# Rank products by Ratings in descending order
ranked_by_ratings <- all_products %>%
  arrange(desc(Ratings))

# View the top-ranked products based on ratings
head(ranked_by_ratings, 150)  # Display the top 10 highest-rated products
```

```r
# Load the CSV file containing the products
all_products <- read.csv("Amazon_products.csv")

# Clean and convert the Price column to numeric (if it includes "$" signs)
all_products$Price <- as.numeric(gsub("\\$", "", all_products$Price))

# Check the structure to ensure Price is correctly formatted
str(all_products)
```

```
## 'data.frame':    150 obs. of  7 variables:
##  $ X           : int  1 2 3 4 5 6 7 8 9 10 ...
##  $ ProductTitle: chr  "1. ASUS TUF Gaming X870-PLUS WiFi AMD AM5 X870 ATX Motherboard, 16+2+1, 80A SI
##  $ Price       : num  290 310 350 370 174 ...
##  $ Category    : chr  "Motherboard" "Motherboard" "Motherboard" "Motherboard" ...
##  $ Ratings     : num  3.7 4.2 4.6 4.4 4.6 4.4 4.6 4.1 4.2 4.5 ...
##  $ Reviews     : chr  "6" "245" "10,695" "1,111" ...
##  $ Description : chr  "ASUS TUF Gaming X870-PLUS WiFi AMD AM5 X870 ATX Motherboard, 16+2+1, 80A SPS I
```

```r
library(dplyr)

# Rank products by Price in ascending order (cheapest first)
ranked_by_price_ascending <- all_products %>%
  arrange(Price)

# Alternatively, rank products by Price in descending order (most expensive first)
ranked_by_price_descending <- all_products %>%
  arrange(desc(Price))

# View the top-ranked products based on price (descending order)
head(ranked_by_price_descending, 150)  # Display top 10 most expensive products
```

```r
# Load necessary libraries
library(rvest)
library(dplyr)
library(stringr)

# Define product links and product names
product_links <- c(
  "https://www.amazon.com/TeckNet-Portable-Wireless-Receiver-Adjustment/dp/B001DHECXA/ref=sr_1_2_sspa",
  "https://www.amazon.com/EVGA-Customizable-Profiles-Ambidextrous-905-W1-12WH-KR/dp/B09QBYWN8T/ref=sr_1_
  "https://www.amazon.com/Logitech-Wireless-Mouse-M185-Swift/dp/B004YAVF8I/ref=sr_1_4",
  "https://www.amazon.com/Logitech-LIGHTSPEED-Wireless-Gaming-Mouse/dp/B07CMS5Q6P/ref=sr_1_5",
  "https://www.amazon.com/Razer-Basilisk-Customizable-Ergonomic-Gaming/dp/B09C13PZX7/ref=sr_1_6",
  "https://www.amazon.com/Micropack-Vertical-Multi-Device-Bluetooth-Adjustable/dp/B0C7GMQRW3/ref=sr_1_1
  "https://www.amazon.com/RAPIQUE-Bluetooth-Wireless-Mouse-Compatibility/dp/B0CYZ2VKGP/ref=sr_1_18",
  "https://www.amazon.com/ASUS-ROG-Wireless-Magnetic-Programmable/dp/B09726KT4R/ref=sr_1_20",
  "https://www.amazon.com/Logitech-910-005867-M575-Latin-America/dp/B08TLYK78K/ref=sr_1_21",
  "https://www.amazon.com/Logitech-Superlight-Lightspeed-Lightweight-Programmable/dp/B09NBWL8J5/ref=sr_
)

product_names <- c(
  "TeckNet Portable Wireless Mouse",
  "EVGA Customizable Ambidextrous Mouse",
  "Logitech Wireless Mouse M185",
  "Logitech LIGHTSPEED Wireless Gaming Mouse",
  "Razer Basilisk Customizable Ergonomic Gaming Mouse",
  "Micropack Vertical Multi-Device Bluetooth Adjustable",
  "RAPIQUE Bluetooth Wireless Mouse Compatibility",
  "ASUS ROG Wireless Magnetic Programmable",
  "Logitech 910-005867 M575 Latin America",
  "Logitech Superlight Lightspeed Lightweight Programmable"
)

category <- "Mouse"

# Initialize an empty dataframe
all_reviews <- data.frame()

# Loop through each product
for (i in seq_along(product_links)) {
  # Get the product link and name
  url <- product_links[i]
  product_name <- product_names[i]

  # Try scraping reviews
  try({
    webpage <- read_html(url)

    # Extract review sections
    reviews <- webpage %>%
      html_nodes(".review")

    # Extract reviewer names
    reviewer_names <- reviews %>%
```

```
    html_nodes(".a-profile-name") %>%
    html_text(trim = TRUE)

  # Extract review dates
  review_dates <- reviews %>%
    html_nodes(".review-date") %>%
    html_text(trim = TRUE)

  # Extract review titles
  review_titles <- reviews %>%
    html_nodes(".review-title span") %>%
    html_text(trim = TRUE)
   review_titles <- review_titles[review_titles != "5.0 out of 5 stars"]
  review_titles <- review_titles[review_titles != "4.0 out of 5 stars"]
  review_titles <- review_titles[review_titles != "1.0 out of 5 stars"]
  review_titles <- review_titles[review_titles != "3.0 out of 5 stars"]
  review_titles <- review_titles[review_titles != "2.0 out of 5 stars"]

  # Extract review comments
  review_comments <- reviews %>%
    html_nodes(".review-text-content span") %>%
    html_text(trim = TRUE)

  # Extract verified purchase labels
  verified_purchases <- reviews %>%
    html_nodes(".review-vp-label") %>%
    html_text(trim = TRUE)

  # Extract star ratings
  star_ratings <- reviews %>%
    html_nodes(".a-icon-alt") %>%
    html_text(trim = TRUE) %>%
    str_extract("\\d\\.\\d") %>%  # Extract the numeric rating
    as.numeric()

  # Limit to the first 20 reviews
  max_reviews <- min(20, length(reviewer_names))
  reviewer_names <- reviewer_names[1:max_reviews]
  review_dates <- review_dates[1:max_reviews]
  review_titles <- review_titles[1:max_reviews]
  review_comments <- review_comments[1:max_reviews]
  verified_purchases <- verified_purchases[1:max_reviews]
  star_ratings <- star_ratings[1:max_reviews]

  # Create a dataframe for this product
  review_data <- data.frame(
    Category = rep(category, max_reviews),
    ProductName = rep(product_name, max_reviews),
    Reviewer = reviewer_names,
    Date = review_dates,
    Title = review_titles,
    Comment = review_comments,
    StarRating = star_ratings,
```

```
      VerifiedPurchase = verified_purchases,
      stringsAsFactors = FALSE
    )

    # Append to the main dataframe
    all_reviews <- bind_rows(all_reviews, review_data)
  }, silent = TRUE)
}

print(all_reviews)

#write.csv(all_reviews, "Mouse_reviews.csv")
```

```
library(rvest)
library(dplyr)
library(stringr)

product_links <- c(
  "https://www.amazon.com/MSI-MAG-B550-TOMAHAWK-Motherboard/dp/B089CWDHFZ/ref=sr_1_1",
  "https://www.amazon.com/JUXIESHI-Motheboard-Support-Desktop-HDMI-Compatible/dp/B0CYBX123R/ref=sr_1_2",
  "https://www.amazon.com/ASUS-ROG-Strix-Gaming-Motherboard/dp/B08F9VP2RC/ref=sr_1_3",
  "https://www.amazon.com/ASUS-TUF-Intel%C2%AE12th-Motherboard-Thunderbolt/dp/B0BQD58D96/ref=sr_1_4",
  "https://www.amazon.com/MSI-B550-Gaming-GEN3-Motherboard/dp/B0BHCL4YVS/ref=sr_1_5",
  "https://www.amazon.com/ASUS-B550-PLUS-Motherboard-DisPlayPort-Addressable/dp/B089CT5GDM/ref=sr_1_6",
  "https://www.amazon.com/GIGABYTE-B650-Warranty-EZ-Latch-Motherboard/dp/B0BH7GTY9C/ref=sr_1_7",
  "https://www.amazon.com/MSI-B760-Motherboard-Supports-Processors/dp/B0C15THTK7/ref=sr_1_8",
  "https://www.amazon.com/GIGABYTE-B760M-DDR4-Motherboard-EZ-Latch/dp/B0D54QJ9CJ/ref=sr_1_9",
  "https://www.amazon.com/MSI-B650-Tomahawk-Motherboard-Processors/dp/B0BHCCNSRH/ref=sr_1_10"
)

product_names <- c(
  "MSI MAG B550 TOMAHAWK Motherboard",
  "JUXIESHI Motherboard Support Desktop HDMI Compatible",
  "ASUS ROG Strix Gaming Motherboard",
  "ASUS TUF Intel® 12th Motherboard Thunderbolt",
  "MSI B550 Gaming GEN3 Motherboard",
  "ASUS B550-PLUS Motherboard",
  "GIGABYTE B650 Warranty EZ-Latch Motherboard",
  "MSI B760 Motherboard Supports Processors",
  "GIGABYTE B760M DDR4 Motherboard EZ-Latch",
  "MSI B650 Tomahawk Motherboard Processors"
)

category <- "Motherboard"

all_reviews <- data.frame()

for (i in seq_along(product_links)) {
  # Get the product link and name
  url <- product_links[i]
  product_name <- product_names[i]

  try({
```

```r
  webpage <- read_html(url)

  reviews <- webpage %>%
    html_nodes(".review")

  reviewer_names <- reviews %>%
    html_nodes(".a-profile-name") %>%
    html_text(trim = TRUE)

  review_dates <- reviews %>%
    html_nodes(".review-date") %>%
    html_text(trim = TRUE)

  review_titles <- reviews %>%
    html_nodes(".review-title span") %>%
    html_text(trim = TRUE)
   review_titles <- review_titles[review_titles != "5.0 out of 5 stars"]
  review_titles <- review_titles[review_titles != "4.0 out of 5 stars"]
  review_titles <- review_titles[review_titles != "1.0 out of 5 stars"]
  review_titles <- review_titles[review_titles != "3.0 out of 5 stars"]
  review_titles <- review_titles[review_titles != "2.0 out of 5 stars"]

  review_comments <- reviews %>%
    html_nodes(".review-text-content span") %>%
    html_text(trim = TRUE)

  verified_purchases <- reviews %>%
    html_nodes(".review-vp-label") %>%
    html_text(trim = TRUE)

  star_ratings <- reviews %>%
    html_nodes(".a-icon-alt") %>%
    html_text(trim = TRUE) %>%
    str_extract("\\d\\.\\d") %>%  # Extract the numeric rating
    as.numeric()

  max_reviews <- min(20, length(reviewer_names))
  reviewer_names <- reviewer_names[1:max_reviews]
  review_dates <- review_dates[1:max_reviews]
  review_titles <- review_titles[1:max_reviews]
  review_comments <- review_comments[1:max_reviews]
  verified_purchases <- verified_purchases[1:max_reviews]
  star_ratings <- star_ratings[1:max_reviews]

  review_data <- data.frame(
    Category = rep(category, max_reviews),
    ProductName = rep(product_name, max_reviews),
    Reviewer = reviewer_names,
    Date = review_dates,
    Title = review_titles,
    Comment = review_comments,
    StarRating = star_ratings,
    VerifiedPurchase = verified_purchases,
```

```
      stringsAsFactors = FALSE
    )

    all_reviews <- bind_rows(all_reviews, review_data)
  }, silent = TRUE)
}

print(all_reviews)

#write.csv(all_reviews, "Motherboard_reviews.csv")
```

```
library(rvest)
library(dplyr)
library(stringr)

product_links <- c(
  "https://www.amazon.com/AmazonBasics-Matte-Keyboard-QWERTY-Layout/dp/B07WJ5D3H4/ref=sr_1_1",
  "https://www.amazon.com/RedThunder-Wireless-Rechargeable-Mechanical-Anti-ghosting/dp/B09BR46F63/ref=sr
  "https://www.amazon.com/SteelSeries-Worlds-Fastest-Mechanical-Keyboard/dp/B0BF64DN6H/ref=sr_1_3",
  "https://www.amazon.com/AmazonBasics-Matte-Keyboard-QWERTY-Layout/dp/B07WJ5D3H4/ref=sr_1_4",
  "https://www.amazon.com/Redragon-S101-Keyboard-Ergonomic-Programmable/dp/B00NLZUM36/ref=sr_1_5",
  "https://www.amazon.com/Logitech-Programmable-Backlighting-Bluetooth-Rechargeable/dp/B0BKW3LB2B/ref=sr
  "https://www.amazon.com/Amazon-Basics-Extension-Keyboard-Transfer/dp/B00NH13Q8W/ref=sr_1_7",
  "https://www.amazon.com/Rii-Multiple-Mechanical-Multimedia-Keyboard/dp/B0CCZWCPRQ/ref=sr_1_8",
  "https://www.amazon.com/Portable-Mechanical-Keyboard-MageGee-Backlit/dp/B098LG3N6R/ref=sr_1_9",
  "https://www.amazon.com/AULA-F2088-Mechanical-Removable-Keyboards/dp/B09DKQWTNC/ref=sr_1_10"
)

product_names <- c(
  "AmazonBasics Matte Keyboard QWERTY Layout",
  "RedThunder Wireless Rechargeable Mechanical Anti-Ghosting",
  "SteelSeries Worlds Fastest Mechanical Keyboard",
  "AmazonBasics Matte Keyboard QWERTY Layout",
  "Redragon S101 Keyboard Ergonomic Programmable",
  "Logitech Programmable Backlighting Bluetooth Rechargeable",
  "Amazon Basics Extension Keyboard Transfer",
  "Rii Multiple Mechanical Multimedia Keyboard",
  "Portable Mechanical Keyboard MageGee Backlit",
  "AULA F2088 Mechanical Removable Keyboards"
)

category <- "Keyboard"

all_reviews <- data.frame()

for (i in seq_along(product_links)) {
  # Get the product link and name
  url <- product_links[i]
  product_name <- product_names[i]

  try({
    webpage <- read_html(url)
```

```r
reviews <- webpage %>%
  html_nodes(".review")

reviewer_names <- reviews %>%
  html_nodes(".a-profile-name") %>%
  html_text(trim = TRUE)

review_dates <- reviews %>%
  html_nodes(".review-date") %>%
  html_text(trim = TRUE)

review_titles <- reviews %>%
  html_nodes(".review-title span") %>%
  html_text(trim = TRUE)
 review_titles <- review_titles[review_titles != "5.0 out of 5 stars"]
review_titles <- review_titles[review_titles != "4.0 out of 5 stars"]
review_titles <- review_titles[review_titles != "1.0 out of 5 stars"]
review_titles <- review_titles[review_titles != "3.0 out of 5 stars"]
review_titles <- review_titles[review_titles != "2.0 out of 5 stars"]

review_comments <- reviews %>%
  html_nodes(".review-text-content span") %>%
  html_text(trim = TRUE)

verified_purchases <- reviews %>%
  html_nodes(".review-vp-label") %>%
  html_text(trim = TRUE)

star_ratings <- reviews %>%
  html_nodes(".a-icon-alt") %>%
  html_text(trim = TRUE) %>%
  str_extract("\\d\\.\\d") %>%  # Extract the numeric rating
  as.numeric()

max_reviews <- min(20, length(reviewer_names))
reviewer_names <- reviewer_names[1:max_reviews]
review_dates <- review_dates[1:max_reviews]
review_titles <- review_titles[1:max_reviews]
review_comments <- review_comments[1:max_reviews]
verified_purchases <- verified_purchases[1:max_reviews]
star_ratings <- star_ratings[1:max_reviews]

review_data <- data.frame(
  Category = rep(category, max_reviews),
  ProductName = rep(product_name, max_reviews),
  Reviewer = reviewer_names,
  Date = review_dates,
  Title = review_titles,
  Comment = review_comments,
  StarRating = star_ratings,
  VerifiedPurchase = verified_purchases,
  stringsAsFactors = FALSE
)
```

```
    all_reviews <- bind_rows(all_reviews, review_data)
  }, silent = TRUE)
}

print(all_reviews)

#write.csv(all_reviews, "Keyboard_reviews.csv")
```

```
library(rvest)
library(dplyr)
library(stringr)

product_links <- c(
  "https://www.amazon.com/Amazon-Basics-Monitor-Compatible-Speakers/dp/B0CP7RZRMD/ref=sr_1_1",
  "https://www.amazon.com/KOORUI-Speakers-Adpitive-Compatible-S01/dp/B0C38FJ6HY/ref=sr_1_2",
  "https://www.amazon.com/PHILIPS-Adaptive-Replacement-Warranty-221V8LB/dp/B0CVM2GJCN/ref=sr_1_3",
  "https://www.amazon.com/Sceptre-DisplayPort-FreeSync-Frameless-E275W-FW100T/dp/B0CHHSFMRL/ref=sr_1_4",
  "https://www.amazon.com/SAMSUNG-27-inch-Border-Less-FreeSync-LF27T350FHNXZA/dp/B08FF3JQ28/ref=sr_1_5",
  "https://www.amazon.com/acer-Zero-Frame-Adaptive-Sync-FreeSync-Compatible/dp/B0D9MK23S7/ref=sr_1_6",
  "https://www.amazon.com/Amazon-Basics-75hz-Panel-Monitor/dp/B08WJ26WP6/ref=sr_1_7",
  "https://www.amazon.com/Sceptre-Curved-Monitor-Speakers-C248W-1920RN/dp/B07KXSR99Y/ref=sr_1_8",
  "https://www.amazon.com/Viewedge-Professional-Monitor-Protection-Adaptive/dp/B0B8BYXXSC/ref=sr_1_9",
  "https://www.amazon.com/Sceptre-24-5-inch-DisplayPort-Speakers-C255B-FWT240/dp/B0BTKJFRDV/ref=sr_1_10
)

product_names <- c(
  "Amazon Basics Monitor Compatible Speakers",
  "KOORUI Speakers Adaptive Compatible S01",
  "PHILIPS Adaptive Replacement Warranty 221V8LB",
  "Sceptre DisplayPort FreeSync Frameless E275W-FW100T",
  "SAMSUNG 27-inch Border-Less FreeSync LF27T350FHNXZA",
  "Acer Zero Frame Adaptive Sync FreeSync Compatible",
  "Amazon Basics 75Hz Panel Monitor",
  "Sceptre Curved Monitor Speakers C248W-1920RN",
  "Viewedge Professional Monitor Protection Adaptive",
  "Sceptre 24.5-inch DisplayPort Speakers C255B-FWT240"
)

category <- "Monitor"

all_reviews <- data.frame()

for (i in seq_along(product_links)) {
  # Get the product link and name
  url <- product_links[i]
  product_name <- product_names[i]

  try({
    webpage <- read_html(url)

    reviews <- webpage %>%
      html_nodes(".review")
```

```
  reviewer_names <- reviews %>%
    html_nodes(".a-profile-name") %>%
    html_text(trim = TRUE)

  review_dates <- reviews %>%
    html_nodes(".review-date") %>%
    html_text(trim = TRUE)

  review_titles <- reviews %>%
    html_nodes(".review-title span") %>%
    html_text(trim = TRUE)
   review_titles <- review_titles[review_titles != "5.0 out of 5 stars"]
  review_titles <- review_titles[review_titles != "4.0 out of 5 stars"]
  review_titles <- review_titles[review_titles != "1.0 out of 5 stars"]
  review_titles <- review_titles[review_titles != "3.0 out of 5 stars"]
  review_titles <- review_titles[review_titles != "2.0 out of 5 stars"]

  review_comments <- reviews %>%
    html_nodes(".review-text-content span") %>%
    html_text(trim = TRUE)

  verified_purchases <- reviews %>%
    html_nodes(".review-vp-label") %>%
    html_text(trim = TRUE)

  star_ratings <- reviews %>%
    html_nodes(".a-icon-alt") %>%
    html_text(trim = TRUE) %>%
    str_extract("\\d\\.\\d") %>%  # Extract the numeric rating
    as.numeric()

  max_reviews <- min(20, length(reviewer_names))
  reviewer_names <- reviewer_names[1:max_reviews]
  review_dates <- review_dates[1:max_reviews]
  review_titles <- review_titles[1:max_reviews]
  review_comments <- review_comments[1:max_reviews]
  verified_purchases <- verified_purchases[1:max_reviews]
  star_ratings <- star_ratings[1:max_reviews]

  review_data <- data.frame(
    Category = rep(category, max_reviews),
    ProductName = rep(product_name, max_reviews),
    Reviewer = reviewer_names,
    Date = review_dates,
    Title = review_titles,
    Comment = review_comments,
    StarRating = star_ratings,
    VerifiedPurchase = verified_purchases,
    stringsAsFactors = FALSE
  )

  all_reviews <- bind_rows(all_reviews, review_data)
}, silent = TRUE)
```

```
}

print(all_reviews)

#write.csv(all_reviews, "Monitor_reviews.csv")
```

```
library(rvest)
library(dplyr)
library(stringr)

product_links <- c(
  "https://www.amazon.com/Microphone-Cancelling-Over-Ear-Headphones-Computer/dp/B0C1GTXY5S/ref=sr_1_1",
  "https://www.amazon.com/OneOdio-PRO1030-DJ-Headphone-Black/dp/B01N6ZJH96/ref=sr_1_2",
  "https://www.amazon.com/JBL-Quantum-400-Headphones-Game-Chat/dp/B084CZDX61/ref=sr_1_3",
  "https://www.amazon.com/Razer-BlackShark-V2-Gaming-Headset/dp/B09PZG4R17/ref=sr_1_4",
  "https://www.amazon.com/Turtle-Gaming-Headset-PlayStation-Nintendo-Console/dp/B00YXO5U40/ref=sr_1_5",
  "https://www.amazon.com/CORSAIR-SURROUND-Multiplatform-Gaming-Headset/dp/B09YHQ3Y61/ref=sr_1_6",
  "https://www.amazon.com/Fachixy-Gaming-Headset-Canceling-Microphone/dp/B09C5MDY2Y/ref=sr_1_7",
  "https://www.amazon.com/BENGOO-G9000-Controller-Cancelling-Headphones/dp/B01H6GUCCQ/ref=sr_1_8",
  "https://www.amazon.com/Stealth-Wireless-Multiplatform-Amplified-Headset-Nintendo/dp/B0CYWFH5Y9/ref=s
  "https://www.amazon.com/Logitech-Wireless-Lightspeed-Headset-Headphone/dp/B081PP4CB6/ref=sr_1_10"
)

product_names <- c(
  "Microphone Cancelling Over-Ear Headphones",
  "OneOdio PRO1030 DJ Headphone",
  "JBL Quantum 400 Headphones Game Chat",
  "Razer BlackShark V2 Gaming Headset",
  "Turtle Gaming Headset PlayStation Nintendo Console",
  "CORSAIR SURROUND Multiplatform Gaming Headset",
  "Fachixy Gaming Headset Canceling Microphone",
  "BENGOO G9000 Controller Cancelling Headphones",
  "Stealth Wireless Multiplatform Amplified Headset",
  "Logitech Wireless Lightspeed Headset"
)

category <- "Headset"

all_reviews <- data.frame()

for (i in seq_along(product_links)) {
  # Get the product link and name
  url <- product_links[i]
  product_name <- product_names[i]

  try({
    webpage <- read_html(url)

    reviews <- webpage %>%
      html_nodes(".review")

    reviewer_names <- reviews %>%
      html_nodes(".a-profile-name") %>%
```

```
      html_text(trim = TRUE)

    review_dates <- reviews %>%
      html_nodes(".review-date") %>%
      html_text(trim = TRUE)

    review_titles <- reviews %>%
      html_nodes(".review-title span") %>%
      html_text(trim = TRUE)
     review_titles <- review_titles[review_titles != "5.0 out of 5 stars"]
    review_titles <- review_titles[review_titles != "4.0 out of 5 stars"]
    review_titles <- review_titles[review_titles != "1.0 out of 5 stars"]
    review_titles <- review_titles[review_titles != "3.0 out of 5 stars"]
    review_titles <- review_titles[review_titles != "2.0 out of 5 stars"]

    review_comments <- reviews %>%
      html_nodes(".review-text-content span") %>%
      html_text(trim = TRUE)

    verified_purchases <- reviews %>%
      html_nodes(".review-vp-label") %>%
      html_text(trim = TRUE)

    star_ratings <- reviews %>%
      html_nodes(".a-icon-alt") %>%
      html_text(trim = TRUE) %>%
      str_extract("\\d\\.\\d") %>%  # Extract the numeric rating
      as.numeric()

    max_reviews <- min(20, length(reviewer_names))
    reviewer_names <- reviewer_names[1:max_reviews]
    review_dates <- review_dates[1:max_reviews]
    review_titles <- review_titles[1:max_reviews]
    review_comments <- review_comments[1:max_reviews]
    verified_purchases <- verified_purchases[1:max_reviews]
    star_ratings <- star_ratings[1:max_reviews]

    review_data <- data.frame(
      Category = rep(category, max_reviews),
      ProductName = rep(product_name, max_reviews),
      Reviewer = reviewer_names,
      Date = review_dates,
      Title = review_titles,
      Comment = review_comments,
      StarRating = star_ratings,
      VerifiedPurchase = verified_purchases,
      stringsAsFactors = FALSE
    )

    all_reviews <- bind_rows(all_reviews, review_data)
  }, silent = TRUE)
}
```

```r
print(all_reviews)

#write.csv(all_reviews, "Headset_reviews.csv")
```