RWorksheet_5a

Jalando-on, Nandin, Palabrica

2024-11-27

MDB

```
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(stringr)
library(polite)
library(kableExtra)
```

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

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

```
library(knitr)
library(tidyverse)
```

```
## -- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
## v forcats   1.0.0     v readr     2.1.5
## v ggplot2   3.5.1     v tibble    3.2.1
## v lubridate 1.9.3     v tidyr     1.3.1
## v purrr     1.0.2
```

```
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter()          masks stats::filter()
## x kableExtra::group_rows() masks dplyr::group_rows()
## x readr::guess_encoding()  masks rvest::guess_encoding()
## x dplyr::lag()             masks stats::lag()
## i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become error
```

```
link = "https://www.imdb.com/chart/toptv/"
page = read_html(link)
session <- bow(link, user_agent = "Educational")
        session
```

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

```
nam <- page %>% html_nodes(".ipc-title__text") %>% html_text()
name <- nam[!grepl("Top 250 TV Shows|IMDb Charts|Recently viewed|More to explore", nam, ignore.case = T
name
```

```
##  [1] "1. Breaking Bad"
##  [2] "2. Planet Earth II"
##  [3] "3. Planet Earth"
##  [4] "4. Band of Brothers"
##  [5] "5. Chernobyl"
##  [6] "6. The Wire"
##  [7] "7. Avatar: The Last Airbender"
##  [8] "8. Blue Planet II"
##  [9] "9. The Sopranos"
## [10] "10. Cosmos: A Spacetime Odyssey"
## [11] "11. Cosmos"
## [12] "12. Our Planet"
## [13] "13. Game of Thrones"
## [14] "14. Bluey"
## [15] "15. The World at War"
## [16] "16. Fullmetal Alchemist: Brotherhood"
## [17] "17. Rick and Morty"
## [18] "18. Life"
## [19] "19. The Last Dance"
## [20] "20. The Twilight Zone"
## [21] "21. The Vietnam War"
## [22] "22. Sherlock"
## [23] "23. Attack on Titan"
## [24] "24. Batman: The Animated Series"
## [25] "25. Arcane"
```

```
rank <- str_extract(name, "^\\d+\\.")
rank
```

```
##  [1] "1."  "2."  "3."  "4."  "5."  "6."  "7."  "8."  "9."  "10." "11." "12."
## [13] "13." "14." "15." "16." "17." "18." "19." "20." "21." "22." "23." "24."
## [25] "25."
```

```
title <- str_replace(name, "^\\d+\\.", "")
title
```

```
##  [1] " Breaking Bad"                    " Planet Earth II"
##  [3] " Planet Earth"                    " Band of Brothers"
##  [5] " Chernobyl"                       " The Wire"
##  [7] " Avatar: The Last Airbender"      " Blue Planet II"
##  [9] " The Sopranos"                    " Cosmos: A Spacetime Odyssey"
## [11] " Cosmos"                          " Our Planet"
## [13] " Game of Thrones"                 " Bluey"
## [15] " The World at War"                " Fullmetal Alchemist: Brotherhood"
## [17] " Rick and Morty"                  " Life"
## [19] " The Last Dance"                  " The Twilight Zone"
## [21] " The Vietnam War"                 " Sherlock"
## [23] " Attack on Titan"                  " Batman: The Animated Series"
## [25] " Arcane"
```

```
yea = page %>% html_nodes(".cli-title-metadata-item") %>% html_text()
year <- str_extract_all(yea, "\\b\\d{4}\\b") %>% unlist()
year
```

```
##  [1] "2008" "2013" "2016" "2006" "2001" "2019" "2002" "2008" "2005" "2008"
## [11] "2017" "1999" "2007" "2014" "1980" "2019" "2023" "2011" "2019" "2018"
## [21] "1973" "1974" "2009" "2010" "2013" "2009" "2020" "1959" "1964" "2017"
## [31] "2010" "2017" "2013" "2023" "1992" "1995" "2021" "2024"
```

```
rating = page %>% html_nodes(".ipc-rating-star--rating") %>% html_text()
rating
```

```
##  [1] "9.5" "9.5" "9.4" "9.4" "9.3" "9.3" "9.3" "9.3" "9.2" "9.2" "9.3" "9.2"
## [13] "9.2" "9.3" "9.2" "9.1" "9.1" "9.1" "9.0" "9.0" "9.1" "9.1" "9.1" "9.0"
## [25] "9.0"
```

```
episode <- page %>% html_nodes(".cli-title-metadata-item") %>% html_text()
episodes <- str_extract_all(episode, "\\d+\\s*eps?\\b") %>% unlist()
episodes
```

```
##  [1] "62 eps"  "6 eps"   "11 eps"  "10 eps"  "5 eps"   "60 eps"  "62 eps"
##  [8] "7 eps"   "86 eps"  "13 eps"  "13 eps"  "12 eps"  "74 eps"  "194 eps"
## [15] "26 eps"  "68 eps"  "78 eps"  "11 eps"  "10 eps"  "156 eps" "10 eps"
## [22] "15 eps"  "98 eps"  "85 eps"  "18 eps"
```

```
vote = page %>% html_nodes(".ipc-rating-star--voteCount") %>% html_text()
vote
```

```
##  [1] " (2.2M)" " (163K)" " (224K)" " (547K)" " (911K)" " (392K)" " (391K)"
##  [8] " (49K)"  " (501K)" " (132K)" " (46K)"  " (54K)"  " (2.4M)" " (34K)"
## [15] " (32K)"  " (210K)" " (629K)" " (44K)"  " (160K)" " (98K)"  " (30K)"
## [22] " (1M)"   " (566K)" " (123K)" " (337K)"
```

```
urls <- c("https://www.imdb.com/title/tt0903747/?ref_=chttvtp_i_1",
          "https://www.imdb.com/title/tt5491994/?ref_=chttvtp_i_2",
          "https://www.imdb.com/title/tt0795176/?ref_=chttvtp_i_3",
          "https://www.imdb.com/title/tt0185906/?ref_=chttvtp_i_4",
          "https://www.imdb.com/title/tt7366338/?ref_=chttvtp_i_5",
          "https://www.imdb.com/title/tt0306414/?ref_=chttvtp_i_6",
          "https://www.imdb.com/title/tt0417299/?ref_=chttvtp_i_7",
          "https://www.imdb.com/title/tt6769208/?ref_=chttvtp_i_8",
          "https://www.imdb.com/title/tt0141842/?ref_=chttvtp_i_9",
          "https://www.imdb.com/title/tt2395695/?ref_=chttvtp_i_10",
          "https://www.imdb.com/title/tt0081846/?ref_=chttvtp_i_11",
          "https://www.imdb.com/title/tt9253866/?ref_=chttvtp_i_12",
          "https://www.imdb.com/title/tt0944947/?ref_=chttvtp_i_13",
          "https://www.imdb.com/title/tt7678620/?ref_=chttvtp_i_14",
          "https://www.imdb.com/title/tt0071075/?ref_=chttvtp_i_15",
          "https://www.imdb.com/title/tt1355642/?ref_=chttvtp_i_16",
          "https://www.imdb.com/title/tt2861424/?ref_=chttvtp_i_17",
          "https://www.imdb.com/title/tt1533395/?ref_=chttvtp_i_18",
          "https://www.imdb.com/title/tt8420184/?ref_=chttvtp_i_19",
          "https://www.imdb.com/title/tt0052520/?ref_=chttvtp_i_20",
          "https://www.imdb.com/title/tt1877514/?ref_=chttvtp_i_21",
          "https://www.imdb.com/title/tt1475582/?ref_=chttvtp_i_22",
          "https://www.imdb.com/title/tt2560140/?ref_=chttvtp_i_23",
          "https://www.imdb.com/title/tt0103359/?ref_=chttvtp_i_24",
          "https://www.imdb.com/title/tt0386676/?ref_=chttvtp_i_25")

user_reviews <- vector("numeric", length(urls))
critic_reviews <- vector("numeric", length(urls))
for (i in seq_along(urls)) {

  session <- bow(urls[i], user_agent = "Educational")

  webpage <- scrape(session)

  reviewz <- webpage %>% html_nodes(".score") %>% html_text()

  if (length(reviewz) >= 2) {

    user_reviews[i] <- ifelse(grepl("K", reviewz[1]),
                              as.numeric(gsub("K", "", reviewz[1])) * 1000,
                              as.numeric(reviewz[1]))
    critic_reviews[i] <- as.numeric(reviewz[2])
  } else {
    user_reviews[i] <- NA
    critic_reviews[i] <- NA
  }
}
```

```
user_reviews
```

```
##  [1] 5100  158  111 1000 3500  787 1000   53  968  205   80  245 5900  369  126
## [16]  468  910   12  542  214  175 1000 2300  219 1700
```

```
critic_reviews
```

```
##  [1] 175   6  10  34  88  77  57   9  93  12   8  15 368   4   5  16  94   9  28
## [20]  85  13 121  64  25  76
```

```
max_length <- max(length(rank), length(title), length(year), length(rating), length(episodes), length(vo
rank <- c(rank, rep(NA, max_length - length(rank)))
title <- c(title, rep(NA, max_length - length(title)))
year <- c(year, rep(NA, max_length - length(year)))
rating <- c(rating, rep(NA, max_length - length(rating)))
episodes <- c(episodes, rep(NA, max_length - length(episodes)))
vote <- c(vote, rep(NA, max_length - length(vote)))
user_reviews <- c(user_reviews, rep(NA, max_length - length(user_reviews)))
critic_reviews <- c(critic_reviews, rep(NA, max_length - length(critic_reviews)))
max_length
```

```
## [1] 38
```

```
movies = data.frame(rank, title, year, rating, episodes, vote, user_reviews, critic_reviews, stringsAsFa
write.csv(movies, "movies.csv")
print(head(movies))
```

```
##   rank              title year rating episodes    vote user_reviews
## 1   1.       Breaking Bad 2008    9.5   62 eps  (2.2M)         5100
## 2   2.   Planet Earth II 2013    9.5    6 eps  (163K)          158
## 3   3.      Planet Earth 2016    9.4   11 eps  (224K)          111
## 4   4.  Band of Brothers 2006    9.4   10 eps  (547K)         1000
## 5   5.         Chernobyl 2001    9.3    5 eps  (911K)         3500
## 6   6.          The Wire 2019    9.3   60 eps  (392K)          787
##   critic_reviews
## 1            175
## 2              6
## 3             10
## 4             34
## 5             88
## 6             77
```

```
movies %>%
  kable("latex", booktabs = TRUE) %>%
  kable_styling(latex_options = "scale_down")
```

```
link2 = "https://www.imdb.com/title/tt0903747/reviews/?ref_=tt_ov_ql_2"
page2 = read_html(link)
session2 <- bow(link, user_agent = "Educational")
        session2
```

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

```
reviews <- page2 %>% html_nodes(".ipc-link--base") %>%
  html_text()
reviews
```

```
## [1] "Learn more about how list ranking is determined."
```

```
date <- page2 %>% html_nodes(".ipc-inline-list__item.review-date") %>%
  html_text()
date
```

```
## character(0)
```

```
user_rating <- page2 %>% html_nodes(".sc-a2ac93e5-4.gyib0i") %>%
  html_text()
user_rating
```

```
## character(0)
```

```
link1 = "https://www.imdb.com/chart/toptv/"
page1 = read_html(link)
session1 <- bow(link1, user_agent = "Educational")
        session1
```

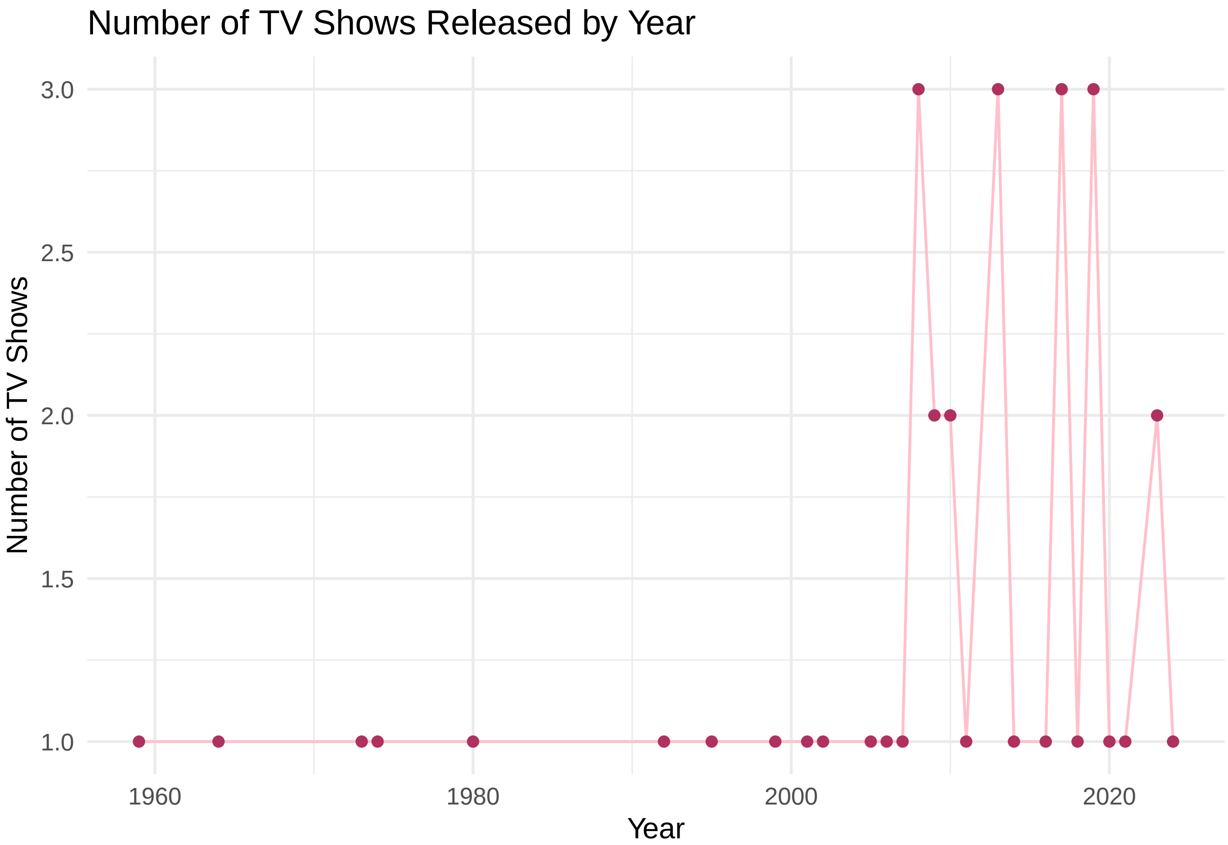
```
## <polite session> https://www.imdb.com/chart/toptv/
##     User-agent: Educational
##     robots.txt: 35 rules are defined for 3 bots
##    Crawl delay: 5 sec
##   The path is scrapable for this user-agent
```

```
user_review = page %>% html_nodes(".score") %>% html_text()
user_review
```

rank	title	year	rating	episodes	vote	user_reviews	critic_reviews
1.	Breaking Bad	2008	9.5	62 eps	(2.2M)	5100	175
2.	Planet Earth II	2013	9.5	6 eps	(163K)	158	6
3.	Planet Earth	2016	9.4	11 eps	(224K)	111	10
4.	Band of Brothers	2006	9.4	10 eps	(547K)	1000	34
5.	Chernobyl	2001	9.3	5 eps	(911K)	3500	88
6.	The Wire	2019	9.3	60 eps	(392K)	787	77
7.	Avatar: The Last Airbender	2002	9.3	62 eps	(391K)	1000	57
8.	Blue Planet II	2008	9.3	7 eps	(49K)	53	9
9.	The Sopranos	2005	9.2	86 eps	(501K)	968	93
10.	Cosmos: A Spacetime Odyssey	2008	9.2	13 eps	(132K)	205	12
11.	Cosmos	2017	9.3	13 eps	(46K)	80	8
12.	Our Planet	1999	9.2	12 eps	(54K)	245	15
13.	Game of Thrones	2007	9.2	74 eps	(2.4M)	5900	368
14.	Bluey	2014	9.3	194 eps	(34K)	369	4
15.	The World at War	1980	9.2	26 eps	(32K)	126	5
16.	Fullmetal Alchemist: Brotherhood	2019	9.1	68 eps	(210K)	468	16
17.	Rick and Morty	2023	9.1	78 eps	(629K)	910	94
18.	Life	2011	9.1	11 eps	(44K)	12	9
19.	The Last Dance	2019	9.0	10 eps	(160K)	542	28
20.	The Twilight Zone	2018	9.0	156 eps	(98K)	214	85
21.	The Vietnam War	1973	9.1	10 eps	(30K)	175	13
22.	Sherlock	1974	9.1	15 eps	(1M)	1000	121
23.	Attack on Titan	2009	9.1	98 eps	(566K)	2300	64
24.	Batman: The Animated Series	2010	9.0	85 eps	(123K)	219	25
25.	Arcane	2013	9.0	18 eps	(337K)	1700	76
NA	NA	2009	NA	NA	NA	NA	NA
NA	NA	2020	NA	NA	NA	NA	NA
NA	NA	1959	NA	NA	NA	NA	NA
NA	NA	1964	NA	NA	NA	NA	NA
NA	NA	2017	NA	NA	NA	NA	NA
NA	NA	2010	NA	NA	NA	NA	NA
NA	NA	2017	NA	NA	NA	NA	NA
NA	NA	2013	NA	NA	NA	NA	NA
NA	NA	2023	NA	NA	NA	NA	NA
NA	NA	1992	NA	NA	NA	NA	NA
NA	NA	1995	NA	NA	NA	NA	NA
NA	NA	2021	NA	NA	NA	NA	NA
NA	NA	2024	NA	NA	NA	NA	NA

```
## character(0)
```

```
library(ggplot2)

movies$year <- as.numeric(movies$year)
year_counts <- movies %>%
  filter(!is.na(year)) %>%
  count(year)

ggplot(year_counts, aes(x = year, y = n)) +
  geom_line(color = "pink") +
  geom_point(color = "maroon") +
  labs(title = "Number of TV Shows Released by Year",
       x = "Year",
       y = "Number of TV Shows") +
  theme_minimal()
```



```
most_releases <- year_counts[which.max(year_counts$n), ]
print(most_releases)
```

```
##    year n
## 14 2008 3
```

AMAZON

```
library(rvest)
library(httr)
library(stringr)
library(dplyr)
library(ggplot2)
```

```
#4. URLs
urls <- c('https://www.amazon.com/s?k=men%27s+clothing',
          'https://www.amazon.com/s?k=men+shoes',
          'https://www.amazon.com/s?k=women+jewelry',
          'https://www.amazon.com/s?k=baby+gifts',
          'https://www.amazon.com/s?k=women+accessories')
```

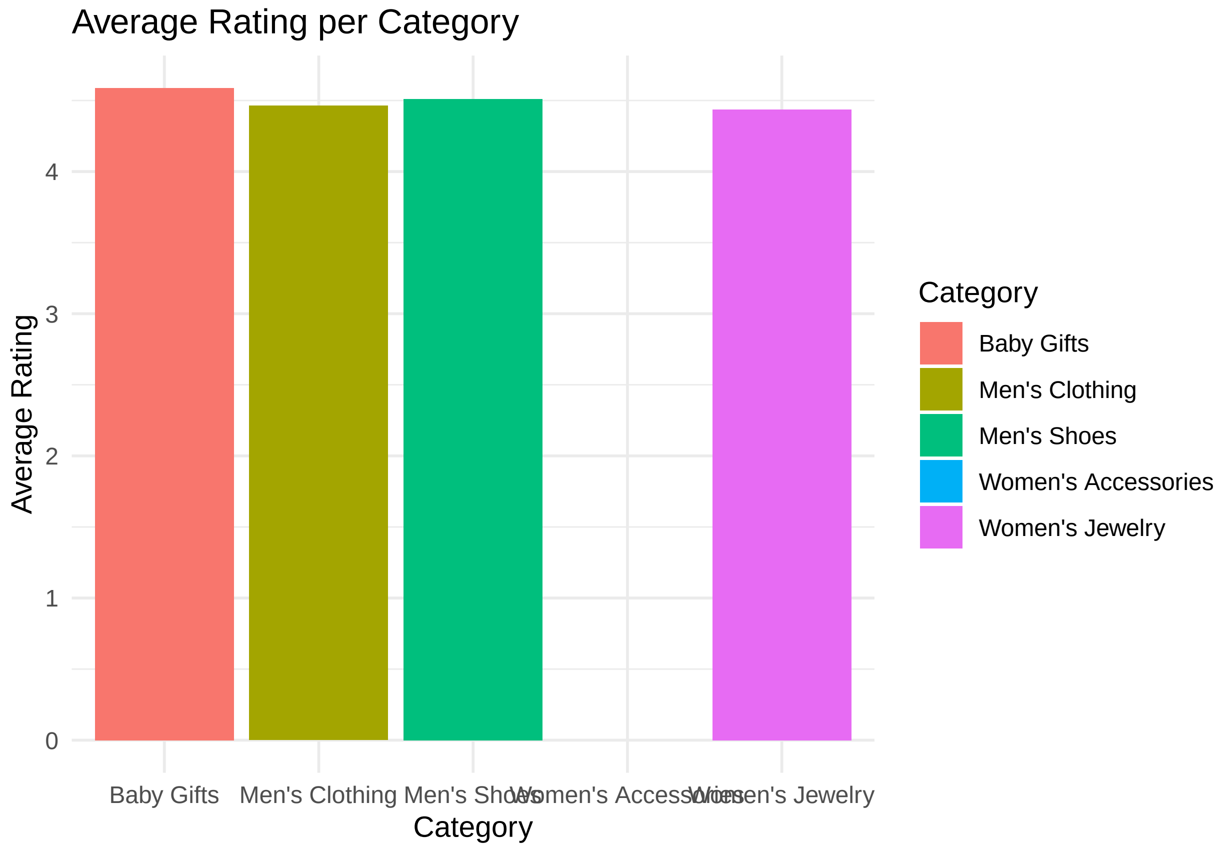
```
#5
df <- list()

for (i in seq_along(urls)) {
  # Read the HTML content of the page
  page <- read_html(urls[i])

  product_name <- page %>%
    html_nodes('h2.a-size-mini') %>%
    html_text(trim = TRUE) %>%
    head(30)

  product_description <- page %>%
    html_nodes('div.productDescription') %>%
    html_text(trim = TRUE) %>%
    head(30)

  product_rating <- page %>%
    html_nodes('span.a-icon-alt') %>%
    html_text(trim = TRUE) %>%
    head(30)
  ratings <- as.numeric(str_extract(product_rating, "\\d+\\.\\d"))

  product_price <- page %>%
    html_nodes('span.a-price') %>%
    html_text(trim = TRUE) %>%
    head(30)
  price <- as.numeric(str_extract(product_price, "\\d+\\.\\d+"))

  product_review <- page %>%
    html_nodes('div.review-text-content') %>%
    html_text(trim = TRUE) %>%
    head(30)

  dfTemp <- data.frame(Product_Name = product_name[1:30],
                       Description = product_description[1:30],
                       Rating = ratings[1:30],
                       Price = price[1:30],
                       stringsAsFactors = FALSE)

  df[[i]] <- dfTemp
}
```

#6. #The code extracts data from Amazon product listing pages based on different search queries, such as “men’s clothing,” “men shoes,” “women jewelry,” “baby gifts,” and “women accessories.” For each URL, the following information is extracted: Product Name along with its description(if available), Rating, and Price.

#7 #This data can be used to compare product popularity, analyze price trends, examine the relationship between price and quality, and conduct market research to inform new product development in each category.

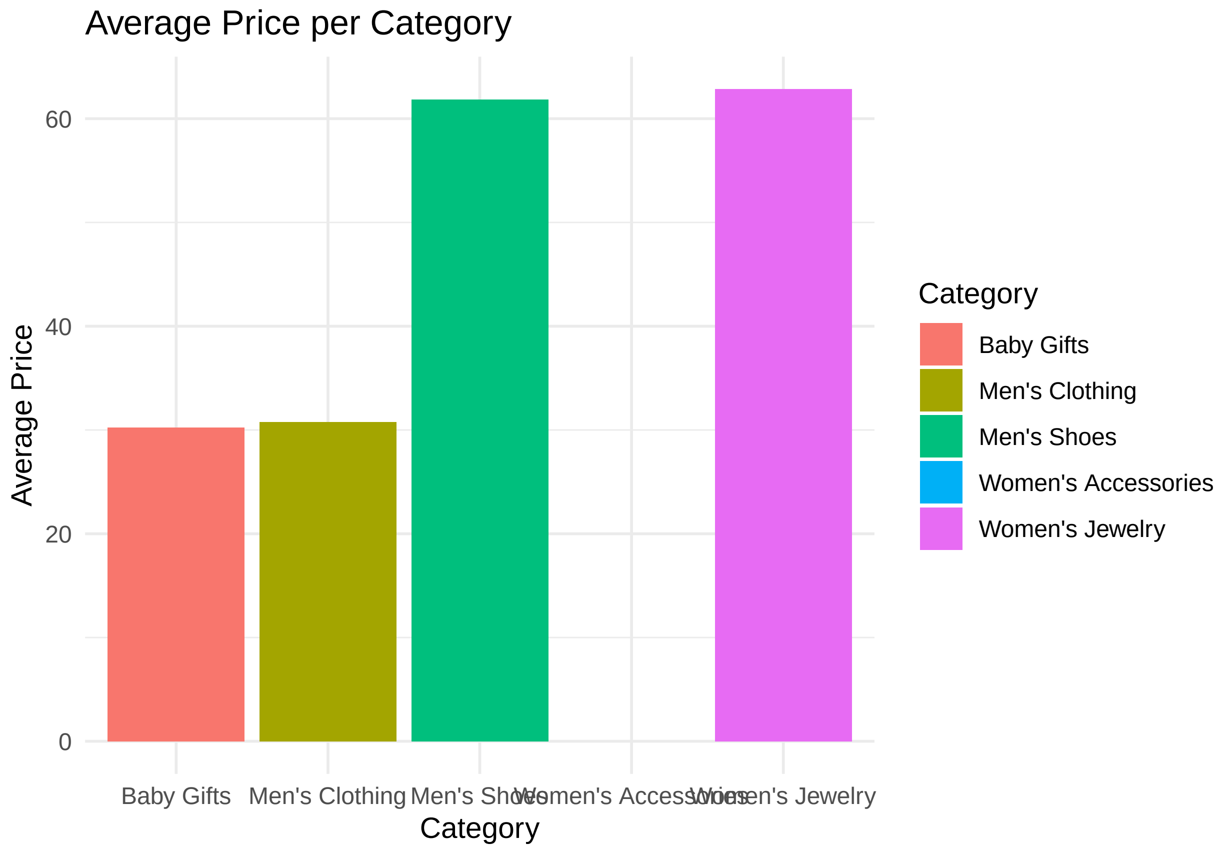
```
#8
combined_df <- do.call(rbind, df)
combined_df$Category <- rep(c("Men's Clothing", "Men's Shoes", "Women's Jewelry", "Baby Gifts", "Women's

avg_rating <- combined_df %>%
  group_by(Category) %>%
  summarize(Average_Rating = mean(Rating, na.rm = TRUE))

ggplot(avg_rating, aes(x = Category, y = Average_Rating, fill = Category)) +
  geom_bar(stat = "identity") +
  labs(title = "Average Rating per Category", x = "Category", y = "Average Rating") +
  theme_minimal()
```

```
## Warning: Removed 1 row containing missing values or values outside the scale range
## (`geom_bar()`).
```



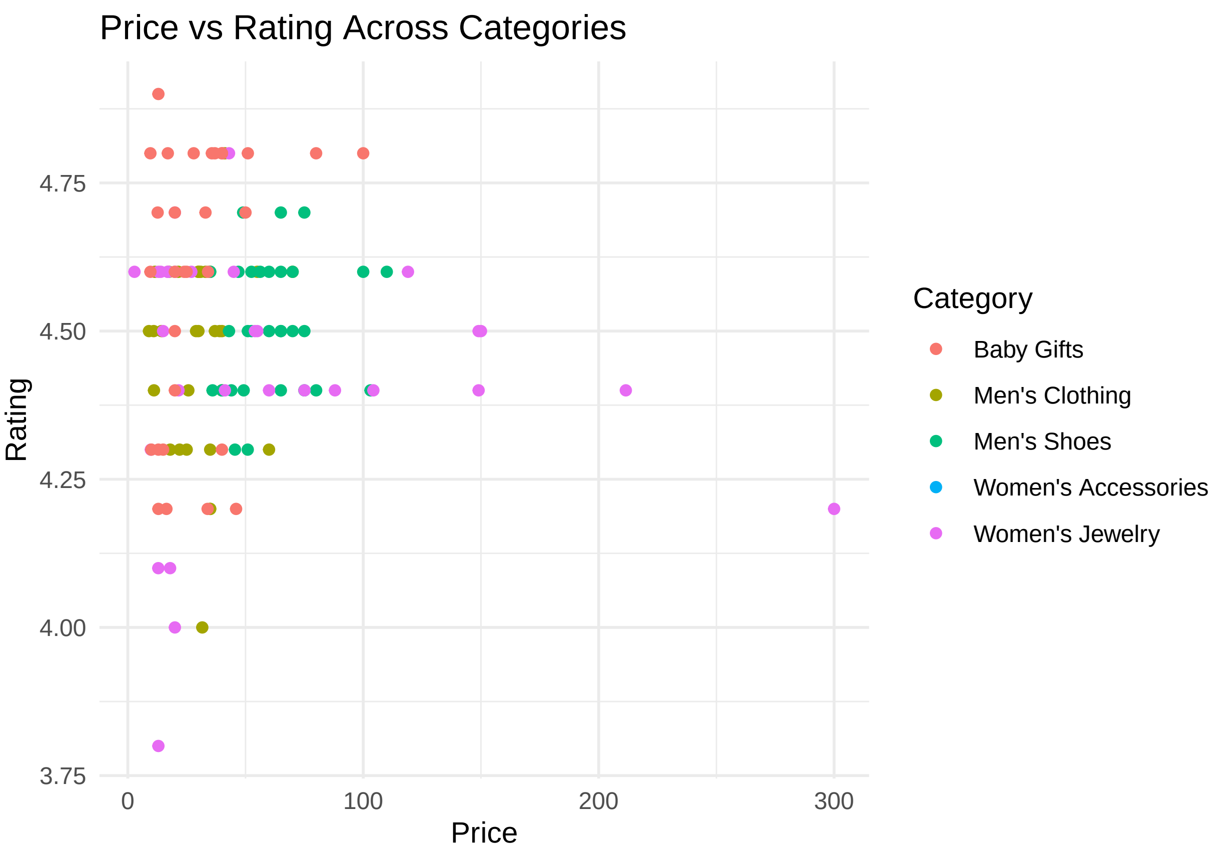
```
avg_price <- combined_df %>%
  group_by(Category) %>%
  summarize(Average_Price = mean(Price, na.rm = TRUE))

ggplot(avg_price, aes(x = Category, y = Average_Price, fill = Category)) +
  geom_bar(stat = "identity") +
  labs(title = "Average Price per Category", x = "Category", y = "Average Price") +
  theme_minimal()
```

```
## Warning: Removed 1 row containing missing values or values outside the scale range
## (`geom_bar()`).
```



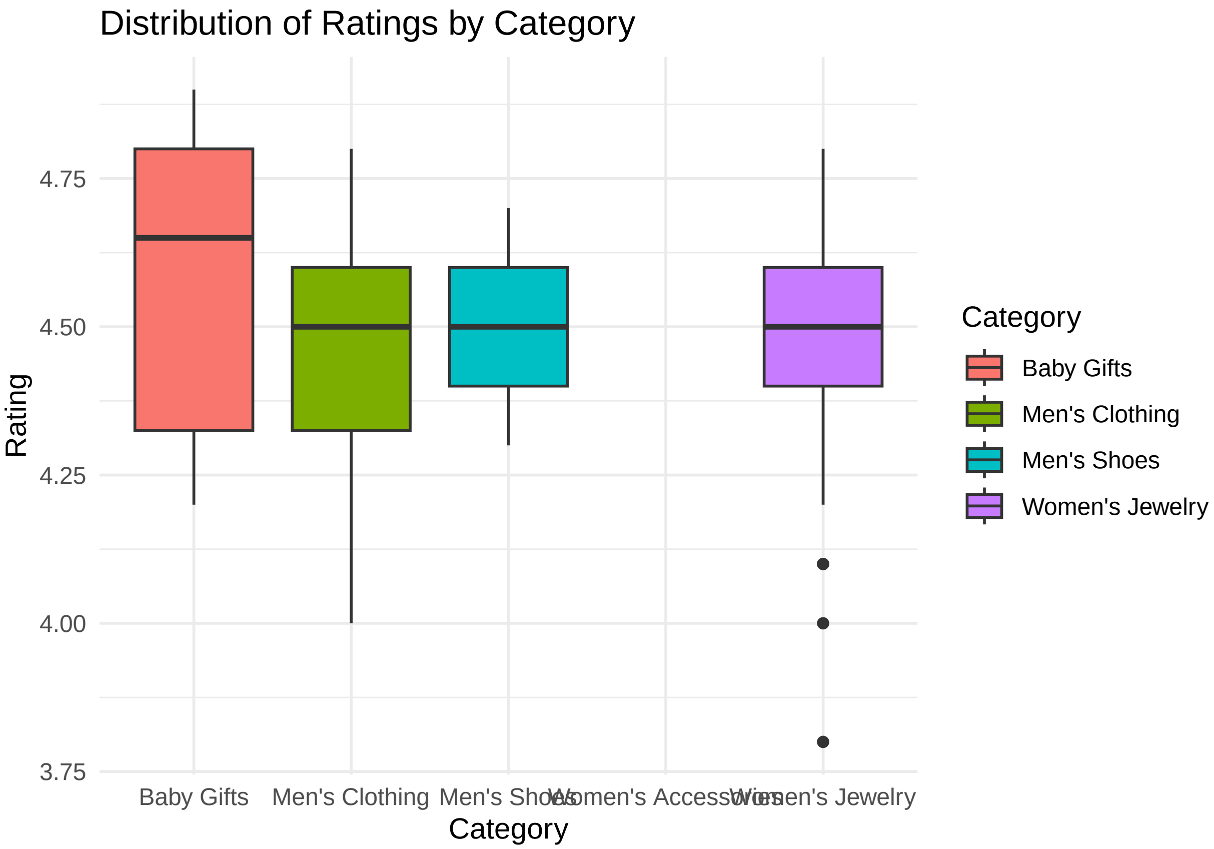
```r
ggplot(combined_df, aes(x = Price, y = Rating, color = Category)) +
  geom_point() +
  labs(title = "Price vs Rating Across Categories", x = "Price", y = "Rating") +
  theme_minimal()
```

```
## Warning: Removed 30 rows containing missing values or values outside the scale range
## (`geom_point()`).
```



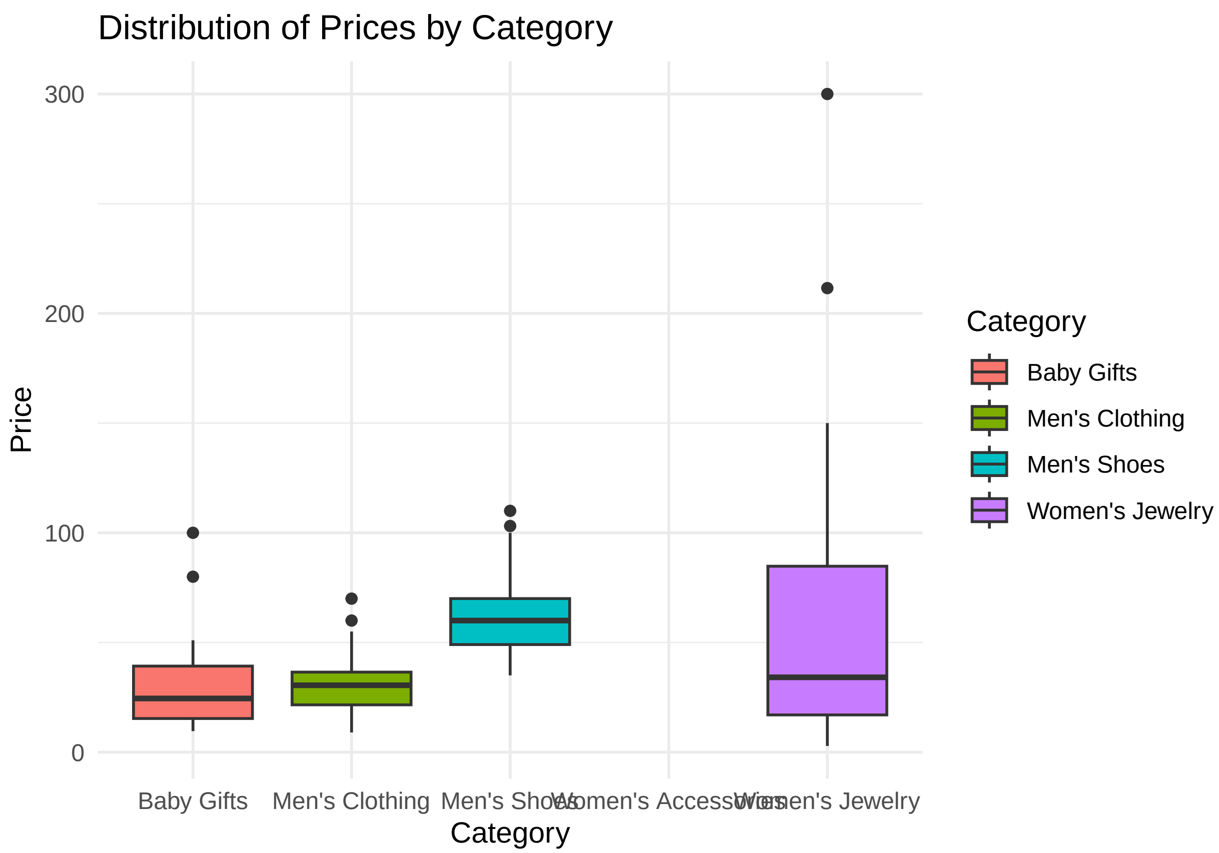
```r
#9
ggplot(combined_df, aes(x = Category, y = Rating, fill = Category)) +
  geom_boxplot() +
  labs(title = "Distribution of Ratings by Category", x = "Category", y = "Rating") +
  theme_minimal()
```

```
## Warning: Removed 30 rows containing non-finite outside the scale range
## (`stat_boxplot()`).
```



```
ggplot(combined_df, aes(x = Category, y = Price, fill = Category)) +
  geom_boxplot() +
  labs(title = "Distribution of Prices by Category", x = "Category", y = "Price") +
  theme_minimal()
```

```
## Warning: Removed 30 rows containing non-finite outside the scale range
## (`stat_boxplot()`).
```



```
#10
ranked_data <- lapply(df, function(df_category) {
  df_category %>%
    arrange(desc(Rating), Price) %>%
    mutate(Rank = row_number()) %>%
    select(Rank, everything())
})

categories <- c("Men's Clothing", "Men's Shoes", "Women's Jewelry", "Baby Gifts", "Women's Accessories"
for (i in seq_along(ranked_data)) {
  ranked_data[[i]]$Category <- categories[i]
}

ranked_combined_df <- do.call(rbind, ranked_data)
ranked_combined_df <- ranked_combined_df %>%
  arrange(Category, Rank) %>%
  group_by(Category) %>%
  slice(1:5)

print(ranked_combined_df)
```

```
## # A tibble: 25 x 6
## # Groups:   Category [5]
##     Rank Product_Name                                Description Rating Price Category
##    <int> <chr>                                       <chr>        <dbl> <dbl> <chr>
##  1     1 Pbooo                                       <NA>           4.9 13.0  Baby Gi~
##  2     2 Baby Books 0-6 Months,Infant Tummy T~ <NA>           4.8  9.59 Baby Gi~
##  3     3 4-in-1 Kickin' Tunes Music and Langu~ <NA>           4.8 17.0  Baby Gi~
##  4     4 beiens                                <NA>           4.8 28.0  Baby Gi~
##  5     5 Nasal Aspirator for Baby, Baby Nose ~ <NA>           4.8 35.7  Baby Gi~
##  6     1 Carhartt                              <NA>           4.8 41.2  Men's C~
##  7     2 Hanes                                 <NA>           4.6 11.5  Men's C~
##  8     3 Men's Quarter Button Sweater Lightwe~ <NA>           4.6 17.5  Men's C~
##  9     4 COOFANDY                              <NA>           4.6 21.4  Men's C~
## 10     5 Mens Hooded Sweatshirt Long Sleeve S~ <NA>           4.6 30.0  Men's C~
## # i 15 more rows
```