Final Project Part 1

Nikhil Dahiya 301449046

2023-03-23

Web Scraping Tutorial in R

Introduction

Web scraping, web harvesting, or web data extraction is data scraping used for extracting data from websites. In this tutorial I will be dicussing how to perform web scraping using R, specifically with the rvest and httr packages. This tutorial is divided into the following sections:

1)Setting up the environment
2)Inspecting a webpage
3)Scraping basic elements
4)Scraping tables
5)Scraping multiple pages

By the end of this tutorial, you should have a good understanding of web scraping with R and be able to extract data from various websites.

1.Setting up the environment

Before we start, please ensure that you have R installed on your computer. If you need to install R, follow the instructions at https://cran.r-project.org/ (https://cran.r-project.org/)

Then, install the required packages. You can do this by running the following command in your R console:

```
# Choose a CRAN mirror
options(repos = c(CRAN = "https://cloud.r-project.org/"))
install.packages(c("rvest", "httr", "magrittr", "tidyverse","janitor"))
```

```
##
## The downloaded binary packages are in
##  /var/folders/1z/vdhl23gd035cj_9m8zyjkg6w0000gn/T//RtmpbqanrG/downloaded_packages
```

Once installed, load the packages:

```
suppressPackageStartupMessages({
library(rvest)#rvest helps you scrape (or harvest) data from web pages
library(httr)#httr is to provide a wrapper for the curl package
library(magrittr)#magrittr provides the pipe
library(tidyverse)#tidyverse helps to transform and better present data.
})
```

2.Inspecting a webpage

Before you start scraping, it's essential to inspect the webpage's structure. This will help you identify the elements you want to extract. You can inspect a webpage using your browser's developer tools. Right-click on the element you want to scrape and select "Inspect" or "Inspect Element" to open the developer tools.

In this tutorial, we will scrape the top 10 movies from the IMDb Top 250 list (https://www.imdb.com/chart/top/ (https://www.imdb.com/chart/top/)). To identify the elements we need, we'll inspect the webpage.

3.Scraping basic elements

After inspecting the IMDb Top 250 list, we can see that the movie information is contained in a table with the class chart.

3.1 Reading a webpage

To start, we need to read the webpage:

```
url <- "https://www.imdb.com/chart/top/"
webpage <- read_html(url)#read_html command creates an R object, basically a list, th
at stores information about the web page
```

3.2 Scrapping the table

```
imdb_table <- webpage %>%
  html_nodes("table.chart") %>%
  html_table(header = TRUE)
imdb_table <- imdb_table[[1]]
#This code scrapes a table with the class "chart" from a webpage and stores it in a d
ata frame called imdb_table. The first line extracts the table from the webpage, whil
e the second line converts it to a data frame.
```

3.3 Extract movie titles, release years, and ratings

Now, we will extract the movie titles, release years, and ratings from the table:

```
titles <- head(imdb_table$`Rank & Title`, 10)
years <- head(gsub("\\D", "", imdb_table$`IMDb Rating`), 10)
ratings <- head(imdb_table$`Your Rating`, 10)
#This code extracts the top 10 values from three columns in the imdb_table data fram
e: "Rank & Title" (titles), "IMDb Rating" with non-digit characters removed (years),
and "Your Rating" (ratings).
```

3.4 Combining the scraped data

We can now combine the movie titles, release years, and ratings into a data frame:

```
movies_df <- tibble(
  Title = titles,
  ReleaseYear = years,
  Rating = ratings
)
movies_df
```

```
## # A tibble: 10 × 3
##    Title                                                        Relea…¹ Rating
##    <chr>                                                        <chr>   <chr>
##  1 "1.\n      The Shawshank Redemption\n        (1994)"          92      "1234…
##  2 "2.\n      The Godfather\n        (1972)"                    92      "1234…
##  3 "3.\n      The Dark Knight\n        (2008)"                  9       "1234…
##  4 "4.\n      The Godfather Part II\n        (1974)"            9       "1234…
##  5 "5.\n      12 Angry Men\n        (1957)"                    9       "1234…
##  6 "6.\n      Schindler's List\n        (1993)"                89      "1234…
##  7 "7.\n      The Lord of the Rings: The Return of the King\n   … 89      "1234…
##  8 "8.\n      Pulp Fiction\n        (1994)"                    88      "1234…
##  9 "9.\n      The Lord of the Rings: The Fellowship of the Ring\… 88      "1234…
## 10 "10.\n      Il buono, il brutto, il cattivo\n        (1966)"  88      "1234…
## # … with abbreviated variable name ¹ReleaseYear
```

```
#This code creates a new data frame (movies_df) with three columns: "Title", "Release Year", and "Rating", using the extracted top 10 values from titles, years, and ratings. It then prints the data frame.
```

4.Scraping tables

In this section, we will scrape an entire table. We will use the Wikipedia page for the list of countries by population (https://en.wikipedia.org/wiki/List_of_countries_and_dependencies_by_population (https://en.wikipedia.org/wiki/List_of_countries_and_dependencies_by_population)) as an example.

4.1 Reading a webpage

First, read the webpage:

```
url <- "https://en.wikipedia.org/wiki/List_of_countries_and_dependencies_by_population"
webpage <- read_html(url)
```

4.2 Identifying the table

Inspect the webpage and identify the table's XPath or CSS selector. In our case, the table has a class called wikitable.

4.3 Extracting the table

We can now extract the table using the html_table() function:

```
population_table <- webpage %>%
  html_nodes("table.wikitable") %>%
  html_table()
population_table <- population_table[[1]]
#This code scrapes a table with the class "wikitable" from a webpage and stores it in a data frame called population_table. The first line extracts the table from the webpage, while the second line converts it to a data frame.
```

4.4 Cleaning the table

First, we need to install and load the janitor package to clean the column names and then read the webpage.

```
library(janitor)
```

```
##
## Attaching package: 'janitor'
```

```
## The following objects are masked from 'package:stats':
##
##     chisq.test, fisher.test
```

```
url <- "https://en.wikipedia.org/wiki/List_of_countries_and_dependencies_by_population"
webpage <- read_html(url)

population_table <- webpage %>%
  html_nodes("table.wikitable") %>%
  html_table()
population_table <- population_table[[1]]
```

Now, use the clean_names() function from the janitor package to clean the column names:

```
population_table <- population_table %>%
  janitor::clean_names() %>%
  select(-rank)
#This code cleans the column names in population_table using the janitor::clean_names() function, and then removes the "rank" column using the select(-rank) function from the dplyr package.
```

Next, clean the population column by removing unnecessary characters and converting it to a numeric data type:

```
population_table$population <- gsub("\\[.*\\]", "", population_table$population)
population_table$population <- gsub(",", "", population_table$population)
population_table$population <- as.numeric(population_table$population)
#This code removes square brackets and their contents, as well as commas, from the population column in population_table. Then, it converts the population column to numeric data type.
```

4.5 Displaying the table

Display the cleaned table:

```
head(population_table)
```

```
## # A tibble: 6 × 6
##   country_dependency   population population_2   date        source_offi…¹ notes
##   <chr>                     <dbl> <chr>          <chr>       <chr>         <chr>
## 1 Country / Dependency         NA % of the world Date        Source (offi… "Not…
## 2 World                10214404000 100%          26 Mar 2023 UN projectio… ""
## 3 China                 1411750000 17.6%         31 Dec 2022 Official est… "Ref…
## 4 India                 1388163000 17.3%         1 Mar 2023  Official pro… "Inc…
## 5 United States          334544000 4.17%         26 Mar 2023 National pop… "Inc…
## 6 Indonesia              275773800 3.44%         1 Jul 2022  Official est… ""
## # … with abbreviated variable name ¹source_official_or_from_the_united_nations
```

```
#This code displays the first six rows of the population_table data frame.
```

5.Scraping multiple pages

In this section, we will demonstrate how to scrape multiple pages. We will use a simple example of scraping quotes from https://www.goodreads.com/list/show/1.Best_Books_Ever (https://www.goodreads.com/list/show/1.Best_Books_Ever) . The website has multiple pages, and we will scrape the book titles and author names from the first five pages of the “Best Books Ever” list on Goodreads.

5.1 Function to extract book details

```
# Function to extract book details from a given URL
scrape_books <- function(url) {
  # Read the HTML content of the URL
  webpage <- read_html(url)

  # Extract book titles
  titles <- webpage %>%
    html_nodes(".bookTitle") %>%
    html_text(trim = TRUE)

  # Extract author names
  authors <- webpage %>%
    html_nodes(".authorName") %>%
    html_text(trim = TRUE)

  # Ensure equal length of titles and authors
  min_length <- min(length(titles), length(authors))

  return(data.frame(Title = titles[1:min_length], Author = authors[1:min_length], str
ingsAsFactors = FALSE))
}
```

5.2 Scraping multiple pages

```
# Initialize an empty data frame to store the results
all_books <- data.frame(Title = character(), Author = character(), stringsAsFactors =
FALSE)

# Scrape books from the first 3 pages
for (i in 1:3) {
  url <- paste0("https://www.goodreads.com/list/show/1.Best_Books_Ever?page=", i)
  books <- scrape_books(url)
  all_books <- rbind(all_books, books)
}
```

5.3 Displaying the result

```
# Print the scraped data
#print(all_books)
head(all_books)# I used the head function to limit the output displayed to the first
6 lines, as the print function was producing too many lines of results, approximately
34 pages.
```

```
##                                                          Title          Author
## 1                      The Hunger Games (The Hunger Games, #1) Suzanne Collins
## 2 Harry Potter and the Order of the Phoenix (Harry Potter, #5)    J.K. Rowling
## 3                                           Pride and Prejudice     Jane Austen
## 4                                        To Kill a Mockingbird      Harper Lee
## 5                                              The Book Thief    Markus Zusak
## 6                               Twilight (The Twilight Saga, #1) Stephenie Meyer
```

Conclusion

In this tutorial, we learned how to perform web scraping using R and the rvest and httr packages. We covered setting up the environment, inspecting a webpage, scraping basic elements, scraping tables, and scraping multiple pages. By now, you should have a good understanding of web scraping with R and be able to extract data from various websites.

Citations

https://www.zenrows.com/blog/web-scraping-r#step-1-install-rvest (https://www.zenrows.com/blog/web-scraping-r#step-1-install-rvest)

https://brightdata.com/blog/how-tos/web-scraping-with-r (https://brightdata.com/blog/how-tos/web-scraping-with-r)

https://medium.com/@Bwhiz/how-to-scrape-tables-from-multiple-pages-using-the-pandas-library-b1ea4f91d764 (https://medium.com/@Bwhiz/how-to-scrape-tables-from-multiple-pages-using-the-pandas-library-b1ea4f91d764)

http://uc-r.github.io/scraping_HTML_tables (http://uc-r.github.io/scraping_HTML_tables)