What is Web Scraping?

Web scraping describes the performance of extracting data from lines of code generated by HTML, CSS, and JavaScript. It is done using a coding language to automate the process.

A webpage is rendered using 3 types of instructions listed below:

HTML (HyperTextMarkupLanguage): describes the web page's infrastructure (heading, paragraphs, etc)

CSS (Cascading Style Sheets): defines the appearance (color, formatting, etc)

JavaScript: decides the behaviour of the page

Having basic knowledge of HTML and CSS is helpful when web scraping, thus we will first begin with a quick run through of how they work.

HTML

Here is an example of an HTML file:

```
<!DOCTYPE html>
<html lang="en">
<body>

<h1 href="https://en.wikipedia.org/wiki/NewJeans"> NewJeans </h1>
<h2> History </h2>
<p> Prior to debuting with NewJeans, several group members were involved in television, music and dance. </p>
<h2> Artistry </h2>
<p> NewJeans music spans genres such as <b>R&B, electropop, and hip hop.</b> </p>

</body>
</html>
```

Lets go over this step by step

Tags are the core of HTML. They have an important role of telling the computer the nature of the codes written inside the tags. For example, `h1` represents heading 1. There are two types of tags:

- **Block tags** : They're used for the overall structure of the page. Examples are <h1> for headings and <p> for paragraphs.
- **Inline tags** : They are used inside block tags to format the text. Examples are <b> for bolding and <i> for italics

Both tags have a starting tag that looks like <h> and an ending tag, </h> to let the program know the start and end of each tag.

You may add optional *attributes* to adjust the behaviour. In the sample code, `href=` is an attribute used to specify that the next line of code is a URL and to present it as a hyperlink.

CSS

Since the output of the HTML code is rather bland, **CSS** is used to stylize the website by adding color, changing the font size, and many more.

CSS selectors are important to understand. Among them, the `.class` selector is the most important. It selects all the elements in the same class. So `.xyz` will select all elements with `class = "xyz"`.

The function `html_nodes` from the `rvest` package (more on that later) may require a CSS selector to find certain elements.

Web Scraping in R

rvest

`rvest` is the most common package used for web scraping in R. Ensure that you have it installed like so:

```
install.packages("rvest")
```

`rvest` allows you to access a web page and elements using CSS selectors and XPath. We will focus on CSS selectors for this tutorial. There are many overlaps with the Tidyverse library such as the use of the pipe operator.

To use the package, we must load the package:

```
library(rvest)
library(tidyverse)
```

Getting data

To get access to the data, we will use `read_html`. In this tutorial, we will be web scraping Amazon's kpop album selection and their prices.

```
link <- "https://www.amazon.ca/s?k=albums+kpop&i=todays-deals&crid=1XCRYSR5O3O65&sprefix=albums+kpop%2Ctodays-deals%2C125&ref=nb_sb_noss_1"
amazon <- read_html(link)

amazon
```

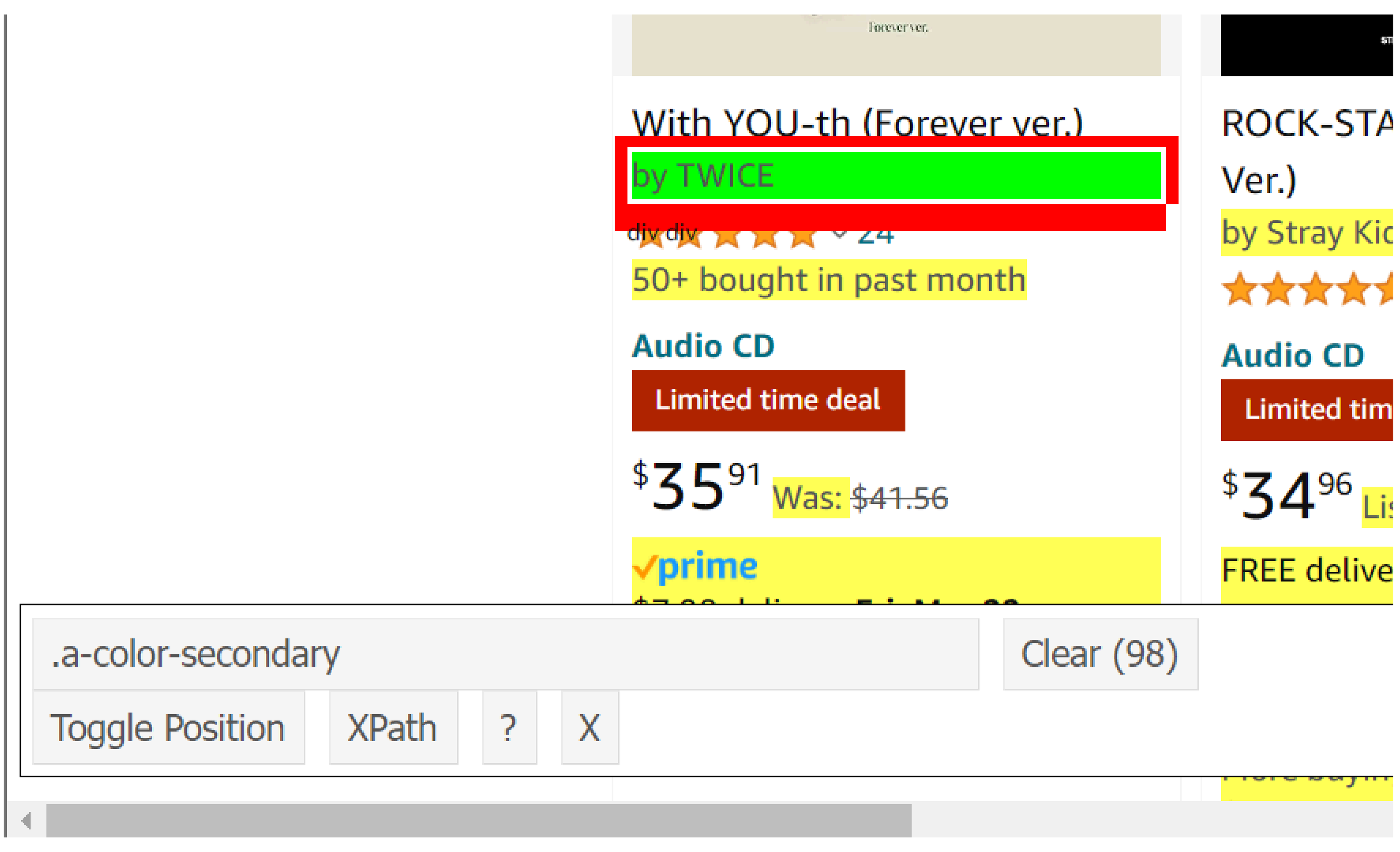
```
## {html_document}
## <html lang="en-ca" class="a-no-js" data-19ax5a9jf="dingo">
## [1] <head>\n<meta http-equiv="Content-Type" content="text/html; charset=UTF-8 ...
## [2] <body class="a-aui_72554-c a-aui_a11y_6_837773-c a-aui_killswitch_csa_log ...
```

`read_html` function read and parses through HTML content given by a URL. It then returns an `html_document` type which lets us use the data from the site for web scraping.

Parsing HTML content

To make it easier to find selectors of a website, downloading the chrome extension *SelectorGadget* is recommended. This is a user friendly way of accessing the CSS selectors.

After running the extension, you will notice that it will highlight texts and images as you hover over the website. To access the selector, simply click on the desired text and copy the selector given at the bottom of the screen. In the screenshot below, the selector is **`.a-color-secondary`**. This indicates that the text "by TWICE" was stylized using the CSS class `.a-color-secondary`.



If for some reason you cannot use SelectorGadget, you may right click anywhere on the website, scroll down to inspect, and it will open a separate tab that shows the code for the website.

Getting the album names and prices

```
album <- amazon %>% html_nodes(".a-color-base.a-text-normal") %>% html_text()

dollar <- amazon %>% html_nodes(".a-price-whole") %>% html_text()

cents <- amazon %>% html_nodes(".a-price-fraction") %>% html_text()

head(album)
```

```
## [1] "NewJeans 2nd EP 'Get Up' (Bunny Beach Bag Ver.)"
## [2] "Skool Luv Affair Special Addition (Cd/2Dvd)"
## [3] "Echo [III]"
## [4] "5th Mini Album Hot (Tin Case Ver.)"
## [5] "TWICE 14th Mini Album [STRATEGY] Digipack STEP4 Version (Random 1ea) +Photocard + Mini Postcard"
## [6] "JENNIE The 1st Studio ExtraL Album [Ruby] Digipack Version"
```

```
head(dollar)
```

```
## [1] "41." "54." "39." "55." "18." "22."
```

```
head(cents)
```

```
## [1] "29" "00" "68" "67" "99" "00"
```

`html_nodes`

This function pulls out the elements that uses the css selector, in this case, it is *.headline* we type inside the parameters. This returns all the nodes which looks something like
<h2 class="a-color-base headline truncate-1line gwm-u-blackjack-typography">Today's deals</h2>.

`html_text`

This extracts the text from the nodes selected using `html_nodes`. The result parses through the nodes to only return the text, which looks like the given output above.

To make it the prices look nicer, we will combine the dollars and cents together by making a data frame.

```
prices = data.frame(dollars = as.integer(dollar), cents = as.integer(cents))

prices$total = prices$dollars + ((prices$cents)/100)
```

Now lets add the album name to our data frame:

```
Album = data.frame(Name = album, Price = prices$total, stringsAsFactors = FALSE)

head(Album)
```

```
##                                                                                           Name
## 1                                              NewJeans 2nd EP 'Get Up' (Bunny Beach Bag Ver.)
## 2                                                  Skool Luv Affair Special Addition (Cd/2Dvd)
## 3                                                                                   Echo [III]
## 4                                                           5th Mini Album Hot (Tin Case Ver.)
## 5 TWICE 14th Mini Album [STRATEGY] Digipack STEP4 Version (Random 1ea) +Photocard + Mini Postcard
## 6                                   JENNIE The 1st Studio ExtraL Album [Ruby] Digipack Version
##   Price
## 1 41.29
## 2 54.00
## 3 39.68
## 4 55.67
## 5 18.99
## 6 22.00
```

Almost done!! Saving the data frame to a table

We will use the function `write.table` to save it to a csv

```
write.table(Album, "Albums.csv", sep= ",", row.names = F, col.names= F)
```

You've reached the end of the tutorial. Thank you for reading!

Citations

Soetewey, A. (2023). Web scraping in R. Stats and R. https://statsandr.com/blog/web-scraping-in-r/

Wickham H., Çetinkaya-Rundel, M., Grolemund, G (2023. 24 Web scraping. In R for Data Science (2e). https://r4ds.hadley.nz/webscraping

Dataslice (2020, May 10). Web Scrape Text from ANY Website - Web Scraping in R (Part 1) [Video]. YouTube. https://www.youtube.com/watch?v=v8Yh_4oE-Fs