web scraper guide

brayden look

Let's say we want to scrape data from Steph Curry's basketball reference page: "https://www.basketball-reference.com/players/c/curryst01.html", specifically the table labeled "Advanced".

The first thing we need to do is understand some basic html/CSS concepts. Every website can be broken down into a tree of things called "nodes" (or "elements"). There are two top-level nodes called the **head** and the **body** from which every other node branches from. The body contains everything we care about in webscraping, so we'll focus there.

Start by going to the basketball reference URL and pressing F12 on your computer (or F_n + F12 on some laptops) to open up the Developer Tools page. This should open up a page on the side of your browser with a bunch of information. Make sure that the "Elements" tab is selected at the top. You should see a bunch of html code, but mainly you should see something labeled "<head> ... </head>" and something labeled "<body> ... </body>". You may see extra information on different types of browsers.

First off, if you hover over **body**, it should highlight the entire page. This is telling you that everything you see on the page is contained inside the **body** node. Click the arrow next to body to open up its branches. Play around with this and open other sub-branches and hover over different elements to see what they highlight on the webpage.

Now on the actual webpage, scroll down to the table that says "Advanced". We want to find the node that corresponds to this table so that we can scrape the data from it. A quick shortcut to find the right node is to highlight something, right click it, and click on **inspect**. Let's do that with the table header that says "Advanced." Highlight it, right click it, and select **inspect**. This should take you directly to the element that corresponds to it in the Elements page.

Now navigate a few lines down on the Elements tab of the Developer Tools to

Click the arrow to open its sub elements, and then go down to

<div class = "table_container tabbed current is_set up", id = "div_advanced">

Under this you'll see something labeled "<table ... >". Hover over this and see that it highlights the table that we want. This is the **node** that we would like to target. Now in R, (after installing the **rvest** and **httr** libraries), run this code chunk:

```
url <- "https://www.basketball-reference.com/players/c/curryst01.html"
headers = "Mozilla/5.0 (Windows NT 10.0; Win64; x64)
            AppleWebKit/537.36 (KHTML, like Gecko)
            Chrome/105.0.0.0 Safari/537.36"

#point at the webpage
document <- GET(url, user_agent(headers))

#access the document and make it readable
html <- document %>%
    read_html()
```

```
#use the table id to access a specific element
table <- html %>% html_node("#advanced")

#convert the element into a dataframe
advanced_data <- table %>% html_table()

advanced_data
```

```
## # A tibble: 15 x 29
##    Season    Age Tm    Lg    Pos       G    MP   PER `TS%` `3PAr`   FTr `ORB%`
##    <chr>   <int> <chr> <chr> <chr> <int> <int> <dbl> <dbl>  <dbl> <dbl>  <dbl>
##  1 2009-10    21 "GSW" NBA   "PG"     80  2896  16.3 0.568  0.332 0.175    1.8
##  2 2010-11    22 "GSW" NBA   "PG"     74  2489  19.4 0.595  0.325 0.216    2.3
##  3 2011-12    23 "GSW" NBA   "PG"     26   732  21.2 0.605  0.409 0.159    2.3
##  4 2012-13    24 "GSW" NBA   "PG"     78  2983  21.3 0.589  0.432 0.21     2.3
##  5 2013-14    25 "GSW" NBA   "PG"     78  2846  24.1 0.61   0.445 0.252    1.8
##  6 2014-15    26 "GSW" NBA   "PG"     80  2613  28   0.638  0.482 0.251    2.4
##  7 2015-16    27 "GSW" NBA   "PG"     79  2700  31.5 0.669  0.554 0.25     2.9
##  8 2016-17    28 "GSW" NBA   "PG"     79  2638  24.6 0.624  0.547 0.251    2.7
##  9 2017-18    29 "GSW" NBA   "PG"     51  1631  28.2 0.675  0.58  0.35     2.7
## 10 2018-19    30 "GSW" NBA   "PG"     69  2331  24.4 0.641  0.604 0.214    2.2
## 11 2019-20    31 "GSW" NBA   "PG"      5   139  21.7 0.557  0.598 0.317    3
## 12 2020-21    32 "GSW" NBA   "PG"     63  2152  26.3 0.655  0.587 0.289    1.5
## 13 2021-22    33 "GSW" NBA   "PG"     64  2211  21.4 0.601  0.613 0.243    1.7
## 14 2022-23    34 "GSW" NBA   "PG"     56  1941  24.1 0.656  0.564 0.248    2.3
## 15 Career     NA ""    NBA   ""      882 30302  23.8 0.627  0.507 0.243    2.2
## # ... with 17 more variables: `DRB%` <dbl>, `TRB%` <dbl>, `AST%` <dbl>,
## #   `STL%` <dbl>, `BLK%` <dbl>, `TOV%` <dbl>, `USG%` <dbl>, `` <lgl>,
## #   OWS <dbl>, DWS <dbl>, WS <dbl>, `WS/48` <dbl>, `` <lgl>, OBPM <dbl>,
## #   DBPM <dbl>, BPM <dbl>, VORP <dbl>
```

That's it! Tables in HTML are really easy to deal with because rvest can turn them into readable dataframes with one line. Let's go through the code chunk to describe what it's doing in more detail.

- First, we run the GET function to point at the webpage. For now, don't worry about what the **headers** variable is doing for us.
- The **read_html** command actually crawls through and gets the html code from the website (basically everything that we saw in the Elements tab of Developer Tools).
- The **html_node** command allows us to access a particular element within the HTML. Inside we put "#advanced". The # symbol tells the function that we are looking for a unique id. If we go back to Developer Tools and look at the <table ...> element again, we'll see that it has an id inside labeled "id = 'advanced' ". Not every element will have an id, but if it does, it is guaranteed to be unique.
- Finally, the **html_table()** function just turns a table into a dataframe.

Let's explore how to access different elements. Let's say we want ALL of the tables from the page. Based on what we just did, we might think that we need to find the id's of each table and loop through each id. Fortunately, rvest makes it much easier:

```
#grab all tables from the page
table <- html %>% html_nodes("table")

#convert the element into a dataframe
all_data <- table %>% html_table()
```

Here we used **html_nodes** rather than **html_node**. **html_node** will only ever return one thing. **html_nodes** can return a list of multiple things. The argument we used was just "table". In English, this code is saying "Find every element labeled 'table' and return them all as a list." Note that we didn't use a # sign. This is because # is used to target a specific id, but we want ALL table elements, regardless of their id. If we ran **html_node**("table"), it would have returned the very first table it saw.

Now let's look at a slightly more complicated and realistic example. Let's say that we want to get the "advanced" data for every active player. First we need to see how players are stored in basketball reference. In the URL for Stephen Curry, we can see that basketball reference has a unique identifier for him "curryst01". Let's strip back that URL to just "https://www.basketball-reference.com/players/". Click on the letter "A" at the top. It takes us to a page with every player whose last name starts with "A". If we hover over a player in this table, it gives us the link to their page, which is what we want. Let's run this chunk to try to get it:

```
url <- "https://www.basketball-reference.com/players/a/"

#point at the webpage
document <- GET(url)

#access the document and make it readable
html <- document %>%
    read_html()

#use the table id to access a specific element
table <- html %>% html_node("table")

#convert the element into a dataframe
player_data <- table %>% html_table()

head(player_data)
```

```
## # A tibble: 6 x 8
##   Player                From    To Pos   Ht       Wt `Birth Date`      Colleges
##   <chr>                <int> <int> <chr> <chr> <int> <chr>             <chr>
## 1 Alaa Abdelnaby        1991  1995 F-C   6-10    240 June 24, 1968     Duke
## 2 Zaid Abdul-Aziz       1969  1978 C-F   6-9     235 April 7, 1946     Iowa Sta~
## 3 Kareem Abdul-Jabbar*  1970  1989 C     7-2     225 April 16, 1947    UCLA
## 4 Mahmoud Abdul-Rauf    1991  2001 G     6-1     162 March 9, 1969     LSU
## 5 Tariq Abdul-Wahad     1998  2003 F     6-6     223 November 3, 1974  Michigan~
## 6 Shareef Abdur-Rahim   1997  2008 F     6-9     225 December 11, 1976 Californ~
```

Since there's only one table on this page, we don't need to worry about an id and can just run **html_node**("table"). However, if we open up our dataframe, it only gave us raw text, not the hyperlink that was underneath each player! What do we do? Let's check the HTML code.

Highlight a name and inspect it. We can see that the link is stored inside an **<a>** element (which just means the element is text) labeled as **href** (which means that it is a hyperlink). This is an **attribute** of that text. But our table function earlier can't access the hyperlink underneath, so we have to be more

explicit. However, we don't even have a unique "id"! So what do we do? The following code chunk will do it. Run it, and then we'll go through what it's doing.

```
links <- html %>% html_node("table") %>% html_nodes("th") %>%
                    html_nodes("a") %>% html_attr("href")
```

The way to read this is from right to left. In English it says: Grab every hyperlink (attributes with **href**) of every text element (things labeled **<a>**) in the main columns (things labeled **<th>**) of the first table. Once it's written out, it's easier to see the logic.

However, we want ACTIVE players, not retired players. Notice that every active player is **bolded** in the table. From what we just did, think about how you would modify the code chunk above to only get active players. What element would you look for? Where in the chain of functions would you place it? Try it on your own before looking below.

Once we have the list of players, we want to loop through each link, grab each player's table of advanced stats, and then store it. We've covered everything needed to do that; the rest is just thinking about code logic and order of operations. Here is the code that will grab the advanced tables of a sample of players (if we run it for every player, it will take too long)

```
#get advanced stats for every player
url_base = "https://www.basketball-reference.com/players/"
headers = "Mozilla/5.0 (Windows NT 10.0; Win64; x64)
           AppleWebKit/537.36 (KHTML, like Gecko)
           Chrome/105.0.0.0 Safari/537.36"

links <- c()

#get all active player's with last names that start with A and store their links
for (letter in letters[1]){
  url = paste(url_base, letter, sep = "")
  document <- GET(url, user_agent(url))
  html <- document %>%  read_html()

  #find all links (hrefs) stored in text (a) that is bolded (strong) in the main column (th) of the tab
  links <- c(links, html %>% html_node("table") %>%
               html_nodes("th") %>%  html_nodes("strong") %>%
               html_nodes("a") %>% html_attr("href"))

  #pause for 4 seconds after each request so that we don't get kicked out
  Sys.sleep(4)
}

url_base = "https://www.basketball-reference.com"
data_frames = list()
#get the first 4 player's advanced data
for (link in links[1:4]){
  url = paste(url_base, link, sep = "")
  document <- GET(url)
  html <- document %>%  read_html()
  table <- html %>% html_node("#advanced")
  data <- table %>% html_table()

  data_frames <- append(data_frames, list(data))
```

```
  Sys.sleep(4)
}
```

data_frames is a list of dataframes of each player's advanced stats table, which is what we wanted. Note that in the code, we use the **Sys.sleep()** function, which pauses for 4 seconds. We do this after each request so that we don't overload the server and get kicked out. Basketball reference has a **rate limit** of one request every 3 seconds, but it's good to add an extra second just to be safe.

These are the basics of scraping. Every website is different and not every method will be straightforward, but these are most of the tools necessary to figure it out.