In September 2017, Its found working on a project that required odds data for football. At the time I didn’t know about resources such as [Football-Data](http://www.football-data.co.uk/) or the [odds-api](https://the-odds-api.com/), so I decided to build a scraper to collect data directly from the bookmakers. However, most of them used JavaScript to display their odds, so I couldn’t collect the data with R and *rvest* alone. In this article, I’ll demonstrate how PhantomJS can be used with R to scrape JS-rendered content from the web.

**Requirements**

In addition to R’s base packages, I’ll need the following for this example:

* [rvest](https://cran.r-project.org/package=rvest) – This package for R is a wrapper for *xml2* and *httr* packages. It allows you to download and extract data from HTML and XML.
* [PhantomJS](http://phantomjs.org/) – A headless browser that can be used to access webpages and extract information from them, among other things.
* [Selector Gadget](https://selectorgadget.com/) – This one is optional, but it makes life easier because it isn’t necessary to go through the source code to find the XPath and/or CSS selectors for extracting the information you need from an HTML document.

**Collecting Data**

For this example I’ll stick to the original intent of the project and collect odds from an online bookmaker. [Paddy Power](https://www.paddypower.com/bet) (no affiliation) uses JavaScript and also doesn’t have an API to my knowledge, so I’ll collect odds from their website.

**Without PhantomJS**

To collect odds data with *rvest*, we need to:

1. Read the HTML file from a URL.
2. Extract the HTML elements containing the odds. (The

elements that contain the data I am interested in have the attribute class="avb-item".)

1. Extract the contents of the elements.
2. Clean the data if necessary.

> withoutJS <- xml2::read\_html("https://www.paddypower.com/football?tab=today") %>%

+ rvest::html\_nodes(".avb-item") %>%

+ rvest::html\_text()

> withoutJS

character(0)

We ended up with an empty character vector because the HTML document contains only JS snippets in the source code instead of the content we see in the browser. That means the CSS selector we can see in the browser does not exist in the document R is trying to extract data from.

**With PhantomJS**

The first step is to save the following script in a separate file:

// scraper\_PaddyPower.js

// Create a webpage object

var page = require('webpage').create();

// Include the File System module for writing to files

var fs = require('fs');

// Specify source and path to output file

var url = 'https://www.paddypower.com/football?tab=today'

var path = 'odds\_PaddyPower.html'

page.open(url, function (status) {

var content = page.content;

fs.write(path,content,'w')

phantom.exit();

});

The purpose of this script is to retrieve the HTML file from the specified URL and store it into a local HTML file, so that R can read contents from that file instead of reading the contents directly from the URL.

We can then use the system() function to invoke the script from R and repeat the steps outlined in the attempt without PhantomJS to extract the data:

> system("/usr/local/phantomjs-2.1.1/bin/phantomjs scraper\_PaddyPower.js")

> withJS <- xml2::read\_html("odds\_PaddyPower.html") %>%

+ rvest::html\_nodes(".avb-item") %>%

+ rvest::html\_text()

***Note for Windows users****: The invocation of the script has the same format as the command for Linux, but you may have to add .exe to phantomjs for it to work.*

The content is currently a mess and the variable is one large chunk of text, i.e. a character vector of length 1. If I remove the white space and split it into individual elements, here’s what I get:

> pp\_odds\_data <- withJS %>%

+ gsub("[\n\t]", "", .) %>%

+ stringr::str\_trim() %>%

+ gsub("\\s+", " ", .) %>%

+ strsplit(" ") %>%

+ unlist()

> pp\_odds\_data[1:40]

[1] "Bahrain" "Tajikistan" "4/7" "13/5" "5/1" "17:00" "More"

[8] "Bets" ">" "Hatayspor" "Genclerbirligi" "1/14" "13/2" "19/1"

[15] "More" "Bets" ">" "Darica" "Genclerbirligi" "Antalyaspor" "4/1"

[22] "23/10" "4/7" "More" "Bets" ">" "Akhisar" "Belediye"

[29] "Fatih" "Karagumrukspor" "1/3" "10/3" "6/1" "16:00" "More"

[36] "Bets" ">" "Giresunspor" "Fenerbahce" "9/2"

There’s still some useless data mixed in, and there is no guarantee that I can turn this vector into a data frame to store and analyze the data. In this particular case, team names can have more than two words, so I should extract information about matches and odds separately:

> matches <- xml2::read\_html("odds\_PaddyPower.html") %>%

+ rvest::html\_nodes(".avb-item .ui-scoreboard-coupon") %>%

+ rvest::html\_text() %>%

+ stringr::str\_trim() %>%

+ gsub("\\s\\s+"," vs ", .)

> odds <- xml2::read\_html("odds\_PaddyPower.html") %>%

+ rvest::html\_nodes(".avb-item .btn-odds") %>%

+ rvest::html\_text() %>%

+ stringr::str\_trim()

> odds\_selector <- seq(1, length(odds), 3)

> pp\_odds\_df <- data.frame(Match = matches,

Home = odds[odds\_selector],

Draw = odds[(odds\_selector + 1)],

Away = odds[(odds\_selector + 2)])

> head(pp\_odds\_df)

Match Home Draw Away

1 Bahrain vs Tajikistan 4/7 13/5 5/1

2 Hatayspor vs Genclerbirligi 1/14 13/2 19/1

3 Darica Genclerbirligi vs Antalyaspor 4/1 23/10 4/7

4 Akhisar Belediye vs Fatih Karagumrukspor 1/3 10/3 6/1

5 Giresunspor vs Fenerbahce 9/2 11/4 1/2

6 Botosani vs Dunarea Calarasi 8/13 5/2 9/2

Now I can further transform the data before storing and analyzing it, e.g. convert fractional odds into decimal, split the Match column into “Home Team” and “Away Team” columns, and so on.

**Final Comments**

Although the script still works today, I should note that the development of PhantomJS is currently suspended and its [git repository](https://github.com/ariya/phantomjs) is archived. However, there are many ways to collect data from a JavaScript-rendered website, so you can also try using [*rvest* and *V8* from R](https://datascienceplus.com/scraping-javascript-rendered-web-content-using-r/) or [Python and Selenium](https://medium.com/@hoppy/how-to-test-or-scrape-javascript-rendered-websites-with-python-selenium-a-beginner-step-by-c137892216aa) to collect the data you need. Good luck!