

# Title: Web Data Scraping Method

Peijie Li

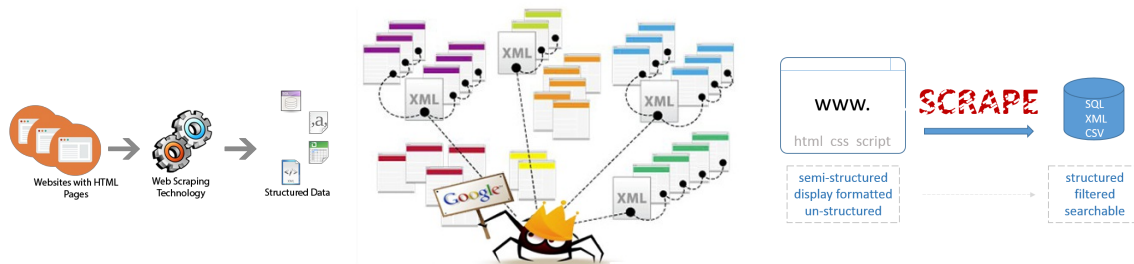
November 26, 2017

## Preface:

This post was originally finished on the week before the professor talked about `rvest` package, which happened to be the **same topic** I introduced originally. So even though the rest of the post was written as an extra knowledge at that time, due to the situation I am going to briefly compared `rvest` with another scraping web package called `XML` packages.

## Introduction:

We have been dealing with data set from the beginning of the semester and learned various ways to manipulate data: with `ggplot` package, `shiny` app and many other functions. However, the data set we were dealing with were pretty much “prepared data set”. Except the last lab where we learned how to clean a data set step by step, we were trained to play with specific data set. In real life, things do not present this neat. So, learning how to scraping data from a website is very useful and practical. In this post, you will be introduced with a new package called “**rvest**”, which was often used with scraping data from html website. Also you will see another web scraping `XML` package comparison. I will present how to scrape the *rate*, *cast* and *poster* of the movie: “The Big Hero 6” and compared with `XML`. I hope after reading this post, you can download and get the data set you want from different messy website and know the differences between these two packages!



## Motivation:

I was pretty struggled when I was trying to think which topic I want to do for the second post. Initially I was thinking to show how to make a map that will show all the ratings in `yelp` for resaurants in Los Angeles. But then I realize even though human can read the ratings easily from the yelp app or website, computers cannot do it by its own. Plus I don't know how to code to scape the ratings, so there's a lot to do before you really making the map. Therefore, I suddenly see the importance of scraping the data we want from a website. Without certain data, we cannot do the analysis and apply the functions we learned from the class. So with this practical problem, I started to search how to code to get the data from the website. Maybe you will meet the same problem in the future like me. I hope this simple introduction post can help you to further road with what you want to do.

*P.S. This topic is containing some other knowledge of CSS and website nodes which could be another long post, so I will just briefly introduce the way to identify them. If there's anything I was wrong or you think there's anything I should add on it, please feel free to tell me and I will be very appreciate!*



## Background:

**rvest** is new package that makes it easy to scrape (or harvest) data from html web pages, inspired by libraries like *beautiful soup*. It is designed to work with `magrittr` so that you can express complex operations as elegant pipelines composed of simple, easily understood pieces. It is a Wrappers around the 'xml2' and 'httr' packages to make it easy to download, then manipulate, HTML and XML. [This is the link](#)

- BugReports: [Here is the link](#)
- Author: Hadley Wickham, RStudio [cph]
- Maintainer: Hadley Wickham ( [hadley@rstudio.com](mailto:hadley@rstudio.com) )
- Repository: CRAN
- Date/Publication: 2016-06-17 08:57:12

**XML** is a file format which shares both the file format and the data on the World Wide Web, intranets, and elsewhere using standard ASCII text. It stands for Extensible Markup Language (XML).

- URL: [Here is the link](#)
- Author: Duncan Temple Lang and the CRAN Team
- Maintainer: Duncan Temple Lang [duncan@r-project.org](mailto:duncan@r-project.org)

- Repository: CRAN
- Date/Publication: 2017-06-19 12:43:32 UTC

## Content:

- 1. Importing html in `rvest` and `XML` packages
- 2. Scraping the rating, actor and poster from website
- 3. Other functions in `rvest` brief introduction

## 1.Importing html in `rvest` and `XML` packages

Let's see when we are extracting the HTML elements, what is the difference between `XML` and `rvest` :

```
library(XML)
srtts<-htmlParse("http://apps.saferoutesinfo.org/legislation_funding/state_apportionment.cfm")
```

In this part, we loaded the `XML` package first. Then we used `htmlParse` function to read the url that we passed to the function.

Let's see the `rvest` package:

```
library(rvest)
```

```
## Loading required package: xml2
```

```
##
## Attaching package: 'rvest'
```

```
## The following object is masked from 'package:XML':
##
## xml
```

```
nba_html <- paste0('https://www.basketball-reference.com', '/leagues/NBA_2017.html')
xml_doc <- read_html(nba_html)
```

In `rvest` package, we used `read_html` function to get the HTML element we want.

Apart from the function we used differently in previous part, the `class` of the read HTML are also different:

- In `XML` packages, the class is “HTMLInternalDocument” “HTMLInternalDocument” “XMLInternalDocument” “XMLAbstractDocument” shown below:

```
class(srtts)
```

```
## [1] "HTMLInternalDocument" "HTMLInternalDocument" "XMLInternalDocument"
## [4] "XMLAbstractDocument"
```

- In `rvest` package, the class is `xml_document` and `xml_node` shown below:

```
class(xml_doc)
```

```
## [1] "xml_document" "xml_node"
```

## 2. Scraping the rating, actors and poster from website

Many of us prefer to check website ratings before we really sit down and watch a movie. Also, some of us will write the comment to the film after watching it and then rate it. Therefore, the ratings in the website represent a certain level of the satisfaction of the audience. So, taking the rates as our data are reliable.

So let's see how do we do it with `rvest`

### Note here:

- How to find the `node` :
  - Find the website you are interested in
  - use the right click in the website and select `View page resources` , then you will see the following vision:

```

4018   if ('csm' in window) {
4019       csm.measure('csm_TitleRecsWidget_finished');
4020   }
4021   </script>
4022   <script>
4023       if (typeof uet == 'function') {
4024           uet("be", "TitleRecsWidget", {wb: 1});
4025       }
4026   </script>
4027   <script>
4028       if (typeof uex == 'function') {
4029           uex("ld", "TitleRecsWidget", {wb: 1});
4030       }
4031   </script>
4032
4033   <script>
4034       if (typeof uet == 'function') {
4035           uet("bb", "TitleCastWidget", {wb: 1});
4036       }
4037   </script>
4038   <script>
4039       if ('csm' in window) {
4040           csm.measure('csm_TitleCastWidget_started');
4041       }
4042   </script>
4043   <div class="article" id="titleCast">
4044       <span class=rightcornerlink >
4045           <a href="/register/login?why=edit&ref_tt_cl"
4046       rel="login">Edit</a>
4047       </span>
4048       <h2>Cast</h2>
4049
4050       <table class="cast_list">
4051           <tr><td colspan="4">Cast overview, first billed only:</td></tr>
4052           <tr class="odd">
4053               <td class="primary_photo">
4054                   <a href="/name/nm0012523/?ref_tt_cl_1"
4055                   ></a>
4058                   <td class="itemprop" itemprop="actor" itemscope itemtype="http://schema.org/Person">
4059                   <a href="/name/nm0012523/?ref_tt_cl_1"
4060                   itemprop="url"> <span class="itemprop" itemprop="name">Scott Adsit</span>
4061                   </a>
4062                   <td class="ellipsis">
4063                       ...
4064                   </td>

```

- Find the part you want to select and identify the `id` or the `xpath` of it
- Write it as an argument in functions introduced above
- And... You are done! :)

## Examples:

### a. Scraping rates

```

#install.packages("rvest")
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

```

```

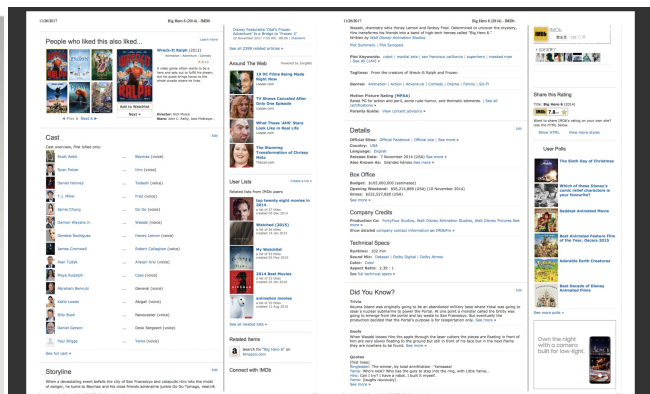
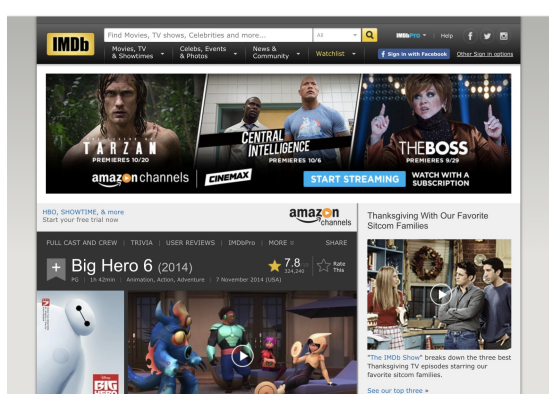
baymax_movie <- read_html("http://www.imdb.com/title/tt2245084/?ref_tt_rec_tti")

rating <- baymax_movie %>%
  html_nodes("strong span") %>%
  html_text() %>%
  as.numeric()
rating

```

```
## [1] 7.8
```

So what did we do here? Let's see the original website:



There are advertisement, videos and many other things. So the first thing to do is turn this visualized website to data that R can read. This is our **first line code** `read_html` followed by the website

Then we want to scrape the rating from this website. In order to read easily for humans, we used `%>%` we learned for the lab. We take the `baymax_movie` (whose class is `xml_document`, `xml_node`, but we don't explain too much here) as our *source*. So we have our **second line code** with functions `html_nodes` and `html_text`

Don't forget to turn it to **numeric** with function `as.numeric`

There for we got the rating is **7.8**

Note the function descriptions here:

- `read_html` : Create an html document from a url, a file on disk or a string containing html
- `html_nodes` :Select nodes from an HTML document
- `html_text` :Extract attributes, text and tag name from html.

## b. Scraping the cast list

```
cast <- baymax_movie %>%  
  html_nodes("#titleCast .itemprop span") %>%  
  html_text()  
cast
```

```
## [1] "Scott Adsit"      "Ryan Potter"      "Daniel Henney"  
## [4] "T.J. Miller"      "Jamie Chung"      "Damon Wayans Jr."  
## [7] "Genesis Rodriguez" "James Cromwell"   "Alan Tudyk"  
## [10] "Maya Rudolph"     "Abraham Benrubi"  "Katie Lowes"  
## [13] "Billy Bush"       "Daniel Gerson"    "Paul Briggs"
```

Same as before, we `source` the `baymax_movie` select the node “#titleCast .itemprop span” as our `css.xpass` which will be introduced in next part.
















So there you can have the cast for the film “Big Hero 6” as shown

Therefor we turned the cast from websit (picture 1) to the data we want (picture 2)

### Cast

[Edit](#)

Cast overview, first billed only:

	Scott Adsit	...	Baymax (voice)
	Ryan Potter	...	Hiro (voice)
	Daniel Henney	...	Tadashi (voice)
	T.J. Miller	...	Fred (voice)
	Jamie Chung	...	Go Go (voice)
	Damon Wayans Jr.	...	Wasabi (voice)
	Genesis Rodriguez	...	Honey Lemon (voice)
	James Cromwell	...	Robert Callaghan (voice)
	Alan Tudyk	...	Alistair Krei (voice)
	Maya Rudolph	...	Cass (voice)
	Abraham Benrubi	...	General (voice)
	Katie Lowes	...	Abigail (voice)
	Billy Bush	...	Newscaster (voice)
	Daniel Gerson	...	Desk Sergeant (voice)
	Paul Briggs	...	Yama (voice)

[See full cast »](#)

```
[1] "Scott Adsit"      "Ryan Potter"  
[3] "Daniel Henney"   "T.J. Miller"  
[5] "Jamie Chung"     "Damon Wayans Jr."  
[7] "Genesis Rodriguez" "James Cromwell"  
[9] "Alan Tudyk"      "Maya Rudolph"  
[11] "Abraham Benrubi" "Katie Lowes"  
[13] "Billy Bush"      "Daniel Gerson"  
[15] "Paul Briggs"
```

## c. Scraping the poster out of the website

```
poster <- baymax_movie %>%
  html_nodes(".poster img") %>%
  html_attr("src")
poster
```

```
## [1] "https://images-na.ssl-images-amazon.com/images/M/MV5BMDliOTIzNmUtOTl1OC00NDU3LWFjNjYtMGMDNDc1YTMxNjYxXkEyXkFqcGdeQXVyNTM3NzExMDQ@._v1_UY268_CR3,0,182,268_AL_.jpg"
```

Here we script the poster from the one which beside the trailer and save it as our poster. The result returned to be a website address and this is the website address for our saved poster.



### 3. Other functions in `rvest` brief introduction

#### 1. `html_table` function

- Description: Parse an html table into a data frame.
- Usage: `html_table(x, header = NA, trim = TRUE, fill = FALSE, dec = ".")`
- Argument:
  - X: A node, node set or document.
  - header: Use first row as header? If NA, will use first row if it consists of " " tags.
  - trim: Remove leading and trailing whitespace within each cell
  - fill: If TRUE, automatically fill rows with fewer than the maximum number of columns with NAs.
  - dec: The character used as decimal mark.
- Examples:

```
tdist <- read_html("http://en.wikipedia.org/wiki/Student%27s_t-distribution")
table <- tdist %>%
  html_node("table.infobox") %>%
  html_table(header = FALSE)
head(table, n=5)
```

```
##                               X1
## 1      Probability density function
## 2 Cumulative distribution function
## 3                               Parameters
## 4                               Support
## 5                               PDF
##
X2
## 1
Probability density function
## 2
Cumulative distribution function
## 3
 $\nu > 0$  degrees of freedom (real)
## 4
 $x \in (-\infty; +\infty)$ 
## 5  $\frac{\Gamma(\frac{\nu+1}{2})}{\Gamma(\frac{\nu}{2})} \frac{1}{\sqrt{\nu\pi}} (1+\frac{x^2}{\nu})^{-\frac{\nu+1}{2}}$ 
 $\frac{\Gamma(\frac{\nu+1}{2})}{\Gamma(\frac{\nu}{2})} \frac{1}{\sqrt{\nu\pi}} (1+\frac{x^2}{\nu})^{-\frac{\nu+1}{2}}$ 
```

#### 2. `html_form` functions

- Description: Parse forms in a page
- Usage: `html_form(x)`
- Argument:
  - X: A node, node set or document

- Examples:

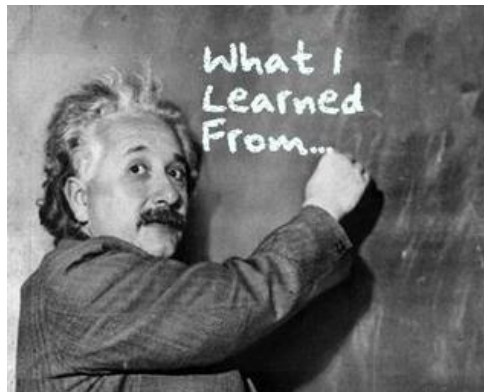
```
html_form(read_html("https://hadley.wufoo.com/forms/libraryrequire-quiz/"))
```

```
## [[1]]
## <form> 'form27' (POST https://hadley.wufoo.com/forms/libraryrequire-quiz/)
## <input hidden> 'Field1':
## <input radio> 'Field1': Loads the class package
## <input radio> 'Field1': I have no idea!
## <input radio> 'Field1': Loads the cluster package
## <input radio> 'Field1': Something else
## <input text> 'Field1_other_Something else':
## <input hidden> 'currentPage': GfqNaMDZduwhzuA9YxErPxIPfa0GJOeczQWDLzZzI=
## <input submit> 'saveForm': Submit
## <textarea> 'comment' [0 char]
## <input hidden> 'idstamp': qRxNeIZQUaB1PFfM+K+FjP+WvMnIre19F6csLMwey0A=
## <input hidden> 'stats': {"errors":0,"startTime":0,"endTime":0,"referer":null}
## <input hidden> 'clickOrEnter':
```

## Conclusion

Therefore, I have shown you the `rvest` package and examples of how to scrape **rate**, **cast** and **poster** from a website. Then introduced `html_table` and `html_form` functions in “`rvest`” package to help you with further scraping willings. The method of identify the `nodes` and `xpath` in the functions above was also introduced briefly but not fully since this is not our major point for this post. So I hope you can get a sense of how the scraping works in real life!

## Take home message



Knowing how to scraping data from website is not only useful but also practical. After reading this post, I hope you know the following points:

- Different class type when extracting HTML elements with `XML` and `rvest` packages
- Get a sense of how to find `nodes` and `xpath`
- How to scrape `rating` of a movie from website
- How to scrape `cast` of a movie from website
- How to scrape `poster` of a movie from website
- How to use function `html_table`
- How to use function `html_form`

## Thanks for your reading!

### References:

- [Github resources](#)
- [rvest documentation on CRAN](#)
- [Github, rvest tutorial](#)
- <https://cran.r-project.org/web/packages/rvest/rvest.pdf>
- <https://www.rdocumentation.org/packages/rvest/versions/0.3.2>
- [https://www.rdocumentation.org/packages/rvest/versions/0.3.2/topics/html\\_nodes](https://www.rdocumentation.org/packages/rvest/versions/0.3.2/topics/html_nodes)
- <https://cssselect.readthedocs.io/en/latest/>
- [http://www.imdb.com/title/tt2245084/?ref\\_=tt\\_rec\\_tti](http://www.imdb.com/title/tt2245084/?ref_=tt_rec_tti)
- <https://blog.rstudio.com/2014/11/24/rvest-easy-web-scraping-with-r/>
- <http://www.reed.edu/data-at-reed/resources/R/rvest.html>