Scraping avec R

Kossi

14/01/2020

La librairie rvest

Le paquet rvest permet d'extraire du contenu des pages web à l'aide de la **syntaxe XPath** ou des **sélecteurs CSS**. On utilisera beaucoup les sélecteurs CSS; les xptah n'étant utilisés qu'en cas de besoin. N'hésutez pas d'exécuter le script suivant qui va installer **rvest** si ce n'est pas encore le cas pour vous.

```
if (! ("rvest" %in% rownames(installed.packages())) )
{install.packages("rvest", dep=TRUE)}
if (! ("httr" %in% rownames(installed.packages())) )
{install.packages("httr", dep=TRUE)}

 require("rvest")
 require("httr")
```

L'interface rvest

rvest est un paquet assez simple d'usage. Le nombre de focntions mises à dispositon de l'utilisateur est reduit mais permet de presque tout faire : extraire les bouts de code **HTML** par tag, par sélecteur CSS, par xpath... **rvest** dispose aussi de quelques fonctions supplémentaires qui permettent de naviguer dans les pages en émulant un navigateur web. Ci-dessous, une liste non complète de fonctions d'extraction qui seront approfondies dans cette partie :

- html()
- html_nodes()
- html_text(), html_attrs(), html_tag()
- html_table

Fonction html()

La fonction html() est généralement la première à être utilisée dans un flux d'extraction car elle permet d'importer en R le contenu d'une page web. La fonction accepte donc deux paramètres, dont le deuxième (encoding) est optionnel. Elle est l'équivalent de `requests.get` en **python**.

```
msn = html("https://www.msn.com/en-us/money/markets/currencies")
```

```
## Warning: 'html' is deprecated.
## Use 'xml2::read_html' instead.
## See help("Deprecated")
```

```
print(msn)
```

```
## {html_document}
## <html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en-US" dir="ltr">
## [1] <head>\n<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">\n<!-- data-info:v:201
```

```
## [2] <body class="currenciespage green">\n \n\n\n<div class="head">\n<div>\n<div id="topnav">\n <ul cl
```

Normalement vous devriez recevoir un message vous informant que la fonction **html** est obsolète. Cette fonction a été rendue obsolète afin de promouvoir une meilleure façon de récupérationn de pages **HTMLL**. En effet, la fonction **html** ne renvoie que le contenu de la page et rien sur le statut et les autres composants de la requête **HTTP**. Avec le paquet **httr**, on lirait récupérerait une page web de la façon suivante:

```
msn = GET("https://www.msn.com/en-us/money/markets/currencies" )
print(paste( "code d'état :",  msn$status_code))
```

```
## [1] "code d'état : 200"
```

```
print(msn)
```

```
## Response [https://www.msn.com/en-us/money/markets/currencies]
##   Date: 2020-01-14 07:24
##   Status: 200
##   Content-Type: text/html; charset=utf-8
##   Size: 113 kB
## <?xml version="1.0" encoding="UTF-8" ?>
## <!DOCTYPE HTML PUBLIC "-//WAPFORUM//DTD XHTML Mobile 1.2//EN" "http://www.openmobilealliance.org/tech
## <html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en-US" dir="ltr">
##
## <head>
##  <!-- data-info:v:20200103_20520765;a:51b8c7f0-aad3-4a84-b0f2-e0b5d65732d3;cn:3;az:{did:8df28e868177
##  <meta name="robots" content="index, follow"/>
##
##
##
## ...
```

Fonction html_nodes()

La fonction **html_nodes()** est celle qui permet de faire l'essentiel du boulot du scrapeur, car elle permet d'extraire des morceaux de code HTML contenant les informations d'intérêt à partir de la page iweb. Pour extraire les données, html_nodes() met à disposition deux moyens : les sélecteurs **xpaths** et **css**. La fonction html_nodes() accepte deux arguments, qui sont tous deux obligatoires. Le format d'appel est donc **html_nodes(page, [css, xpath])** L'argument `page` représente le code HTML de la page et le deuxième argument est un critère de sélection.

```
page =  content(msn)
# Extraire les paragraphes de la page
html_nodes(page, "p")
```

```
## {xml_nodeset (121)}
##  [1] <p class="truncated-string" title="Open">Open</p>
##  [2] <p class="truncated-string" title="0.8974">0.8974</p>
##  [3] <p class="truncated-string" title="Change">Change</p>
##  [4] <p class="truncated-string" title="-0.0005">-0.0005</p>
##  [5] <p class="truncated-string" title="Change%">Change%</p>
##  [6] <p class="truncated-string" title="-0.0557%">-0.0557%</p>
##  [7] <p class="truncated-string" title="52 Week High">52 Week High</p>
##  [8] <p class="truncated-string" title="0.9191">0.9191</p>
##  [9] <p class="truncated-string" title="52 Week Low">52 Week Low</p>
## [10] <p class="truncated-string" title="0.8642">0.8642</p>
## [11] <p class="truncated-string" title="Major Currencies">Major Currencies</p>
## [12] <p class="truncated-string" title="Price">Price</p>
```

```
## [13] <p class="truncated-string" title="Change">Change</p>
## [14] <p class="truncated-string" title="Change%">Change%</p>
## [15] <p class="truncated-string" title="52 Week High">52 Week High</p>
## [16] <p class="truncated-string" title="52 Week Low">52 Week Low</p>
## [17] <p class="truncated-string" title="Euro">Euro</p>
## [18] <p class="truncated-string" title="USD/EUR">USD/EUR</p>
## [19] <p class="truncated-string" title="0.8977" aria-label="Price 0.8977">0.8977</p>
## [20] <p class="truncated-string" title="-0.0002" aria-label="Change -0.0002">-0.0002</p>
## ...
```

On voit bien que l'équivalent **python** de **html_nodes()** est soit **bs.find_all()** ou **bs.select**. Il existe aussi un équivalent de **bs.find** ou **bs.select_one** en **R** : c'est **html_node** .

Fonctions html_text(), html_attrs(), et html_name()

Ces fonctions permettent d'avoir accès aux différents composants d'un noeud **html** extrait à partir des fonctions **html_node** ou **html_nodes**.

- **html_text(x, ...)** : extraire le texte de l'élément (passer l'argument trim = TRUE pour supprimer les espaces de début et de fin)
- **html_attr(x), html_attrs(x)** : extraire les attributs du noeud x
- **html_name(x)** : obtenir le nom de l'élément

```
currency_class <- ".mjrcurrncsitem"
cur = html_node(page, currency_class)

thead_class = '.mjrcurrncsrow.tblheaderrow'
header = html_node(cur,  thead_class)
headers = html_nodes(header, ".mctblheading") # Avez-vous pu retrouver la classe `mctblheading`  de vou
header_values = c()
i = 0
for (header in headers){
    header_values[i] = html_node(header, "p")%>%html_attr("title")
    i = i + 1
}
print(header_values)
```

```
## [1] "Price"        "Change"       "Change%"      "52 Week High" "52 Week Low"
```

Résumé

Dans cette quatrième partie du cours portant sur le scraping avec **R**, nous avons abordé :

- la librairie **rvest** et ses différentes interfaces html(), html_node(), html_attr(), html_text(), html_tag()
- la librairie **httr** qui permet de recupérer de façon plus fiable une page web
- les différentes "relations d'équivalence" entre les interfaces **R** et celles de **python**