Webscrapping avec R

Lise Vaudor, Sébastien Rey-Coyrehourcq, Fabien Pfaender

Web scraping with R

Lise Vaudor / Sébastien Rey-Coyrehourcq / Fabien Pfaender

Today !

- Webscraping, what is this ?
- Some ethicals considerations
- A theoritical HTML page
  - *Read Html using rvest*
  - *How manipulate data efficiently (purrr, string)*
  - *How-to extract link & data from HTML (Regex, CSS, XPath)*
- A cat and mouse game :)
- Tutorial & Uses cases

Webscraping

- definition
- some elements of web architecture

Ethics, some elements of reflexion.

In two words

- Grey Zone !! Heavily dependent of your country and the site you scrap !
- Law is clearly not so clear, but now RGPD include **derogations** for researchers
- Contact CNIL (France) or similar institute in your country, they will help you !
- **Don’t publish any data on the web** before anonymisation !
- Collect data limited to your project
- Protect personnal data very seriously (data-center, encrypted, firewall protection, etc.)

In more than two words …

- Read new European Law : GPDR or RGPD (not a joke, not so hard to read)

- Read this starting paper writed on this subject for this school : https://hackmd.io/g9JrWURjTHyLr9EWQqKoiQ?both
- Be inspirated by existing case of scrapping or other discipline (medecine)

Packages

- 

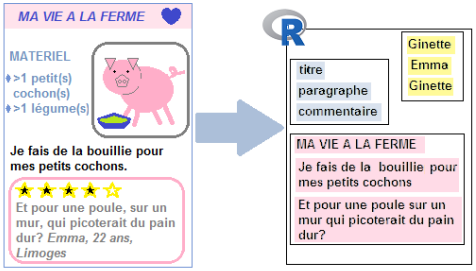
`rvest`: web scraping
- 

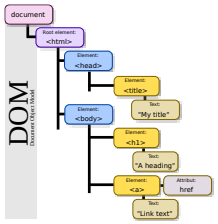
`dplyr`: table-wrangling
- 

`stringr`: strings
- 

`purrr`: iteration

A theoretical example of Web scraping





HTML Document

Understanding html documents' structure

Tools **SelectorGadget** or html inspector

Package **rvest**:

To **collect the contents of a web page**, one has to:

- **read an html page** in R
- **get the element of interest** in the page (basic html, SelectorGadget)
- **parse** the page
- **select an element** of the page
- **get the contents** of the element

html tags

```
<html>
<style>
h1 {background-color: powderblue;}
.image {margin-left:50%;}
.comment{border-style:solid; background-c
.comment-author{font-style: italic;}
</style>
<h1> MA VIE A LA FERME </h1>
<div class="ingredients">
  <b> INGREDIENTS</b>
  <ul>
    <li> >1 cochon(s) </li>
    <li> >1 légume(s) </li>
  </ul>
</div>
<div class="right"><div class="image"><im
<p> Je fais de la bouillie pour mes petit
<p> Pour un cochon, pour deux cochons, po
<b>comments</b>
<div class="comment">Et pour une poule su
</div>
<div class="comment-author">Emma, 22 ans,
<div class="comment">Je vois que vousête
<div class="comment-author">Michel, 56 an
</html>
```

MA VIE A LA FERME

INGREDIENTS

- >1 cochon(s)
- >1 légume(s)

Je fais de la bouillie pour mes petits cochons.

Pour un cochon, pour deux cochons, pour trois cochons, pour quatre, puis pour cinq, pour six, pour sept, pour huit, pour neuf, boeuf!

Commentaires

Et pour une poule sur un mur qui picoterait du pain dur? c'est bien beau de nourrir les petits cochons mais qu'en est-il des autres?

Emma, 22 ans, Limoges

Je vois que vous êtes, telle la petite poule rousse, bien aimable. Avez-vous pu compter sur l'aide du chat et du canard (et des cochons eux-mêmes!) pour semer vos 5 grains de blé? Les cochons se sont-ils régalés?

Michel, 56 ans, P

Read an html page

```
<html>
<style>
h1 {background-color: powderblue;}
.image {margin-left:50%;}
.comment{border-style:solid; background-c
.comment-author{font-style: italic;}
</style>
<h1> MA VIE A LA FERME </h1>
<div class="ingredients">
  <b> INGREDIENTS</b>
  <ul>
    <li> >1 cochon(s) </li>
    <li> >1 légume(s) </li>
  </ul>
</div>
<div class="right"><div class="image"><im
<p> Je fais de la bouillie pour mes petit
<p> Pour un cochon, pour deux cochons, po
<b>comments</b>
<div class="comment">Et pour une poule su
</div>
<div class="comment-author">Emma, 22 ans,
<div class="comment">Je vois que vous ête
<div class="comment-author">Michel, 56 an
</html>
```

Read html page in R:

```
library(rvest)
html=read_html("data/blog_de_ginette.ht
html
```

```
## {xml_document}
## <html>
## [1] <head>\n<meta http-equiv="Conten
## [2] <body>\n<h1> MA VIE A LA FERME <
```

Extract some elements in the html page

```
<html>
<style>
h1 {background-color: powderblue;}
.image {margin-left:50%;}
.comment{border-style:solid; background-c
.comment-author{font-style: italic;}
</style>
<h1> MA VIE A LA FERME </h1>
<div class="ingredients">
  <b> INGREDIENTS</b>
  <ul>
    <li> >1 cochon(s) </li>
    <li> >1 légume(s) </li>
  </ul>
</div>
<div class="right"><div class="image"><im
<p> Je fais de la bouillie pour mes petit
<p> Pour un cochon, pour deux cochons, po
<b>comments</b>
<div class="comment">Et pour une poule su
</div>
<div class="comment-author">Emma, 22 ans,
<div class="comment">Je vois que vous ête
<div class="comment-author">Michel, 56 an
</html>
```

Extract some elements (“nodes” or “nodesets”):

```
html_nodes(html,"b")
```

```
## {xml_nodeset (2)}
## [1] <b> INGREDIENTS</b>
## [2] <b>Commentaires</b>
```

```
html_nodes(html,".comment-author")
```

```
## {xml_nodeset (2)}
## [1] <div class="comment-author">Emma
## [2] <div class="comment-author">Mich
```

```
html_nodes(html,".ingredients") %>%
  html_children()
```

```
## {xml_nodeset (2)}
## [1] <b> INGREDIENTS</b>
## [2] <ul>\n<li> &gt;1 cochon(s) </li>
```

Extract the type of some elements

```
<html>
<style>
h1 {background-color: powderblue;}
.image {margin-left:50%;}
.comment{border-style:solid; background-c
.comment-author{font-style: italic;}
</style>
<h1> MA VIE A LA FERME </h1>
<div class="ingredients">
  <b> INGREDIENTS</b>
  <ul>
    <li> >1 cochon(s) </li>
    <li> >1 légume(s) </li>
  </ul>
</div>
<div class="right"><div class="image"><im
<p> Je fais de la bouillie pour mes petit
<p> Pour un cochon, pour deux cochons, po
<b>comments</b>
<div class="comment">Et pour une poule su
</div>
<div class="comment-author">Emma, 22 ans,
<div class="comment">Je vois que vous ête
<div class="comment-author">Michel, 56 an
</html>
```

<name attr.name="attr.value"> text </name>

html_name( ) ➡ name

html_text( ) ➡ text

html_attrs( ) ➡ attr.name "attr.value"

html_attr(,"attr.name") ➡ "attr.value"

Extract the type of nodes or nodesets:

```
html_nodes(html,".image") %>%
  html_name()

## [1] "div"
```

Extract the content of some elements

```
<html>
<style>
h1 {background-color: powderblue;}
.image {margin-left:50%;}
.comment{border-style:solid; background-c
.comment-author{font-style: italic;}
</style>
<h1> MA VIE A LA FERME </h1>
<div class="ingredients">
  <b> INGREDIENTS</b>
  <ul>
    <li> >1 cochon(s) </li>
    <li> >1 légume(s) </li>
  </ul>
</div>
<div class="right"><div class="image"><im
<p> Je fais de la bouillie pour mes petit
<p> Pour un cochon, pour deux cochons, po
<b>comments</b>
<div class="comment">Et pour une poule su
</div>
<div class="comment-author">Emma, 22 ans,
<div class="comment">Je vois que vous ête
<div class="comment-author">Michel, 56 an
</html>
```

<name attr.name="attr.value"> text </name>

html_name( ) ➡ name

html_text( ) ➡ text

html_attrs( ) ➡ attr.name "attr.value"

html_attr(,"attr.name") ➡ "attr.value"

Extract the content of nodes or nodesets:

```
html_nodes(html,"b") %>%
  html_text()
```

```
## [1] " INGREDIENTS" "Commentaires"
```

Extract the attributes of some elements

```
<html>
<style>
h1 {background-color: powderblue;}
.image {margin-left:50%;}
.comment{border-style:solid; background-c
.comment-author{font-style: italic;}
</style>
<h1> MA VIE A LA FERME </h1>
<div class="ingredients">
  <b> INGREDIENTS</b>
  <ul>
    <li> >1 cochon(s) </li>
    <li> >1 légume(s) </li>
  </ul>
</div>
<div class="right"><div class="image"><im
<p> Je fais de la bouillie pour mes petit
<p> Pour un cochon, pour deux cochons, po
<b>comments</b>
<div class="comment">Et pour une poule su
</div>
<div class="comment-author">Emma, 22 ans,
<div class="comment">Je vois que vous ête
<div class="comment-author">Michel, 56 an
</html>
```

<name attr.name="attr.value"> text </name>

html_name( ) ➡ name

html_text( ) ➡ text

html_attrs( ) ➡ attr.name "attr.value"

html_attr(,"attr.name") ➡ "attr.value"

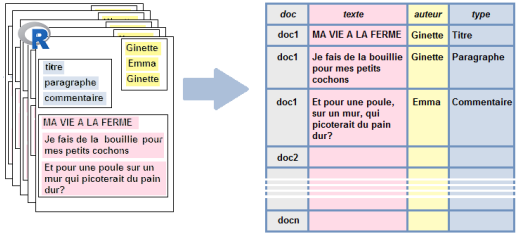
Extract the attributes of nodes or nodesets:

```
html_nodes(html,"div") %>%
  html_attrs()
```

```
## [[1]]
##         class
## "ingredients"
##
## [[2]]
##   class
## "right"
##
## [[3]]
##   class
## "image"
##
## [[4]]
##     class
## "comment"
##
## [[5]]
##            class
## "comment-author"
```

```
##
## [[6]]
##     class
## "comment"
##
## [[7]]
##            class
## "comment-author"
```

Rectangular format, and function-ization



doc	texte	auteur	type
doc1	MA VIE A LA FERME	Ginette	Titre
doc1	Je fais de la bouillie pour mes petits cochons	Ginette	Paragraphe
doc1	Et pour une poule, sur un mur, qui picoterait du pain dur?	Emma	Commentaire
doc2			
docn			

Extract data and make it into a table:

```
page="data/blog_de_ginette.htm"
html=read_html(page, encoding="UTF-8")
texte=html %>% html_nodes(".comment") %>%
auteur=html %>% html_nodes(".comment-auth
tib_comments=tibble(texte,auteur)
tib_comments
```

```
## # A tibble: 2 x 2
##   texte
##   <chr>
## 1 Et pour une poule sur un mur qui pic
## 2 Je vois que vous êtes, telle la peti
```

It is actually a good idea to make all this into a fonction that would have the page's url as intput and the tibble as output:

```
extract_comments=function(page){
  html=read_html(page, encoding="UTF-8"
  texte=html %>% html_nodes(".comment")
  auteur=html %>% html_nodes(".comment-
  tib_comments=tibble(doc=rep(page,leng
                      texte,
                      auteur)
  return(tib_comments)
}

extract_comments("data/blog_de_ginette.
```

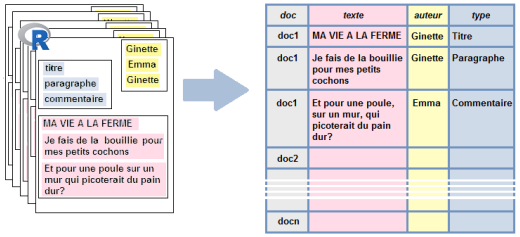
```
## # A tibble: 2 x 3
##   doc                     texte
##   <chr>                   <chr>
## 1 data/blog_de_ginette.htm Et pour u
## 2 data/blog_de_ginette.htm Je vois q
```

```
extract_comments("data/blog_de_jean-mar
```

```
## # A tibble: 6 x 3
##   doc                       texte
##   <chr>                     <chr>
## 1 data/blog_de_jean-marc.htm Les tho
```

```
## 2 data/blog_de_jean-marc.htm Pourquo
## 3 data/blog_de_jean-marc.htm Tout ça
## 4 data/blog_de_jean-marc.htm Je préf
## 5 data/blog_de_jean-marc.htm On ne c
## 6 data/blog_de_jean-marc.htm Et pend
```

Iteration

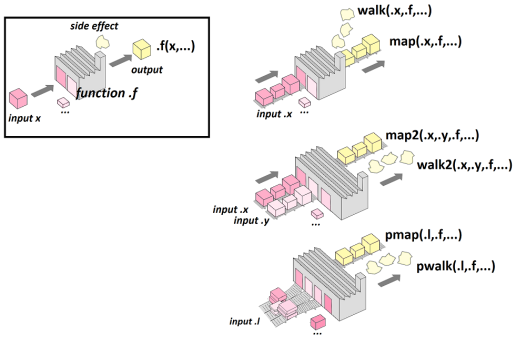


doc	texte	auteur	type
doc1	MA VIE A LA FERME	Ginette	Titre
doc1	Je fais de la bouillie pour mes petits cochons	Ginette	Paragraphe
doc1	Et pour une poule, sur un mur, qui picoterait du pain dur?	Emma	Commentaire
doc2			
docn			

Now, let's imagine that we actually have to deal with several pages **with a common structure**.

We would like to apply `extract_comments()` iteratively to all these pages.

The `purrr` package enables us to **apply a function iteratively** to all elements of a list or of a vector (… of course this is just a more straightforward way to loop through a `for` structure…).



Iteration with purrr

```
pages=c("data/blog_de_ginette.htm",
        "data/blog_de_jean-marc.htm",
        "data/blog_de_norbert.htm")

list_comments=map(pages, extract_comments
list_comments
```

```
## [[1]]
## # A tibble: 2 x 3
##   doc                      texte
##   <chr>                    <chr>
## 1 data/blog_de_ginette.htm Et pour une
## 2 data/blog_de_ginette.htm Je vois que
##
## [[2]]
## # A tibble: 6 x 3
##   doc                        texte
##   <chr>                      <chr>
## 1 data/blog_de_jean-marc.htm Les thons
## 2 data/blog_de_jean-marc.htm Pourquoi
## 3 data/blog_de_jean-marc.htm Tout ça m
## 4 data/blog_de_jean-marc.htm Je préfèr
## 5 data/blog_de_jean-marc.htm On ne com
## 6 data/blog_de_jean-marc.htm Et pendan
##
## [[3]]
## # A tibble: 4 x 3
##   doc                      texte
##   <chr>                    <chr>
## 1 data/blog_de_norbert.htm "A quel mom
## 2 data/blog_de_norbert.htm Norbert, mo
## 3 data/blog_de_norbert.htm L'ambiance
## 4 data/blog_de_norbert.htm Vous devez
```

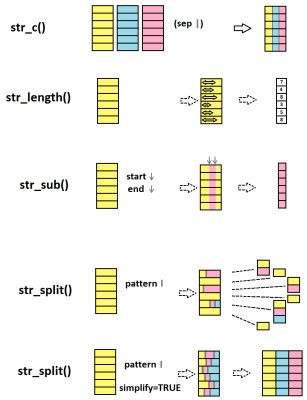
```
tibtot_comments <- list_comments %>%
  bind_rows()
tibtot_comments
```

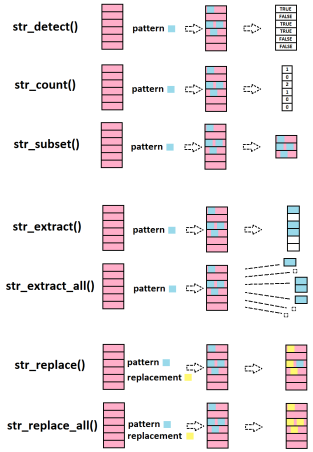
```
## # A tibble: 12 x 3
##    doc                      texte
##    <chr>                    <chr>
##  1 data/blog_de_ginette.htm Et pou
##  2 data/blog_de_ginette.htm Je voi
##  3 data/blog_de_jean-marc.htm Les th
##  4 data/blog_de_jean-marc.htm Pourqu
##  5 data/blog_de_jean-marc.htm Tout ç
##  6 data/blog_de_jean-marc.htm Je pré
##  7 data/blog_de_jean-marc.htm On ne
##  8 data/blog_de_jean-marc.htm Et pen
##  9 data/blog_de_norbert.htm   "A que
## 10 data/blog_de_norbert.htm   Norber
## 11 data/blog_de_norbert.htm   L'ambi
## 12 data/blog_de_norbert.htm   Vous d
```

Manipulate strings: package stringr

Package **stringr** 

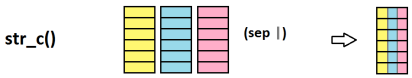
Blog post (in French) here.





Strings: concatenate, replace pattern

str_c() to combine strings




```
str_c("abra","ca","dabra")
```

```
## [1] "abracadabra"
```

```
str_c("Les jeux","de mots laids","sont po
```

```
## [1] "Les jeux de mots laids sont pour
```

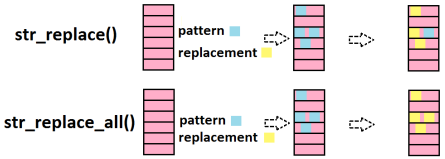
str_detect() detects a pattern


```
str_detect(c("Quarante","carottes","cru
             "croient","que","croquer",
             "crée","des","crampes."),
           pattern="cr")
```

```
## [1] FALSE FALSE  TRUE  TRUE FALSE  T
```

Strings: replace pattern, extract pattern

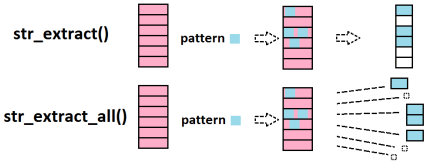
str_replace() replaces a pattern with another




```
str_replace(c("All we hear is",
              "Radio ga ga",
              "Radio goo goo",
              "Radio ga ga"),
            pattern="goo",
            replacement="ga")
```

```
## [1] "All we hear is" "Radio ga ga"
```

str_extract() extracts the pattern (if it's there!)

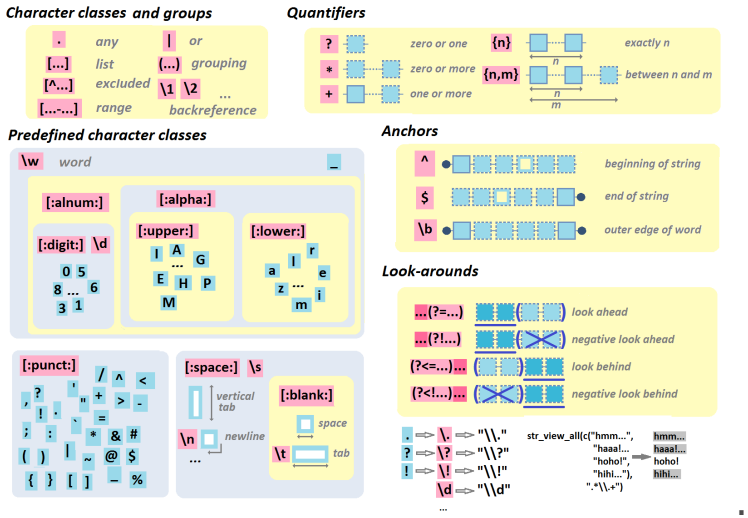



```
str_extract(c("L'âne","Trotro","trotte"
          pattern="tr")
```

```
## [1] NA   "tr" "tr" NA   "tr" "tr"
```

Regular expressions

Regular expressions are used to define **patterns** through **rules of construction**.A tutorial here



Regexp: character classes and groups

`.`	*any*	`|`	*or*
`[...]`	*list*	`(...)`	*grouping*
`[^...]`	*excluded*	`\1` `\2`	*...*
`[...-...]`	*range*		*backreference*

A **character class** corresponds to the notation `[...]`.

For instance, to detect all voyels in the string:

```
str_view_all("youp la boum",
             "[aeiou]")
```

`y`o`u`p l`a` b`o`u`m`

See the difference:

```
str_view_all("A132-f445-e34-C308-M9-E18
             "[308]")
```

A1`3`2-f445-e`3`4-C`3`0`8`-M9-E1`8`

```
str_view_all("A132-f445-e34-C308-M9-E18
             "308")
```

A132-f445-e34-C`308`-M9-E18

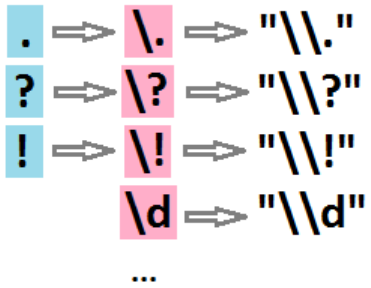
Any character can be noted `.`.

For instance, the pattern “any character followed by a letter” can be searched through

```
str_view_all("32a-B44-552-98eEf",
             ".[a-z]")
```

3`2a`-B44-552-9`8e`E`f`

Regexp: special characters



To find dots, question marks, exclamation marks:

```
str_view_all(c("Allô, John-John? Ici Joe
               "[\\.\\?\\!]")
```

```
Allô, John-John? Ici Joe la frite. Surprise!
```

Note that we don't write `"[.?!]"`, but `"[\\.\\?\\!]"`.

`.` (as we saw before), as well as ? and ! are **special characters**. So, to point out *actual* dots or interrogation/exclamation marks, one has to use the escape character `\`. The regular expression hence becomes `[\.\?\!]`

…

But it does not end here… as it is not *directly* the regular expression that is passed to the function, but a *string* that is *interpreted as a regular expression*. Thus each escape character `\` has to be escaped through a `\`.

So that the pattern passed to the function is actually `"[\\.\\?\\!]"`.

Regexp: excluded characters, character ranges

A character class can be defined as all characters **excluding the ones listed**. This is noted as `[^...]`:

For instance, all characters that are neither a vowel nor a blank space:

```
str_view_all("turlututu chapeau pointu",
             "[^aeiou ]")
```

turlututu chapeau pointu

Ranges of characters are noted `[...-...]`

For instance, all numbers between 1 and 5:

```
str_view_all(c("3 petits cochons", "101
               "[1-5]")
```

3 petits cochons

101 dalmations

77unains

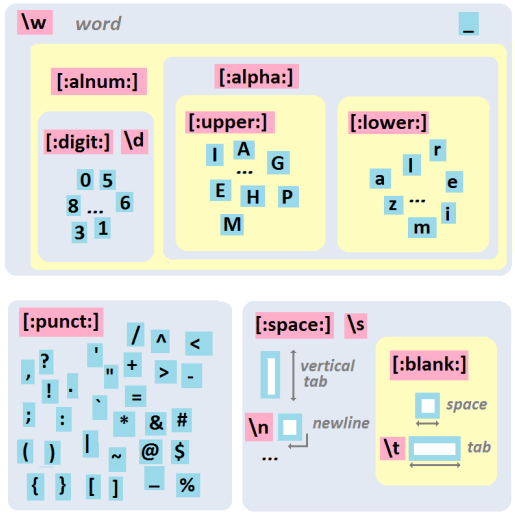
... or all those between A-F or a-e:

```
str_view_all("A132-f445-e34-C308-M2244-
             "[A-Fa-e]")
```

A132-f445-e34-C308-M2244-Z449-E18

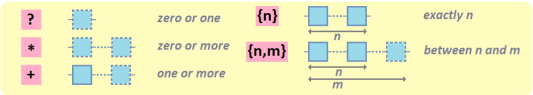
Regexp: predefined classes

Some character classes are predefined for instance **digits**, **punctuation characters**, **lower-case alphabetical characters**, etc.



Regexp: quantifiers

Quantifiers are used to specify **how many consecutive times** a particular class or group occurs.



zero or one: the pattern of interest is followed by ?.

```
str_view_all(c("file1990-fileB1990-fileAb
               "file\\d?")
```

file1990-fileB1990-fileAbis2005

zero or more: O the pattern of interest is followed by *.

```
str_view_all(c("file1990-fileB1990-file
               "file\\d*")
```

file1990-fileB1990-fileAbis2005

one or more : the pattern of interest is followed by +.

```
str_view_all(c("file1990-fileB1990-file
               "file\\d+")
```

file1990-fileB1990-fileAbis2005

Xpath

XPATH is another way to query the DOM, it's very powerfull, but also a little complex. Use some cheatsheet to remember principal operators.

Some examples :

```
html_nodes(html, xpath = "//div[@class='ingredients']//ul//li")
```

```
## {xml_nodeset (2)}
## [1] <li> &gt;1 cochon(s) </li>
## [2] <li> &gt;1 légume(s) </li>
```

```
html_nodes(html, xpath = "//text()[contains(.,'cochons') and not(contains(.,'poule'))]")
```

```
## {xml_nodeset (2)}
## [1]  Je fais de la bouillie pour mes petits cochons.
## [2]  Pour un cochon, pour deux cochons, pour trois cochons, pour quatre, ...
```

```
html_nodes(html, xpath = "//div[@class='comment']")
```

```
## {xml_nodeset (2)}
## [1] <div class="comment">Et pour une poule sur un mur qui picoterait du  ...
## [2] <div class="comment">Je vois que vous êtes, telle la petite poule ro ...
```

A 🐱 and 🐭 game (1)

How to defend from webscraping :

Legend : {⭐ : cost, difficulty to implement } , {★ : dificulty to bypass } , { ☻ or ☹ : user happiness }

- change or randomize some internal structure of HTML pages (★ ★ / ⭐ ⭐ ⭐ / ☻)
- sample, limits data displayed in time/quantity (★ ★ ★/ ⭐ / ☹)
- use really weird Captcha (★ / ⭐ / ☹)
- insert all informations into image, remove textual information (★ / ⭐ ⭐ / ☹)
- ask authentification before display any informations (★ / ⭐ / ☻)
- add a signature invisible in the content (honey pot) (★ ★/ ⭐ ⭐/ ☻)
- slowing at infinite the download of the content (★ ★ / ⭐ ⭐ / ☻)

- device/browser fingerprinting (⭐⭐ / 🌟 / ☻)
- pay company to do part of this, ex: cloudflare (⭐ / 🌟 / ☻)

A  and  game (2)

How to attack :

One rule : *If you see it on your browser, so you can get it*

Basics :

- using lot of slave people in parallel (joke!?)
- analyze webpage structure and network data (search XHR/API)
- use random (for time delay, query, location, etc.)
- use random user-agent

Advanced :

- use your imagination (limited to 1000 result for any reason ? => np, find another query with no limitation)
- use OCR / AI (image to text ? no problem !)
- use Proxy/VPN or TOR network (fast Hiding ! fast Jumping !)

- use cookie
- pay for online tools made by others
- ask dev community

- use (headless) browsers (to simulate human behavior)
- use a framework with lot of options (Scrapy & Python !)
- use a framework + lot of headless browser on the cloud (an army of false human)

Uses cases

- First tutorial (beginer), open project `projet_leboncoin.Rproj` or directly the html page online

- Second tutorial (advanced), open project `scrap_flightradar.Rmd` or directly the html page online

!Warning! You need Docker installed