Web Scraping with rvest: : CHEAT SHEET

Rvest

Rvest package can help us wrapper around the ‘xml2’ and ‘httr’ packages to make it easy to download, then manipulate, HTML and XML. It helps you scrape information from web pages. It is designed to work with ‘magrittr’ to make it easy to express common web scraping tasks, inspired by libraries like beautiful soup.

It has several important function, we take some of them shown below:

- Create an html document from a url, a file on disk or a string containing html
- Parse tables into data frames
- Extract attributes, text and tag name from html
- Fetch, modify and submit forms

Get page’s html

Save x as the link of the website

Get the html of the website

html(x, ..., encoding = "")

But now the html function is deprecated we use read_html() instead.

read_html(x, encoding = "", ..., options = c("RECOVER", "NOERROR", "NOBLANKS"))

web <- read_html(‘http://news.sina.com.cn/china/’)

Detect web encoding

Figure out the real encoding

guess_encoding(x)

fix character vectors after the fact

repair_encoding(x, from = NULL)

Process on html data

Save x as a node, node set or document

Select nodes from an HTML document

When for one element

html_node(x, css, xpath)

for a list of elements:

html_nodes(x, css, xpath)

web <- read_html(‘http://news.sina.com.cn/china/’)

html_nodes(web, "center")

Simulate a session in an html browser

html_session(url, ...)

s <- html_session("http://hadley.nz")

Parse an html table into a data frame

html_table(x, header = NA, trim = TRUE, fill = FALSE, dec = ".")

```
sample1 <- minimal_html("<table>
  <tr><th>Col A</th><th>Col B</th></tr>
  <tr><td>1</td><td>x</td></tr>
  <tr><td>4</td><td>y</td></tr>
  <tr><td>10</td><td>z</td></tr>
</table>")
sample1 %>%
  html_node(“table”) %>% html_table()
```

Process with XML file

Save x as the link of the website

Get the html of the website

xml(x, ..., encoding = "")

xml_tag(x)

xml_node(x, css, xpath)

xml_nodes(x, css, xpath)

Supply one of css or xpath depending on whether you want to use a CSS or XPath 1.0 selector.

Extract attributes, text and tag name

‘name’ is name of attribute to retrieve, and if trim is TRUE, it will trim leading and trailing spaces

CODE

```
html_text(x, trim = FALSE)
html_name(x)
html_children(x)
html_attrs(x)
html_attr(x, name, default =
NA_character_)
```

EXAMPLE

```
movie <- read_html(“http://www.
imdb.com/title/tt1490017/")
cast <- html_nodes(movie,
"#titleCast span.itemprop")
html_text(cast)
html_name(cast)
html_attrs(cast)
html_attr(cast, "class")
```

Processing example

Get the movie link from rotten tomatoes

Detect whether the path is allowed

paths_allowed("https://www.rottentomatoes.com/browse/box-office/")

Get html

rotten <- read_html(“https://www.rottentomatoes.com/browse/box-office/")

Get the link we want

```
rotten_links <- rotten %>%
html_nodes("td.left a") %>%
html_attr('href')
```

Scarping on Multiple pages

You may want to navigate to multiples pages while you scarping data. Here x is the session.

Navigate to a new URL

jump_to(x, URL)

Navigate to a link given expression

follow_link(x, i, css, xpath)

Jump back to previous URL

back(x)

Navigate history

session_history(x)

Process form

Set values and submit forms

Set the values of a form

set_values(form, …)

Submit a form back to the server

submit_form(session, form, submit=Null, ..)

Get a google form by id

google_form(id)

EXAMPLE

```
test <- google_form(“1M9B8DsYNFy
DjpwSK6ur_bZf8Rv_04ma3rmaaBiveoUI”
)
f1 <- set_values(html_form(test)[[1]],
entry.564397473=“abc”)
submit_form(session=x,
form=html_form(test))
```