2nd_ClassEX_thomas_fedra_simo

Fedra Ippolito

30/1/2021

WEB SCRAPING

The first part of the script is about web scraping. We have assigned the url of the article to an object called url, and then we have checked the robots.txt file. It said to us that we are not allowed to scrape the admin section of the web site, for the rest there were not problems. After this quick check we have made a polite request with the funciron getURL by the package RCurl:: to identify ourself to the server. In conclusion of this fist phase we have downloaded the html file to work in local to get the 105 links in the page with function getHTMLlinks by the package XML::.

```
library(tidyverse)
library(rvest)
library(stringr)
#dir.create("File download")
#dir.create("Scripts")
#dir.create("Report")

url <- "http://www.beppegrillo.it/un-mare-di-plastica-ci-sommergera/"
#browseURL(url)

#robots.txt check
#browseURL("http://www.beppegrillo.it/robots.txt")

#file downloading
page <- RCurl::getURL(url,
               useragent = str_c(R.version$platform,
                   R.version$version.string,sep = ", "),
                  httpheader = c(From = "xxxx@xxxx.xxx")) #your e-mail address

#Saving the page
download.file(url, destfile = "plastica.html")

#Get links
links <- as_tibble(XML::getHTMLLinks(page))
```

FOR LOOP

Now the game starts to get complex, but we never give up. Now we had to get all the links to the articles of the year 2016. The amount of them is 470 articles distributed homogeniuxly in 47 pages: 10 articles per page. The first step was to get a list of the links of the pages. To do that we have played with the links.

In fact, each link differs from the others just by the number of the page at the end of the link. Therefore, we get those links with the str_c function by the package stringr::. Easy, but after this point the problems started. We had to make a loop to make the process of getting the links automated. To do this we hat to create an empty list of lists with the function of container. With a combination of the rep(replicate) and vector functions we have created an empty list ("sut") containing 47 sublists. In each list of the "magic 47" we have created 10 empty spaces in which put the links to the articles.

```
link_to_pages <- str_c("https://www.beppegrillo.it/category/archivio/2016/page/",
                       1:47, "/")
link_to_pages
sut <-  rep(list(vector(mode ="list", length = 10)), 47)
sut
```

At this point it was time to create the for loop to get the links. We spliced it into two parts. The fist part was about the download of the 47 pages to work later in local. Therefore we have used the download.file function in combination with str_c function to download the pages and to get them a progressive numeration. In addition we have used the Sys.sleep function, set to one second, for politeness reasons. Came at this point we were ready to get all the links we need with another for loop. In the end we have unslisted the object and we obtained a vector with all the 470 links.

```
for (i in seq_along(link_to_pages)) {

  download.file(url = link_to_pages[i],
                destfile = here::here("File download", str_c("art", i, ".html")))
  Sys.sleep(1)

}

for(x in 1:47){
  sut[[x]][1:10] <- read_html(here::here("File download",
                                         str_c("art", x, ".html"))) %>%
  html_nodes(".td_module_10 .td-module-title a") %>%
  html_attr("href")
}

art <- unlist(sut)
art
```

To do the last part of the exercise and get the main texts of each article we took a similar way to the previous one with the creation of an empty list of lists. A preliminary consideration is that we have interpreted the wording "main text" as the part of the page where the text of the articles is actually contained. Often in this part of the web pages there are You Tube players. They will appear in the text as links contained in the main text of the articles. Moreover, it is possible that if in some articles there are not text part, they would be substituted by the YT players which will appear in our list as links.

```
out <- rep(list(vector(mode ="list", length = 10)), 47)

for(z in 1:470){
  out[[z]] <- read_html(art[z]) %>%
    html_nodes(".rs_preserve+ div") %>%
    html_text()
}

out
```

NOTE: outputs have been hidden for easier reading. In fact, often they are long lists of characters or articles, as in the case just above, that would have made it tedious to scroll through the report.

What does it means to "crawl" and what is a web spider?

The verb crawl means to collect web pages, but a web crawler can also perform data extraction during crawling. Usually web spiders are programs that automatically browse and download pages by following hyperlinks in a methodical and automated manner. Web crawler and web spider are used like synonyms.

How is it different from scraping?

The activity of web scraping concerns only the extraction of data from a website, in fact our scraping was built from two steps: parsing and extracting contents from URLs. This process was manually done, instead the crawling is an automated method. Moreover we have tryed to authomize the process of web scraping adding the for loop.

Try to build a spider scraper: what functions should you use?

Rcrawler(Website = "https://www.beppegrillo.it/", no_cores = 4, no_conn = 4, dataUrlfilter ="./category/archivio/2016/", crawlUrlfilter="./category/archivio/2016/", ExtractCSSPat = c(".td_module_10 .td-module-title a",".td-excerpt"), PatternsNames = c("Title","Content"))

dataUrlfilter , crawlUrlfilter <- are useful to filter URLs to be crawled and collected by Regex