

Introduction to Programming

Extra Lecture: Web-scrapping with R

Clément Mazet-Sonilhac

clement.mazet@sciencespo.fr

SciencesPo Paris - Collège de France

Preliminary stuff

Install and call packages

- `install.packages("rvest")`
- `install.packages("tidyverse")`
- Install a CSS selector on you web browser : **Scrapemate** if you're using Firefox or **SelectorGadget** if you use Google Chrome

Web-scrapping

What and why ?

- **Web scrapping**: technique for converting the data present in unstructured format (HTML tags) over the web to the structured format which can easily be accessed and used.

Web-scraping

What and why ?

- **Web scraping**: technique for converting the data present in unstructured format (HTML tags) over the web to the structured format which can easily be accessed and used.
- **Why do we need web scraping ?** \Rightarrow Find data that does not exist elsewhere !

Web-scrapping

What and why ?

- Davis and Dingell (JPE, 2019) : use Yelp to look *racial* segregation in consumption
- Cavallo and Rigobon (JEP, 2016) : “Billion Prices Project” collects prices from online retailers to look at macro price changing issues; also Cavallo (ReStat, 2018) “Scraped Data and Sticky Prices”
- Halket and Pignatti (JME, 2015) : scrape Craigslist to better understand US rental market
- Many papers on eBay / Alibaba
- Edleman, B. “Using Internet Data for Economic Research” (JEP, 2012) : useful discussion of many issues

Web-scrapping

How ?

- **Human Copy-Paste** : This is a slow and inefficient way of scraping data from the web. This involves humans themselves analyzing and copying the data to local storage.

Web-scrapping

How ?

- **Human Copy-Paste** : This is a slow and efficient way of scraping data from the web. This involves humans themselves analyzing and copying the data to local storage.
- **Text pattern matching** : Another simple yet powerful approach to extract information from the web is by using regular expression matching facilities of programming languages.

Web-scrapping

How ?

- **Human Copy-Paste** : This is a slow and efficient way of scraping data from the web. This involves humans themselves analyzing and copying the data to local storage.
- **Text pattern matching** : Another simple yet powerful approach to extract information from the web is by using regular expression matching facilities of programming languages.
- **DOM Parsing** : By using web browsers, programs can retrieve the dynamic content generated by client-side scripts.

Preliminary stuff

What is a web page ?

- Web-page : combination of (i) [HTML](#) and (ii) [CSS](#)

Preliminary stuff

What is a web page ?

- Web-page : combination of (i) [HTML](#) and (ii) [CSS](#)
 - [HTML](#) is the standard markup language for creating Web pages.
 - [CSS](#) is the language used to style a Web page. Describes how HTML elements are to be displayed on screen
- Quick example : [click here](#)

Scrapping a web page using R

First step

- Lets parse the website of the French newspaper Le Monde!

```
#Loading the rvest package:
```

```
library('rvest')
```

```
#Specifying the url for desired website to be scraped
```

```
url = 'https://www.lemonde.com'
```

```
#Reading the HTML code from the website
```

```
webpage <- read_html(url)
```

Scrapping a web page using R

Second step

- Get the title of the 10 most read articles !
- Use the `CSS selector` to get the markup (i.e., the name of the element) you're interested in [\[tuto here\]](#)

Scraping a web page using R

Third step

- Get the title of the 10 most read articles !

```
# Get the value of the accurate markup :
title = html_nodes(webpage, '.old__top-article
.article__title')
# Extract the text:
text_title = html_text(title)
# Show result:
text_title
```

Scrapping a web page using R

- **Exercise 1** : do the same with the *Opinions* titles of the New York Times (<https://www.nytimes.com/>)

Scraping a web page using R

- **Exercise 2** : find the markups for price, surface and location of the PAP website (<https://www.pap.fr>) or by clicking [\[here\]](#)

Scrapping a web page using R

- **Exercise 2** : find the markups for price, surface and location of the PAP website (<https://www.pap.fr>) or by clicking [\[here\]](#)
- price ?

Scraping a web page using R

- **Exercise 2** : find the markups for price, surface and location of the PAP website (<https://www.pap.fr>) or by clicking [\[here\]](#)
- price? \Rightarrow `.item-price`
- surface?

Scraping a web page using R

- **Exercise 2** : find the markups for price, surface and location of the PAP website (<https://www.pap.fr>) or by clicking [\[here\]](#)
 - price? \Rightarrow `.item-price`
 - surface? \Rightarrow `li~ li+ li`
 - location?

Scraping a web page using R

- **Exercise 2** : find the markups for price, surface and location of the PAP website (<https://www.pap.fr>) or by clicking [\[here\]](#)
 - price? \Rightarrow `.item-price`
 - surface? \Rightarrow `li~ li+ li`
 - location? \Rightarrow `.h1`

Scrapping movies data

- We want to gather data about the Top 50 most popular movies in 1998 (US box office).
- Use the website IMDb [[link here](#)]

Scrapping movies data

- Same process as before :

```
# If not already done, load the rvest package:
library('rvest')
#Specifying the url for desired website to be scraped
url = 'https://www.imdb.com/search/title_[...].com'
#Reading the HTML code from the website
webpage <- read_html(url)
```

Scrapping movies data

- Step 1 : Get the ranking of the movies :

```
# Get the value of the accurate markup :
rank_html = html_nodes(webpage, '.text-primary')
# Extract the text:
rank = html_text(rank_html)
# Show result:
head(rank)
# Transform into numeric data:
rank = as.numeric(rank)
```

Scrapping movies data

- Step 2 : Get all relevant information we need
 - Gross revenue (**gross**) : the amount of money earned
 - Run time (**runtime**) : how long is the movie ?
 - Genre (**genre**) : Drama, action, adventure ?
 - IMDb rating (**rating**) : proxy for movie's quality

Scrapping movies data

- Step 3 : Clean the data. Example with `gross` :

```
#Data-Preprocessing: removing '$' and 'M' signs:
```

```
gross<-gsub("M","",gross)
```

```
gross<-substring(gross,2,6)
```

```
gross = as.numeric(gross)
```


Scrapping movies data

- Step 3 : Another example with `genre` :

```
# Data-cleaning: removing \n
genre<-gsub("\n","",genre)

# Data-cleaning: removing excess spaces
genre<-gsub(" ","",genre)

# Taking only the first genre of each movie
genre<-gsub(",.*","",genre)

# Converging each genre from text to factor
genre<-as.factor(genre)
```

Scrapping movies data

- Step 4 : put everything all together in a dataframe

```
# Combining all the lists to form a data frame:
```

```
movies_df<-data.frame(Rank = rank,  
                      Runtime = runtime,  
                      Genre = genre, Rating = rating,  
                      Revenue = gross)
```

```
# Check your final dataframe:
```

```
str(movies_df)
```

Scrapping movies data

- Step 5 : Explore, plot, regress !

Scrapping movies data

- Step 5 : Explore, plot, regress !
 - **Question :** is it profitable to make good movies ?