ECPR Methods Summer School: Automated Collection of Web and Social Data

Pablo Barberá

School of International Relations University of Southern California pablobarbera.com

Networked Democracy Lab www.netdem.org

Course website:

github.com/pablobarbera/ECPR-SC103

Scraping the web

Scraping the web: what?

An increasing amount of data is available on the web:

- Speeches, sentences, biographical information...
- Social media data, newspaper articles, press releases...
- Geographic information, conflict data...

These datasets are often provided in an unstructured format.

Web scraping is the process of extracting this information automatically and transforming it into a structured dataset.

Scraping the web: why?

- Copy & pasting is time-consuming, boring, prone to errors, and impractical for large datasets
- In contrast, automated web scraping:
 - 1. Scales well for large datasets
 - Is reproducible
 - 3. Involved adaptable techniques
 - Facilitates detecting and fixing errors
- When to scrape?
 - Trade-off between your time today and your time in the future. Invest in your future self!
 - 2. Computer time is cheap; human time is expensive

Scraping the web: two approaches

Two different approaches:

- Screen scraping: extract data from source code of website, with html parser and/or regular expressions
 - rvest package in R
- 2. Web APIs (application programming interfaces): a set of structured http requests that return JSON or XML data
 - httr package to construct API requests
 - Packages specific to each API: weatherData, WDI,
 Rfacebook... Check CRAN Task View on Web Technologies and Services for examples
 - More on APIs tomorrow

The rules of the game

- Respect the hosting site's wishes:
 - Check if an API exists or if data are available for download
 - Keep in mind where data comes from and give credit (and respect copyright if you want to republish the data!)
 - ▶ Some websites disallow scrapers on robots.txt file
- Limit your bandwidth use:
 - Wait one or two seconds after each hit
 - Scrape only what you need, and just once (e.g. store the html file in disk, and then parse it)
- 3. When using APIs, read documentation
 - Is there a batch download option?
 - Are there any rate limits?
 - Can you share the data?

The art of web scraping

Workflow:

- 1. Learn about structure of website
- 2. Choose your strategy
- 3. Build prototype code: extract, prepare, validate
- 4. Generalize: functions, loops, debugging
- Data cleaning

The art of web scraping



ingilp con

Three main scenarios

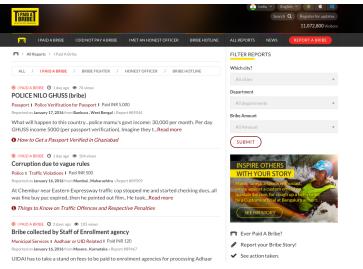
1. Data in table format



List of international courts [edit]

Name	+	Scope \$	Years active	Subject matter \$
International Court of Justice		Global	1945-present	General disputes
International Criminal Court		Global	2002-present	Criminal prosecutions
Permanent Court of International Justice		Global	1922-1946	General disputes
Appellate Body		Global	1995-present	Trade disputes within the WTO
International Tribunal for the Law of the Sea		Global	1994-present	Maritime disputes
African Court of Justice		Africa	2009-present	Interpretation of AU treaties
African Court on Human and Peoples' Rights		Africa	2006-present	Human rights
COMESA Court of Justice		Africa	1998-present	Trade disputes within COMESA
ECOWAS Community Court of Justice		Africa	1996-present	Interpretation of ECOWAS treaties
East African Court of Justice		Africa	2001-present	Interpretation of EAC treaties
SADC Tribunal		Africa	2005-2012	Interpretation of SADC treaties
0.21 0.11.0		^ ···		

2. Data in unstructured format



www.ipaidabribe.com/reports/paid

3. Data hidden behind web forms



Candidates on 2015 Venezuelan parliamentary election

- 1. Data in table format
 - Automatic extraction with rvest
- 2. Data in unstructured format
 - ▶ Element identification with selectorGadget
 - Automatic extraction with rvest
- 3. Data hidden behind web forms [TOMORROW]
 - Automation of web browser behavior with selenium

HTML: a primer

Hypertext Markup Language (HTML): hidden standard behind every website.

HTML is text with marked-up structure, defined by tags:

What you see in your browser is an interpretation of the HTML document

HTML: a primer

- Some common tags:
 - ▶ Document elements: <head>, <body>, <footer>...
 - ▶ Document components: <title>,<h1>, <div>...
 - ► Text style: , <i>, ...
 - Hyperlinks: <a>
- An example: www.pablobarbera.com

Beyond HTML

Cascading Style Sheets (CSS): describes formatting of HTML components (e.g. <h1>, <div>...), useful for us!



Javascript: adds functionalities to the website (e.g. change content/structure after website has been loaded)

Parsing HTML code

First step in webscraping: read HTML code in R and parse it

- Parsing = understanding structure
- How? rvest package in R:
 - read_html: parse HTML code into R
 - html_text: extract text from HTML code
 - html_table: extract tables in HTML code
 - html_nodes: extract components with CSS selector
 - html_attrs: extract attributes of nodes
- ► How to identify relevant CSS selectors?
 selectorGadget extension for Chrome and Firefox.