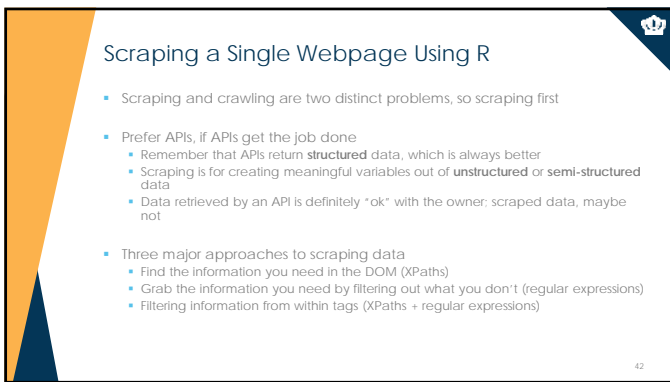




41

Executing a Web Scraping Project

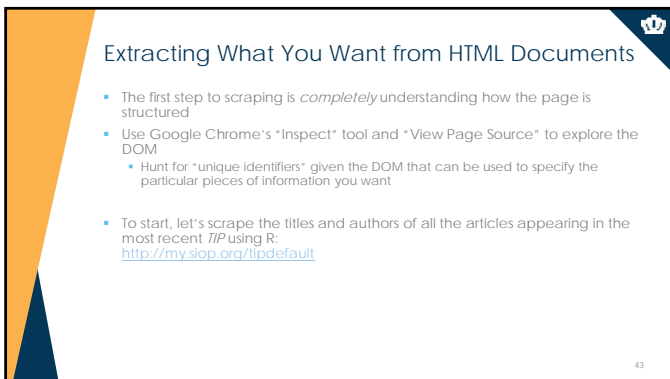
How do I scrape webpages?
How do I crawl webpages?



42

Scraping a Single Webpage Using R

- Scraping and crawling are two distinct problems, so scraping first
- Prefer APIs, if APIs get the job done
 - Remember that APIs return **structured** data, which is always better
 - Scraping is for creating meaningful variables out of **unstructured** or **semi-structured** data
 - Data retrieved by an API is definitely "ok" with the owner; scraped data, maybe not
- Three major approaches to scraping data
 - Find the information you need in the DOM (XPath)
 - Grab the information you need by filtering out what you don't (regular expressions)
 - Filtering information from within tags (XPath + regular expressions)



43

Extracting What You Want from HTML Documents

- The first step to scraping is *completely* understanding how the page is structured
- Use Google Chrome's "Inspect" tool and "View Page Source" to explore the DOM
 - Hunt for "unique identifiers" given the DOM that can be used to specify the particular pieces of information you want
- To start, let's scrape the titles and authors of all the articles appearing in the most recent *TiP* using R:
<http://my.slop.org/tip/default>

Crawling Across Multiple Documents

- Crawling refers to the page-by-page traversal of a particular target set of webpages (also called spidering)
 - Can be very specific, e.g., a list of webpages to consider
 - Can be very general, e.g., a domain name
 - For maximum data quality with the least headaches, you usually want the most specific criteria that get you all the data you want
- If possible, generate a list of specific pages
- If not, you'll need to create an algorithm
 - Involves recursively scraping all of the links on every page of a target site
 - Usually includes both inclusionary criteria and exclusionary criteria

44

Crawling the Current Issue of TIP

- Starting at <http://my.slop.org/tipdefault>, how would you develop rules for inclusion and exclusion?
 - First, determine inclusionary criteria
 - *Mouseover* all links to the sorts of pages you're interested in, and see what's in common between them
 - Alternatively, scrape all the links on a single page and look at them
 - You've already done it! Let's look at that CSV again
 - Second, determine exclusionary criteria
 - Most common when you have modified links for printing or special views, e.g., <http://somewhere.com/link.asp?id=1232312> vs <http://somewhere.com/link.asp?id=1232312&print=TRUE>
- Let's try it in *R*

45

Crawling then Scraping

- This was the easiest type of crawling: there is a single link of URLs that you can scrape individually
- Recursive crawling is the hardest: any webpage you crawl may contain *new* links that in turn need to be crawled. To do this, you'll need to:
 - Crawl an initial set of webpages/link
 - Within each of those webpages, scrape all embedded links
 - Process links according to inclusionary/exclusionary rules
 - Create a new list of "scrape next" links
 - Return to step 1 with new list

46

This is Why You Want an API

- Crawling/scraping is more complicated than API requests because you are restricted by:
 - Often poorly written webpages that are non-compliant with the HTML standards (to see if you're crazy, check <https://validator.w3.org>)
 - Nonsensical pagination and naming conventions
 - Dynamic webpages that don't create distinct URLs (<http://www.slop.org/obnet/default.aspx>)
 - Server-side restrictions, such as crawling speed
 - Your own coding skill, attention to detail, and patience
- R is also not particularly well-suited for crawling
 - This is where I suggest you turn to the *scrapy* library in Python



47

To Learn More Technical Bits

- For general information about both *R* and *Python*, I strongly recommend <http://datacamp.com>
- General Crawling/Scraping Frameworks
 - To learn how to use *scrapy* with Python, I recommend my tutorial: <http://flanders.net/scrapy/tutorial.html>
 - The other big library for web crawling/scraping in Python is *Beautiful Soup*: <https://www.crummy.com/software/BeautifulSoup/>
- Parsing
 - To learn basic HTML and CSS: <https://www.codecademy.com/learn/web>
 - To learn how to use XPath: <http://www.w3schools.com/xpath/>
 - To learn how to use regular expressions: <https://regexone.com/>

48
