Web Scraping & Web Services

Duncan Temple Lang, UC Davis
Deborah Nolan, UC Berkeley

Sponsored by
National Science Foundation
Data Sciences Initiative, UC Davis
Institute for Social Sciences, UC Davis
Welcome

Students from many different institutions

Opportunities and Challenges.

So many sources of data on the Web

We can ask new questions with these data

We can use them as auxiliary data when analyzing other data, e.g. weather information, median household income, census data, geolocating IP addresses, ...
Automation & Repeatability.

You can manually cut and paste data from Web pages.

Our focus today is to get data programmatically, i.e., *not* us manually finding the relevant data on each page and repeating the process, but the computer doing this.

We automate accessing data over 10s, 100s, 1000s… of pages.

This makes things feasible, less error-prone, repeatable - by both us and others.
Schedule & Topics

Lot to pack into a single day. So we will introduce a broad range of topics that cover almost all you will encounter when accessing data from the Web.

We won’t go into as much depth as we would like, but enough to get you started on any of the common approaches, and you can find additional information and details in books (e.g., XML and Web Technologies for Data Science with R, by us), and Web pages, blogs, etc.
Semi-structured data (HTML) & Structured data (Web Services/APIs) using XML and JSON

Core Technologies
- HTML - hyper text markup language
- XPath
- Simple CSS-related concepts.
- JavaScript
- HTTP/HTTPS - hyper text transfer protocol & Web requests
- REST - Representational State Transfer.
- XML - eXtensible Markup Language
- JSON - JavaScript Object Notation
- Authentication - OAuth 1.0 & 2.0
Tools are not just for scraping

In almost everything we do, there are two steps

Retrieving a document from a remote Web server

Processing its contents locally.

Increasingly common, we can retrieve entire collections of documents in a single step, and then process them locally, omitting the first step.

These formats & tools (XML, JSON, XPath, ...) apply to local data, not just scraping.

Even apply to Word processing, Spreadsheet, presentation documents.
Before you scrape…

Scraping from HTML pages is semi-structured, at best!

Unstable over time - format & structure of a Web site can change and where the data are on the page is different.

So try to find the data in a better form, e.g. CSV files, SQL database format, collection of XML or JSON files, or a Web Service.

Before scraping lots of pages on the same site, ask the owners if they can share the data in a better format.

They don’t want you making lots of requests for individual pages, occupying their Web server.
Speaking of occupying the server...

Before you scrape pages, make certain you are allowed to do this. Check the Terms of Service (TOS) of the site. Many prohibit “programmatic” access of the site.

It is easy for them to detect non-human requests and they can block your account, IP address, or even the entire sub-network your machine is on.

Some sites will let you scrape, but will limit the number of requests per day, per 15 minutes, etc.

Have to make your code capable of dealing with “temporary rejection”.
Make certain your code is robust before scraping many pages from a site

Handle different structure on pages and errors.

Otherwise, you may scrape lots of pages and get no data

Then you have to fix the code and make the same requests!
Has this already been done?

Before digging into the details, see if somebody has already done this and either use that or adapt it (and then share the updates with the community.)

For different Web services, there may already be an R package (or Python module, …) that provides a convenient interface to the API.

Check CRAN, rOpenSci, blogs that illustrate the basic steps for a particular Web site,…

When you do something that others might find useful, collect the code into an R package and submit to CRAN, or put on github.com
Topics

- Reading data from Tables in an HTML document
- Reading Links from HTML documents
- Scraping less-structured, less-generic data from HTML
- Anatomy of HTML documents
- Parsing HTML documents into a tree/hierarchy
- Accessing elements in the tree
  - XPath for querying collections of matching elements
- Manipulating individual elements, e.g., R functions `xmlGetAttr()`, `xmlValue()`, ..
Separating retrieving and parsing HTML
Understanding & Debugging HTTP requests - tools
HTML forms - GET and POST requests
  getForm(), postForm() (… httpPUT())
  Customizing queries with HTTP settings
    followlocation, useragent, cookies, userpwd, ssl certificates, httpheader
Dynamic HTML with JavaScript using RSelenium
Web Services
  REST concepts
  Public/anonymous APIs
  Using a personal key
  Full-fledged authentication - OAuth 1.0 and 2.0
Examples

Tables: Wikipedia population of country page


Scraping generally
  text: Supreme court nomination transcripts.
  walmart, cybercoders job posting*

HTML forms:
  CPI http://www.rateinflation.com/consumer-price-index/usa-historical-cpi
  RITA http://www.transtats.bts.gov/DL_SelectFields.asp?Table_ID=236

Complex: PEMS
  login form
    http://pems.dot.ca.gov/?dnode=VDS&content=detector_health&tab=dh_raw&station_id=401079
Dynamic HTML via JavaScript - Walmart???

Public/anonymous REST -
   http://climatedataapi.worldbank.org/climateweb/rest/v1

Personal Key API - Zillow, NYTimes, NOAA

OAuth 1.0 - Yelp API

OAuth 2.0 - Twitter’s newer API.
Let’s get started on the topics....
Understanding & Debugging HTTP Requests

How can we determine what we need to send to the Web server?

Use the Web browser and watch what it does.

Enable the Developer Tools in the Web browser.
We may know what the server expects, but how do we see what the Web server sees when we send a request?

We could run a Web server ourselves and a receiver that accepts the input.

Done generally by http://requestb.in
HTML Forms

We have been looking at getting the contents of a URL

Often we can specify additional inputs to customize the contents of the resulting page, e.g., start and end dates, search query string, name of state.

2 typical types of HTTP requests when submitting a form
- GET and POST (also PUT)

GET - appends the name-value parameters as base_url?name1=value&name2=value&name3=value..