NAVAL
POSTGRADUATE
SCHOOL

OS4118
Statistical and Machine Learning

Web Scraping

Prof. Sam Buttrey

Fall AY 2020

Some Admin Notes

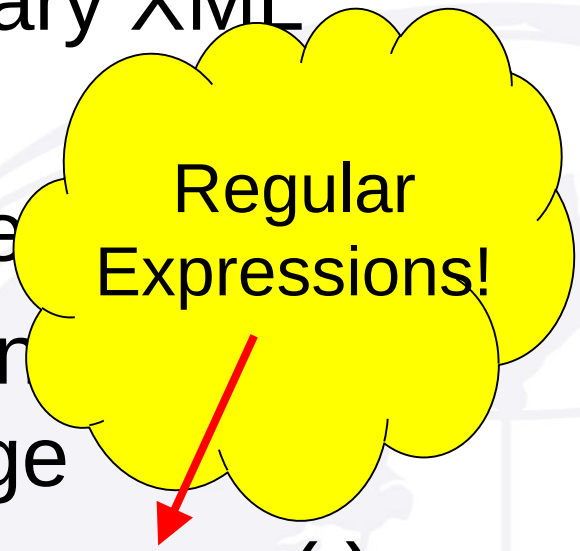
- Late homeworks are okay…
  - …up to a point
  - This is week 8 of 10; expect a homework after the last meeting

Parsing a Table

- `readHTMLTable` from library XML works – or doesn't
- A custom solution is often needed
- Page layouts rarely documented, presumably subject to change
- When needed, in R, I use `regexpr()` to seek tags, then home in on the target
  - E.g. `gs.reader (url = game.list.20142015[123])`

Parsing a Table

- `readHTMLTable` from library XML works – or doesn't
- A custom solution is often ne
- Page layouts rarely documen presumably subject to change
- When needed, in R, I use `regexpr()` to seek tags, then home in on the target
  - E.g. `gs.reader (url = game.list.20142015[123])`



Acquiring data files (cont'd)

- `wget` runs in the background, can follow links, supports retrying on drop…lots of options (for R, must be on executable PATH)
  - For servers, not ordinary machines on the cluster – for those, use `sftp`
  - Alternatives: widely-used `libcurl`, in R {RCurl}, {curl}, {httr}, Python (pycurl); R's internal `download.file()`

Dynamic Data: Web Services

- Term for procedures to communicate between programs via the web
- Two Major Approaches
  - **SOAP** (“simple object access protocol”)
    - Not, in fact, simple
  - **REST** (“representational state transfer”)
    - Build on web-based APIs, “application programming interfaces” that describe permitted transactions

SOAP

- Big, complex standard protocol; typically for large enterprises
  - Interface definition in XML
  - Applications send messages which can ask for data but can also execute “remote procedure calls”
  - Server may maintain “state” – that is, know which transactions have gone before
  - Rigid spec. suitable for formal interaction

REST

- An architecture, not a specific standard
  - Often informal
- For data rather than function calls
  - Usually delivered as HTML, XML or JSON
- Conducted over the web using http
- Server is stateless – the client maintains all information about its session
- System is scalable – new servers can be added invisibly

REST via HTTP

- REST is conducted via HTTP requests
  - GET and POST are the two we use
  - PUT, DELETE and a few others available
- A formal API will lay out whether we should GET or POST and what parameters should be specified
  - These will be name-value pairs
- If no spec is available, we have to deduce that information

GET vs POST

- In a **GET** request, the name-value pairs are sent in the URL
  - More insecure, limited to ~2048 bytes
  - Cached by browser, stored in history
  - Reload okay; ASCII only; bookmarkable
- In a **POST**, n-v pairs go in the body
  - Any size; binary data permitted
  - Nothing saved or cached
  - Reload will re-submit data (w/warning)

HTML Forms

- Enclosed in `<form></form>`
- Include standard interface “widgets” (building blocks): drop-down lists, radio buttons, checkboxes…
- These items define name=value pairs that get sent to the server:
  - name= selected `<option>` for drop-downs
  - name=ON for checkboxes
  - name=value for radio, submit, hidden

Scripts in HTML

- HTML scripts are normally in JavaScript
- Invoked with methods like `onSubmit()`, `onClick()` etc.
- Allow you (the client) to process input before contacting the server
  - Ensure data is complete and correct
- JavaScript is a complete language that is no relation to Java

Client and Server

- The client is the machine requesting
- Client's browser displays a page with HTML including one or more `<form>`s
- Frequently there will be `<script>`s (probably in JavaScript), for error checking etc.
- The form plus the scripts construct a request to the server with parameters specified as name=value pairs

The Request

- (In our world) the request is either a GET or a POST, specified in the `<form>` tag
- E.g. CDC: `<form action= "mmwrmorb2.asp" method="get">`
- We need to know which type it is, but otherwise don't really care
- In either case, name=value pairs are specified by the form controls

Parameters

- Name=value pairs can be passed in two ways
- For GET, they are in the “query string,” following a question mark in a web addr.
  - Pairs separated by &; special characters encoded (in one of two different ways!)
- For POST a string of the same format is sent in the request proper

Simple Example (client)

```
<form name="f" action="http:/srver.com/script"
  method= "get">
Your state? <select name="a">
  <option>CA</option><option>Not</option>
  </select>
Type:<input type="text" name="tt" size=10>
<input type="hidden" name="Bunnies"
  value="23">
<input type="submit" value="Go!"></form>
```

- Embed in HTML, click on “submit” and…

Server

- The web server receives the request and processes it, typically with a CGI (“Common Gateway Interface”) script
- These are written in VBScript, Perl…
- This script “unpacks” the request and processes it, possibly by calling other resources (database server, …)
- The item returned to the caller is (typically) a complete HTML page

The Request

- Suppose I choose `Not` for a, Test for `tt`
- The name-value pairs of this GET request are visible in the URL bar:

```
http://server.com/script?a=Not&tt=Test&Bunnies=23
```

- This URI acts like a regular web page and can be stored and re-accessed

Simple Example (server)

- PERL language example
- Grab the name, value pairs and do something with them (here we just return them)
- Return a complete HTML page (plus a content line at the top) by writing, e.g., `Content-type: text/html\n\n` + `<html>`, `<body>`...`</html>` etc. tags

Calling from R

- The `getForm()`, `postForm()` `{Rcurl}`, `GET()`, `POST()` `{httr}` functions emulate clicking on `Submit` buttons
- `getURI()`, `getURLContent()` retrieves the material at a URI
- Either way, substantial processing usually remains to be done!

REST API

- A REST Applications Programmer Interface (API) is the specification for fetching data from a REST source
- Check out, e.g., ProgrammableWeb.com
- Read, understand and respect the Terms of Use!

Example: REST API (2)

- Example: Gov't API for energy costs
  - National Renewable Energy Lab
- Sign up in advance for API access
- GET from `httr` with authenticate, sending the API Key in every request
  - Response in JSON or XML
  - Needs parsing
  - Example!

More Issues

- Lots of the data on the web is in PDF or Excel – programmatically handling it is harder than HTML, XML, or JSON
  - DoD likes PowerPoint as a data tool (!)
- Flash, video, graphics are beyond me…
- …But we also see a lot of HTML pages
- Unpacking is often not trivial

Case Study (cont'd)

- CDC Mortality data : `http://wonder.cdc.gov/mmwr/mmwrmort.asp`
- Each call produces mortality data for all locations for some times (up to 2016), or all times (up to 2016)for some locations
- How do we get data for **all** locations, **all** years?