High Powered Data and Development Economics

Scraping the Web to Generate Unique Datasets

Damian Clarke

November 24, 2013

Why Python?

- Free
- Power over the *whole* operating system
  - Imagine if Stata had control over Firefox, image editing, Google Earth, better scientific libraries, . . .
- Quite easy to get up and scraping the web (we'll do it in 20 mins)
- If you decide you like it, it can do everything for you
  - Kevin Sheppard's course, John Stachurski and Sargent's course
- Signalling?

What Do You Need?

- Unix or OS X: nothing!
- Windows: In many distributions Python is not installed by default
    - For complete packages, install Anaconda (http://continuum.io/)
- It may also be useful to install a stand alone text editor with syntax highlighting (ie gedit)

How to Run Python

- A number of ways: from the command line, interactively, using ipython
- For the interests of time, we'll just run from the command line
  - However, if you're going to run this frequently, ipython is worth checking out
- If you're interested in following along online (without downloading Python to your local machine), go to http://py-ide-online.appspot.com/

What is Web Scraping?

Essentially, the process of harvesting data that is directly stored on the web in an irregular or highly disperse format.

- When undertaking econometric analysis, we of course want very regular data, formatted into lines and columns
- Generally two steps:
  - Looping through nested urls to get to (many) source html pages
  - Taking html and formatting into a useful structure
- There are a number of tools people use for this sort of analysis: Python, R, RapidMiner, even Matlab ...

Why do we care?

- Often (particularly in developing country settings) data is not stored directly as a csv
- In some cases, data does not yet exist in any centralised form
- This opens up many entirely different types of data we mightn't have previously thought about
- The majority of economics papers are now using 'novel' data (ie not survey based)

What can we do with it?

- It has come in handy for me many times
  - Download, unzip and merge 1000+ DHS surveys, up to date at the second that scraping takes place
  - Download all (30,000+) papers on NBER for text analysis
  - Download election results: India, Philippines
  - Repeated calls to World Bank Data Bank
- And turns up frequently in cool development papers
  - Looking at effects of natural disasters
  - Looking at effects of ports
  - Night lights, geography, bombs, weather, ...

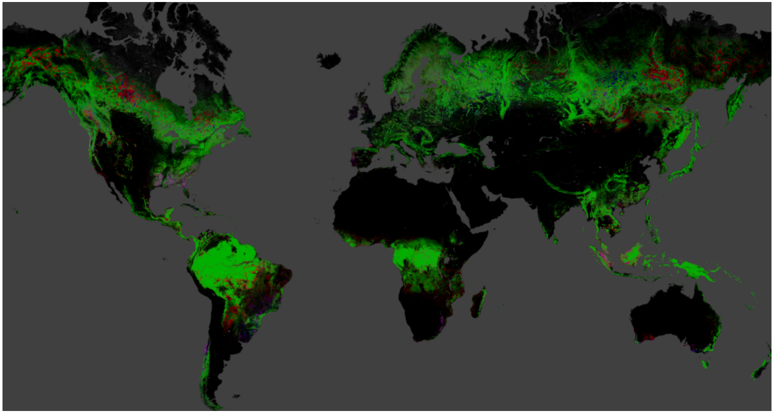

Figure 1: And it can look quite cool...

Hansen, M.C. et al (2013) High-Resolution Global Maps of 21st-Century Forest Cover Change. *Science* 342 (6160) 850-853.

Coding

We will go through a relatively simple (and contrived) example.

- For this process, there are a number of tools we will use:
    - Ideally, a web browser that lets us look at source code (pretty much any of them)
    - Regular Expressions (Python's `re`)
    - If this is a big job, we should think about error capture (Python's `try` command)

Basic Code

```
#  Scrape_xkcd 0.01             damiancclarke               yyyy-mm-dd:2013-11-21
#---|----1----|----2----|----3----|----4----|----5----|----6----|----7----|----8
#

#********************************************************************************
# (1) Import required packages, set-up names used in urls
#********************************************************************************
import urllib2
import re

target = 'http://www.xkcd.com'

#********************************************************************************
# (2) Scrape target url and print source code
#********************************************************************************
response = urllib2.urlopen(target)
print response
```

If you want to download the source code for the example we'll go through, go to http://users.ox.ac.uk/~ball3491/Python/

Complete Code

```python
# (1) Import required packages, set-up names used in urls
import urllib2
import re
target = 'http://www.xkcd.com'

# (2) Scrape target url and find the last comic number (num)
response = urllib2.urlopen(target)

for line in response:
    search = re.search('Permanent link to this comic:', line)
    if search!=None:
        lastcomic=re.findall('\d*', line)

for item in lastcomic:
    if len(item)>0:
        num = int(item)

# (3) Loop through all comics, finding each comic's title or capturing errors
for append in range(1, num+1):
    url = target + '/' + str(append)
    response = urllib2.urlopen(url)
    for line in response:
        search = re.search('ctitle',line)
        if search!=None:
            print line[17:-7]
```

Or, With Error Capture

```
#*******************************************************************************
# (3) Loop through all comics, finding each comic's title or capturing errors
#*******************************************************************************
for append in range(1, num+1):
    url = target + '/' + str(append)
    try:
        response = urllib2.urlopen(url)
        for line in response:
            search = re.search('ctitle',line)
            if search!=None:
                print line[17:-7]
    except urllib2.HTTPError, e:
        print('%s has http error' % url)
    except urllib2.URLError, e:
        print('%s has url error' % url)
```

Exporting Our 'Data'

Python is extremely capable at editing text to create output files:

```python
#*******************************************************************************
# (3) Loop through all comics, finding each comic's title or capturing errors
#*******************************************************************************
output = open('xkcd_names.txt', 'w')
output.write('Comic, Number, Title \n')

for append in range(1, num+1):
    url = target + '/' + str(append)
    response = urllib2.urlopen(url)
    for line in response:
        search = re.search('ctitle',line)
        if search!=None:
            print line[17:-7]
            output.write('xkcd,' + str(append) + ',' + line[17:-7] + '\n')

output.close()
```

Where to From Here

- You can actually get remarkably far with Python + a web browser + Regular Expressions!
- Some times you may want a more structured approach: Beautiful Soup
- Python can do much, much, much more
- Further applied examples at: bitbucket.org/damiancclarke
- Questions/comments?