Stat 133, Fall 2016, Web Scraping

Gaston Sanchez

November 18, 2016

Getting Started with Web Scraping

The purpose of this lab is to provide you with some web scraing practice using the R package XML:

```
# load XML
library(XML)
```

You'll be working with the **Mail Code List** pages from *UC Berkeley Mail Services*:

http://mailservices.berkeley.edu/incoming/mailcodes/list

Note that the first page shows the departments "90.7 FM, AAS, ..., AHMA" and their respective codes. Then, at the bottom of the page you'll see a series of links numbered `1, 2, 3, 4, 5, 6, 7, 8, 9, ..., next> last>>`

The purpose of this lab is to scrape all the deparments and their mail codes.

Looking at the HTML source code

Let's inspect the source html code. If you are using Chrome as web browser, one way to see the html source code is to click "View" on the menu bar, then click on "Developer", and select "View Source".

The first 3 departments (90.7 FM, AAS, and ACADEMIC ACHIEVEMENT PROGRAMS) should appear in lines 209-211 within nodes like these (inside an html table):

```
<tr class="odd">
  <td class="view-field view-field-node-title active">90.7 FM</td>
 <td class="view-field view-field-node-data-field-code-field-code-value">5650</td> </tr>
<tr class="even">
  <td class="view-field view-field-node-title active">AAS</td>
 <td class="view-field view-field-node-data-field-code-field-code-value">2572</td> </tr>
<tr class="odd">
 <td class="view-field view-field-node-title active">ACADEMIC ACHIEVEMENT PROGRAMS</td>
 <td class="view-field view-field-node-data-field-code-field-code-value">2410</td>
</tr>
```

Get a copy of the *mailcodes list* page

When doing web scraping, as with any other programming and data computing tasks, **always start small** (e.g. with some "toy" example). In particular, when dealing data from the web, whenever possible, I recommend you to first get a local copy of the xml/html document you want to work with:

```r
# mailcodes
url <- 'http://mailservices.berkeley.edu/incoming/mailcodes/list'

# download html file to your working directory
download.file(url, 'mailcodes.html')
```

Now let's **parse** the file `"mailcodes.html"` with the function `htmlParse()`

```r
# parsing html content
page1 <- htmlParse('mailcodes.html')

class(page1)
```

```
## [1] "HTMLInternalDocument" "HTMLInternalDocument" "XMLInternalDocument"
## [4] "XMLAbstractDocument"
```

Notice that `page1` is an object of class `"HTMLInternalDocument"`. This means that it is stored as an internal C-level object (NOT an R object). This is important because we are going to use **XPath** expressions which require an `"Internal"` type of object.

XPath expressions

- XPath is a language to navigate through the elements and attributes in an XML/HTML document.
- XPath uses path expressions to select nodes in an XML document.
- XPath has a computational model to identify sets of nodes (node-sets)

XPath expressions have a syntax similar to the way files are located in a hierarchy of directories/folders in a computer file system. For instance, we can match each department's name node with the following **XPath** pattern:

`'//td[@class="view-field view-field-node-title active"]'`

Likewise, we can match each department's code node with the following **XPath** pattern:

`'//td[@class="view-field view-field-node-data-field-code-field-code-value"]'`

Selecting Nodes

The package `"XML"` provides the function `getNodeSet()` which allows you to select html nodes that have a specific XPath pattern. For example, here's how you can get the `td` nodes with `class="view-field view-field-node-title active"`

```r
dept_names <- getNodeSet(page1, '//td[@class="view-field view-field-node-title active"]')
```

If you take a look at `dept_names`, you will see a list that contains the html code of the `td` elements (i.e. nodes) of the department names.

To extract the actual names of the departments, you can use `sapply()`, with the function `xmlValue()`

```r
sapply(dept_names, xmlValue)
```

```
##  [1] "90.7 FM"
##  [2] "AAS"
##  [3] "ACADEMIC ACHIEVEMENT PROGRAMS"
##  [4] "ACADEMIC COUNCIL NEWSLETTER"
##  [5] "ACADEMIC FACILITIES OFFIC"
##  [6] "ACADEMIC PERSONNEL OFFICE"
##  [7] "ACADEMIC SENATE"
##  [8] "ACADEMIC TALENT DEVELOPMENT PROGRAM"
##  [9] "ACQUISITIONS"
## [10] "ACROSS THE SEA"
## [11] "ACUC"
## [12] "ADMISSIONS & ENROLLMENT ASSOC VC"
## [13] "AEROSPACE STUDIES"
## [14] "AF ROTC DETACHMENT 85"
## [15] "AFR AM STUD ASSOC OF PSYCHOLOGY"
## [16] "AFRICAN AMERICAN HOUSE"
## [17] "AFRICAN AMERICAN STUDIES"
## [18] "AFRICAN STUDIES - CENTER FOR"
## [19] "AFRIKAN STUDENT LITERARY MAGAZINE"
## [20] "AFRO AMERICAN STUDIES"
## [21] "AG ECON"
## [22] "AGRICULTURAL AND RESOURCE ECONOMICS"
## [23] "AH&I OFFICE"
## [24] "AHMA"
## [25] "AIESEC"
```

Option 2

Another way to extract the names of the departments, is to directly use the function `xpathSApply()` on the object of class "HTMLInternalDocument". Instead of extracting the nodes, you simply specify the XPath expression, and pass the function `xmlValue`

```r
# department names
xpathSApply(
  doc = page1,
  path = '//td[@class="view-field view-field-node-title active"]',
  fun = xmlValue)
```

```
##  [1] "90.7 FM"
##  [2] "AAS"
##  [3] "ACADEMIC ACHIEVEMENT PROGRAMS"
##  [4] "ACADEMIC COUNCIL NEWSLETTER"
##  [5] "ACADEMIC FACILITIES OFFIC"
##  [6] "ACADEMIC PERSONNEL OFFICE"
##  [7] "ACADEMIC SENATE"
##  [8] "ACADEMIC TALENT DEVELOPMENT PROGRAM"
##  [9] "ACQUISITIONS"
## [10] "ACROSS THE SEA"
## [11] "ACUC"
## [12] "ADMISSIONS & ENROLLMENT ASSOC VC"
## [13] "AEROSPACE STUDIES"
## [14] "AF ROTC DETACHMENT 85"
## [15] "AFR AM STUD ASSOC OF PSYCHOLOGY"
## [16] "AFRICAN AMERICAN HOUSE"
## [17] "AFRICAN AMERICAN STUDIES"
## [18] "AFRICAN STUDIES - CENTER FOR"
## [19] "AFRIKAN STUDENT LITERARY MAGAZINE"
## [20] "AFRO AMERICAN STUDIES"
## [21] "AG ECON"
## [22] "AGRICULTURAL AND RESOURCE ECONOMICS"
## [23] "AH&I OFFICE"
## [24] "AHMA"
## [25] "AIESEC"
```

Your turn

How would you extract the Mail Code numbers of each department? Use both approaches:

- using `getNodeSet()`

```
# extracting dept mail codes via getNodeSet()
```

- using `xpathSApply()`

```
# extracting dept mail codes via xpathSApply()
```

Scraping the first 5 pages

Now that you have some working code, you can start generalizing it. Instead of just scraping the first page, try to crawl through the first 5 pages: 1, 2, 3, 4, 5

Your mission is to obtain a data frame with two columns: `department` and `code`, and as many rows as departments.

Brainstorm with your neighbors:

- How would you scrape the first 5 pages?
- Think about all the steps you need to perform
- Think about the data objects you may use to store the scraped content

---

Scraping all the pages

Instead of just scraping the first 5 pages, you'll have to crawl through every single page: 1, 2, 3, 4, …

- How many pages you have to crawl over?
- Try to write functions that help you get the job done

```
# scraping UC Berkeley Mail Codes
```

Optional Challenge

If you are done with this lab. Try work on the other file: `"lab11-more-xml-practice.Rmd"`

Solutions

You can match each department's code node with the following **XPath** pattern:

```r
# department codes via getNodeSet
mail_codes <- getNodeSet(
  page1,
  '//td[@class="view-field view-field-node-data-field-code-field-code-value"]'
)

sapply(mail_codes, xmlValue)


# department codes via xpathSApply
xpathSApply(
  doc = page1,
  path = '//td[@class="view-field view-field-node-data-field-code-field-code-value"]',
  fun = xmlValue)
```

Crawling first 5 pages

```r
# empty vectors to store results
deps <- character(0)
codes <- numeric(0)

for (i in 0:4) {
  # parsing each page
  page <- htmlParse(paste0('http://mailservices.berkeley.edu/incoming/mailcodes/list?page=', i

  # department names
  deps <- c(deps, xpathSApply(
    doc = page,
    path = '//td[@class="view-field view-field-node-title active"]',
    fun = xmlValue))

  # department codes
  codes <- c(codes, xpathSApply(
    doc = page,
    path = '//td[@class="view-field view-field-node-data-field-code-field-code-value"]',
    fun = xmlValue))
}

# assembling a data.frame
dep_code <- data.frame(
  department = deps,
  code = codes)
```