Digital Data Collection - The Hard Way

Rolf Fredheim

10/03/2015

Logging on

Before you sit down: - Do you have your MCS password? - Do you have your Raven password? - If you answered **'no'** to either then go to the University Computing Services (just outside the door) NOW! - Are you registered? If not, see me!

Download these slides

Follow link from course description on the SSRMC pages or go directly to `http://fredheir.github.io/WebScraping/`

Download the R file to your computer

Install the following packages:

- XML
- RCurl
- lubridate
- plyr
- stringr

Recap

Week1 - Basic principles of data collection - Basics of text manipulation in R - Simple scraping example

Week2 - Utility functions - JSON and APIs

What's the problem with APIs?

- Overkill
- Rate limiting
- Authentication
- Web content may be richer
- Deprecation
- `https://developers.google.com/youtube/2.0/developers_guide_protocol_deprecated`
- `https://blog.twitter.com/2013/api-v1-is-retired`

Today we will

Work with html to - extract tables - extract links - extract information using class and id tags - write a two-stage scraper to download newspaper articles

For that we will need

To use Google Chrome of Mozilla Firefox.

No Internet Explorer or Safari please!

Load the packages

```
require(lubridate)
require(plyr)
require(stringr)
require(XML)
require(RCurl)
```

Getting to know HTML structure

- `http://en.wikipedia.org/wiki/Euromaidan`
- `http://en.wikipedia.org/wiki/Boris_Nemtsov`

Let's look at this webpage

- Headings
- Images
- links
- references
- tables

To look at the code (in Google Chrome), right-click somewhere on the page and select 'inspect element'

Tree-structure (parents, siblings)

Back to Wikipedia

HTML tags.

They come in pairs and are surrounded by these guys: <>

e.g. a heading might look like this:

<h1>MY HEADING</h1>

MY HEADING

Which others do you know or can you find?

Extracting a table

can be a bit fiddly to format, but quite accessible:

```
url='http://en.wikipedia.org/wiki/Elections_in_Russia'
tables<-readHTMLTable(url)
head(tables[[6]])
```

```
            Candidates       Nominating parties      Votes     %
1       Vladimir Putin            United Russia 45,513,001 63.64
2     Gennady Zyuganov         Communist Party 12,288,624 17.18
3    Mikhail Prokhorov           self-nominated  5,680,558  7.94
4 Vladimir Zhirinovsky Liberal Democratic Party  4,448,959  6.22
5       Sergey Mironov            A Just Russia  2,755,642  3.85
6          Valid votes               70,686,784      98.84  <NA>
```

Download html

Dont print a webpage like this to the console!

Download using readLines() or getURL() (RCurl)

Rstudio will probably crash

use head(), str(), summary(), tail() etc. to inspect data.

```
url  <- "http://en.wikipedia.org/wiki/Boris_Nemtsov"
raw <-  getURL(url,encoding="UTF-8") #Download the page
#this is a very very long line. Let's not print it. Instead:
substring (raw,1,200)
```

```
[1] "<!DOC
```

```
PARSED <- htmlParse(raw) #Format the html code d
```

Accessing HTML elements in R with XPath

Is a language for querying XML

Reading and examples: - `http://www.w3schools.com/xpath/xpath_intro.asp` - `http://www.w3schools.com/xml/xml_xpath.asp`

we can use XPath expressions to extract elements from HTML

The element in quotes below is an *XPath expression* that selects the indicated node

```
xpathSApply(PARSED, "//h1")
```

```
[[1]]
<h1 id="firstHeading" class="firstHeading" lang="en">Boris Nemtsov</h1>
```

Extract content

Not so pretty. But! Specifying xmlValue strips away the surrounding code and returns only the content of the tag

```
xpathSApply(PARSED, "//h1",xmlValue)
```

```
[1] "Boris Nemtsov"
```

Fundamental XPath Syntax

- / Select from the root
- // Select anywhere in document
- @ Select attributes. Use in square brackets

```
xpathSApply(PARSED, "//h1",xmlValue)
```

```
[1] "Boris Nemtsov"
```

HTML tags

- <html>: starts html code
- <head> : contains meta data etc
- <script> : e.g. javascript to be loaded
- <style> : css code
- <meta> : denotes document properties, e.g. author, keywords
- <title> :
- <body> :

HTML tags2

- , :these are used to break up a document into sections and boxes
- <h1>,<h2>,<h3>,<h4>,<h5> Different levels of heading
- <p> : paragraph
-
 : line break
- and others: <a>, <ul>, <tbody>, <th>, <td>, <ul>, <ul>,

What about other headings?

```
xpathSApply(PARSED, "//h3",xmlValue)
```

```
 [1] "§Nemtsov's fears[edit]"
 [2] "§The assassination (27 February 2015)[edit]"
 [3] "§Aftermath, context and accusations[edit]"
 [4] "§Reactions[edit]"
 [5] "Personal tools"
 [6] "Namespaces"
 [7] "Variants"
 [8] "Views"
 [9] "More"
[10] "\n\t\t\t\t\t\t\tSearch\n\t\t\t\t\t\t\t"
[11] "Navigation"
[12] "Interaction"
[13] "Tools"
[14] "Print/export"
[15] "Languages"
```

Line ten... 'search'. Remember the clever str_trim() function?

```
temp=xpathSApply(PARSED, "//h3",xmlValue)

str_trim(temp[10])
```

```
[1] "Search"
```

Extracting links

and links

```
length(xpathSApply(PARSED, "//a/@href"))
```

```
[1] 800
```

That's hundreds of links!!! That's hopeless. We want to be more selective. Back to the drawing board

CSS and XPath

web-designers use Cascading Style Sheets to determine the way a webpage looks

Like variables: change the style, rather than the every item on a page

I use CSS for these slides, check out the code for this page

CSS allows us to make better selections, by latching onto tags

Xpath allows us to move up and down the html tree structure

CSS can be an html **attribute**

Principles of scraping

- Identify the tag
- Download the web-page
- Extract content matching the tag
- Save the content
- Optional: repeat

Get references

Content of the references

```
head(xpathSApply(PARSED, "//span[@class='reference-text']",xmlValue))
```

```
[3] "Birnbaum, Michael; Branigan, William (28 February 2015). \"Putin critic, Russian
[4] "http://www.foxnews.com/world/2015/03/04/crossing-kremlin-nemtsov-latest-in-long-
[5] "Amos, Howard; Millward, David (27 February 2015). \"Leading Putin critic gunned
[6] "Kramer, Andew E., \"Boris Nemtsov, Putin Foe, Is Shot Dead in Shadow of Kremlin,
```

URLS

```
head(as.character(xpathSApply(PARSED, "//span[@class=
  'reference-text']/span/a/@href")))
```

```
[1] "http://www.livelib.ru/author/208578"
[2] "http://www.kommersant.ru/doc/2678842"
[3] "/wiki/Kommersant"
[4] "http://www.washingtonpost.com/world/europe/russian-opposition-leader-boris-nemts
[5] "http://www.telegraph.co.uk/news/worldnews/europe/russia/11441466/Veteran-Russian
[6] "http://www.rosbalt.ru/moscow/2015/02/28/1372887.html"
```

Sanity test

Test that these work, using browseURL()

```
links <- (xpathSApply(PARSED, "//span[@class='reference-text']/span/a/@href"))
browseURL(links[1])
```

So if you wanted to, you could scrape these links in turn.

tree-structure is navigated a bit like that on your computer (c:/windows/system)

How it works

In this example, we select all elements of 'span' . . . Which have an **attribute** "class" of the value "citation news" . . . then we select all links . . . and return all attributes labeled "href" (the urls)

```
head(xpathSApply(PARSED, "//span[@class='reference-text']/span/a/@href"))
```

XPath2

Like in R, we use square brackets to make selections

What does this select?

```
head(xpathSApply(PARSED, "//span[@class='reference-text'][17]/span/a/@href"))
```

```
NULL
```

Wildcards

We can also use wildcards:

- * selects any node or tag
- @* selects any attribute (used to define nodes)

```
(xpathSApply(PARSED, "//*[@class='citation news'][17]/a/@href"))
```

```
NULL
```

```
(xpathSApply(PARSED, "//span[@class='citation news'][17]/a/@*"))
```

```
NULL
```

Scrape Telegraph

http://www.telegraph.co.uk/search/?queryText=nemtsov&sort=relevant

Note: this is a get request (see the ? and & characters?)

```
url <- 'http://www.telegraph.co.uk/search/?queryText=nemtsov&sort=relevant'
raw <-  getURL(url)#,encoding="UTF-8")
PARSED <- htmlParse(raw) #Format the html code d
links<-xpathSApply(PARSED, "//a/@href")
length(links)
```

```
[1] 211
```

```
links<-xpathSApply(PARSED, "//div[@class='searchresults']//a/@href")
length(links)
```

```
[1] 73
```

```
length(unique(links))
```

```
[1] 37
```

```
links<-unique(links)

links<-links[grep('http',links)]
```

Daily Mail

```
url <- 'http://www.dailymail.co.uk/home/search.html?sel=site&searchPhrase=nemtsov'
raw <-  getURL(url)#,encoding="UTF-8")
PARSED <- htmlParse(raw) #Format the html code d
links<-xpathSApply(PARSED, "//a/@href")
length(links)
```

```
[1] 284
```

```
links<-xpathSApply(PARSED, "//div[@class='sch-results']//a/@href")
length(links)
```

```
[1] 210
```

```
length(unique(links))
```

```
[1] 51
```

```
links<-unique(links)
```

dt2

```
head(links)
```

```
[1] "#"
[2] "/news/article-2987259/Litvinenko-s-killer-honoured-Putin-Russian-president-gives
[3] "/wires/reuters/article-2986593/Russias-Putin-honours-suspect-Litvinenko-poisonir
[4] "/wires/ap/article-2986551/Putin-honors-Chechen-leader-praised-Nemtsov-suspect.ht
[5] "/wires/reuters/article-2986264/Friend-says-Islamist-motive-killing-Nemtsov-nonse
[6] "/wires/ap/article-2986228/Putin-bestows-awards-Chechen-leader-Litvinenko-suspect
```

```
paste('http://www.dailymail.co.uk',links,sep='')
```

```
 [1] "http://www.dailymail.co.uk#"
 [2] "http://www.dailymail.co.uk/news/article-2987259/Litvinenko-s-killer-honoured-Pu
 [3] "http://www.dailymail.co.uk/wires/reuters/article-2986593/Russias-Putin-honours-
 [4] "http://www.dailymail.co.uk/wires/ap/article-2986551/Putin-honors-Chechen-leader
 [5] "http://www.dailymail.co.uk/wires/reuters/article-2986264/Friend-says-Islamist-m
 [6] "http://www.dailymail.co.uk/wires/ap/article-2986228/Putin-bestows-awards-Cheche
 [7] "http://www.dailymail.co.uk/wires/reuters/article-2986027/Nemtsov-friend-says-Is
 [8] "http://www.dailymail.co.uk/wires/ap/article-2985778/10-Things-Know-Monday--9-Ma
 [9] "http://www.dailymail.co.uk/news/article-2985383/Suspect-Nemtsov-killing-devout-
[10] "http://www.dailymail.co.uk/video/news/video-1165510/Suspects-in-killing-of-Bori
[11] "http://www.dailymail.co.uk/video/news/video-1165505/Boris-Nemtsov-shooting-susp
[12] "http://www.dailymail.co.uk/wires/reuters/article-2985103/Russian-court-orders-t
[13] "http://www.dailymail.co.uk/news/article-2985051/Five-suspects-court-assassinati
[14] "http://www.dailymail.co.uk/wires/reuters/article-2985011/Russian-authorities-ch
[15] "http://www.dailymail.co.uk/wires/ap/article-2984912/Nemtsov-killing-suspects-wh
```

Scrape BBC

Do the same for this page:

`http://www.bbc.co.uk/search?q=nemtsov`

Solution

```
url <- 'http://www.bbc.co.uk/search?q=nemtsov'
raw <-  getURL(url,encoding="UTF-8")
PARSED <- htmlParse(raw) #Format the html code d
links<-xpathSApply(PARSED, "//a/@href")
length(links)
```

```
[1] 114
```

```
links<-xpathSApply(PARSED, '//ol[@class="search-results results"]//a/@href')
length(links)
```

```
[1] 30
```

```
length(unique(links))
```

```
[1] 10
```

```
links<-unique(links)
```

Make function to get links

```
getBBCLinks <- function(url){
  raw <-  getURL(url,encoding="UTF-8")
  PARSED <- htmlParse(raw) #Format the html code d
  links<-unique(xpathSApply(PARSED,
      '//ol[@class="search-results results"]//a/@href'))
  return (links)
}


url='http://www.bbc.co.uk/search?q=putin'
links <-getBBCLinks(url)
links
```

```
 [1] "http://www.bbc.co.uk/news/world-europe-31794742"
 [2] "http://www.bbc.co.uk/news/world-europe-31796226"
 [3] "http://www.bbc.co.uk/news/world-europe-31792991"
 [4] "http://www.bbc.co.uk/news/world-europe-31794523"
 [5] "http://www.bbc.co.uk/news/uk-politics-31787910"
 [6] "http://www.bbc.co.uk/news/uk-politics-31787908"
 [7] "http://www.bbc.co.uk/news/election-2015-31787285"
 [8] "http://www.bbc.co.uk/news/uk-politics-31786906"
 [9] "http://www.bbc.co.uk/news/world-europe-31728623"
[10] "http://www.bbc.co.uk/news/uk-31809453"
```

BBC article scraper

```
url <- "http://www.bbc.co.uk/news/world-europe-26333587"
# Specify encoding when dealing with non-latin characters
SOURCE <-  getURL(url,encoding="UTF-8")
PARSED <- htmlParse(SOURCE)
```

Get the headline:

```
(xpathSApply(PARSED, "//h1[@class='story-header']",xmlValue))
```

```
[1] "Ukraine crisis: Turchynov warns of 'separatism' risk"
```

Get the date:

```
(xpathSApply(PARSED, "//span[@class='date']",xmlValue))
```

```
[1] "25 February 2014"
```

Make a scraper

```
bbcScraper <- function(url){
  date=''
  title=''
  SOURCE <-  getURL(url,encoding="UTF-8")
  PARSED <- htmlParse(SOURCE)
  title=xpathSApply(PARSED, "//title",xmlValue)
  date=xpathSApply(PARSED, "//meta[@name='OriginalPublicationDate']/@content")
  d=data.frame(url,title,date)
  return(d)
}

url='http://www.bbc.co.uk/search?q=putin'
links <-getBBCLinks(url)

dat <- ldply(links,bbcScraper)
```

Guardian

start with the bbc scraper as a base, then change the necessary fields

```
url <- "http://www.theguardian.com/environment/2015/mar/08/
  how-will-everything-change-under-climate-change"
  SOURCE <-  getURL(url,encoding="UTF-8")
  PARSED <- htmlParse(SOURCE)
  xpathSApply(PARSED, "//h1[contains(@itemprop,'headline')]",xmlValue)
```

```
[1] "\nHow will everything change under climate change?\n"
```

```
  xpathSApply(PARSED, "//a[@rel='author']",xmlValue)
```

```
[1] "Naomi Klein"
```

```
  xpathSApply(PARSED, "//time[@itemprop='datePublished']",xmlValue)
```

```
[1] "\nSunday 8 March 2015 12.00 GMT\n"
```

Guardian continued

```
  xpathSApply(PARSED, "//time[@itemprop='datePublished']/@datetime")
```

```
                  datetime
"2015-03-08T12:00:04+0000"
```

```
  xpathSApply(PARSED, "//li[@class='inline-list__item ']/a",xmlValue)
```

```
[1] "\nClimate change\n"          "\nGreenhouse gas emissions\n"
[3] "\nFossil fuels\n"            "\nEnergy\n"
```

```
  xpathSApply(PARSED, "//div[@itemprop='articleBody']",xmlValue)
```

```
[1] "\nThe alarm bells of the climate crisis have been ringing in our ears for years
```

Guardian scraper

```
guardianScraper <- function(url){
  SOURCE <-  getURL(url,encoding="UTF-8")
  PARSED <- htmlParse(SOURCE)
  headline = author =date = tags = body =''
  headline<-xpathSApply(PARSED, "//h1[contains(@itemprop,'headline')]",xmlValue)
  author<-xpathSApply(PARSED, "//a[@rel='author']",xmlValue)[1]
  date<-as.character(xpathSApply(PARSED,
      "//time[@itemprop='datePublished']/@datetime"))
  tags<-xpathSApply(PARSED, "//li[@class='inline-list__item ']/a",xmlValue)
  tags<-paste(tags,collapse=',')
  body<-xpathSApply(PARSED, "//div[@itemprop='articleBody']",xmlValue)
  d<- tryCatch(
    {
      d=data.frame(url,headline,author,date,tags,body)
    },error=function(cond){
      print (paste('failed for page',url))
      return(NULL)
    }
    )
}
```

Get Guardian links

scraping is getting harder

```
url='http://www.theguardian.com/world/russia?page=7'
getGuardianLinks <- function(url){
  raw <-  getURL(url,encoding="UTF-8")
  PARSED <- htmlParse(raw) #Format the html code d
  links<-unique(xpathSApply(PARSED, '//div[@class="fc-item__container"]//a/@href'))
  return (links)
}

links <- getGuardianLinks(url)
dat<-ldply(links[1:8],guardianScraper)
```

```
[1] "failed for page http://get.adobe.com/flashplayer/"
[1] "failed for page http://whatbrowser.org/"
[1] "failed for page http://www.theguardian.com/world/video/2015/feb/10/
    rockets-hit-residential-area-kramatorsk-ukraine-video"
[1] "failed for page http://www.theguardian.com/sport/2015/feb/10/
    marat-safin-denies-tax-evasion-allegations"
```

```
head(dat[,1:5])
```

```
1                    http://www.theguardian.com/world/2015/feb/10/minsk-summit-talks
2   http://www.theguardian.com/politics/2015/feb/10/uk-government-defends-role-in-ukr
3 http://www.theguardian.com/world/2015/feb/10/litvinenko-inquiry-suspect-drank-porn-
4    http://www.theguardian.com/world/2015/feb/10/ukraine-draft-dodgers-jail-kiev-str
```

Tidy the data

```
gsub('\n','',dat$tags)
as.Date(dat$date)
gsub('\n','',dat$headline)

dat$tags <-gsub('\n','',dat$tags)
dat$date <-as.Date(dat$date)
dat$headline <-gsub('\n','',dat$headline)
```

lubridate for dates

```
time="2015-02-09T14:59:32+0000"
ymd_hms(time)
```

```
[1] "2015-02-09 14:59:32 UTC"
```

```
time="\nMonday 23 February 2015\n"
dmy(time)
```

```
[1] "2015-02-23 UTC"
```

```
time='06/12/2013'
dmy(time)
```

```
[1] "2013-12-06 UTC"
```

```
mdy(time)
```

```
[1] "2013-06-12 UTC"
```

Practice time

write a scraper for

- `http://www.mirror.co.uk/`
- `http://www.telegraph.co.uk/`
- `http://www.independent.co.uk`

Solutions (1: Mirror)

Can you turn these into scraper functions? Mirror

```
#MIRROR
url <- "http://www.mirror.co.uk/news/world-news/
    oscar-pistorius-trial-murder-reeva-3181393"
SOURCE <-  getURL(url,encoding="UTF-8")
PARSED <- htmlParse(SOURCE)
title <- xpathSApply(PARSED, "//h1",xmlValue)
author <- xpathSApply(PARSED, "//li[@class='author']",xmlValue)
time  <- xpathSApply(PARSED, "//time[@itemprop='datePublished']/@datetime")
```

Telegraph

```
#Telegraph
url <- "http://www.telegraph.co.uk/news/uknews/terrorism-in-the-uk/10659904/
    Former-Guantanamo-detainee-Moazzam-Begg-one-of-four-arrested-on-suspicion
    -of-terrorism.html"
SOURCE <-  getURL(url,encoding="UTF-8")
PARSED <- htmlParse(SOURCE)
title <- xpathSApply(PARSED, "//h1[@itemprop='headline name']",xmlValue)
author <- xpathSApply(PARSED, "//p[@class='bylineBody']",xmlValue)
time  <- xpathSApply(PARSED, "//p[@class='publishedDate']",xmlValue)
```

Independent

```
#Independent
url <- "http://www.independent.co.uk/news/people/emma-watson-issues-feminist-
    response-to-prince-harry-speculation-marrying-a-prince-is-not-prerequisite-
    to-being-a-princess-10064025.html"
SOURCE <-  getURL(url,encoding="UTF-8")
PARSED <- htmlParse(SOURCE)
title <- xpathSApply(PARSED, "//h1",xmlValue)
author <- xpathSApply(PARSED, "//span[@class='authorName']",xmlValue)
time  <- xpathSApply(PARSED, "//p[@class='dateline']",xmlValue)
as.Date(str_trim(time),"%d %B %Y")
```

Where do we go from here

Scale up, store data, retrieve:

- R + sql + data.table
- python + mongodb
- python + selenium -> javascript and logons.

R for tables, vector operations, numbers.

Python for (irregular) objects, loops, text processing.