Web Data Collection with R

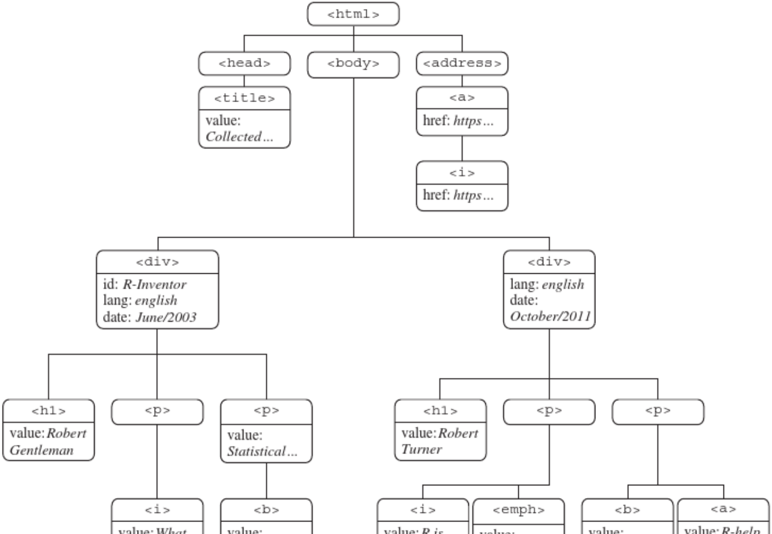
Xpath

Peter Meißner / 2016-02-29 – 2016-03-04 / ECPR WSMT

HTML/XML tree structure again

HTML/XML tree structure, nodes and attributes

http://www.r-datacollection.com/materials/html/fortunes.html



Running Example

running example

```r
require(rvest)
require(stringr)
```

```r
url <-
"http://pmeissner.com/downloads/fortunes.html"
fname <- basename(url)

if(!file.exists(fname)){
  download.file(url, fname)
}

html <- read_html(fname)
```

running example

How XPath works . . .

XPath? What is it all about?

- XPath is a query language for XML (Extensible Markup Language) documents
- XML examples are: XML, HTML, SVG, GML, KML, EPUB, RSS, Office Open XML, OpenDocument
- in XPath on selects nodes describing the paths that lead to that path

How XPath Works . . .

- builds on
  - **hierarchy** (select parent, child, sibling, . . . node)
  - **node names** (select node by name)
  - **node values** (select node by value)
  - **attribute name and value** (select node on attribute value)
  - **further functions** (select depending on more complex derivates of the above)
    - e.g. name, string_length, contains, count, position, . . .

How CSS-Selectors Work ...

How CSS-Selectors Work ...

- CSS-Selectors were designed to apply Styles to HTML elements
- While XPath is build around the idea of hierarchy and tree-structure first and foremost meaning that paths lead to data, with CSS-S selection is more set-like.
- CSS-S is used and written for Web-Designers so it might be less-powerful-complete-systematic than XPath but it is also less intimidating and easier to write
- selection on class and id attributes is super easy
  - **name** (select nodes by name)
  - **id** (select node id attribute)
  - **node values** (select node by value)
  - **attribute name and value** (select node on attribute value)
  - **hierarchy** (select depending on the position in path)

selecting nodes by name

```
html_nodes(html, "p")
```

Selector Gadget and Developer Tools to the Rescue

Selector Gadget and Developer Tools to the Rescue

- building Xpath (CSS-S) expressions is an art
- and easily and quickly becomes mind buggling and complicated
- there are however some tools that might help lessen the burden:
  - selectorgadget : `http://selectorgadget.com/`
  - developer tools

R-Packages and Functions

rvest and XML

rvest (httr + xml2 + selectr)

- scraping centered package (download and extraction)
- HTML / XML
- XPath / CSS-S
- very handy and slick
- we use this

XML (xml)

- XML centered package (parsing and extraction)
- XPath
- much more powerful in terms of parsing (also SAX for LARGE documents)
- goes back to 1999 (according to README; you know just after the internet became a thing)
- two good sources cover that one: Nolan & Temple-Lang (2013): *XML and Web Technologies for Data Sciences with R*; Munzert et al (2014): *Automated Data Collection with R*

rvest / xml2

(important) XML handling functions

function | description ——— | ——— `read_html()` | parse HTML (file); all others based on `html_structure()` | shows the structure of an HTML (doc) `as_list()` | transform parsed XML / HTML to list (doc) html_attr

doc: parsed document; ns: node set or node; file: un-parsed XML document

`write_xml` `xml_attr` `xml_attrs`
`xml_children` `xml_contents` `xml_find_all` `xml_find_one`
`xml_has_attr` `xml_length` `xml_name` `xml_ns`
xml_ns_rename‘`xml_parent` `xml_parents` `xml_path`
`xml_siblings` `xml_structure` `xml_text` `xml_type`
`xml_url`
`url_absolute` `url_escape` `url_parse` `url_relative`
`url_unescape`