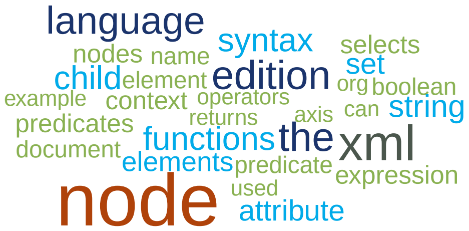
Introduction to Web Scraping with R

XPath, Part I



Simon Munzert | IPSDS

Accessing the HTML tree with R

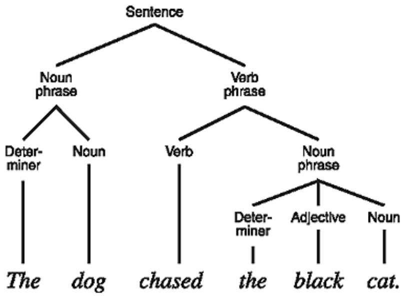
Accessing the HTML tree with R

- HTML documents are human-readable
- HTML tags structure the document
- **web user perspective**: the browser interprets the code and renders the page
- **web scraper perspective**: use the tags to locate information; document has to be parsed first

Parsing

Parsing

Parsing originally describes the syntactic analysis of text according to grammatical rules; analysis of the relationship between single parts of text. In programming, the input has to be interpreted (e.g., by R) to process the command.



HTML parsing with R

Tools

- the `xml2` package allows us to parse XML-style documents
- the `rvest` package, which we will mainly use for scraping, wraps the `xml2` package, so we rarely have to load it manually
- HTML is a "flavor" of XML, so we can use the package to parse HTML
- one high-level function: `read_html()`
- `read_html()` represents the HTML in a list-style fashion
- we could also import HTML files via `readLines()`, but this is not parsing—the document's structure is not retained

HTML parsing with R

Parsing a website is straightforward

R code

```r
library(rvest)
parsed_doc <- read_html("https://google.com")
parsed_doc
```

```
{xml_document}
<html itemscope="" itemtype="http://schema.org/WebPage" lang="de">
[1] <head>\n<meta content="text/html; charset=UTF-8" http-equiv="Content ...
[2] <body bgcolor="#fff">\n<script>(function(){var src='/images/nav_logo ...
```

end

Functions to inspect the parsed document - better use the browser instead

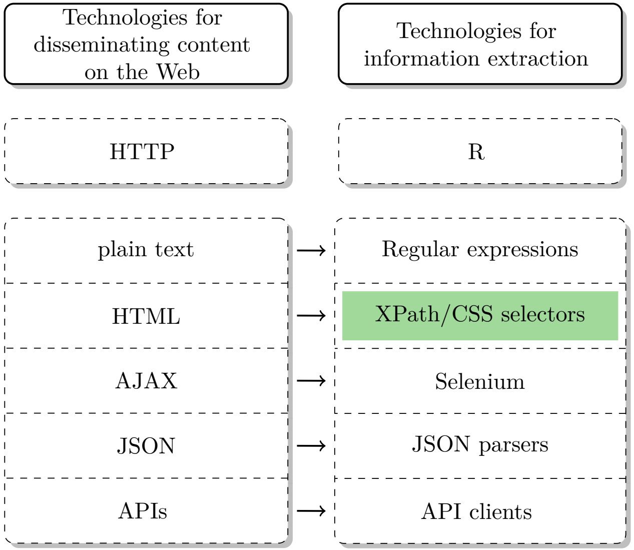
R code

```r
html_structure(parsed_doc)
as_list(parsed_doc)
```

end

XPath

Technologies of the World Wide Web



What's XPath?

Definition

- XML Path language, a W3C standard
- query language for XML-based documents ($\rightarrow$ HTML)
- access node sets and extract content

What’s XPath?

Definition

- XML Path language, a W3C standard
- query language for XML-based documents (→ HTML)
- access node sets and extract content

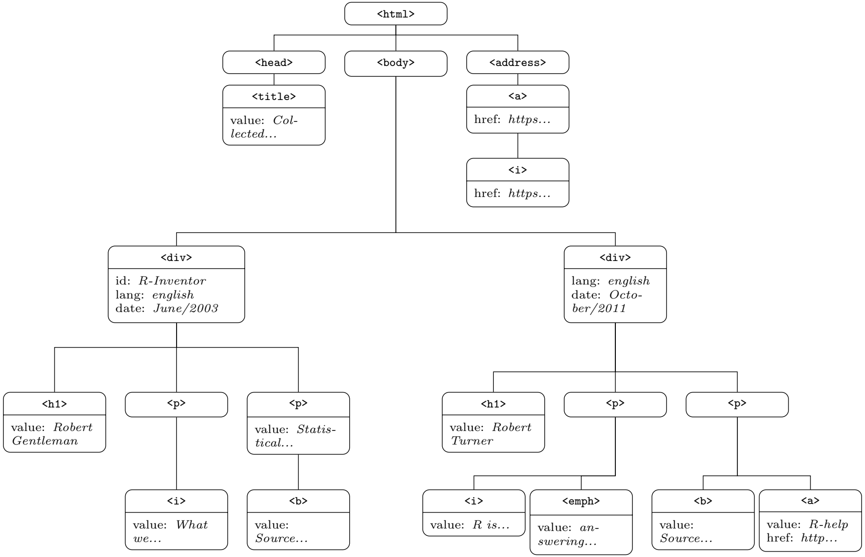
Why XPath for web scraping?

- source code of webpages (HTML) structures both layout and content
- not only content, but context matters!
- enables us to extract content based on its location in the document and (usually) regardless of its shape

Example

```html
<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML//EN">
<html> <head>
<title>Collected R wisdoms</title>
</head>
<body>
<div id="R Inventor" lang="english" date="June/2003">
  <h1>Robert Gentleman</h1>
  <p><i>'What we have is nice, but we need something very different'</i></p>
  <p><b>Source: </b>Statistical Computing 2003, Reisensburg</p>
</div>
<div lang="english" date="October/2011">
  <h1>Rolf Turner</h1>
  <p><i>'R is wonderful, but it cannot work magic'</i> <br><emph>answering a request for automatic
        generation of 'data from a known mean and 95% CI'</emph></p>
  <p><b>Source: </b><a href="https://stat.ethz.ch/mailman/listinfo/r-help">R-help</a></p>
</div>
</body>
<address><a href="http://www.r-datacollection.com"><i>The book homepage</i><a/></address>
</html>
```

Example


<html>
<head>
<body>
<address>
<title>
value: *Collected...*
<a>
href: *https...*
<i>
href: *https...*
<div>
id: *R-Inventor*
lang: *english*
date: *June/2003*
<div>
lang: *english*
date: *October/2011*
<h1>
value: *Robert Gentleman*
<p>
<p>
value: *Statistical...*
<h1>
value: *Robert Turner*
<p>
<p>
<i>
value: *What we...*
<b>
value: *Source...*
<i>
value: *R is...*
<emph>
value: *answering...*
<b>
value: *Source...*
<a>
value: *R-help*
href: *http...*

Example

Applying an XPath expression in R

- load package rvest
- parse document with read_html()
- query document with XPath expression using html_nodes()
- rvest can process XPath queries as well as CSS selectors
- in this course, we'll focus on XPath

R code

```r
library(rvest)
parsed_doc <- read_html("../materials/fortunes.html")
html_nodes(parsed_doc, xpath = "//div[last()]/p/i")
```

```
{xml_nodeset (1)}
[1] <i>'R is wonderful, but it cannot work magic'</i>
```

end

Grammar of XPath

Basic rules

1. we access nodes by writing down the hierarchical structure in the DOM that locates the node set of interest
2. a sequence of nodes is separated by slash symbols
3. the easiest localization of a node is given by the absolute path (but often not the most efficient one!)
4. apply XPath on document in R with the `html_nodes()` function

R code

```
html_nodes(parsed_doc, xpath = "//div[last()]/p/i")
{xml_nodeset (1)}
[1] <i>'R is wonderful, but it cannot work magic'</i>
```

end

Grammar of XPath

Absolute vs. relative paths

- absolute paths start at the root node and follow the whole way down to the target node (with simple slashes, '/')
- relative paths skip nodes (with double slashes, '//')

R code

```
html_nodes(parsed_doc, xpath = "/html/body/div/p/i")
{xml_nodeset (2)}
[1] <i>'What we have is nice, but we need something very different'</i>
[2] <i>'R is wonderful, but it cannot work magic'</i>

html_nodes(parsed_doc, xpath = "//body//p/i")
{xml_nodeset (2)}
[1] <i>'What we have is nice, but we need something very different'</i>
[2] <i>'R is wonderful, but it cannot work magic'</i>
```

end

Grammar of XPath

When to use absolute, when relative paths?

- relative paths faster to write
- relative paths often more comprehensive (but less robust)
- relative paths consume more computing time, as the whole tree has to be parsed, but this is usually of less relevance for reasonably small documents

R code

```r
html_nodes(parsed_doc, xpath = "//i")
```

```
{xml_nodeset (3)}
[1] <i>'What we have is nice, but we need something very different'</i>
[2] <i>'R is wonderful, but it cannot work magic'</i>
[3] <i>The book homepage</i>
```

end

Grammar of XPath

Wildcard operator

- meta symbol *
- matches any node
- works only for one arbitrary node
- far less important than wildcards in regular expressions

R code

```
html_nodes(parsed_doc, xpath = "/html/body/div/*/i")
{xml_nodeset (2)}
[1] <i>'What we have is nice, but we need something very different'</i>
[2] <i>'R is wonderful, but it cannot work magic'</i>
# this does not work:
html_nodes(parsed_doc, xpath = "/html/body/*/i")
{xml_nodeset (0)}
```

end

Grammar of XPath

Navigational operators '.' and '..'

- `.` accesses nodes at the same level ('self axis')
- useful when working with predicates
- `..` accesses nodes at a higher hierarchical level

R code

```
html_nodes(parsed_doc, xpath = "//title/..")
{xml_nodeset (1)}
[1] <head>\n<meta http-equiv="Content-Type" content="text/html; charset= ...
```

end

Grammar of XPath

Pipe operator

- combines several paths

R code

```
html_nodes(parsed_doc, xpath = "//address | //title")
{xml_nodeset (2)}
[1] <title>Collected R wisdoms</title>
[2] <address>\n<a href="http://www.r-datacollection.com"><i>The book hom ...
```

end

Summary

- XPath is a little language that lets you query specfic parts of an XML-style document
- it has its own grammar (logic) and vocabulary
- in this session, you learned the basics of XPath
- in the next session, you will learn more advanced, powerful XPath expressions

Source: https://commons.wikimedia.org/wiki/File:Mozie_Law_path_junction_-_geograph.org.uk_-_1131.jpg (Andy Stephenson)