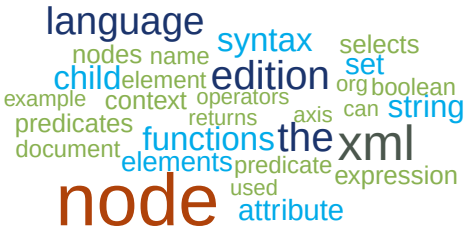
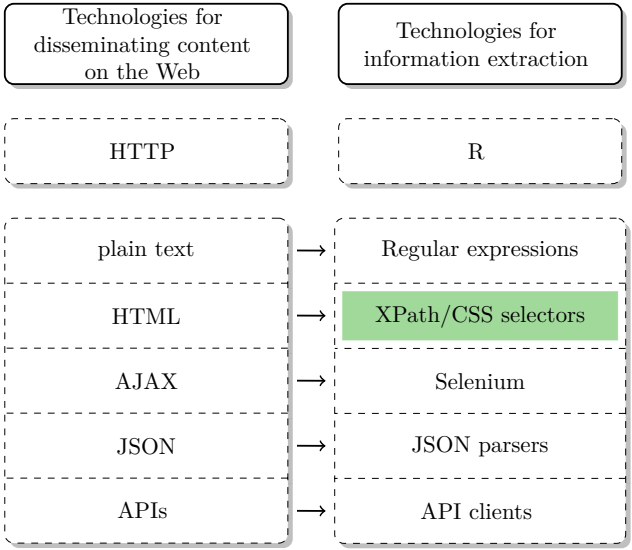
Introduction to Web Scraping with R

XPath, Part II



Simon Munzert | IPSDS

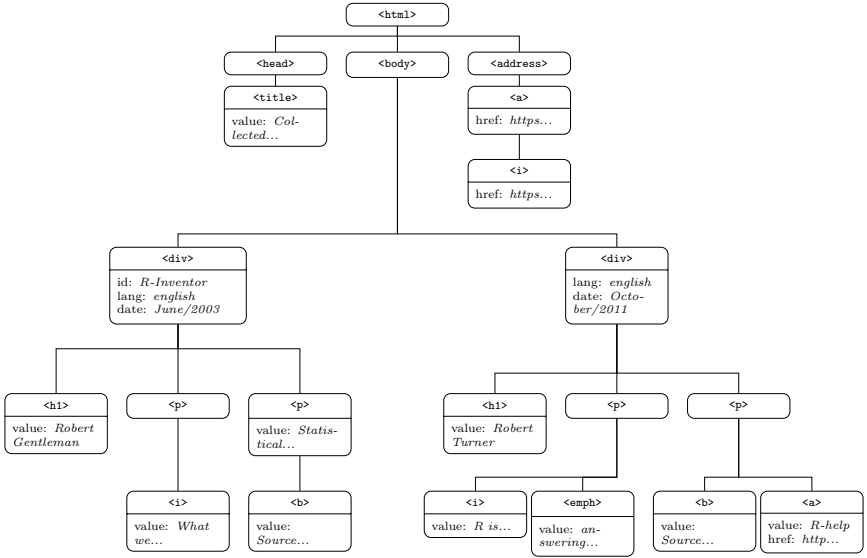
Technologies of the World Wide Web



Example

```
<!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...

id: R-Inventor
lang: english
date: June/2003

lang: english
date: October/2011
<h1>
value: Robert Gentleman
<p>
<p>
value: Statistical...
<i>
value: What we...
<b>
value: Source...
<h1>
value: Robert Turner
<p>
<p>
<i>
value: R is...
<emph>
value: answering...
<b>
value: Source...
<a>
value: R-help
href: http...

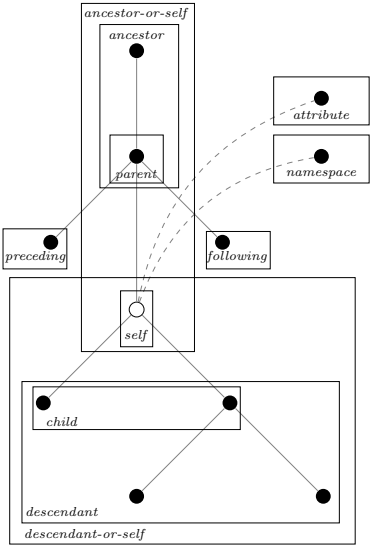
Node relations in XPath

Node relations in XPath

‘Family relations’ between nodes

- the tools learned so far are sometimes not sufficient to access specific nodes without accessing other, undesired nodes as well
- relationship statuses are useful to establish unambiguity
- can be combined with other elements of the grammar
- basic syntax: `node1/relation::node2`
- we describe *relation* of *node2* to *node1*
- *node2* is to be extracted—we **always** extract the node at the end

Node relations in XPath



Node relations in XPath

Axis name	Result
ancestor	all ancestors (parent, grandparent, etc.) of the current node
ancestor-or-self	all ancestors of the current node and the current node itself
attribute	all attributes of the current node
child	all children of the current node
descendant	all descendants (children, grandchildren, etc.) of the current node
descendant-or-self	all descendants of the current node and the current node itself
following	everything in the document after the closing tag of the current node
following-sibling	all siblings after the current node
namespace	all namespace nodes of the current node
parent	the parent of the current node
preceding	all nodes that appear before the current node in the document, except ancestors, attribute nodes and namespace nodes
preceding-sibling	all siblings before the current node
self	the current node

Node relations in XPath

Example: access the `<div>` nodes that are ancestors to an `<a>` node:

R code

```
1  html_nodes(parsed_doc, xpath = "//a/ancestor::div")
   {xml_nodeset (1)}
   [1] <div lang="english" date="October/2011">\n  <h1>Rolf Turner</h1>\n   ...
```

end

Node relations in XPath

Example: access the `<div>` nodes that are ancestors to an `<a>` node:

R code

```
3 html_nodes(parsed_doc, xpath = "//a/ancestor::div")
{xml_nodeset (1)}
[1] <div lang="english" date="October/2011">\n  <h1>Rolf Turner</h1>\n   ...
```

end

Another example: Select all `<h1>` nodes that precede a `<p>` node:

R code

```
4 html_nodes(parsed_doc, xpath = "//p/preceding-sibling::h1")
{xml_nodeset (2)}
[1] <h1>Robert Gentleman</h1>
[2] <h1>Rolf Turner</h1>
```

end

Predicates

Predicates

Predicates

- Conditions based on a node's features (`true`/`false`)
- applicable to a variety of features: name, value, attribute
- basic syntax:

  node[predicate]

Commands in predicates

Function	Returns...
`name(<node>)`	name of `<node>` or the first node in a node set
`text(<node>)`	value of `<node>` or the first node in a node set
`@attribute`	value of a node's *attribute*
`string-length(str1)`	length of `str1`. If there is no string argument, it length of the string value of the current node
`translate(str1, str2, str3)`	`str1` by replacing the characters in `str2` with the characters in `str3`
`contains(str1,str2)`	`TRUE` if `str1` contains `str2`, otherwise `FALSE`
`starts-with(str1,str2)`	`TRUE` if `str1` starts with `str2`, otherwise `FALSE`
`substring-before(str1,str2)`	start of `str1` before `str2` occurs in it
`substring-after(str1,str2)`	remainder of `str1` after `str2` occurs in it
`not(arg)`	`TRUE` if the Boolean value is `FALSE`, and `FALSE` if the boolean value is `TRUE`
`local-name(<node>)`	name of the current `<node>` or the first node in a node set – without the namespace prefix
`count(<node>)`	count of a nodeset `<node>`
`position(<node>)`	index position of `<node>` that is processed
`last()`	number of items in the processed node list `<node>`

Predicates

Numeric predicates

- indicate positions, counts, etc.

Example: Select all first `<p>` nodes that are children of a `<div>` node:

R code

```
5 html_nodes(parsed_doc, xpath = "//div/p[position()=1]")
```

```
{xml_nodeset (2)}
[1] <p><i>'What we have is nice, but we need something very different'</ ...
[2] <p><i>'R is wonderful, but it cannot work magic'</i> <br><emph>answe ...
```

end

Further examples:

R code

```
6 html_nodes(parsed_doc, xpath = "//div/p[last()-1]")
7 html_nodes(parsed_doc, xpath = "//div[count(./@*)>2]")
8 html_nodes(parsed_doc, xpath = "//*[string-length(text())>50]")
```

end

Predicates

Textual predicates

- describe text features
- applicable on: node name, content, attributes, attribute values
- XPath 2.0 supports (simplified) regex, however, R (the `rvest` package and the underlying `xml2` package) does not support it

Example: Select all `<div>` nodes that contain an attribute named '`October/2011`':

R code

```
9  html_nodes(parsed_doc, xpath = "//div[@date='October/2011']")
   {xml_nodeset (1)}
   [1] <div lang="english" date="October/2011">\n  <h1>Rolf Turner</h1>\n   ...
```

end

Predicates

Partial matching

- rudimentary string matching
- 'content contains' (`contains()`), 'content begins with' (`starts-with()`), 'content ends with' (`ends-with()`), 'content contains after split' (`substring-after()`)

R code

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

end

Further examples:

R code

```
11 html_nodes(parsed_doc, xpath = "//div[starts-with(./@id, 'R')]")
12 html_nodes(parsed_doc, xpath = "//div[substring-after(./@date, '/')='2003']//i")
```

end

Content extraction

Extraction of text and attributes

Extraction

- until now: query of complete nodes
- common scenario: only parts of the node are interesting, e.g., content (value)
- additional extraction operations from a selected node set possible with additional extractor functions

Function	Argument	Return value
`html_text`		node value
`html_attr`	`name`	node attribute
`html_attrs`		(all) node attributes
`html_name`	`trim`	node name
`html_children`		node children

Extraction of node elements

Values/text

R code

```
13 html_nodes(parsed_doc, xpath = "//title") %>% html_text()
   [1] "Collected R wisdoms"
```

end

Attributes

R code

```
14 html_nodes(parsed_doc, xpath = "//div") %>% html_attrs()
   [[1]]
             id         lang         date
   "R Inventor"    "english"  "June/2003"

   [[2]]
             lang           date
        "english" "October/2011"
```

end

Extraction of node elements

Attribute values

R code

```
15  html_nodes(parsed_doc, xpath = "//div") %>% html_attr("lang")
    [1] "english" "english"
```

end

A silver lining

Do I really have to construct XPath expressions all by my own?

Do I really have to construct XPath expressions all by my own?

No!

Do I really have to construct XPath expressions all by my own?

No!

XPath creator tools

- *Selectorgadget*: http://selectorgadget.com/. Browser plugin that constructs XPath statements via a point-and-click approach. The generated expressions are not always efficient though
- *Web Developer Tools*: internal browser functionality which return XPath statements for selected nodes
- you will learn how to use these tools in upcoming sessions