Cologne Center for Comparative Politics

A Primer to Web Scraping with R

Introduction

Simon Munzert
Hertie School of Governance, Berlin

November 12, 2018

Welcome

Welcome!

Thank you for your interest in the course!

Welcome!

Thank you for your interest in the course!

I'm glad that I will have the opportunity to give you a gentle introduction to web scraping with R.

Welcome!

Thank you for your interest in the course!

I'm glad that I will have the opportunity to give you a gentle introduction to web scraping with R.

Just a few words on my personal background:

- Lecturer in Political Data Science at the Hertie School of Governance
- political scientist by training
- working with web-based data since about 2010
- what I do with web data: measure public awareness, news consumption, political behavior

Your background and interests

“Please let me know if there is any topic that you definitely want me to cover.”

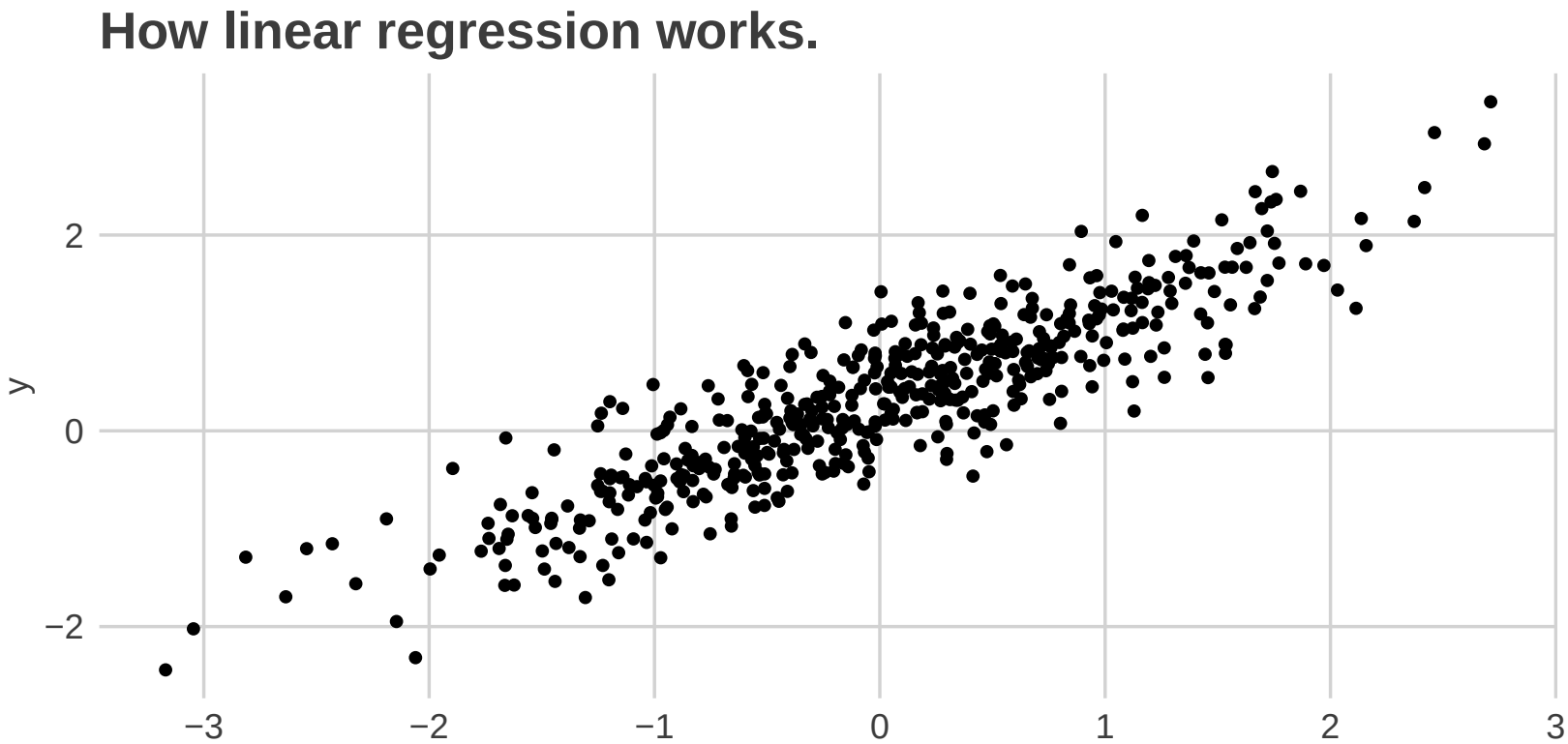
Your background and interests

“Please let me know if there is any topic that you definitely want me to cover.”

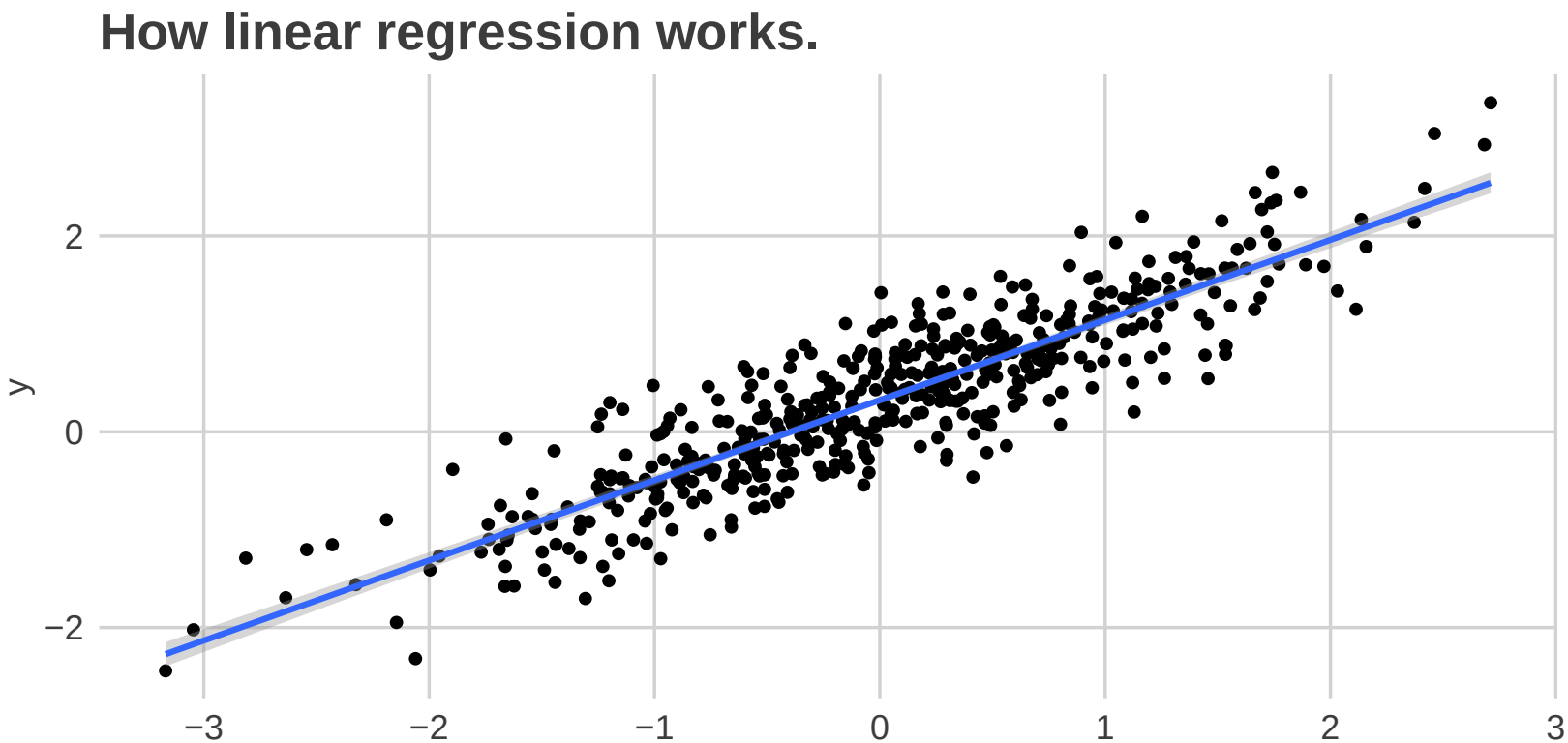
- ”Scraping Flash pages.”
- ”rather interested in general introduction”
- ”How to scrape Java scripts”
- ”I am just getting acquainted with R studio”
- ”linear regression”

I want all of you to be happy, so...

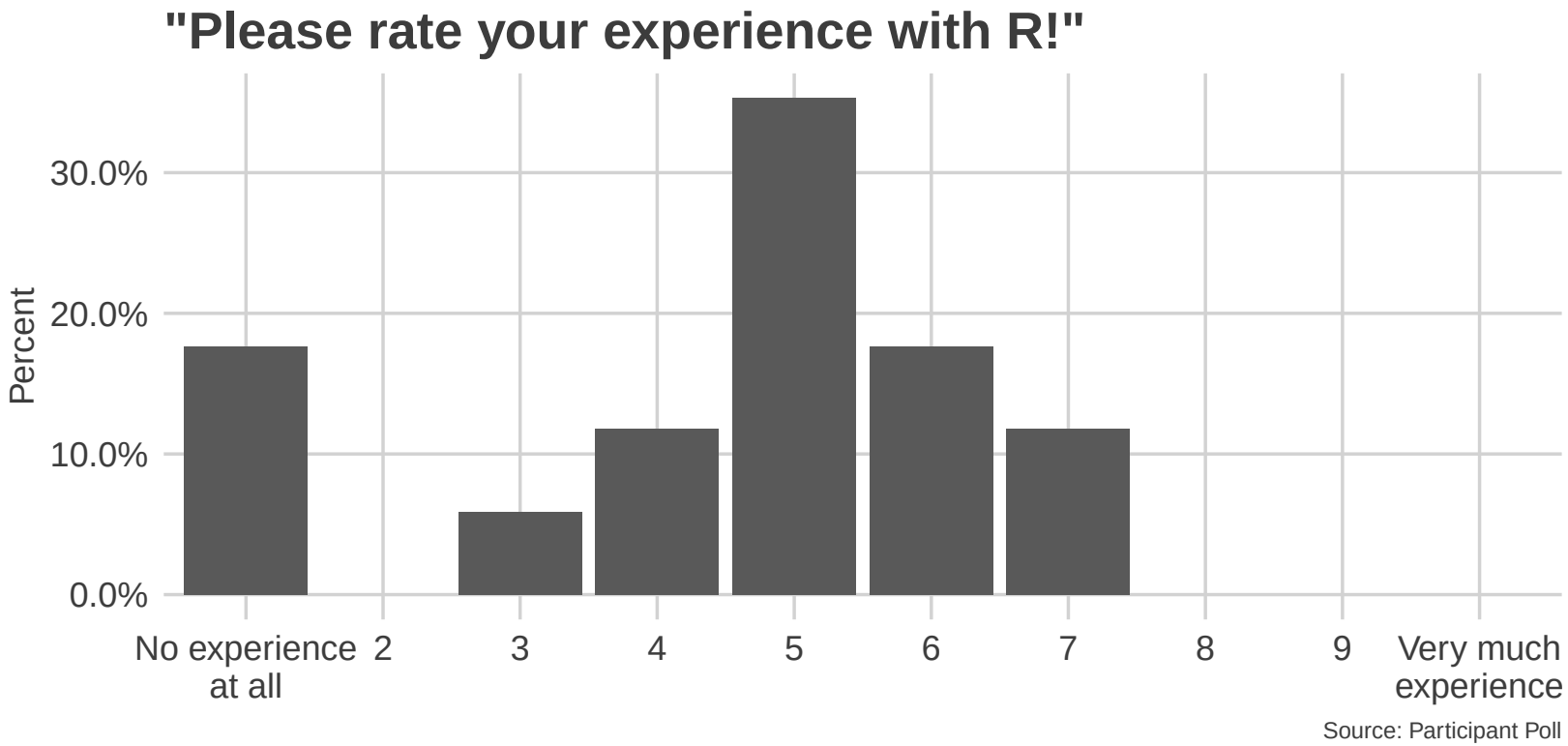
I want all of you to be happy, so...



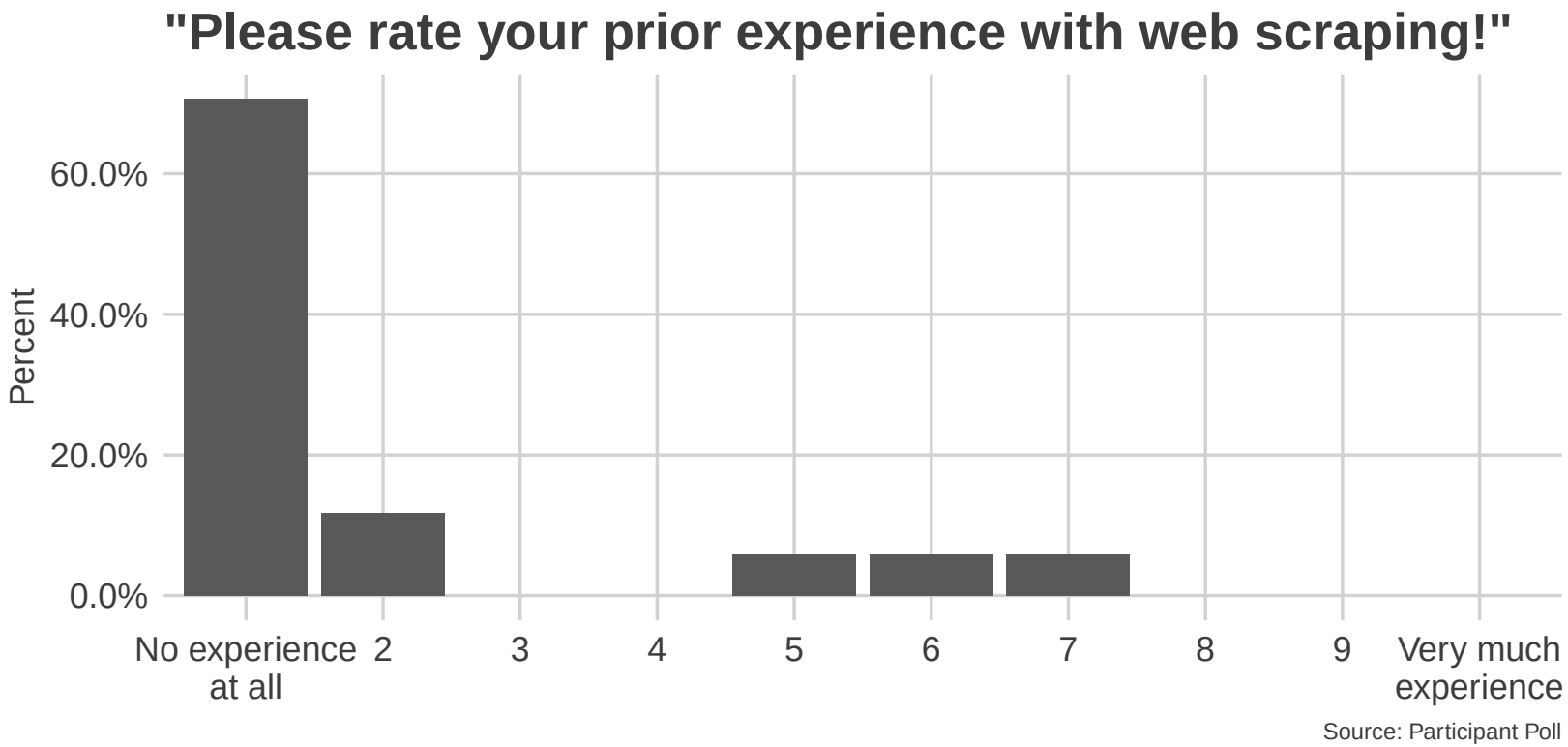
I want all of you to be happy, so...



Your background and interests



Your background and interests



Goals and outline

Goals

After attending this course, ...

Goals

After attending this course, ...

- you have acquired basic knowledge of web technologies

Goals

After attending this course, ...

- you have acquired basic knowledge of web technologies
- you are able to scrape information from static and dynamic websites using R

Goals

After attending this course, ...

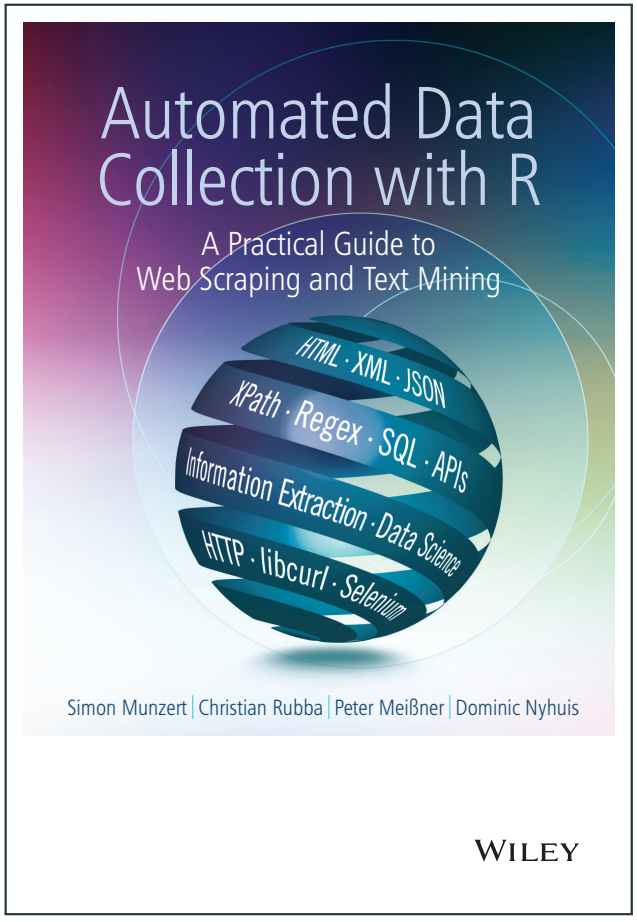
- you have acquired basic knowledge of web technologies
- you are able to scrape information from static and dynamic websites using R
- you are able to access web services (APIs) with R

Course outline

Time	Topic
09:00 - 09:45	Introduction; a first encounter with the Web using R
09:45 - 12:15	Scraping static webpages
12:15 - 13:15	*Lunch break*
13:15 - 14:00	Scraping dynamic webpages
14:00 - 15:30	Tapping APIs
15:30 - 16:30	Scraping ethics and workflow

The accompanying book

- contains most of which I tell you during the course (but much more, and at times more accurate)
- written between 2012 and 2014 → not entirely up-to-date anymore, but I will provide updated material during the course
- homepage with materials: www.r-datacollection.com
- if you find any errors in the book, please let me know!



Cologne Center for Comparative Politics

A Primer to Web Scraping with R

Overview

Simon Munzert
Hertie School of Governance, Berlin

November 12, 2018

Web scraping with R

Web scraping. What? Why?

Web scraping

A.k.a. screen scraping, is the business of

- getting (unstructured) data from the web and
- bringing it into shape (e.g., clean, make tabular format)

Web scraping. What? Why?

Web scraping

A.k.a. screen scraping, is the business of

- getting (unstructured) data from the web and
- bringing it into shape (e.g., clean, make tabular format)

A data analyst's view

- data abundance online
- social interaction online
- services track social behavior

Web scraping. What? Why?

Web scraping

A.k.a. screen scraping, is the business of

- getting (unstructured) data from the web and
- bringing it into shape (e.g., clean, make tabular format)

A data analyst's view

- data abundance online
- social interaction online
- services track social behavior

A pragmatist's view

- financial resources
- time resources
- reproducibility
- updateability

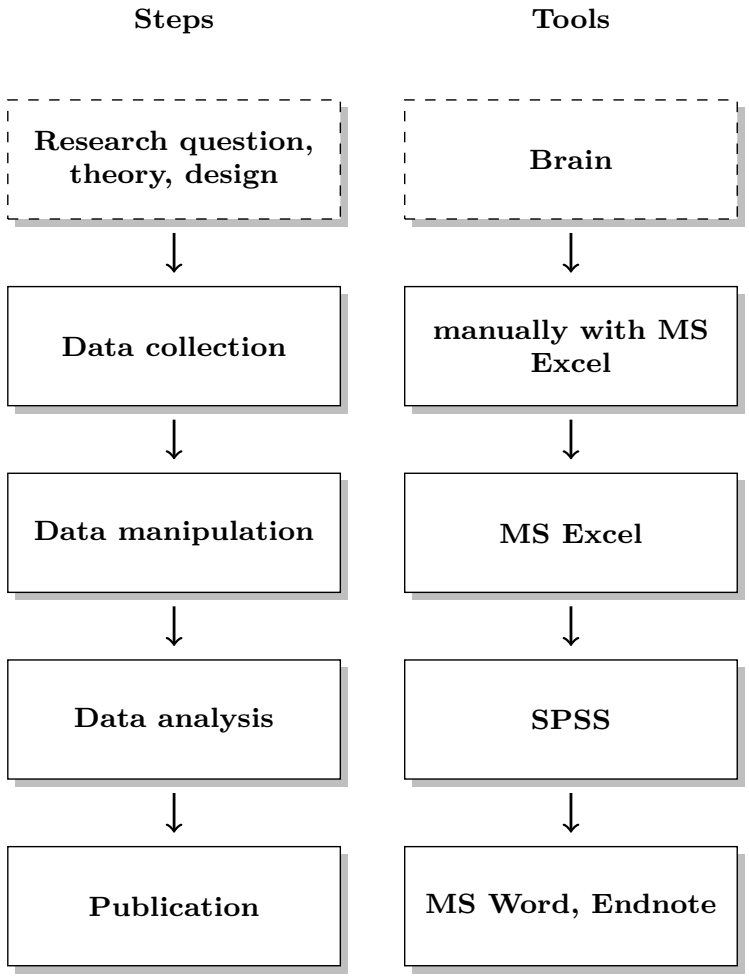
Why R?

Why R?

- free
- open source
- large community
- powerful tools for statistical analysis
- powerful tools for visualization
- flexible in processing all kinds of data/languages
- useful in every step of the workflow

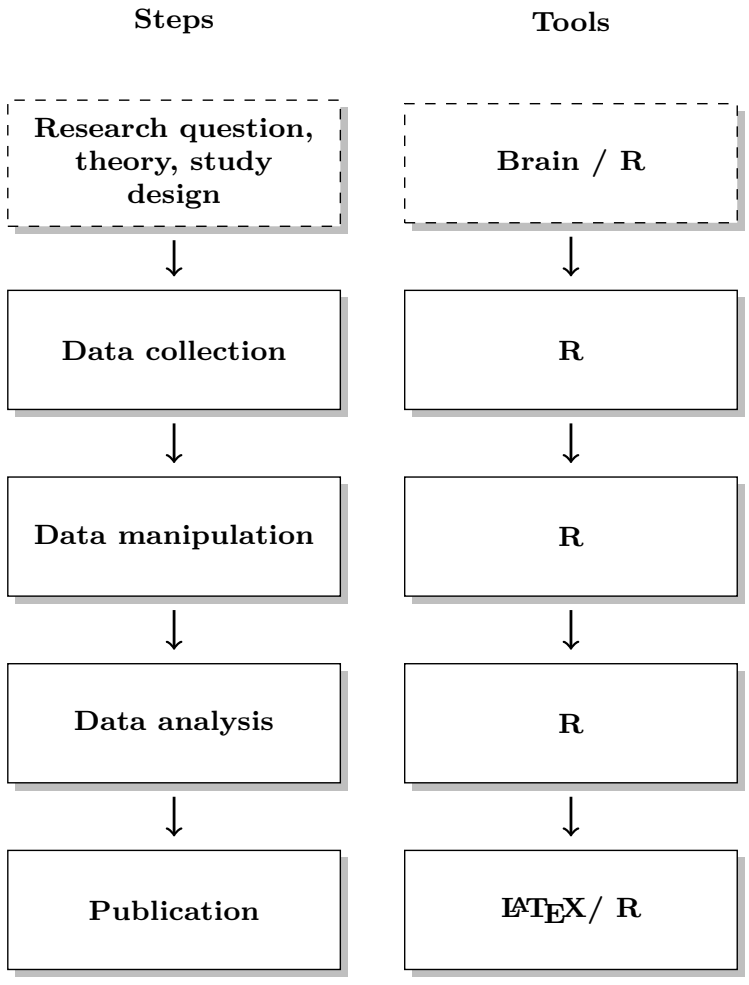
Why R?

- free
- open source
- large community
- powerful tools for statistical analysis
- powerful tools for visualization
- flexible in processing all kinds of data/languages
- useful in every step of the workflow



Why R?

- free
- open source
- large community
- powerful tools for statistical analysis
- powerful tools for visualization
- flexible in processing all kinds of data/languages
- useful in every step of the workflow

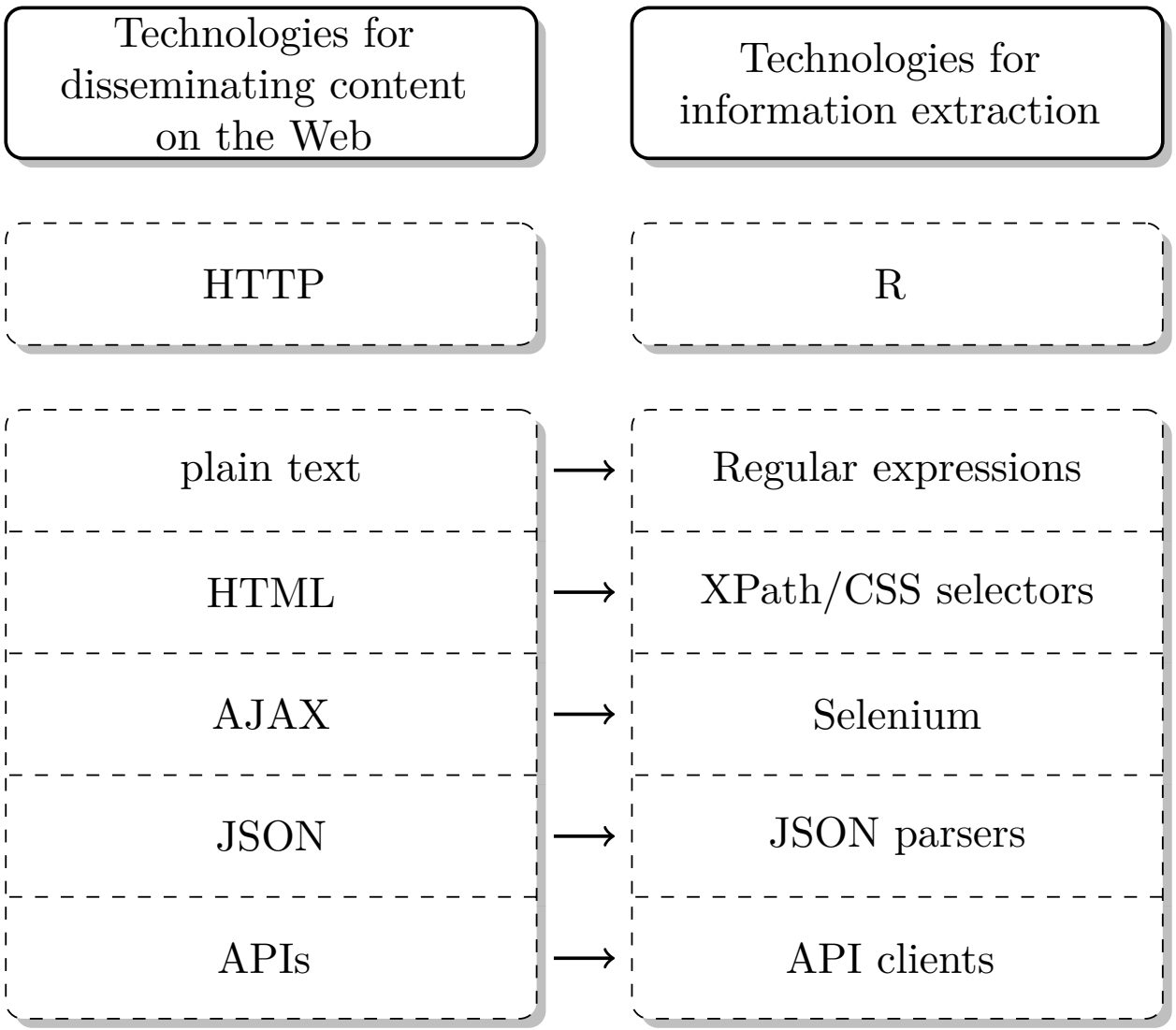


The philosophy behind web data collection with R

- no point-and-click procedure
- script the entire process from start to finish
- automation of
  - downloading
  - classical screen scraping
  - tapping APIs
  - parsing
  - data tidying, text data processing
- scaling up scraping procedures
- scheduling of scraping tasks

Technologies of the World Wide Web

Technologies of the World Wide Web



Technical setup

1. make sure that the newest version of R is installed on your computer (available here: https://cran.r-project.org/)
2. install the newest stable version of *RStudio Desktop* (available here: https://www.rstudio.com/products/rstudio/download)

Cologne Center for Comparative Politics

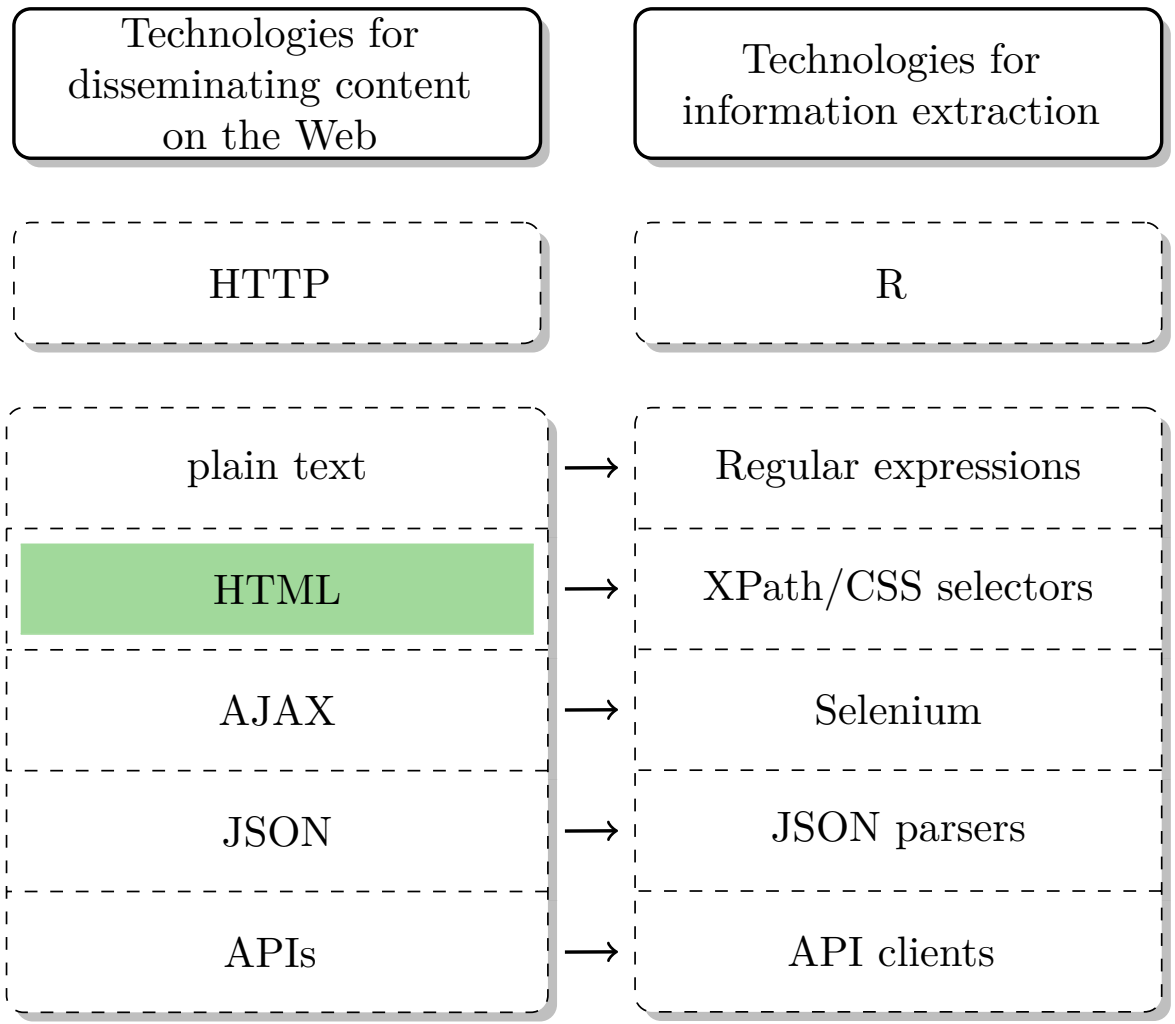
A Primer to Web Scraping with R

HTML

Simon Munzert
Hertie School of Governance, Berlin

November 12, 2018

Technologies of the World Wide Web



HTML basics

HTML – a quick primer

What's HTML?

- **H**yper**T**ext **M**arkup **L**anguage
- markup language = plain text + markups
- W3C standard for the construction of websites
- lies underneath of what you see in your browser

HTML – a quick primer

What's HTML?

- **H**yper**T**ext **M**arkup **L**anguage
- markup language = plain text + markups
- W3C standard for the construction of websites
- lies underneath of what you see in your browser

Why is this important to us?

- it determines where and how information is stored
- a basic understanding of HTML helps us locate the information we want to retrieve
- relax. A passive understanding of HTML is sufficient

HTML in the wild

HTML in the wild

borough's mayors make up the council of mayors (*Rat der Bürgermeister*), which is led by the city's Governing Mayor and advises the Senate. The neighborhoods have no local government bodies.

Twin towns – sister cities [edit]

See also: List of twin towns and sister cities in Germany

Berlin maintains official partnerships with 17 cities.[100] Town twinning between Berlin and other cities began with its sister city Los Angeles in 1967. East Berlin's partnerships were canceled at the time of German reunification but later partially reestablished. West Berlin's partnerships had previously been restricted to the borough level. During the Cold War era, the partnerships had reflected the different power blocs, with West Berlin partnering with capitals in the Western World, and East Berlin mostly partnering with cities from the Warsaw Pact and its allies.

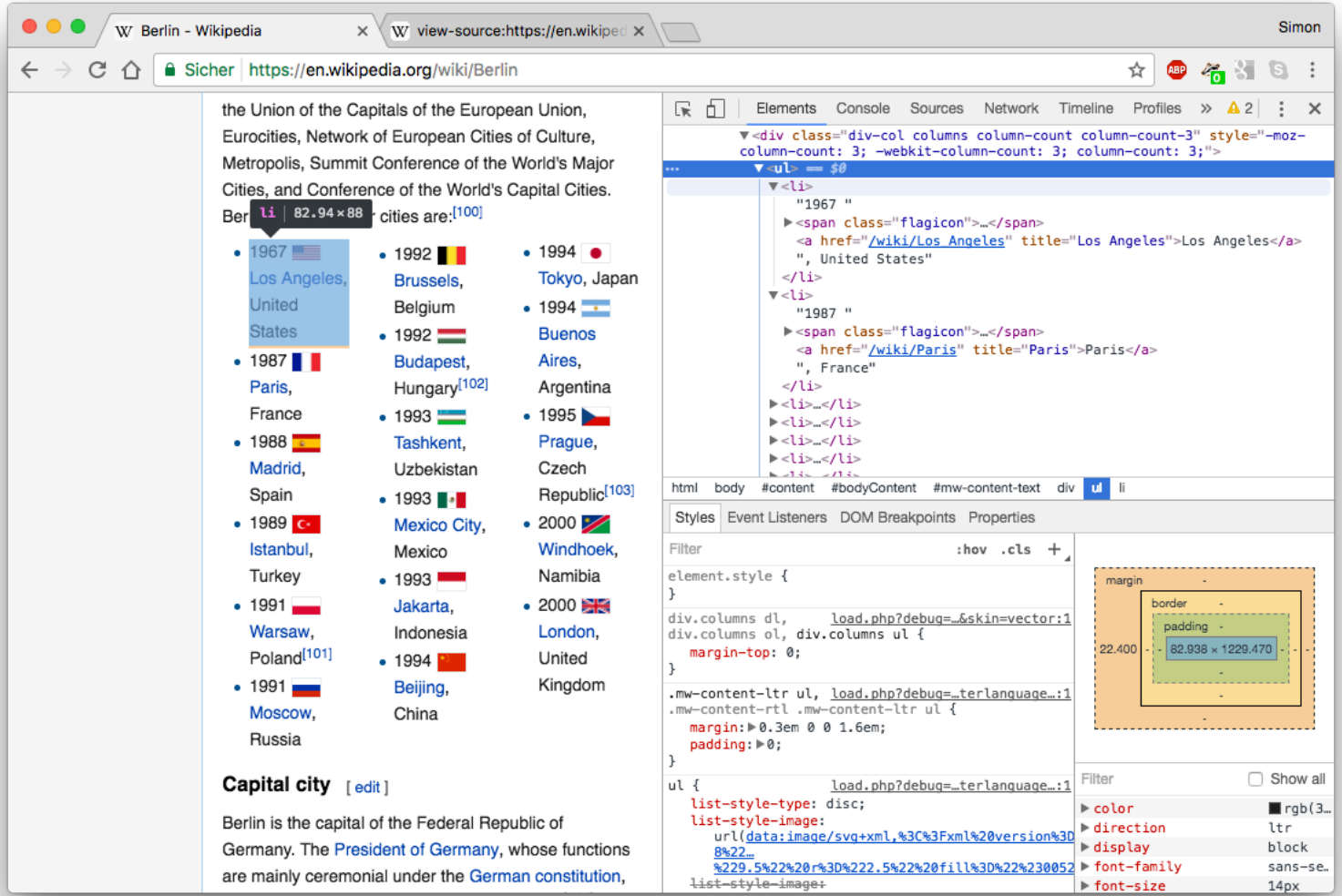
There are several joint projects with many other cities, such as Beirut, Belgrade, São Paulo, Copenhagen, Helsinki, Johannesburg, Mumbai, Oslo, Shanghai, Seoul, Sofia, Sydney, New York City and Vienna. Berlin participates in international city associations such as the Union of the Capitals of the European Union, Eurocities, Network of European Cities of Culture, Metropolis, Summit Conference of the World's Major Cities, and Conference of the World's Capital Cities. Berlin's official sister cities are:[100]

- 1967 Los Angeles, United States
- 1987 Paris, France
- 1988 Madrid, Spain
- 1989 Istanbul, Turkey
- 1991 Warsaw, Poland[101]
- 1991 Moscow, Russia
- 1992 Brussels, Belgium
- 1992 Budapest, Hungary[102]
- 1993 Tashkent, Uzbekistan
- 1993 Mexico City, Mexico
- 1993 Jakarta, Indonesia
- 1994 Beijing, China
- 1994 Tokyo, Japan
- 1994 Buenos Aires, Argentina
- 1995 Prague, Czech Republic[103]
- 2000 Windhoek, Namibia
- 2000 London, United Kingdom

Capital city [edit]

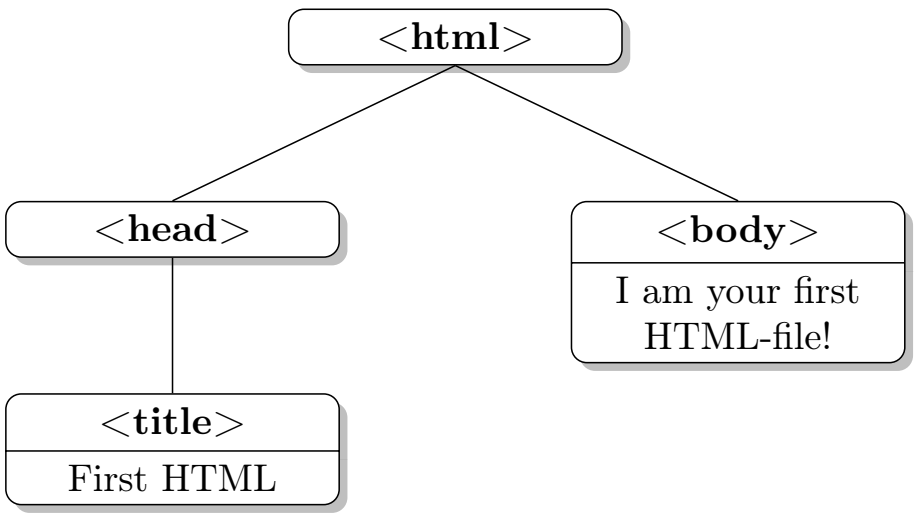
Berlin is the capital of the Federal Republic of Germany. The President of Germany, whose functions are mainly ceremonial under the German constitution, has his official residence in Schloss Bellevue.[104] Berlin is the seat of the German executive, housed in the Chancellery, the *Bundeskanzleramt*. Facing the Chancellery is the Bundestag, the German Parliament, housed in the renovated Reichstag building since the government relocated to Berlin in 1998. The Bundesrat ("federal council", performing the function of an upper house) is the representation of the Federal States (*Bundesländer*) of Germany and has its

HTML in the wild



Tree structure

```
<!DOCTYPE html>
  <html>
    <head>
      <title id=1>First HTML</title>
    </head>
    <body>
      I am your first HTML file!
    </body>
  </html>
```



Elements and attributes

Elements and attributes

Elements

Elements are a combination of *start tags*, content, and *end tags*.

Example:

```
1 <title>First HTML</title>
```

Syntax

element title	`title`
start tag	`<title>`
end tag	`</title>`
value	`First HTML`

Elements and attributes

Attributes

Attributes describe elements and are stored in the start tag. In HTML, there are specific attributes for specific elements.

Example:

```
1 <a href="http://www.r-datacollection.com/">Link to Homepage</a>
```

Syntax

- name-value pairs: `name="value"`
- simple and double quotation marks possible
- several attributes per element possible

Important tags and attributes

Why tags and attributes are important

- tags structure HTML documents
- everything that structures a document can be used to extract information
- in the following, we get to know some important tags which are useful when scraping information from the Web

Important tags and attributes

Anchor tag `<a>`

- links to other pages or resources
- classical links are always formatted with an anchor tag
- the `href` attribute determines the target location
- the value is the name of the link

Link to another resource:

```
<a href="en.wikipedia.org/wiki/List_of_lists_of_lists">Link with absolute path</a>
```

Reference in a document:

```
<a id="top">Reference Point</a>
```

Link to a reference:

```
<a href="#top">Link to Reference Point</a>
```

Important tags and attributes

Heading tags <h1>, <h2>, ..., and paragraph tag <p>

- structure text and paragraphs
- heading tags range from level 1 to 6
- paragraph tag induces line break

Examples:

```
<p>This text is going to be a paragraph one day and separated from other
text by line breaks.</p>
```

```
<h1>heading of level 1 -- this will be BIG</h1>
...
<h6>heading of level 6 -- the smallest heading</h6>
```

Important tags and attributes

Listing tags <ul>, <ol> and <dl>

- the <ol> tag creates a numeric list, <ul> an unnumbered list, <dl> a definition list
- list elements are indicated with the <li> tag

Example:

```
<ul>
^^I<li>Dogs</li>
^^I<li>Cats</li>
^^I<li>Fish</li>
</ul>
```

Important tags and attributes

Organizational tags and

- grouping of content over lines ( ) or within lines ()
- do not change the layout themselves but work together with CSS

Example of CSS definition

```
1  div.happy { color:pink;
2              font-family:"Comic Sans MS";
3              font-size:120% }
4  span.happy { color:pink;
5              font-family:"Comic Sans MS";
6              font-size:120% }
```

In the HTML document

```
1  <div class="happy"><p>I am a happy styled paragraph</p></div>
2  non-happy text with <span class="happy">some happiness</span>
```

Important tags and attributes

Form tag `<form>`

- allows to incorporate HTML forms
- client can send information to the HTTP server via forms
- whenever you type something into a field or click on radio buttons in your browser, you are interacting with forms

Example:

```
1  <form name="submitPW" action="Passed.html" method="get">
2    password:
3    <input name="pw" type="text" value="">
4    <input type="submit" value="SubmitButtonText">
5  </form>
```

Important tags and attributes

Table tags `<table>`, `<tr>`, `<td>`, and `<th>`

- standard HTML tables always follow a standard architecture
- the different tags allow to define the table as a whole, individual rows (including the heading), and cells
- if the data is hidden in tables, scraping will be straightforward

Example:

```
<table>
  <tr> <th>Rank</th> <th>Nominal GDP</th> <th>Name</th> </tr>
  <tr> <th></th> <th>(per capita, USD)</th> <th></th> </tr>
  <tr> <td>1</td> <td>170,373</td> <td>Lichtenstein</td> </tr>
  <tr> <td>2</td> <td>167,021</td> <td>Monaco</td> </tr>
  <tr> <td>3</td> <td>115,377</td> <td>Luxembourg</td> </tr>
  <tr> <td>4</td> <td>98,565</td> <td>Norway</td> </tr>
  <tr> <td>5</td> <td>92,682</td> <td>Qatar</td> </tr>
</table>
```

Summary

Summary

- HTML is the *lingua franca* on the web
- content on webpages is structured by HTML tags that are nested in a tree structure
- to break open information, we will have to locate it in the HTML tree
- for web scraping purposes, a mostly passive knowledge of HTML is sufficient

Source: https://xkcd.com/1144/ (Randall Munroe)

Cologne Center for Comparative Politics

A Primer to Web Scraping with R

XPath, Part I

Simon Munzert
Hertie School of Governance, Berlin

November 12, 2018

Accessing the HTML tree with R

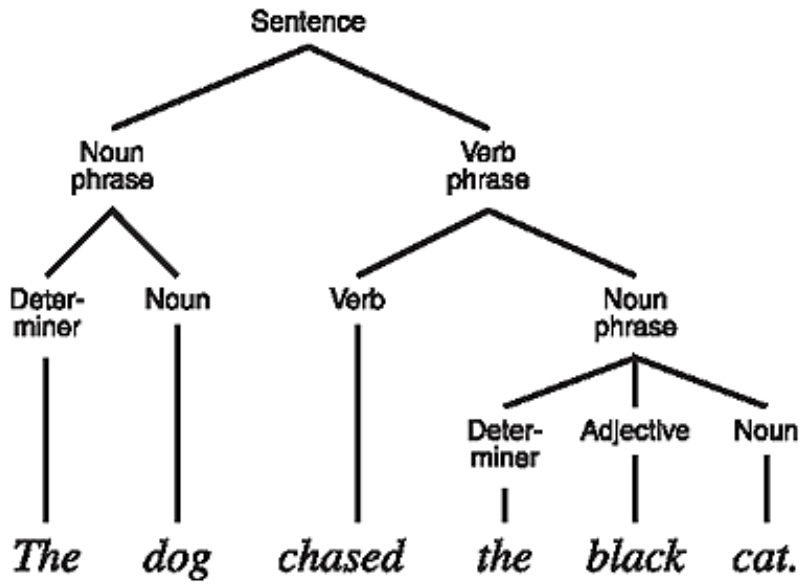
Accessing the HTML tree with R

- 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")
```

end

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

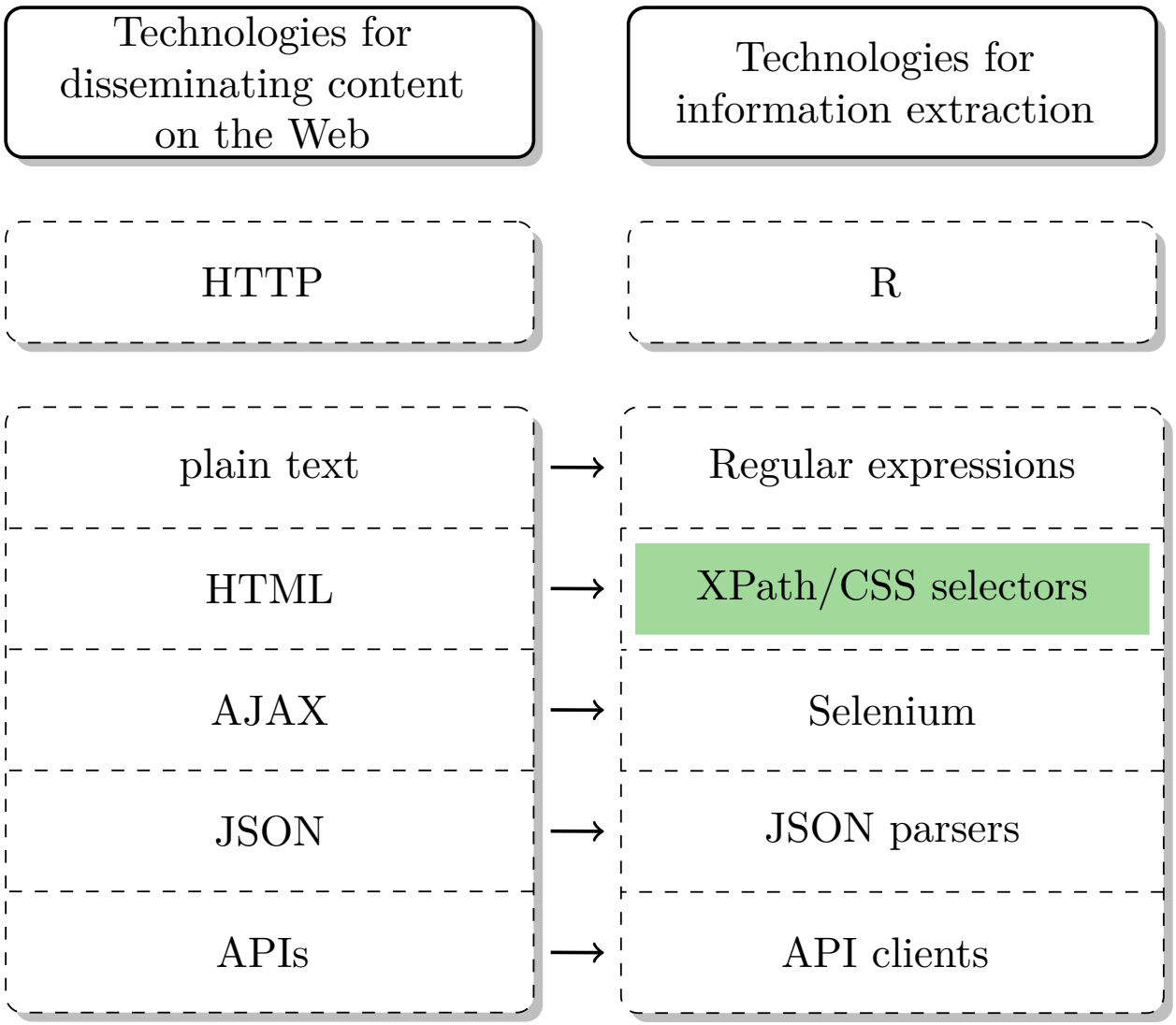
R code

```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 (→ 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

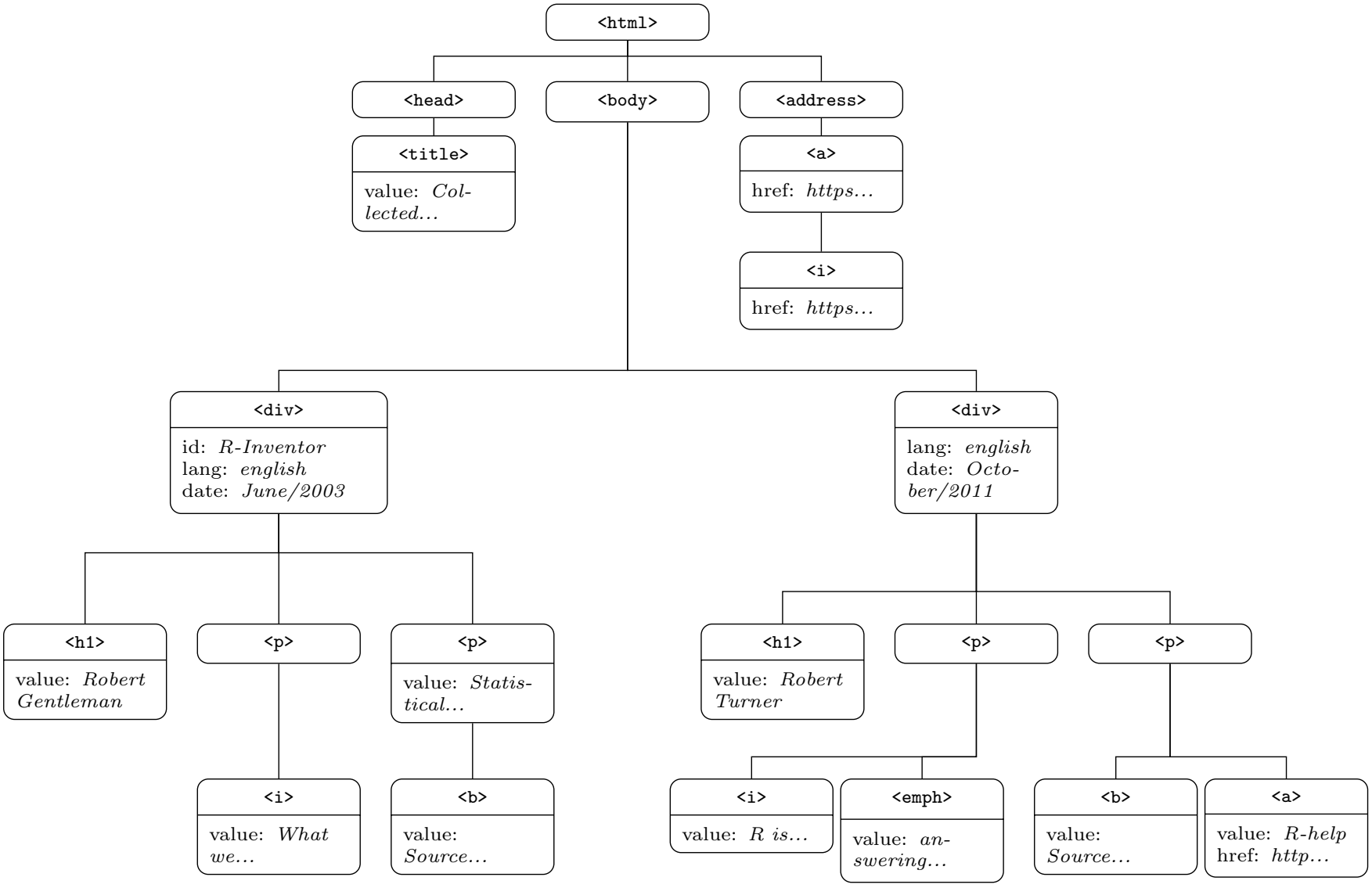
Why XPath for web scraping?

- 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

```
<!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: Col-
lected...
<a>
href: https...
<i>
href: https...

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

lang: english
date: Octo-
ber/2011
<h1>
value: Robert
Gentleman
<p>
<p>
value: Statis-
tical...
<h1>
value: Robert
Turner
<p>
<p>
<i>
value: What
we...
<b>
value:
Source...
<i>
value: R is...
<emph>
value: an-
swering...
<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

```
5  library(rvest)
6  parsed_doc <- read_html("../materials/fortunes.html")
7  html_nodes(parsed_doc, xpath = "//div[last()]/p/i")
   {xml_nodeset (1)}
   [1] <i>'R is wonderful, but it cannot work magic'</i>
```

end

The grammar of XPath

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

```
8  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

```
9  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>

10 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

```
11  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

```
12  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>
13  # this does not work:
14  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

```
15  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

```
16  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

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)

Cologne Center for Comparative Politics

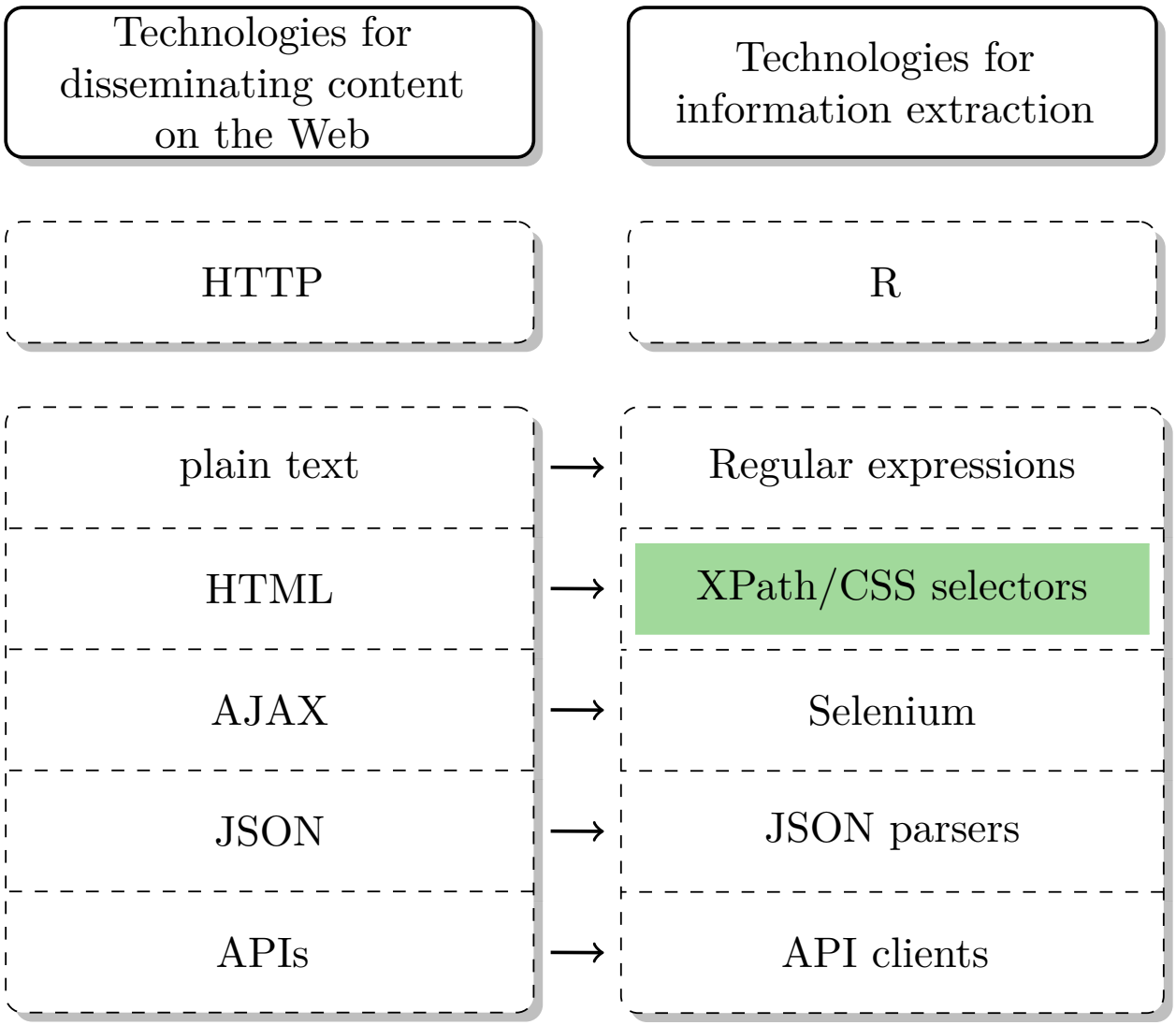
A Primer to Web Scraping with R

XPath, Part II

Simon Munzert
Hertie School of Governance, Berlin

November 12, 2018

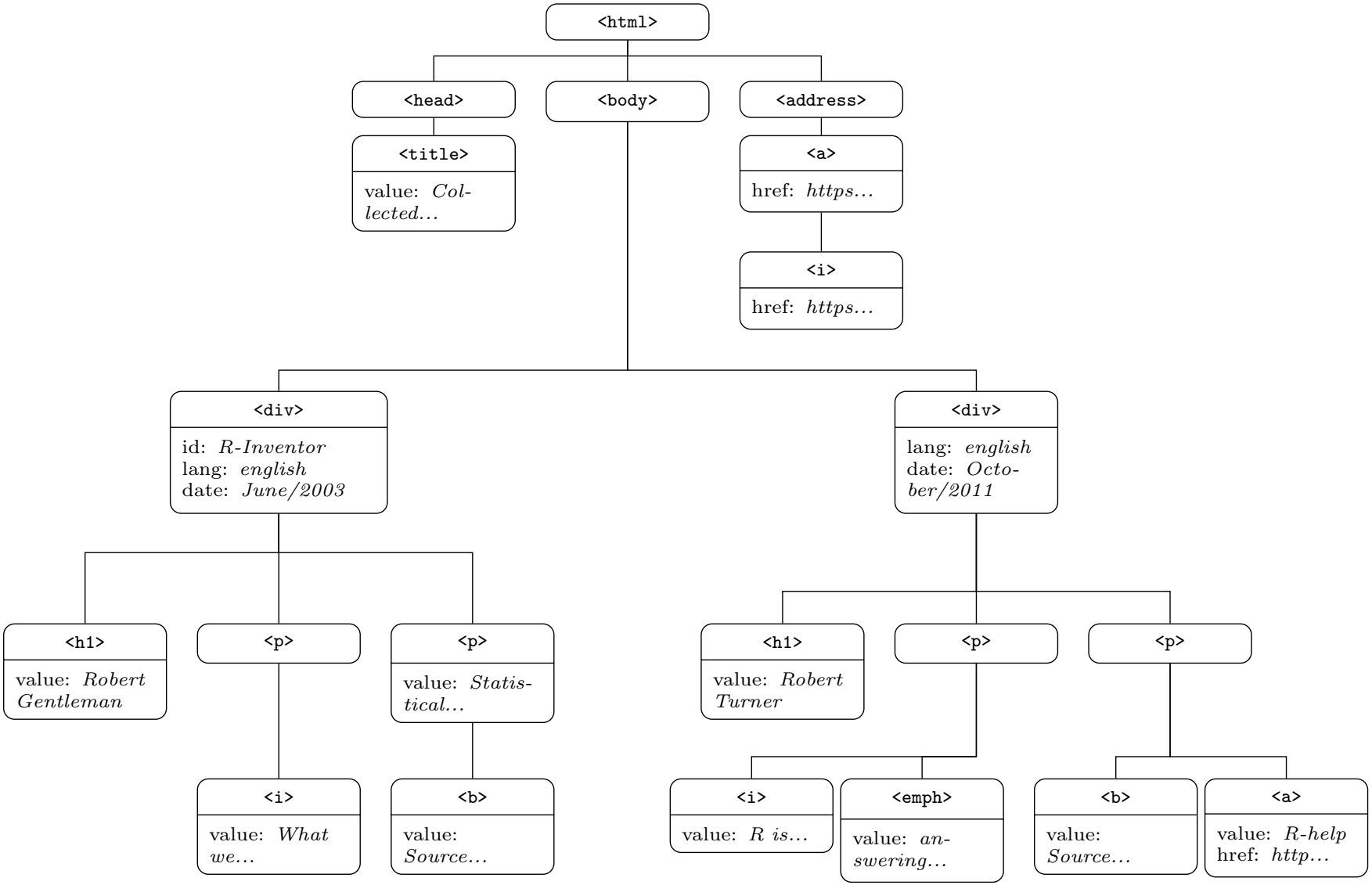
Technologies of the World Wide Web



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: Col-
lected...
<a>
href: https...
<i>
href: https...

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

lang: english
date: Octo-
ber/2011
<h1>
value: Robert
Gentleman
<p>
<p>
value: Statis-
tical...
<h1>
value: Robert
Turner
<p>
<p>
<i>
value: What
we...
<b>
value:
Source...
<i>
value: R is...
<emph>
value: an-
swering...
<b>
value:
Source...
<a>
value: R-help
href: http...

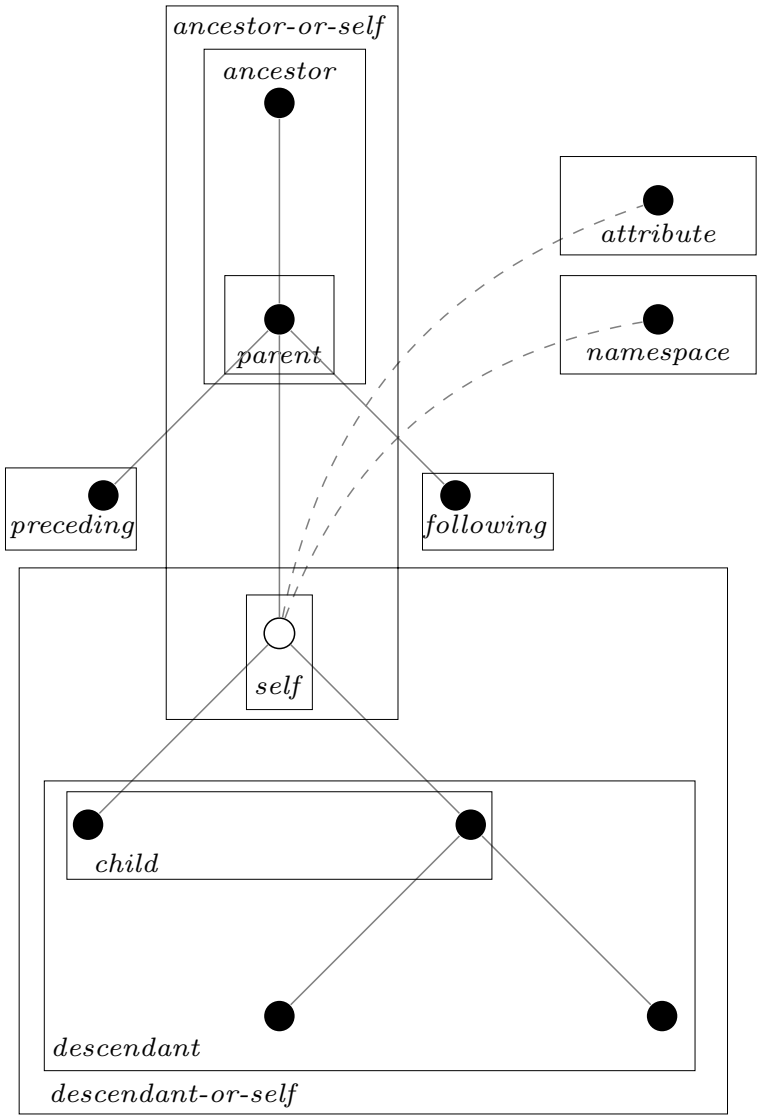
Node relations in XPath

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"
```

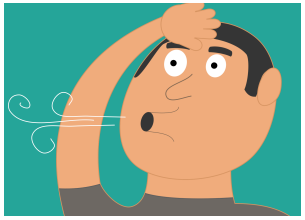
end

A silver lining

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

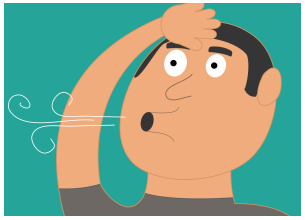
Do I really have to construct XPath expressions all by my own?

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

Cologne Center for Comparative Politics

A Primer to Web Scraping with R

The Scraping Workflow

Simon Munzert
Hertie School of Governance, Berlin

November 12, 2018

Where you stand now

You have learned the main tools necessary to scrape static webpages with R

Where you stand now

You have learned the main tools necessary to scrape static webpages with R

1. you are able to inspect HTML pages in your browser using the web developer tools

Where you stand now

You have learned the main tools necessary to scrape static webpages with R

1. you are able to inspect HTML pages in your browser using the web developer tools

2. you are able to parse HTML into R with `rvest`

Where you stand now

You have learned the main tools necessary to scrape static webpages with R

1. you are able to inspect HTML pages in your browser using the web developer tools
2. you are able to parse HTML into R with `rvest`
3. you are able to speak XPath

Where you stand now

You have learned the main tools necessary to scrape static webpages with R

1. you are able to inspect HTML pages in your browser using the web developer tools
2. you are able to parse HTML into R with `rvest`
3. you are able to speak XPath
4. you are able to apply XPath expressions with `rvest`

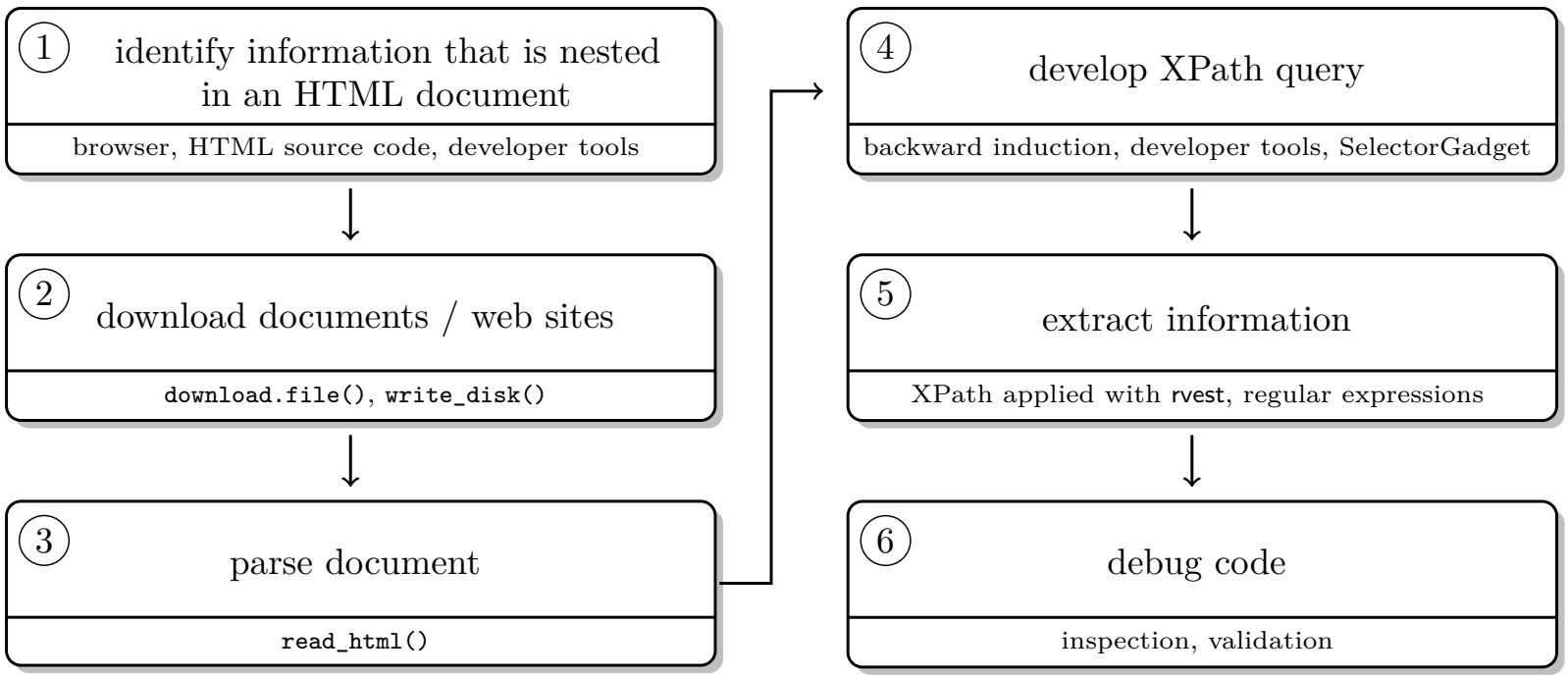
Where you stand now

You have learned the main tools necessary to scrape static webpages with R

1. you are able to inspect HTML pages in your browser using the web developer tools
2. you are able to parse HTML into R with `rvest`
3. you are able to speak XPath
4. you are able to apply XPath expressions with `rvest`
5. you are able to tidy web data with your R skills and regular expressions

The scraping workflow

The scraping workflow



Downloading HTML files

Stay modest when accessing lots of data

- content on the web is publicly available, ...
- but accessing the data causes server traffic
- stay polite by querying resources as sparsely as possible

Downloading HTML files

Stay modest when accessing lots of data

- content on the web is publicly available, ...
- but accessing the data causes server traffic
- stay polite by querying resources as sparsely as possible

Two easy-to-implement practices

1. do not bombard the server with requests—and if you have to, do at a reasonable speed
2. download HTML files first, then parse

Downloading HTML files

R code

```
1  for (i in 1:length(list_of_urls)) {
2      if (!file.exists(paste0(folder, file_names[i]))) {
3          download.file(list_of_urls[i], destfile = paste0(folder, file_
   names[i]))
4          Sys.sleep(runif(1, 1, 2))
5      }
6  }
```

end

Looping over a list of URLs

- `!file.exists()` checks whether a file does not yet exist in the local folder
- `download.file()` downloads the file to a folder; file name has to be specified
- `Sys.sleep()` suspends the execution of R code for a given time interval. Here: random interval between 1 and 2 seconds

Staying identifiable

Don’t be a phantom

- downloading massive amounts of data may arouse attention from server adminstrators
- assuming that you’ve got nothing to hide, you should stay identifiable beyond your IP address

Staying identifiable

Don't be a phantom

- downloading massive amounts of data may arouse attention from server adminstrators
- assuming that you've got nothing to hide, you should stay identifiable beyond your IP address

Two easy-to-implement practices

1. personally get in touch with website owners
2. use HTTP header fields `From` and `User-Agent` to provide information about yourself

Staying identifiable

R code

```
7  url <- "http://a-totally-random-website.com"
8  session <- html_session(url, add_headers(From = "my@email.com", `User-
   Agent` = R.Version()$version.string))
9  headlines <- session %>% html_nodes(xpath = "p//a") %>%  html_text()
```

end

The code snippet explained

- `rvest`'s `html_session()` creates a session object that responds to HTTP and HTML methods
- here, we provide our email address and the current R version as User-Agent information
- this will pop up in the server logs—the webpage adminstrator has the chance to easily get in touch with you

Summary

Summary

- the basic scraping workflow with R is straightforward
- with great power comes great responsibility: stay polite on the web when scraping lots of data!
- more complexity is added when you want to gather data from multiple websites, or when dynamic elements such as forms or JavaScript content is involved
- we will consider such cases in upcoming sessions

Cologne Center for Comparative Politics

A Primer to Web Scraping with R

Dynamic Webpages

Simon Munzert
Hertie School of Governance, Berlin

November 12, 2018

Static webpages

The simple world of static webpages

Static webpages

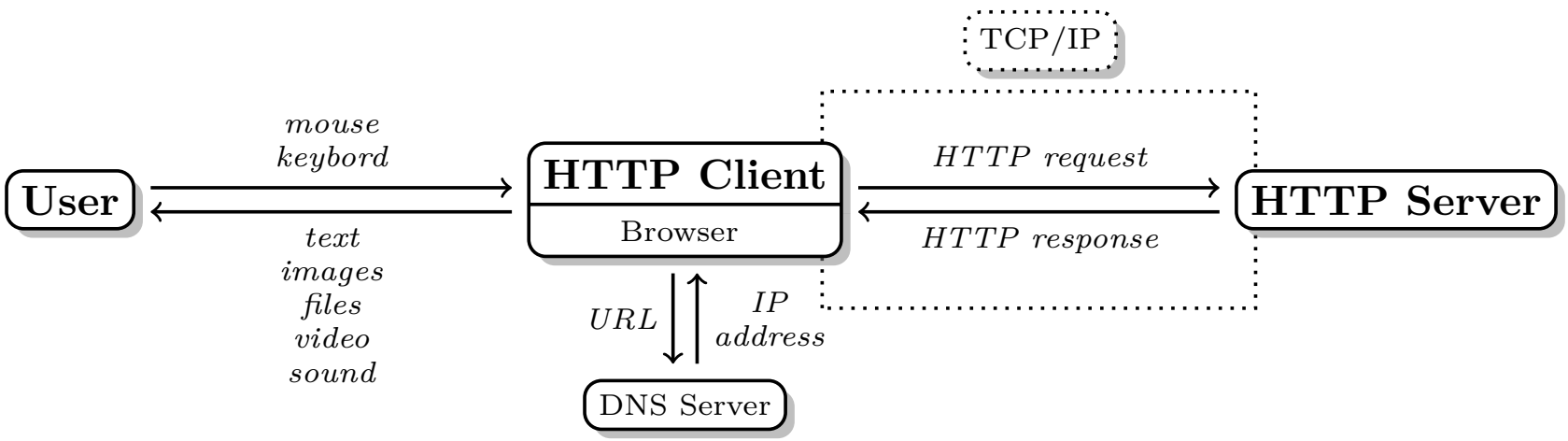
- every user who visits the site gets the same content (unless the developer edits the source code)
- example: https://www.jstatsoft.org
- can contain ”dynamic features”, such as video or sound, but the page itself is not dynamic

Screenshot of Amazon.com in 1997

A tiny bit of HTTP

Client-server communication with HTTP

- HTTP, the **H**yper**t**ext **T**ransfer **P**rotocol, is a stateless protocol
- no information is retained by either sender or receiver
- makes interaction with websites straightforward, but not very exciting



A tiny bit of HTTP

Client-server communication with HTTP

1. Establishing connection

```
About to connect() to www.r-datacollection.com port 80 (#0)
  Trying 173.236.186.125... connected
Connected to www.r-datacollection.com (173.236.186.125) port 80 (#0)
Connection #0 to host www.r-datacollection.com left intact
```

2. HTTP request

```
GET /index.html HTTP/1.1
Host: www.r-datacollection.com
Accept: */*
```

A tiny bit of HTTP

Client-server communication with HTTP

3. HTTP response

```
HTTP/1.1 200 OK
Date: Thu, 27 Feb 2014 09:40:35 GMT
Server: Apache
Vary: Accept-Encoding
Content-Length: 131
...

<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML//EN">
<html> <head>
<title></title>
</head>
...
```

4. Closing connection

```
Closing connection #0
```

The ”problem” of static webpages

Static webpages reconsidered

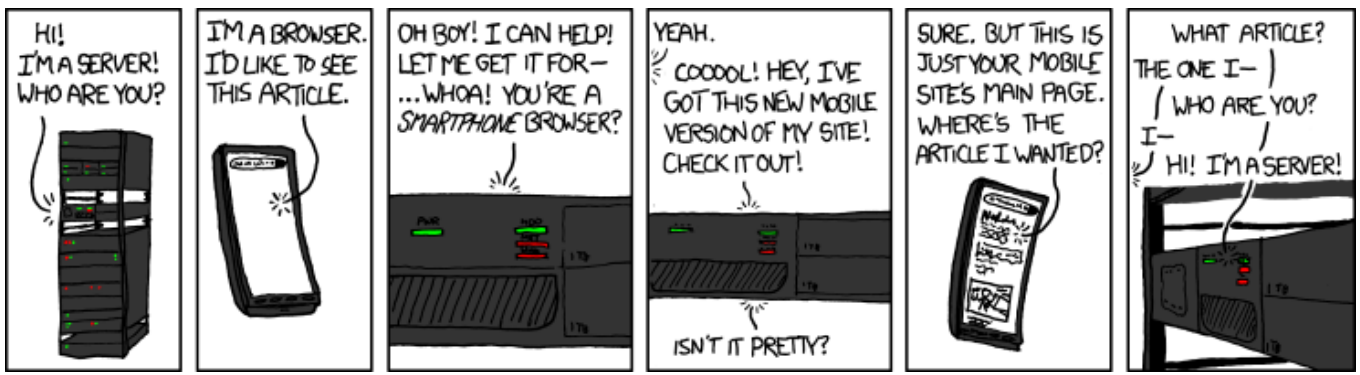
- HTML/HTTP are used for static display of content → same content for every visitor
- in order to display dynamic content, they lack
  1. mechanisms to detect user behavior in the browser (and not only on the server)
  2. a scripting engine that reacts on this behavior
  3. a mechanism for asynchronous queries

Dynamic webpages

The not so simple world of dynamic webpages

Dynamic webpages

- with dynamic webpages, the displayed content can differ between users, even if the source code is the same
- things that can cause the display of different content:
  - operating system, browser, device
  - user actions on the page (mouse movements, scrolling, clicks, keyboard strokes)
  - conditions on the server-side



Source: https://xkcd.com/869/ (Randall Munroe)

The not so simple world of dynamic webpages

Dynamic webpages

- massively used in modern webpage design and architecture
- (are thought to) enhance the user experience
- allow for much more ways to interact with webpage content

The not so simple world of dynamic webpages

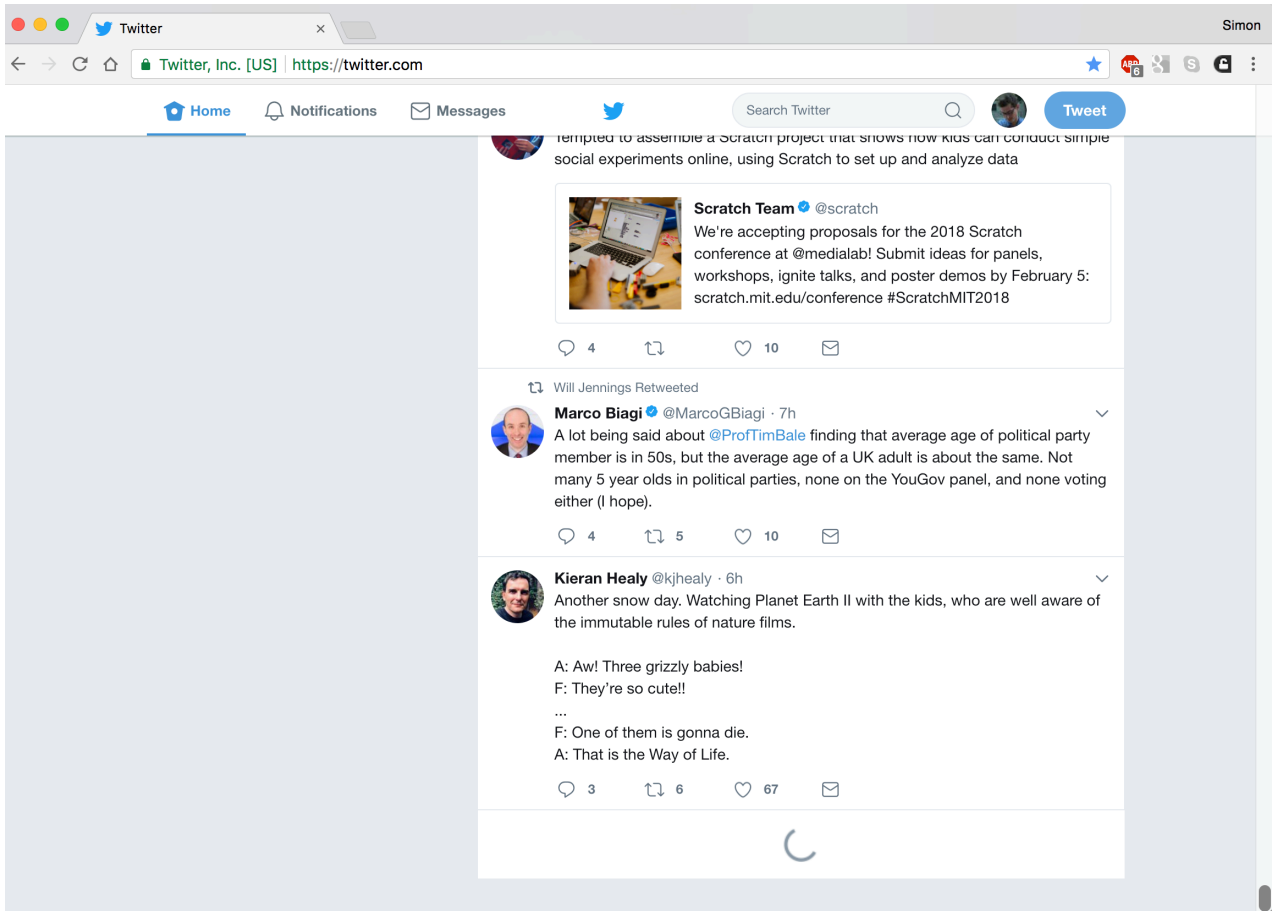
Dynamic webpages

- massively used in modern webpage design and architecture
- (are thought to) enhance the user experience
- allow for much more ways to interact with webpage content

Technologies

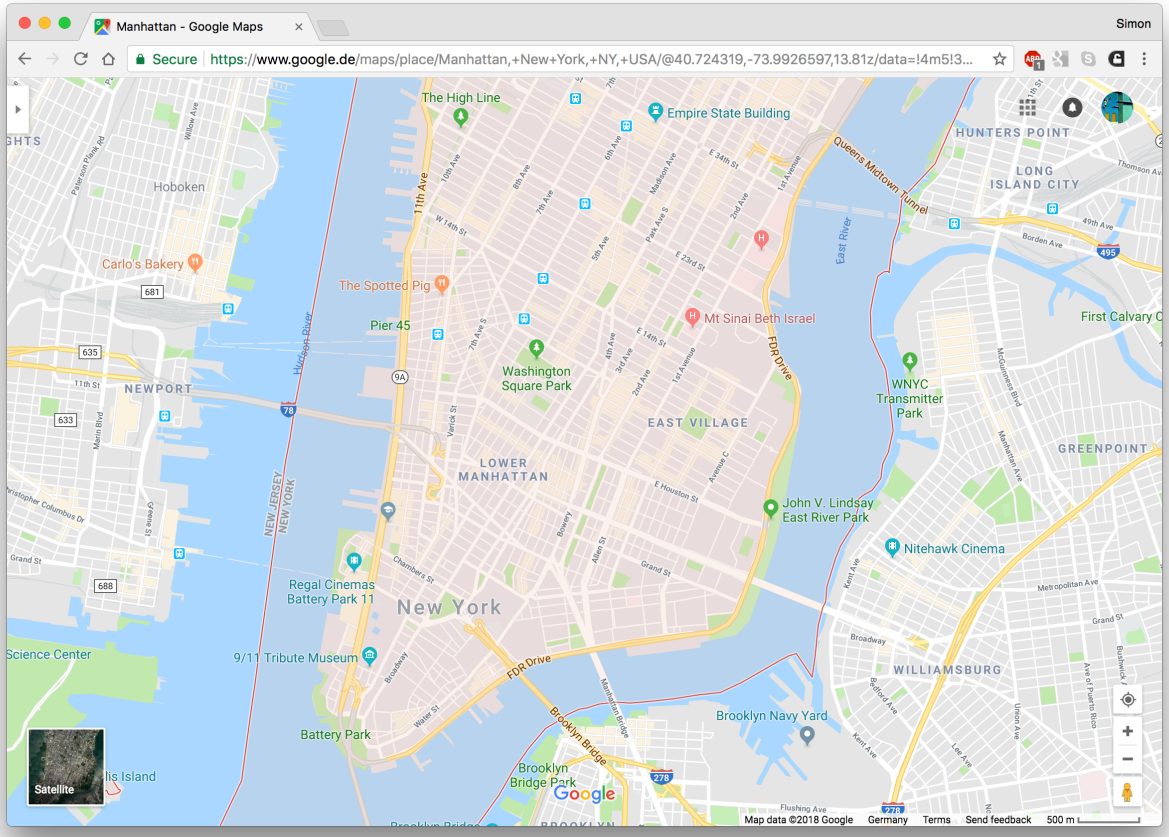
- in the case of **server-side** dynamic webpages, scripts on the server control webpage construction (e.g., PHP script reacts to user input to a form and returns subsets of a database)
- in the case of **client-side** dynamic webpages, (typically) JavaScript embedded in HTML determine what is displayed and how the DOM is changed
- a combination of these technologies is **A**synchronous **J**avaScript **a**nd **X**ML (AJAX)

Examples of dynamic webpages



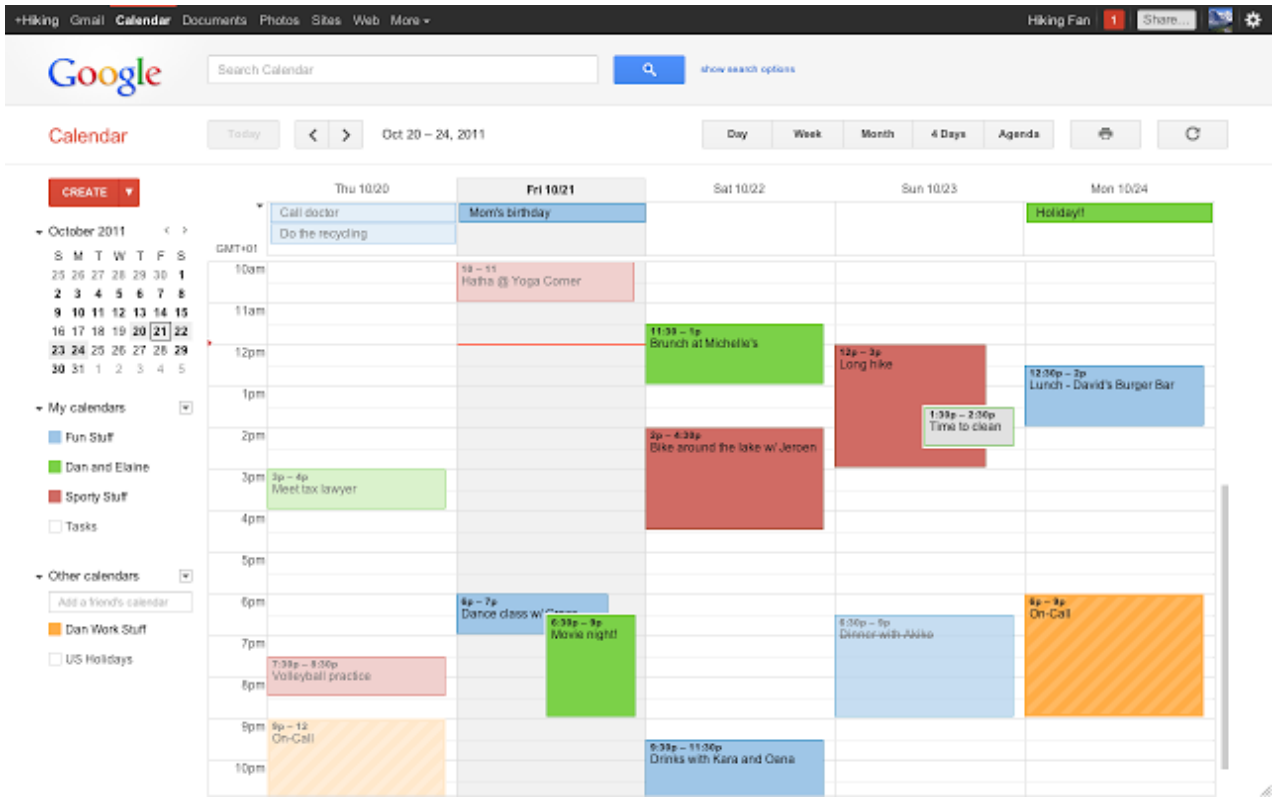
Your Twitter or Facebook feed that automatically gets updated when you scroll down

Examples of dynamic webpages



The map application that automatically loads new content when you zoom in

Examples of dynamic webpages



The calendar app that displays personal information and lets you interact a lot

The problem of dynamic webpages

The problem of dynamic webpages

The problem of dynamic webpages

- the tools you have encountered so far operated on the static source code: you access a page, download it, parse it

The problem of dynamic webpages

- the tools you have encountered so far operated on the static source code: you access a page, download it, parse it
- but what if content on the page changes, but the source code does not?

The problem of dynamic webpages

- the tools you have encountered so far operated on the static source code: you access a page, download it, parse it
- but what if content on the page changes, but the source code does not?
- dynamic webpages make classical screen scraping more difficult if not impossible

The problem of dynamic webpages

- the tools you have encountered so far operated on the static source code: you access a page, download it, parse it
- but what if content on the page changes, but the source code does not?
- dynamic webpages make classical screen scraping more difficult if not impossible
- and: dynamically rendered webpages become more and more common

The problem of dynamic webpages

- the tools you have encountered so far operated on the static source code: you access a page, download it, parse it
- but what if content on the page changes, but the source code does not?
- dynamic webpages make classical screen scraping more difficult if not impossible
- and: dynamically rendered webpages become more and more common
- before we learn about tools that can help us extract data in such scenarios, we will briefly consider the underlying processes, in particular AJAX technologies

Cologne Center for Comparative Politics

A Primer to Web Scraping with R

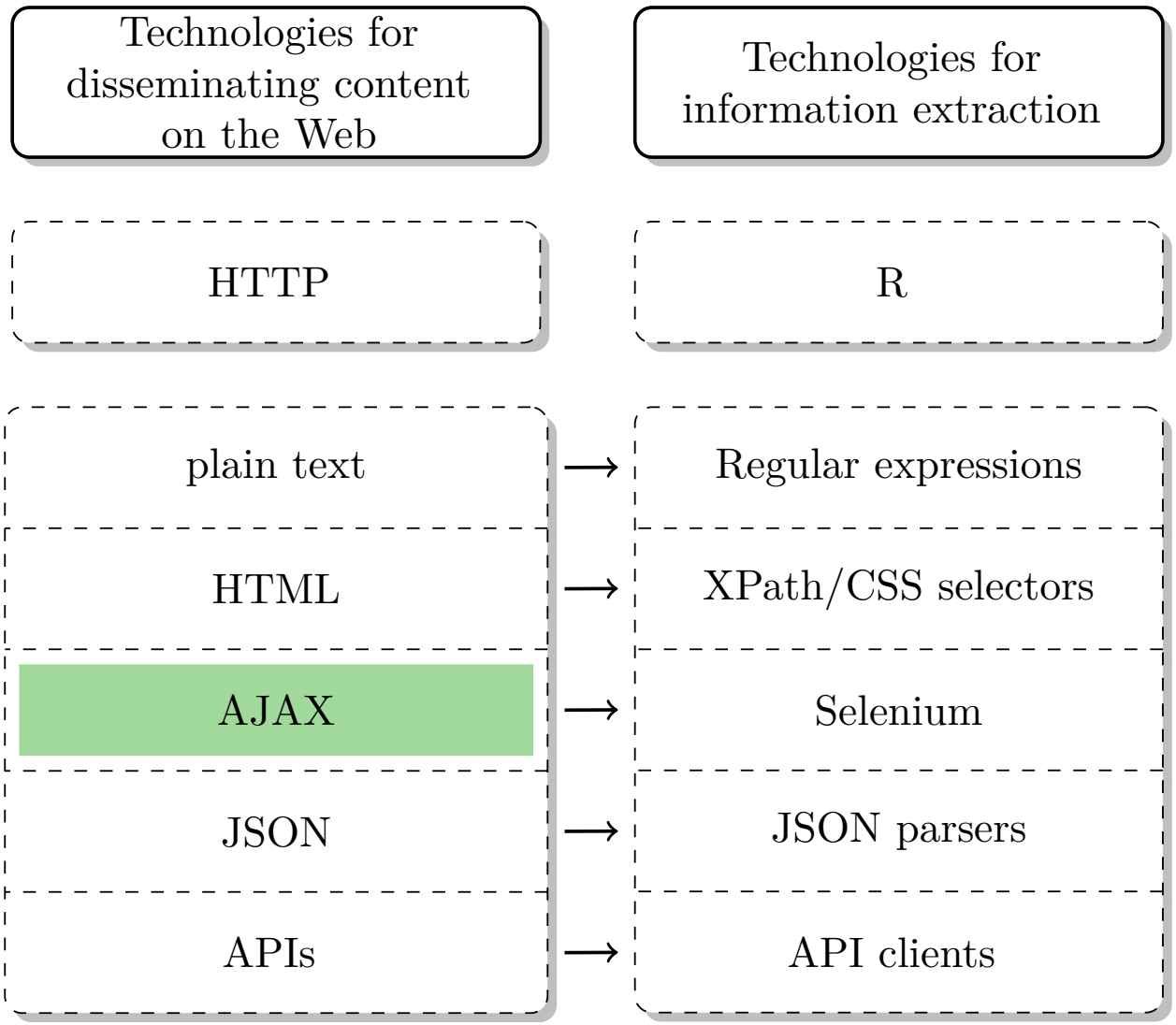
AJAX Technologies

Simon Munzert
Hertie School of Governance, Berlin

November 12, 2018

AJAX

Technologies of the World Wide Web



What's AJAX?

Purpose

- recall: HTML/HTTP lack
  1. mechanisms to detect user behavior in the browser
  2. a scripting engine that reacts on this behavior
  3. a mechanism for asynchronous queries
- **A**synchronous **J**avaScript **a**nd **X**ML is a set of technologies that serve these purposes

What's AJAX?

Purpose

- recall: HTML/HTTP lack
  1. mechanisms to detect user behavior in the browser
  2. a scripting engine that reacts on this behavior
  3. a mechanism for asynchronous queries
- **A**synchronous **J**avaScript **a**nd **X**ML is a set of technologies that serve these purposes

Components

- HTML/CSS for presentation
- Document Object Model (DOM; "tree structure") to interact with data
- JSON/XML for data interchange
- XMLHttpRequest for asynchronous communication
- JavaScript as a scripting language

JavaScript

JavaScript

What's JavaScript? I

- Programming language that connects well to web technologies
- W3C web standard
- native browser support (built-in JS engine)
- nowadays employed on the majority of websites
- extensible by libraries, e.g. *jQuery*

JavaScript

What's JavaScript? I

- Programming language that connects well to web technologies
- W3C web standard
- native browser support (built-in JS engine)
- nowadays employed on the majority of websites
- extensible by libraries, e.g. *jQuery*

What's JavaScript? II

- technology to make webpages interactive ('dynamic HTML")
- allows to...
  - animate page elements
  - offer interactive content (games, video, ...)
  - manipulate page content and communication with the server without reloading the page

JavaScript on the Web

How's JavaScript code embedded in HTML?

- between `<script>` tags
- as an external reference in the `src` attribute of a `<script>` element
- directly in certain HTML attributes ('event handler')

JavaScript on the Web

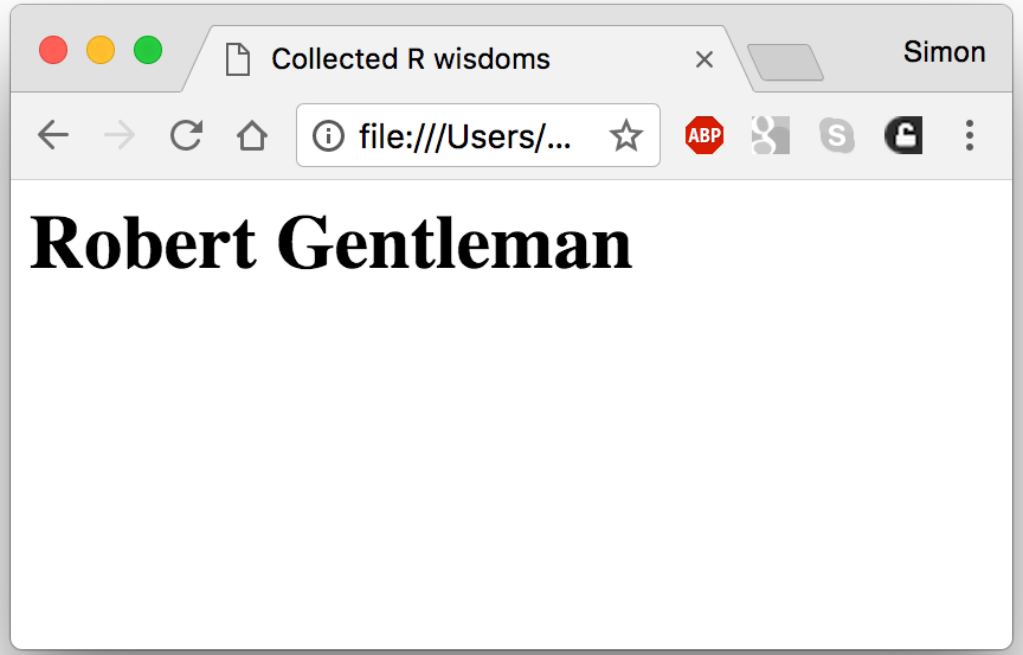
DOM manipulation with JavaScript

- adding/removing HTML elements
- changing attributes
- modification of CSS styles
- ...

Example:

```
1 <script type="text/javascript" src="jquery-1.8.0.min.js"></script>
2 <script type="text/javascript" src="script1.js"></script>
```

JavaScript on the Web



JavaScript on the Web

```
1  <!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML//EN">
2  <html>

4  <script type="text/javascript" src="jquery-1.8.0.min.js"></script>
5  <script type="text/javascript" src="script1.js"></script>

7  <head>
8  <title>Collected R wisdoms</title>
9  </head>

11 <body>
12 <div id="R Inventor" lang="english" date="June/2003">
13   <h1>Robert Gentleman</h1>
14   <p><i>'What we have is nice, but we need something very different'</i>
         </p>
15   <p><b>Source: </b>Statistical Computing 2003, Reisensburg</p>
16 </div>
17 </body>
18 </html>
```

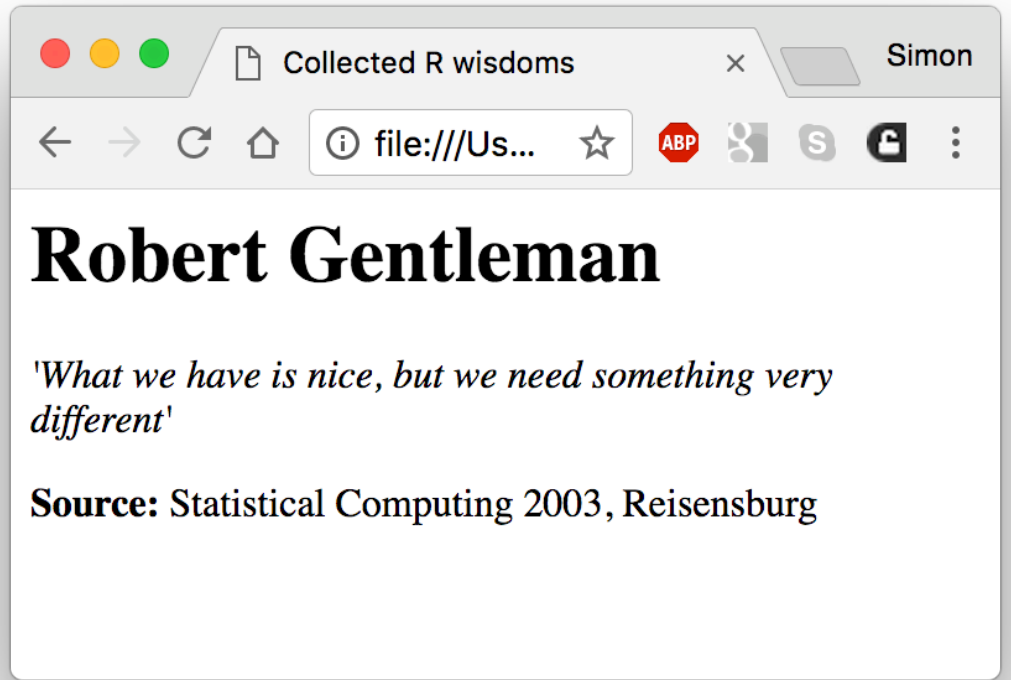
JavaScript on the Web

A JavaScript code snippet

```
1 $(document).ready(function() {
2     $("p").hide();
3     $("h1").click(function(){
4         $(this).nextAll().slideToggle(300);
5     });
6 });
```

- `$()` operator: addresses DOM elements
- `ready()`: JavaScript execution starts when the complete DOM is ready, i.e. fetched from the server
- `hide()`: element is hidden at first place
- `click` event: identifies mouse click and executes a certain action
- `nextAll()`: all subsequent elements in the DOM are addressed
- `slideToggle()`: Toggle effect, 300 milli-seconds

JavaScript on the Web



Summary

Summary

- AJAX technologies allow for a dynamic manipulation of the HTML tree
- what the user sees in the browser is not necessarily what's in the static HTML source code
- elements can be added, removed, or changed
- the "live" DOM tree that you saw in the Web Developer Tools takes track of such changes
- but you cannot simply access this live HTML with R and `rvest`
- we need another technology that helps us mimic a session in the browser to render all dynamic content

Cologne Center for Comparative Politics

A Primer to Web Scraping with R

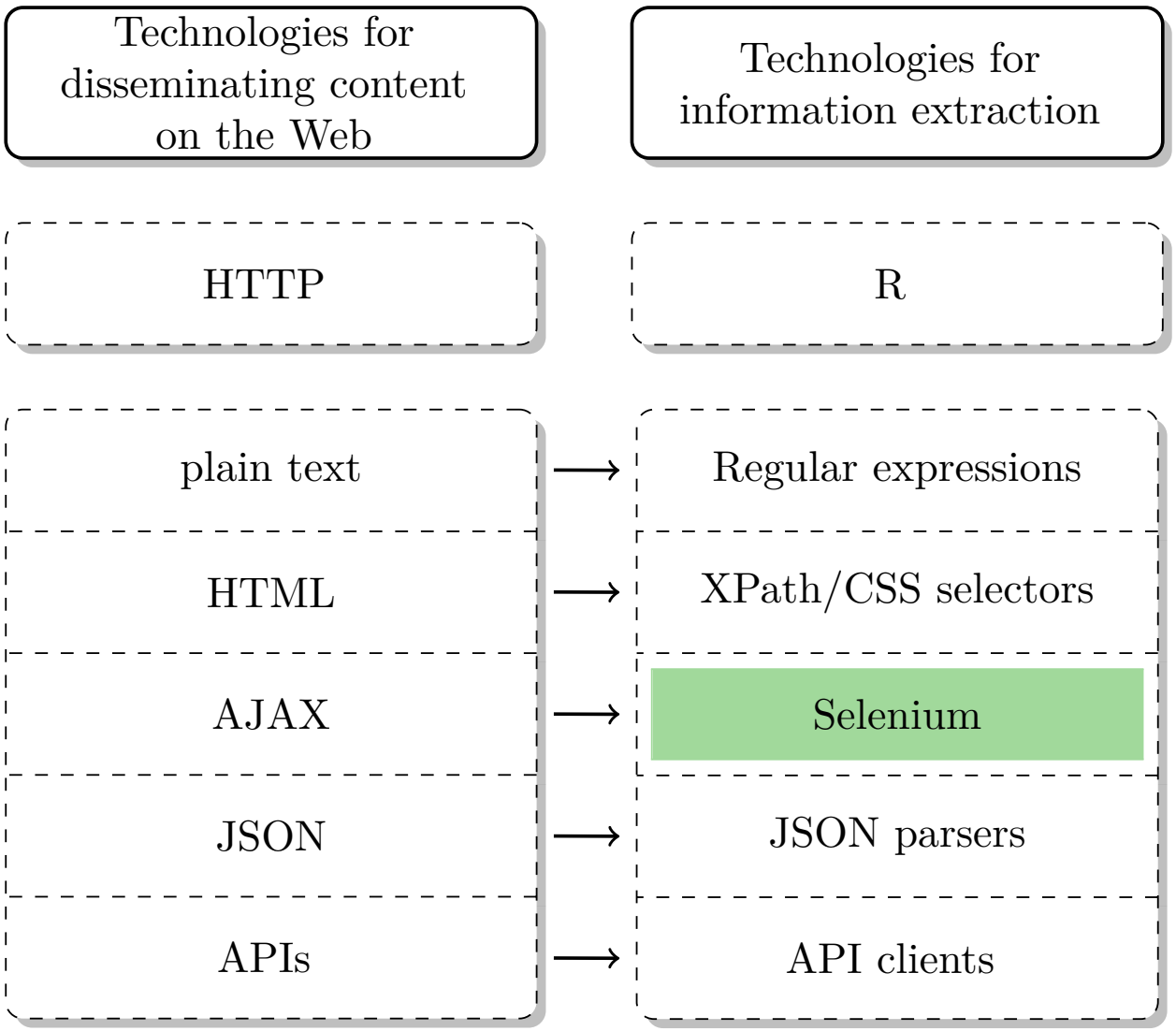
Selenium: Basics

Simon Munzert
Hertie School of Governance, Berlin

November 12, 2018

Selenium

Technologies of the World Wide Web



Scraping dynamic webpages

The problem reconsidered

- dynamic data requests are not stored in the static HTML page
- therefore, we cannot access them with classical methods and packages (`httr`, `rvest`, `download.file()`, etc.)
- R is not a browser with a JavaScript rendering engine

Scraping dynamic webpages

The problem reconsidered

- dynamic data requests are not stored in the static HTML page
- therefore, we cannot access them with classical methods and packages (`httr`, `rvest`, `download.file()`, etc.)
- R is not a browser with a JavaScript rendering engine

The solution

- initiate and control a web browser session with R
- let the browser do the JavaScript interpretation work and the manipulations in the live DOM tree
- access information from the web browser session

Selenium

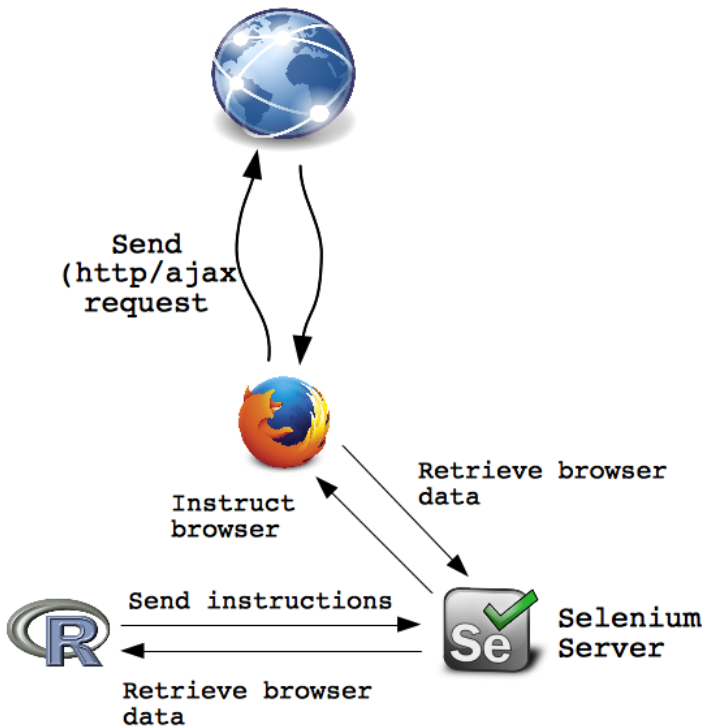
What's Selenium?

- free software environment for automated web application testing
- webpage: http://www.seleniumhq.org
- several modules for different tasks; most important for our purposes: Selenium WebDriver
- enables automated browsing via scripts

Selenium and R

The scraping workflow with Selenium and R

1. Selenium—an external, Java-based program—is launched
2. via the R package `RSelenium`, we remote-control Selenium and let it start a virtual server
3. we let Selenium start a browser session (e.g., Chrome)
4. everything we do in the browser—open a page, click on links or buttons, enter information—is described in R and sent to the server via the "remote-control" Selenium
5. we can gather the live HTML tree at any instance



Selenium and R

Software requirements

- Java, https://www.java.com/de/download/
- Selenium Standalone Server, newest version available here: http://www.seleniumhq.org/download/ or via RSelenium and `rsDriver()`
- browser (on most systems, both Chrome and Firefox seem to work reliably)
- `RSelenium` package

Cologne Center for Comparative Politics

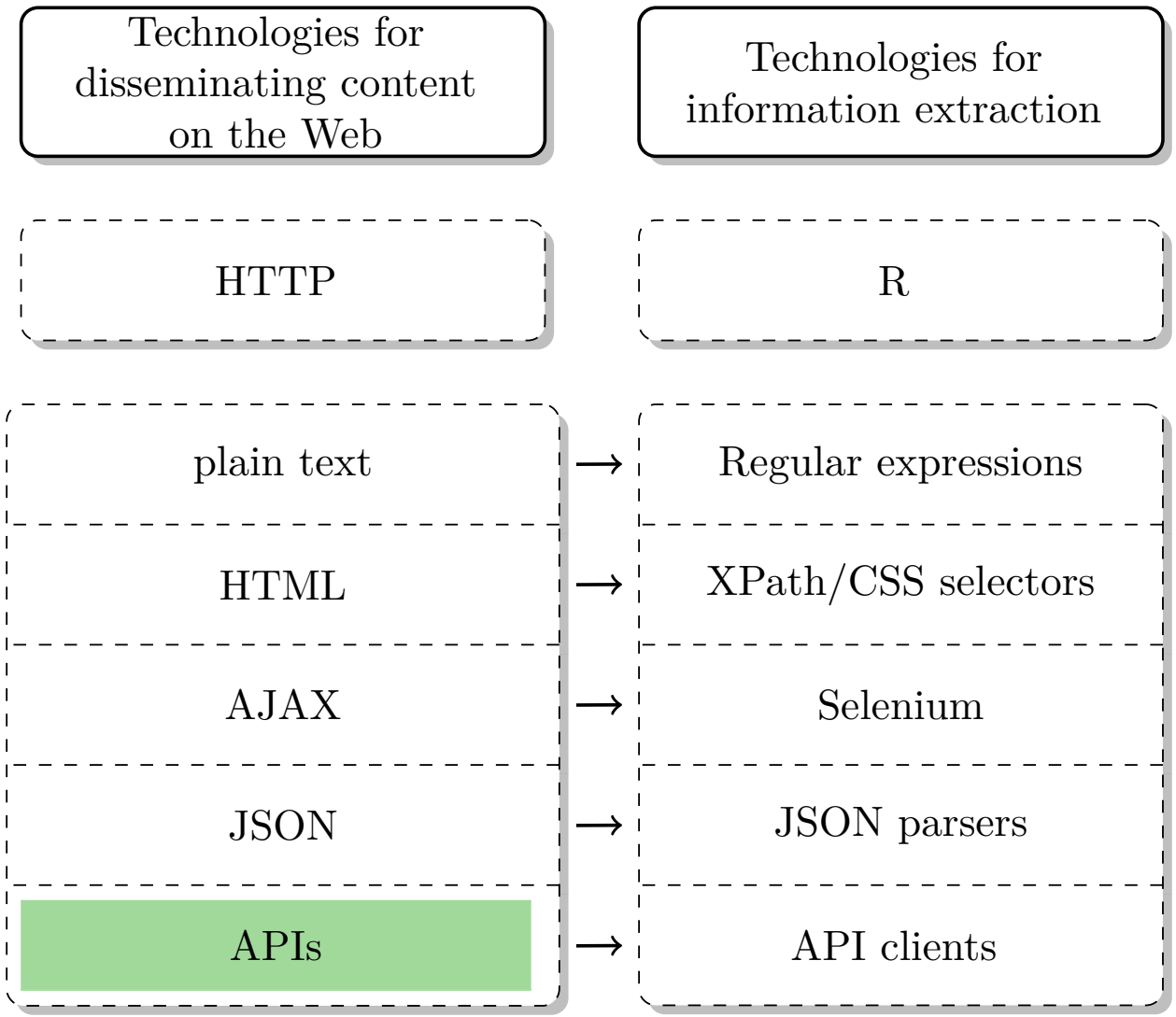
A Primer to Web Scraping with R

APIs

Simon Munzert
Hertie School of Governance, Berlin

November 12, 2018

Technologies of the World Wide Web



What are APIs?

What are APIs?

Definition

- **A**pplication **P**rogramming **I**nterface
- ”data search engine”: you pose a request, the API answers with a bulk of data
- let you/your program query a provider for specifc data
- common data formats: XML, JSON
- many popular web services provide APIs (Twitter, Google, Facebook, Wikipedia, ...)

What are APIs?

Definition

- **A**pplication **P**rogramming **I**nterface
- ”data search engine”: you pose a request, the API answers with a bulk of data
- let you/your program query a provider for specifc data
- common data formats: XML, JSON
- many popular web services provide APIs (Twitter, Google, Facebook, Wikipedia, ...)

Why we should care about APIs

- provide instant access to clean data
- free us from building manual scrapers
- API usage implies mutual data collection agreement

Example

Example

Google Maps API

- Google provides access to powerful location services
- free service (at least when used modestly)
- input/output: places, names, coordinates, maps, ...
- see also: https://developers.google.com/maps/documentation/

Example

Google Maps API

- Google provides access to powerful location services
- free service (at least when used modestly)
- input/output: places, names, coordinates, maps, ...
- see also: https://developers.google.com/maps/documentation/

Potential use cases

- geocode observations based on address, city, post code, ...
- calculate distances between observations
- map observations

Example

Access the API with R

- the `ggmap` package provides high-level functions to access API
- in this case, all we have to do is to figure out how the R function works – the communication with the API is processed completely in the background

Example

Access the API with R

- the ggmap package provides high-level functions to access API
- in this case, all we have to do is to figure out how the R function works – the communication with the API is processed completely in the background

R code

```
3  library(ggmap)
4  geocode("Berlin, Germany")
   Information from URL : http://maps.googleapis.com/maps/api/geocode/json?
   address=Berlin,%20Germany&sensor=false
         lon      lat
   1 13.40495 52.52001
```

end

Example

Excerpt from raw JSON behind the call


```
{
   "results" : [
      {
         "address_components" : [
            {
               "long_name" : "Berlin",
               "short_name" : "Berlin",
               "types" : [ "locality", "political" ]
            },
         ],
         "formatted_address" : "Berlin, Germany",
            "location" : {
               "lat" : 52.52000659999999,
               "lng" : 13.404954
            }
         },
         "place_id" : "ChIJAVkDPzdOqEcRcDteW0YgIQQ",
         "types" : [ "locality", "political" ]
      }
   ]
}
```

Example

Map the location

R code

```
5  get_googlemap("Berlin, Germany",
    zoom = 12, maptype = "hybrid")
   %>% ggmap()
```

end

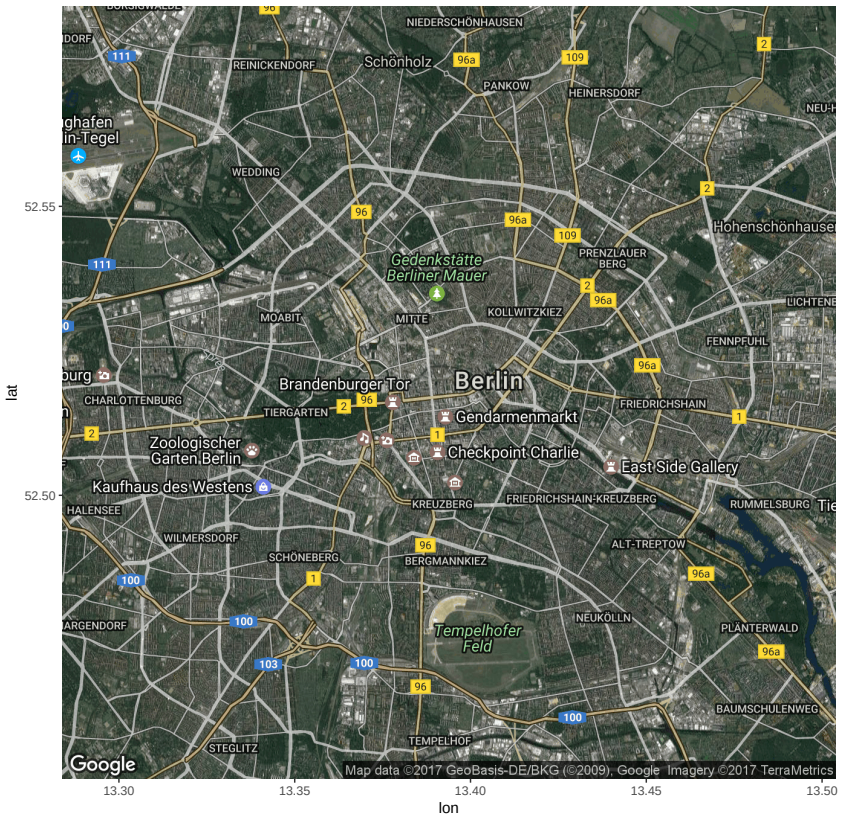
Example

Map the location

R code

```
6  get_googlemap("Berlin, Germany",
    zoom = 12, maptype = "hybrid")
   %>% ggmap()
```

end



Summary

Summary

Advantages of API use

- collecting data from the web using APIs provided by the data owner represents **the gold standard of web data retrieval**
- pure data collection without 'layout waste'
- standardized data access
- de facto automatic agreement
- robustness of calls

http://maxpixel.freegreatpicture.com/photo-1461569

Summary

Advantages of API use

- collecting data from the web using APIs provided by the data owner represents **the gold standard of web data retrieval**
- pure data collection without 'layout waste'
- standardized data access
- de facto automatic agreement
- robustness of calls

http://maxpixel.freegreatpicture.com/photo-1461569

Disadvantages of API use

- (sometimes) requires knowledge of API architecture
- dependent upon API suppliers
- use not always free

Cologne Center for Comparative Politics

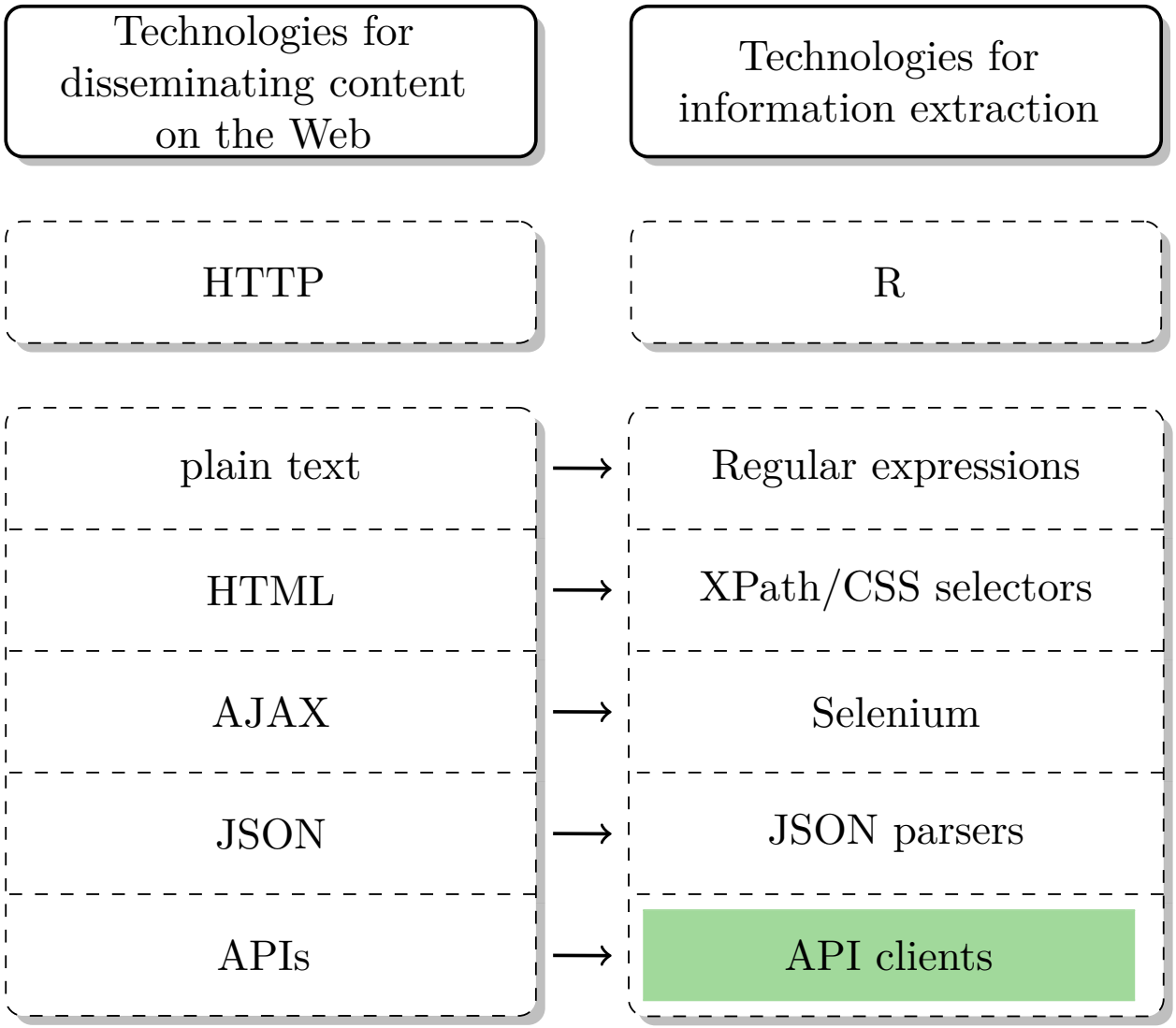
A Primer to Web Scraping with R

API Clients

Simon Munzert
Hertie School of Governance, Berlin

November 12, 2018

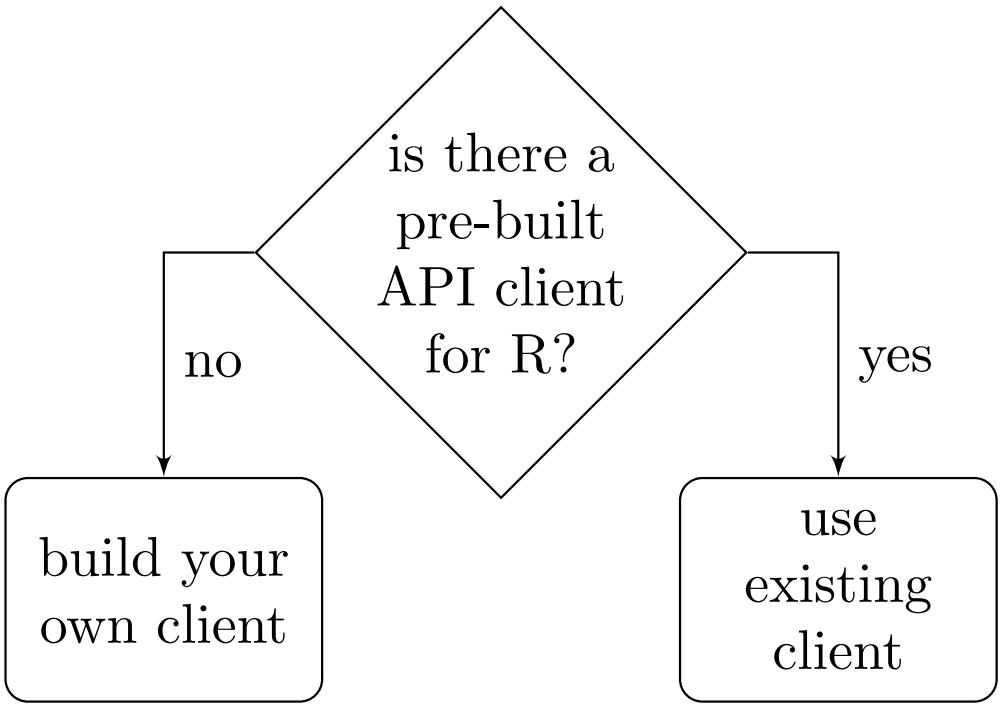
Technologies of the World Wide Web



Accessing APIs with R

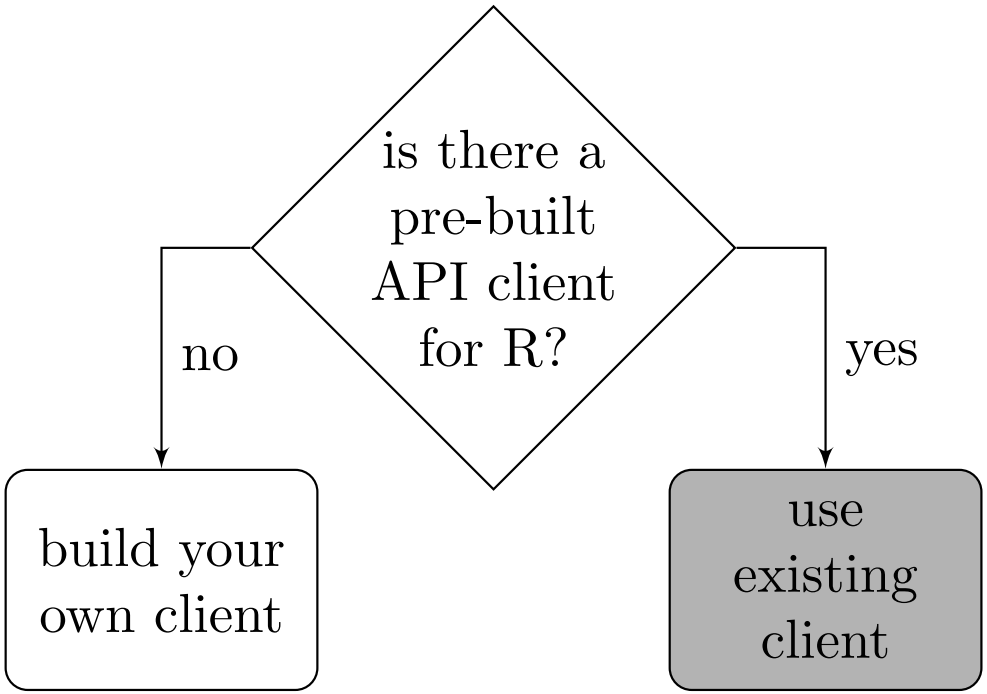
API access with R

There are basically **two scenarios:**



API access with R

There are basically **two scenarios:**



Here, we talk about the case where a client already exists.

API access with R

API clients

- provide interface to APIs
- hide API back-end
- let you stay in your programming environment

API access with R

API clients

- provide interface to APIs
- hide API back-end
- let you stay in your programming environment

Example

- the `rtweet` package provides an R client for Twitter
- lets you query data from the Twitter API with R commands
- you don't have to work with unfamiliar data formats (JSON) that is provided by the API—the client automatically transforms the incoming data into R objects

Finding API clients on the Web

General resources

List of APIs: http://www.programmableweb.com/apis

rOpenSci – collection of R API clients:
https://github.com/ropensci/opendata

CRAN Task View:
http://cran.r-project.org/web/views/WebTechnologies.html

Finding API clients on the Web

General resources

List of APIs: http://www.programmableweb.com/apis

rOpenSci – collection of R API clients:
https://github.com/ropensci/opendata

CRAN Task View:
http://cran.r-project.org/web/views/WebTechnologies.html

How to find the API client you need

- google "R package + name of website"
- search on the website for a "Developer" or "API" section (only if you don't find an R package that works)

Popular R API clients

package name	access to	more info
`rtweet`	Twitter Stream and REST API	http://rtweet.info/
`Rfacebook`	Facebook API	https://github.com/pablobarbera/Rfacebook
`ipapi`	ip-api.com's API	https://github.com/hrbrmstr/ipapi
`ggmap`	Google Maps/OpenStreetMap APIs	https://github.com/dkahle/ggmap
`eurostat`	Eurostat database	https://github.com/ropengov/eurostat
`rtimes`	New York Times APIs	https://cran.rstudio.com/web/packages/rtimes/index.html

Summary

Summary

- if the website you want to use as a data basis for your research provides access via an API, you can consider yourself lucky
- if there is a working R-based API client available, you hit the jackpot

Summary

- if the website you want to use as a data basis for your research provides access via an API, you can consider yourself lucky
- if there is a working R-based API client available, you hit the jackpot

Final considerations

- please always cite package authors when you use their work. run `citation("<package name>")` to see how
- sometimes API clients are not up to date. Consider to notify the author or help update the package
- sometimes API clients provide functionality for only a fraction of the API's capability. It's always worth to check out the API documentation if you are looking for specific features

Cologne Center for Comparative Politics

A Primer to Web Scraping with R

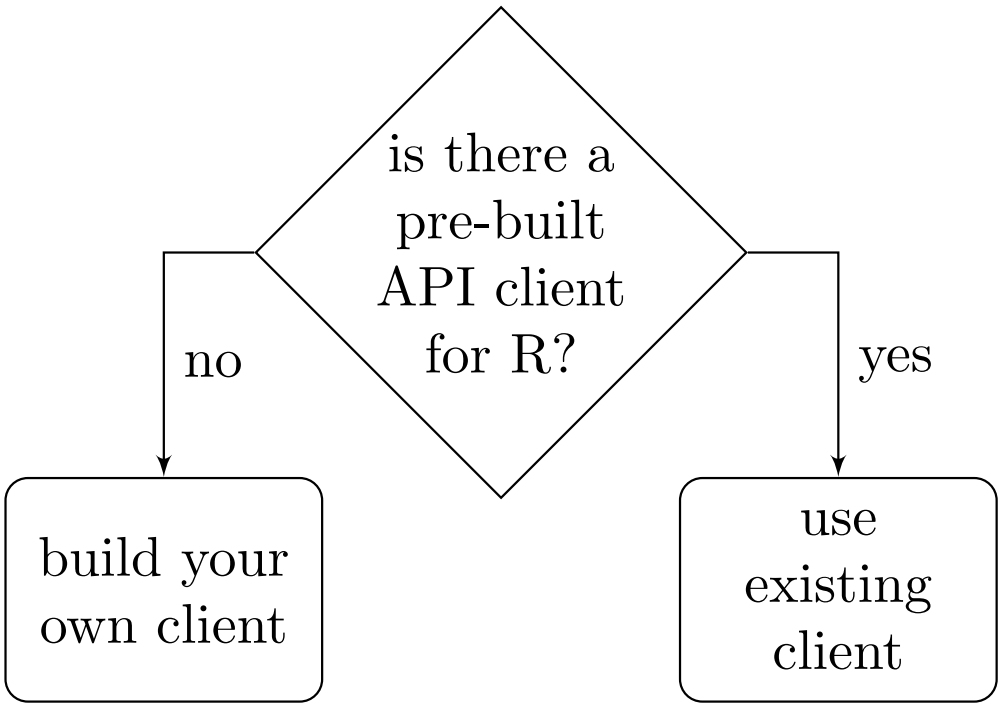
Accessing APIs from Scratch

Simon Munzert
Hertie School of Governance, Berlin

November 12, 2018

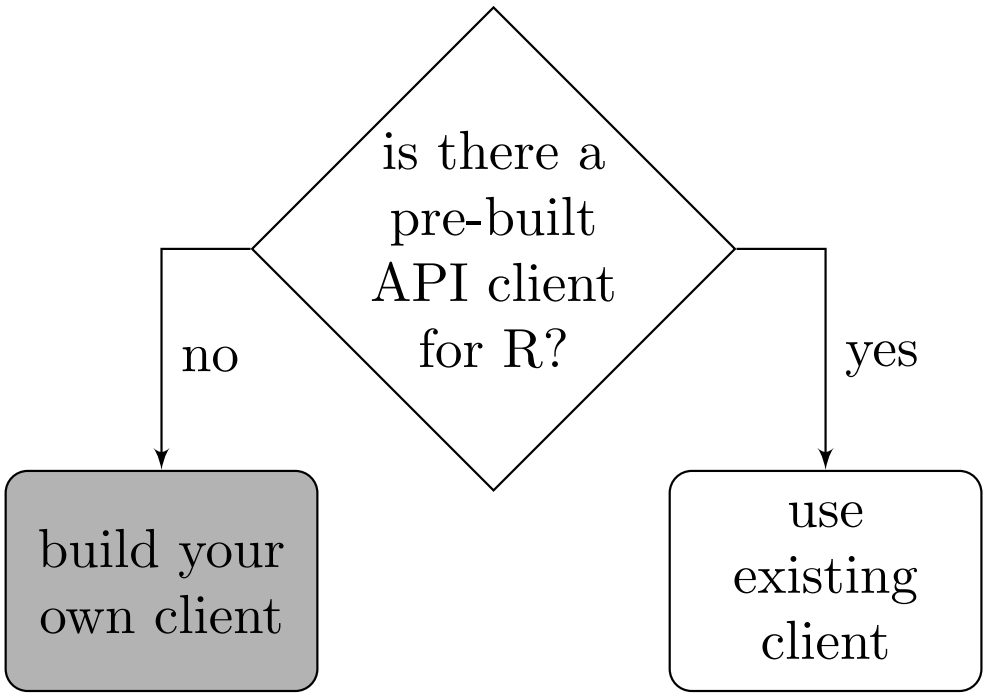
API access with R

There are basically **two scenarios:**



API access with R

There are basically **two scenarios:**



Here, we talk about the case where you have to build your own API binding.

Accessing APIs from Scratch

Accessing APIs from Scratch

Steps to be taken

Accessing APIs from Scratch

Steps to be taken

1. figure out how the API works: every API has a human-readable documentation for developers

Accessing APIs from Scratch

Steps to be taken

1. figure out how the API works: every API has a human-readable documentation for developers
2. build access to the API via R

Accessing APIs from Scratch

Steps to be taken

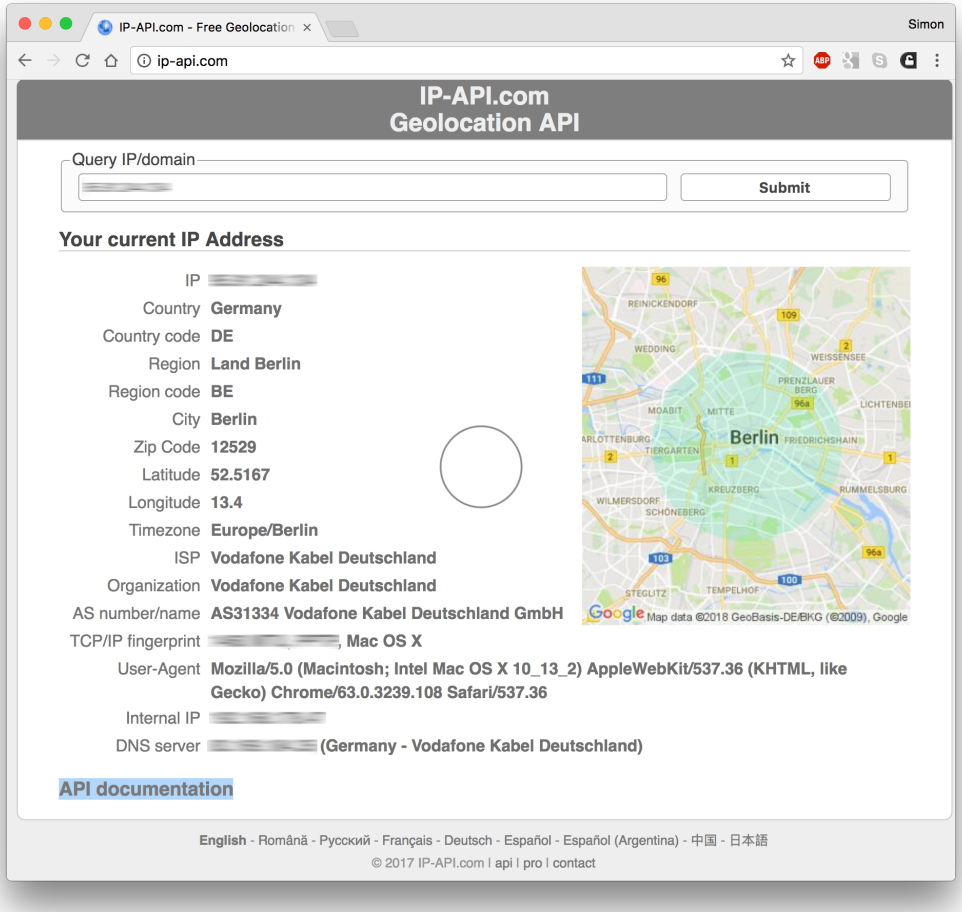
1. figure out how the API works: every API has a human-readable documentation for developers
2. build access to the API via R
3. build functionality that processes the data output (e.g., turns it into R objects)

Example

Example

The IP API

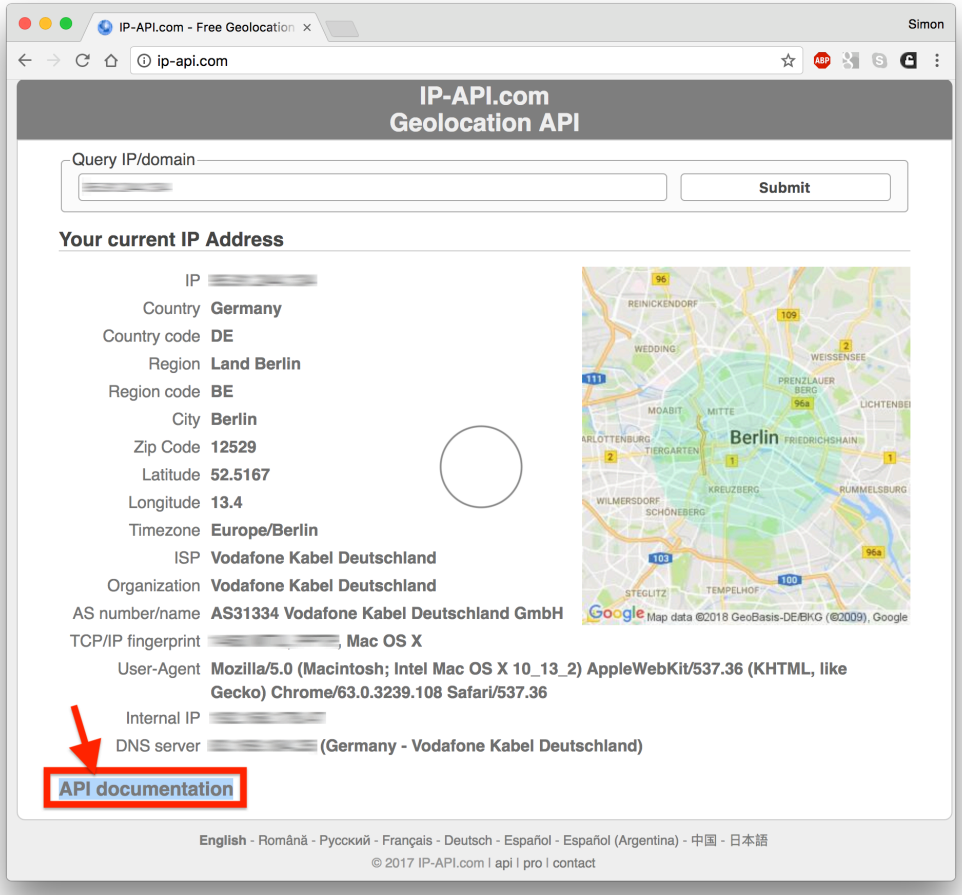
- available at http://ip-api.com/
- its purpose: parses your IP (or a bunch of given IPs) and returns some values, including guessed location, service provider, and organization behind the IP



Example

The IP API

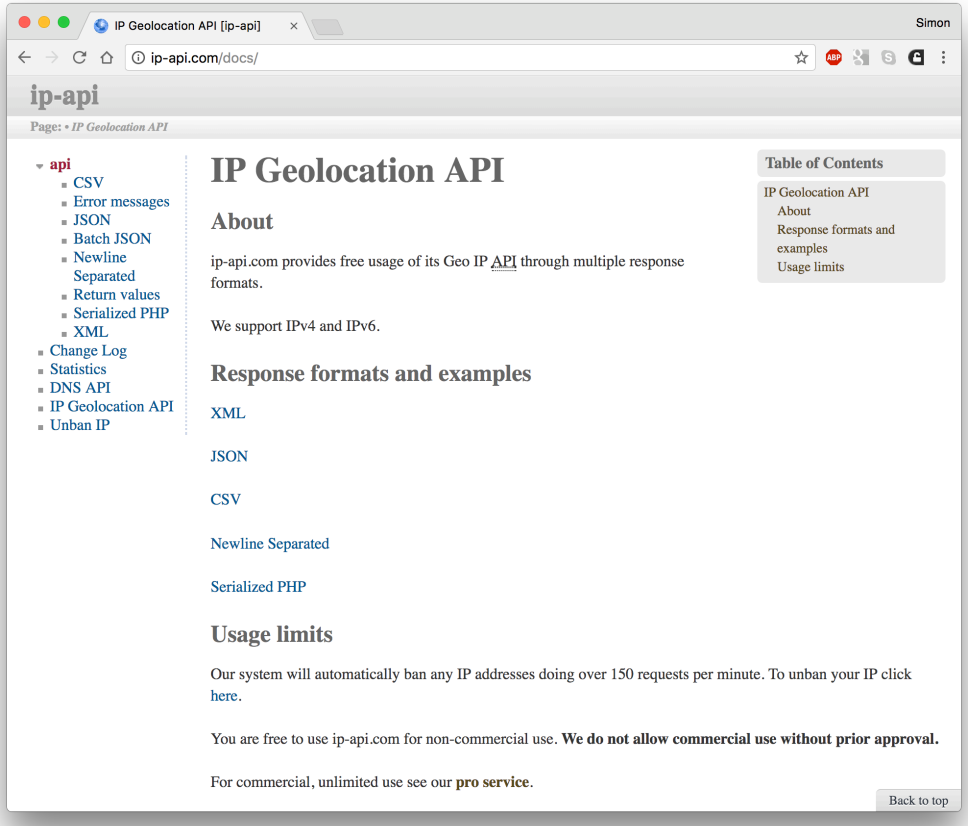
- available at http://ip-api.com/
- its purpose: parses your IP (or a bunch of given IPs) and returns some values, including guessed location service provider, and organization behind the IP
- check out API documentation



Example

The IP API

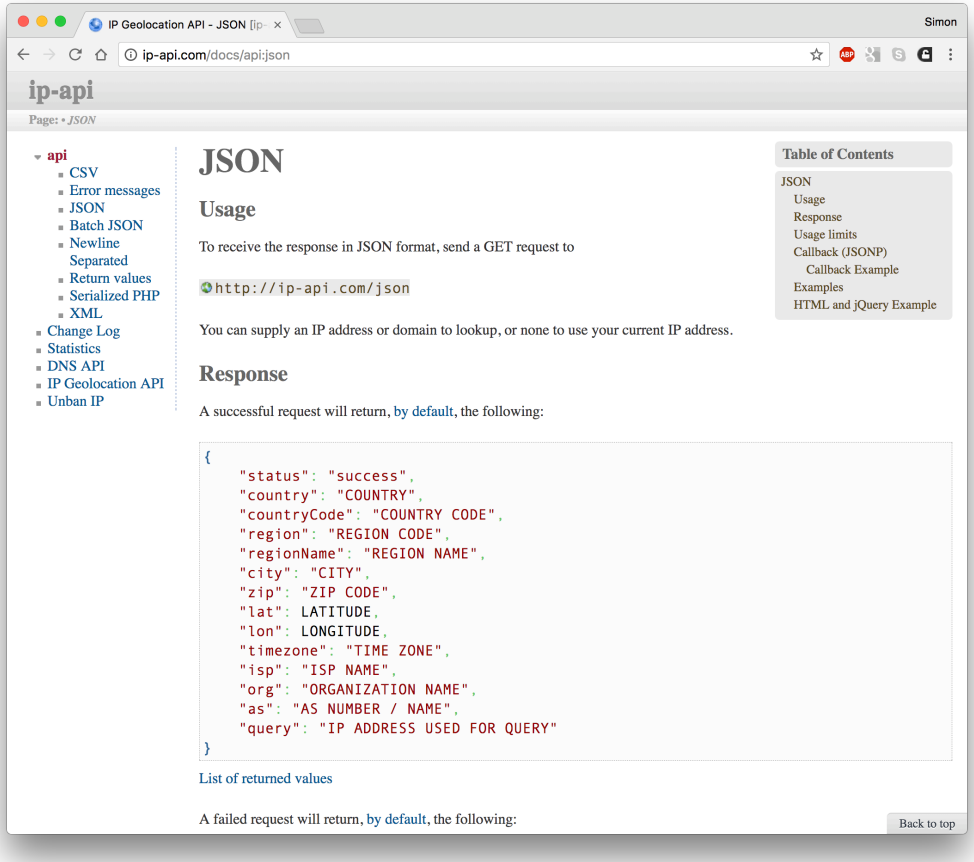
- access is free for non-commercial use (up to 150 requests per minute)
- various response formats available
- paid pro service available



Example

The IP API

- to access the API, we have to send a `GET` request to http://ip-api.com/json
- we can provide any IP we want
- the response is raw JSON code



Example

IP API access in R

A `GET` request essentially means that we assemble a URL using

- the API **endpoint** (a basic URL) and
- **parameters-value pairs** that are added to the URL
- here, we only add the literal IP address
- we use `fromJSON` to pose the request and directly parse the data from the API

R code

```
1  url <- "http://ip-api.com/json"
2  url_ip <- paste0(url, "/208.80.152.201")
3  ip_parsed <- jsonlite::fromJSON(url_ip)
```

end

Example

Investigating the output

R code

```
4  names(ip_parsed)
 [1] "as"          "city"        "country"     "countryCode" "isp"
 [6] "lat"         "lon"         "org"         "query"       "region"
[11] "regionName"  "status"      "timezone"    "zip"
5  ip_parsed
$as
[1] "AS14907 Wikimedia Foundation Inc."

$city
[1] "San Francisco (South Beach)"

$country
[1] "United States"

$countryCode
[1] "US"
```

Example

Bringing it into shape

- in our case, `fromJSON()` has created an R list object
- we turn it into a data frame

R code

```
6  ip_parsed %>% as.data.frame(stringsAsFactors = FALSE)
                                    as                          city
1 AS14907 Wikimedia Foundation Inc. San Francisco (South Beach)
        country countryCode                  isp    lat    lon
1 United States          US Wikimedia Foundation 37.787 -122.4
                    org          query region regionName  status
1 Wikimedia Foundation 208.80.152.201        California success
             timezone   zip
1 America/Los_Angeles 94105
```

end

Summary

Summary

- accessing APIs from scratch can be almost as easy as using a readily available R client
- often however, APIs are more complex (provide much more parameters and varations in data output)
- ideally, you build functions around your API calls to make it even more accessible

Cologne Center for Comparative Politics

A Primer to Web Scraping with R

Legal Issues

Simon Munzert
Hertie School of Governance, Berlin

November 12, 2018

Is web scraping legal?

Is web scraping legal?

Disclaimer

Obviously, I am not a lawyer. Do not rely on any of my comments on this topic. If you are seriously worried about the legality of your scraping work, please consult a legal expert.

Is web scraping legal?

Scraping vs. crawling

- **Web scraping**: downloading data from a very specific page in a (semi-)automated manner
- **Web crawling**: automatically downloading webpage data, extracting hyperlinks, following links, downloading webpage data, ... (e.g.: Googlebot, BaiduSpider)

This course mostly is scraping, not crawling or web harvesting!

Is web scraping legal?

Scraping vs. crawling

- **Web scraping**: downloading data from a very specific page in a (semi-)automated manner
- **Web crawling**: automatically downloading webpage data, extracting hyperlinks, following links, downloading webpage data, ... (e.g.: Googlebot, BaiduSpider)

This course mostly is scraping, not crawling or web harvesting!

- web scraping per se is not illegal
- there is no unambiguous **yes** or **no** for specific applications in any country according to current jurisdiction
- so far, legal cases (especially in the US) often (but not always) dealt with commercial interest, crawling applications, and often (but not always) huge masses of data, e.g., *eBay vs. Bidder's Edge, AP vs. Meltwater, Facebook vs. Pete Warden*

Is web scraping legal?

Why web scraping can be problematic

- violation of copyrights, Terms of Service, consumption of bandwith

Is web scraping legal?

Why web scraping can be problematic

- violation of copyrights, Terms of Service, consumption of bandwith

Some counter-arguments

- data is publicly accessible
  → but the website as a ”creative arrangement” might be copyrighted
- this is fair use → depends on your use
- it’s the same what my browser does
  → not exactly, and many ToS prohibit automated uses of their data
- this is unfair—Google’s business model is built on crawling the whole web
  → true, but you are not Google

Is web scraping legal?

Some interesting comments by Pablo Hoffman, co-founder of Scrapinghub

- As long as they don't crawl at a disruptive rate, scrapers do not breach any contract (in the form of terms of use) or commit a crime (as defined in the Computer Fraud and Abuse Act).
- Website's user agreement is not enforceable as a browse-wrap agreement because companies do not provide sufficient notice of the terms to site visitors.
- Scrapers accesses website data as a visitor, and by following paths similar to a search engine. This can be done without registering as a user (and explicitly accepting any terms).

(found on https://goo.gl/jTt4ER)

Action from the other side

Things webpage administrators can do to prevent you from scraping massive amounts of data from their pages

- block your IP address
- identify your approximate geolocation from your IP address, then block
- move content exclusively to web services / APIs
- block bots with a particular user agent string (more on that later)
- challenge-response tests like **C**ompletely **A**utomated **P**ublic **T**uring test to tell **C**omputers and **H**umans **A**part (CAPTCHA)
- obfuscation of data
- frequent changes in HTML/CSS

Summary

Summary

- there is no unconditional ”legal” or ”illegal” status of web scraping
- your use of the data can violate the data owner’s rights
- targeted scraping efforts with limited traffic usually do not cause any problems

Cologne Center for Comparative Politics

A Primer to Web Scraping with R

Good Practice

Simon Munzert
Hertie School of Governance, Berlin

November 12, 2018

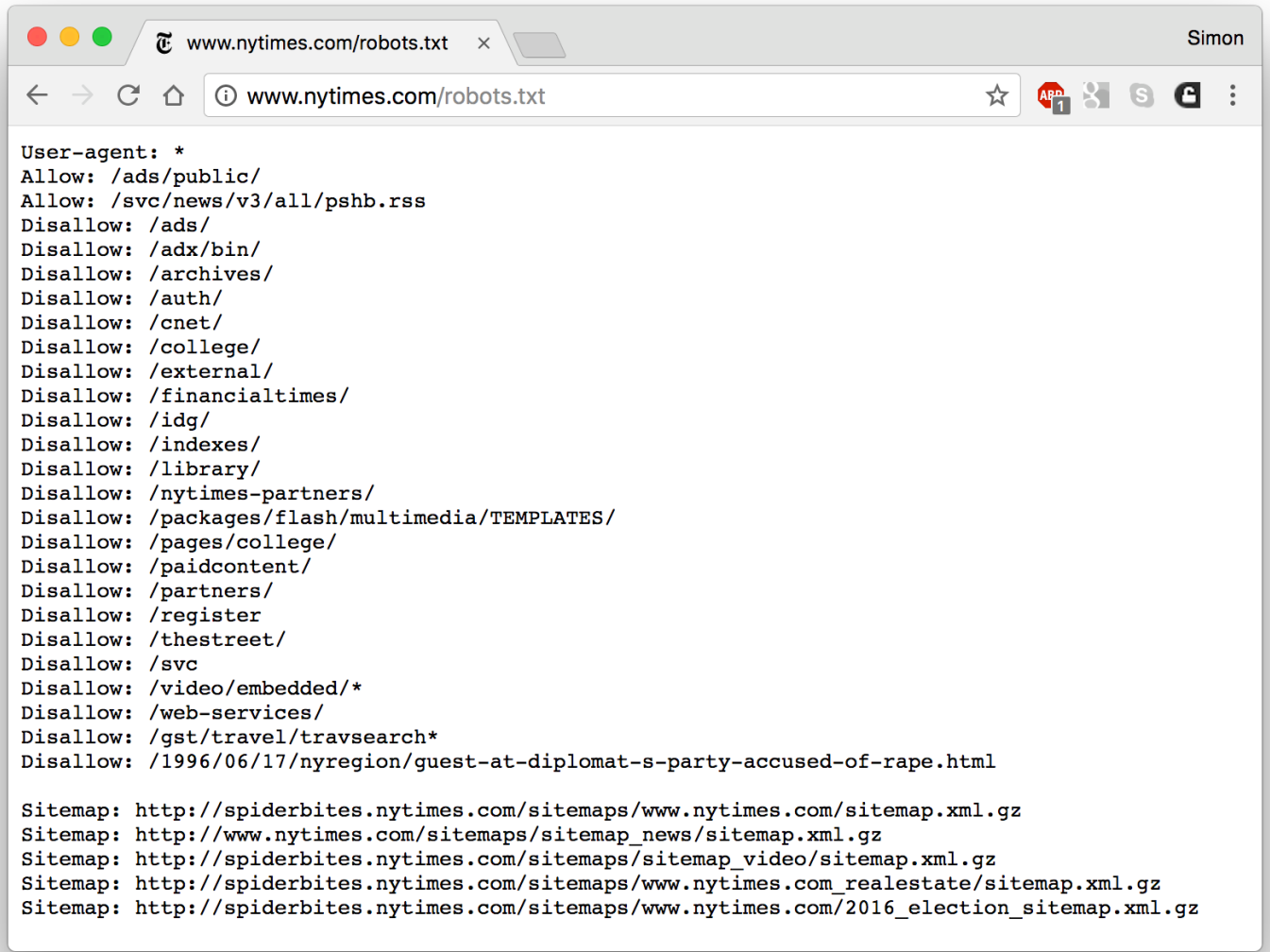
Understanding `robots.txt`

robots.txt

What's robots.txt?

- 'Robots Exclusion Protocol', informal protocol to prohibit web robots from crawling content
- located in the root directory of a website (e.g., http://www.google.com/robots.txt)
- documents which bot is allowed to crawl which resources (and which not)
- not a technical barrier, but a sign that asks for compliance

Example: NY Times's `robots.txt`




```
User-agent: *
Allow: /ads/public/
Allow: /svc/news/v3/all/pshb.rss
Disallow: /ads/
Disallow: /adx/bin/
Disallow: /archives/
Disallow: /auth/
Disallow: /cnet/
Disallow: /college/
Disallow: /external/
Disallow: /financialtimes/
Disallow: /idg/
Disallow: /indexes/
Disallow: /library/
Disallow: /nytimes-partners/
Disallow: /packages/flash/multimedia/TEMPLATES/
Disallow: /pages/college/
Disallow: /paidcontent/
Disallow: /partners/
Disallow: /register
Disallow: /thestreet/
Disallow: /svc
Disallow: /video/embedded/*
Disallow: /web-services/
Disallow: /gst/travel/travsearch*
Disallow: /1996/06/17/nyregion/guest-at-diplomat-s-party-accused-of-rape.html

Sitemap: http://spiderbites.nytimes.com/sitemaps/www.nytimes.com/sitemap.xml.gz
Sitemap: http://www.nytimes.com/sitemaps/sitemap_news/sitemap.xml.gz
Sitemap: http://spiderbites.nytimes.com/sitemaps/sitemap_video/sitemap.xml.gz
Sitemap: http://spiderbites.nytimes.com/sitemaps/www.nytimes.com_realestate/sitemap.xml.gz
Sitemap: http://spiderbites.nytimes.com/sitemaps/www.nytimes.com/2016_election_sitemap.xml.gz
```

Syntax in `robots.txt`

Syntax

- not an official W3C standard, partly inconsistent syntax
- rules listed bot by bot
- general, bot-independent rules under '*' (most interesting entry for R-based crawlers)
- directories/folders listed separately

```
User-agent: Googlebot
Disallow: /images/
Disallow: /private/
```

```
User-agent: *
Disallow: /private/
```

Syntax in `robots.txt`

Universal ban

```
User-agent: *
Disallow: /
```

Separation of bots by empty line

```
User-agent: Googlebot
Disallow: /images/

User-agent: Slurp
Disallow: /images/
```

`Allow` declaration

```
User-agent: *
Disallow: /images/
Allow: /images/public/
```

Syntax in `robots.txt`

Crawl-delay (in seconds)

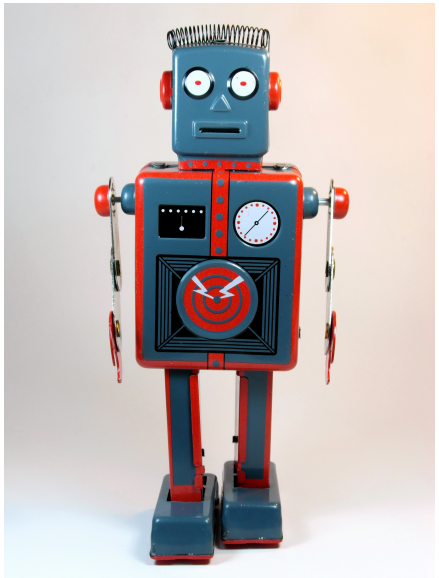
```
User-agent: *
Crawl-delay: 2
User-Agent: Googlebot
Disallow: /search/
```

Robots `<meta>` tag

```
<meta name="robots" content="noindex,nofollow" />
```

How to deal with `robots.txt`?

- not clear if robots.txt is legally binding or not, and if yes for which activities
- originally not thought of as protection against small-scale web scraping applications, but against large-scale indexing bots
- guide to a webmaster's preferences with regards to visibility of content
- my advice: take **`robots.txt`** into account! If the data you are interested in are excluded from crawling: contact webmaster
- there is even an R package, **`robotstxt`**, that helps you parse **`robots.txt`** documents and stick to the rules. It's available here: https://github.com/ropenscilabs/robotstxt



By D J Shin - My Toy Museum, https://commons.wikimedia.org/wiki/File:QSH_Tin_Wind_Up_Mechanical_Robot_(Giant_Easelback_Robot)_Front.jpg

Scraping etiquette

Some advice for your work

1. You take all the responsibility for your web scraping work.

Some advice for your work

1. You take all the responsibility for your web scraping work.
2. Take all copyrights of a country's jurisdiction into account.

Some advice for your work

1. You take all the responsibility for your web scraping work.
2. Take all copyrights of a country's jurisdiction into account.
3. If you publish data, do not commit copyright fraud.

Some advice for your work

1. You take all the responsibility for your web scraping work.
2. Take all copyrights of a country's jurisdiction into account.
3. If you publish data, do not commit copyright fraud.
4. If possible, stay identifiable.

Some advice for your work

1. You take all the responsibility for your web scraping work.
2. Take all copyrights of a country's jurisdiction into account.
3. If you publish data, do not commit copyright fraud.
4. If possible, stay identifiable.
5. If in doubt, ask the author/creator/provider of data for permission—if your interest is entirely scientific, chances aren't bad that you get data.

Some advice for your work

1. You take all the responsibility for your web scraping work.
2. Take all copyrights of a country's jurisdiction into account.
3. If you publish data, do not commit copyright fraud.
4. If possible, stay identifiable.
5. If in doubt, ask the author/creator/provider of data for permission—if your interest is entirely scientific, chances aren't bad that you get data.
6. Consult current jurisdiction, e.g. on http://blawgsearch.justia.com or from a laywer specialized on internet law.

Scraping etiquette