Web Scraping

Fundamentals of Computing and Data Display

Christoph Kern[1]

c.kern@uni-mannheim.de

09/16/2019

[1]Thanks to Malte Schierholz (BA, IAB)

Outline

Introduction

(Big) Web data for social science research

- Data originally related to some non-research purpose ("found" data)
- "Naturally occurring" as byproducts from various processes
  - e.g. social media data, personal data (tracking devices), sensor data, transactional data
- Case rich, variable poor

"The term 'Big Data' is an imprecise description of a rich and complicated set of characteristics, practices, techniques, ethical issues, and outcomes all associated with data." (Japec et al. 2015)

The three V's

Big Data Characteristics

- High volume
  - As a result of the increasing number of data-collection instruments
- High velocity 速度
  - New information is being added at high rates
- Great variety
  - Various sources create a heterogeneous set of data points

Advantages & Challenges

Advantages

- Less effort to collect the data
- "Naturally occurring" data can be more accurate
- Large volume allows detailed (subgroup) analysis

Challenges

- Substantial effort to prepare and clean the data
- Relevant variables may not naturally appear in data source
- Large volume can be a challenge for data processing
- **Inference and data quality**

→ AAPOR Task Force Report (Japec et al. 2015)

Advantages & Challenges

Quality of scraped web data

- **Integration:** Are there unique identifiers by which the scraped data can be linked with other data sources?
- **Coverage:** Are the scraped data complete (enough) for the question of interest?
- **Missing values?**
- **Credibility** and correctness of scraped data?

Web scraping

- Extracting/ converting (text)data from (HTML) websites into tables/ datasets
  - Process of utilizing structure of page code to grab data pieces
- Static HTML
  - Basic websites, data directly accessible
- Dynamic HTML
  - Interactive websites, scrolling and clicking needed to access data
- APIs
  - Data access is provided via an interface

与因特网相连的端系统提供了一个应用程序接口（英語：Application Programming Interface，缩写：API；又称为应用程序编程接口

First try API before HTML scraping

HTML

```
<!DOCTYPE html>
<html>
  <head>
    <title>Page Title</title>
    <style>
      table, td {border: 1px solid black;}
    </style>
  </head>

  <body>
    <h1>Table: Name and Age</h1>
    <a href='https://wikipedia.org/'>Click</a>
    <table>
      <tr>
        <td>Andrew Smith</td>
        <td>50</td>
      </tr>
      <tr>
        <td>Thomas Jackson</td>
        <td>94</td>
      </tr>
    </table>

  </body>
</html>
```

Start-Tag < >

End-Tag </ >

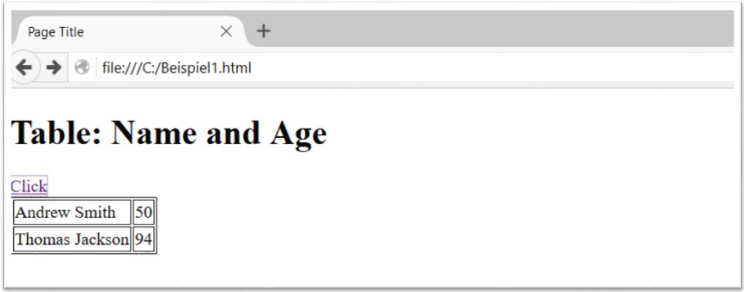
Type of document: HTML

Metadata
Usually not important for Web scraping

Visible content of the website
Includes our desired information

HTML

HTML page as displayed in a web browser



Read source code with R

```
install.packages( xml2 )
install.packages( rvest )
```

Install `xml2` and `rvest`

```
library( xml2 )
library( rvest )
```

Load `xml2` and `rvest`

```
src <- read_html( "C:/Beispiel1.html" )
```

Load/ parse the source code into `src`

Navigate a HTML document

```
<!DOCTYPE html>
<html>
  <head>
    <title>Page Title</title>
    <style>
      table, td {border: 1px solid black;}
    </style>
  </head>

  <body>
    <h1>Table: Name and Age</h1>
    <a href='https://wikipedia.org/'>Click</a>
    <table>
      <tr>
        <td>Andrew Smith</td>
        <td>50</td>
      </tr>
      <tr>
        <td>Thomas Jackson</td>
        <td>94</td>
      </tr>
    </table>

  </body>
</html>
```

XPath

Used to address elements in a HTML document

Usage of absolute paths

```
nds <- html_nodes( src,
xpath =
"/html/body/table/tr/td")

  {xml_nodeset (4)}
   [1] <td>Andrew Smith</td>
   [2] <td>50</td>
   [3] <td>Thomas Jackson</td>
   [4] <td>94</td>
```

```
html_text( nds )
   [1] "Andrew Smith" "50"
  "Thomas Jackson" "94"
```

`html_text` selects the text between the start- <> and the end-tag < / >

Navigate a HTML document

```
<!DOCTYPE html>
<html>
  <head>
    <title>Page Title</title>
    <style>
      table, td {border: 1px solid black;}
    </style>
  </head>

  <body>
    <h1>Table: Name and Age</h1>
    <a href='https://wikipedia.org/'>Click</a>
    <table>
      <tr>
        <td>Andrew Smith</td>
        <td>50</td>
      </tr>
      <tr>
        <td>Thomas Jackson</td>
        <td>94</td>
      </tr>
    </table>

  </body>
</html>
```

XPath

Used to address elements in a HTML document

Usage of relative paths

```
nds <- html_nodes( src,
xpath = "//a")
```

The //-operator indicates that all a-Tags are searched

```
html_text( nds )
  [1] "Click"
```

```
html_attr( nds, "href" )
  [1] "https://wikipedia.org/"
```

`html_attr` selects the text of an attribute

Navigate a HTML document

```
<!DOCTYPE html>
<html>
  <head>
    <title>Page Title</title>
    <style>
      table, td {border: 1px solid black;}
    </style>
  </head>

  <body>
    <h1>Table: Name and Age</h1>
    <a href='https://wikipedia.org/'>Click</a>
    <table>
      <tr>
        <td>Andrew Smith</td>
        <td>50</td>
      </tr>
      <tr>
        <td>Thomas Jackson</td>
        <td>94</td>
      </tr>
    </table>

  </body>
</html>
```

XPath

Used to address elements in a HTML document

Read a table

```
html_table( src )

[[1]]
                X1 X2
1   Andrew Smith 50
2 Thomas Jackson 94
```

`html_table` automatically selects all tables in the HTML document and represents them in a tabular format in R

Simple example with HTML

Wikipedia: List of the german Nobel prize winners

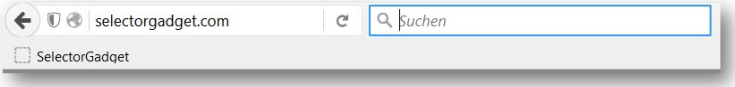
Name	Jahr	Kategorie	
Gustav Stresemann	1926	Friedensnobelpreis	Annäherung an Frankreich zur Sicherung
Ludwig Quidde	1927	Friedensnobelpreis	Organisation von Friedenskonferenzen
Carl von Ossietzky	1935	Friedensnobelpreis	Einsatz gegen den deutschen Militarismu
Albert Schweitzer*	1952	Friedensnobelpreis	Einsatz gegen die atomare Aufrüstung
Willy Brandt	1971	Friedensnobelpreis	Ostpolitik
Henry Kissinger*	1973	Friedensnobelpreis	Verhandlung einer Waffenruhe im Vietna
Theodor Mommsen	1902	Nobelpreis für Literatur	Römische Geschichte
[illegible]	[illegible]	[illegible]	„auf Grund des ernsten Suchens nach W

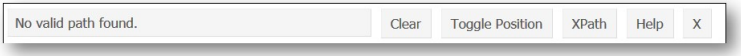
Read the HTML table

```
src <- read_html(
   "https://de.wikipedia.org/wiki/Liste_der_deutschen_Nobelpre
   istr%C3%A4ger" )
nobel <- html_table( src )
nobel[[1]][,1:3]
```

```
                                Name      Jahr
1 Stresemann! Gustav Stresemann 1926!1926 Fried
2          Quidde!Ludwig Quidde 1927!1927 Fried
3  Ossietzky!Carl von Ossietzky 1935!1935 Fried
4 Schweitzer!Albert Schweitzer* 1952!1952 Fried
5           Brandt!Willy Brandt 1971!1971 Fried
```

SelectorGadget

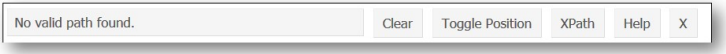
1. `http://selectorgadget.com/`
2. Or drag this link to your bookmark bar: SelectorGadget (updated August 7, 2013)
3. 

4. 

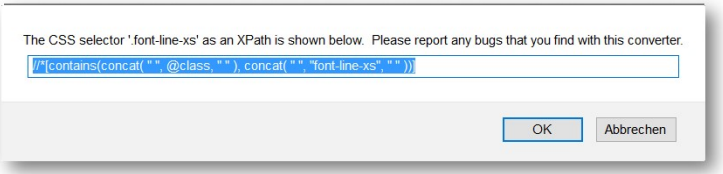
5. Click the element that should be read, everything highlighted in yellow is read out

SelectorGadget

6. Click XPath



7. Copy the path from the Pop-Up (`Strg+C`)



8. Use copied path in the `html_nodes` command as xpath-path

XML

While HTML is used to build Web sites, Web data are often stored in a separate data format

HTML

```
<!DOCTYPE html>
<html>
  <head> … </head>

 <body>
    <table>
      <tr>
        <td>Andrew Smith</td>
        <td>50</td>
      </tr>
      <tr>
        <td>Thomas Jackson</td>
        <td>94</td>
      </tr>
    </table>
  </body>
</html>
```

XML

```
<?xml version="1.0"
encoding="windows-1252"
standalone="no"?>

<persons>
  <person>
    <name>Smith</name>
    <first>Andrew</first>
    <age type="Years">50</age>
  </person>
  <person>
    <name>Jackson</name>
    <first>Thomas</first>
    <age type="Years">94</age>
   </person>
</persons>
```

XML

HTML

- Used to build Web sites
- Tags are used to structure the code
- Predefined tags (e.g.)
- HTML is often less structured than XML and thus more difficult to read

XML

- Describes the structure of Web data
- XML resembles HTML because it also uses tags
- Tags are not predefined and can be defined as required
- Usually XML has a simple structure and is thus easier to read

HTML and XML

```
<?xml version="1.0"
encoding="windows-1252"
standalone="no"?>

<persons>
  <person>
    <name>Smith</name>
    <first>Andrew</first>
    <age type="Years"> 50
    </age>
  </person>

  <person>
    <name>Jackson</name>
    <first>Thomas</first>
    <age type="Years"> 94
    </age>
   </person>
</persons>
```

How to access XML content

```
src <- read_html(
   "C:/Beispiel2.xml" )
nds_name <- html_nodes( src,
   xpath = "//name")
html_text( nds_name )
   [1] "Smith" "Jackson"
```

- Because XML is similar to HTML, the same commands can be used
- Because the structure is simpler, the content is easier to access
- XPath knowledge is necessary because SelectorGadget cannot be used

JSON

Another frequently used data format is JSON

XML

```
<?xml version="1.0"
encoding="windows-1252"
standalone="no"?>

<persons>
  <person>
    <name>Smith</name>
    <first>Andrew</first>
    <age type="Years">
       50</age>
  </person>

  <person>
    <name>Jackson</name>
    <first>Thomas</first>
    <age type="Years">
       94</age>
   </person>
</persons>
```

JSON

```
{
 "note": "UTF-8 Codierung",


  "persons": [
    {
       "name": "Smith",
       "first": "Andrew",
       "age": {"type": "Years",
               "value" : 50}
    },

    {
       "name": "Jackson",
       "first": "Thomas",
       "age": {"type": "Years",
               "value" : 94}
    }
  ]
}
```

JSON

```
{
  "note": "UTF-8 Codierung",


  "persons": [
    {
        "name": "Smith",
        "first": "Andrew",
        "age": {"type": "Years",
                "value" : 50}
    },

    {
        "name": "Jackson",
        "first": "Thomas",
        "age": {"type": "Years",
                "value" : 94}
    }
  ]
}
```

How to access JSON content

```
library(jsonlite)
src <-
    fromJSON("C:/Beispiel3.json")
str(src)
List of 2
 $ note : chr "UTF-8 Codierung"
 $ persons:'data.frame': 2 obs. of
   ..$ name : chr [1:2] "Smith" "Jackson"
src$persons$name
[1] "Smith" "Jackson"
```

- When JSON data are loaded, R converts them into a nested list
- With the "str"-command the structure of the loaded data can be displayed

JSON and XML

JSON

- Format for online data exchange
- Small file size for faster online transmission
- Valid JavaScript, a script language used to display interactive Web sites
- Xpath is not available to access elements, possibly more effort needed for programming

XML

- Language to describe arbitrary data structures
- Files are larger due to start- and end-tags
- Document type definitions (DTD) allow to define what makes an XML-file valid
- Xpath allows easy access to the desired elements

APIs

Application Programming Interfaces (APIs)

- RESTful APIs allow transferring data using web protocols
- Enables programmatic access to data

API Endpoint: root URL + data query

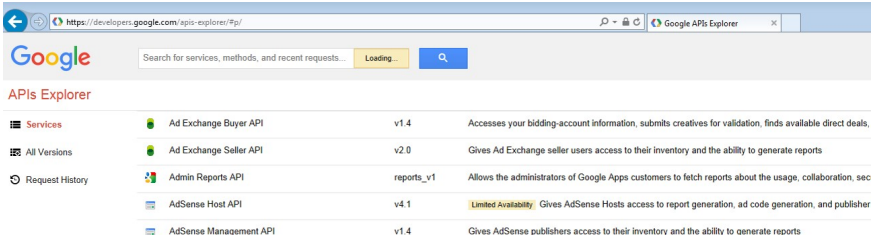
- A unique URL that requests a data piece
- Allows HTTP client to interact with data resources
- Response/ data are often in JSON or XML format

APIs

Many online services offer interfaces (APIs) to make selected data available.

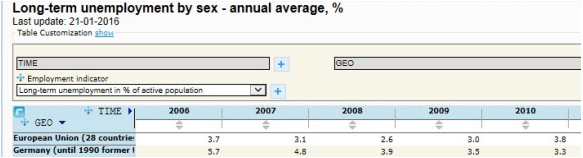
- Target group are mostly programmers and app developers
- Access to resources may be limited
- Consent of the respective user may be necessary (e.g., Facebook, LinkedIn)
- Fees may be required

Google example



APIs: Eurostat example

How to access the Eurostat API



1. Create an URL through the web interface to read the desired data

Result: `http://ec.europa.eu/eurostat/wdds/rest/data/v2.1/json/en/une_ltu_a?sex=F&sex=M&sex=T&geo=DE&geo=EU28&precision=1&sinceTimePeriod=2012&unit=PC_ACT&indic_em=LTU&age=Y20-64`

1. Open file with R

```
library(jsonlite)
data <- fromJSON("path_to_file/une_ltu_a.json", flatten = FALSE)
```

Regular Expressions

Scraped text is often formatted poorly.

```
<cl title="Kategorie:Arbeitsmarkt" ns="14"/>
<cl title="Kategorie:Forschungsinstitut in Nürnberg" ns="14"/>
<cl title="Kategorie:Gegründet 1967" ns="14"/>
<cl title="Kategorie:Ressortforschungseinrichtung" ns="14"/>
```

```
                                Name      Jahr
1 Stresemann! Gustav Stresemann 1926!1926 Fried
2          Quidde!Ludwig Quidde 1927!1927 Fried
3  Ossietzky!Carl von Ossietzky 1935!1935 Fried
4 Schweitzer!Albert Schweitzer* 1952!1952 Fried
5            Brandt!Willy Brandt 1971!1971 Fried
```

Solution: Search and replace using Regular Expressions:

```
txt <- c("Kategorie:Arbeitsmarkt",
    "Kategorie:Forschungsinstitut",
    "Kategorie:Gegründet 1967")

## Search for „Kategorie:“ and
## only keep what is
gsub("Kategorie:(.*)", "\\1", txt)
"Arbeitsmarkt"
"Forschungsinstitut"
"Gegründet 1967"
```

```
txt <- c("Quidde!Ludwig Quidde",
  "Ossietzky!Carl von Ossietzky")

## Only keep the text following !

gsub(".*?!(.*?)", "\\1", txt)
"Ludwig Quidde"
"Carl von Ossietzky"
```

Summary

- Lots of data available are available online that can be useful for analyses
- HTML, XML and JSON are particulary common web-formats, that can be used with little effort
- Some Web sites prohibit scraping, others explicitly allow it through APIs
- Text can be formatted using Regular Expressions
- Further challenges often exist for particular Web sites, e.g.
  - password protection, cookies
  - Sequential scraping of many sites 顺序scraping
- Web Scraping can be chaotic and other options might be preferable

Summary

Web sites are not designed for Web scraping

- It can be difficult to find the desired contents/tags automatically
- Providers can take technical measures to make Web scraping more difficult
- Web sites can collapse if they receive too many requests
- Some Web sites prohibit web scraping in their terms and conditions

robots.txt

- Lists permissions for webbots, crawlers for a given webpage
- `https://cran.r-project.org/web/packages/robotstxt/`

Stay polite!

robots.txt

Figure: Snippet of `https://en.wikipedia.org/robots.txt`

```
# robots.txt for http://www.wikipedia.org/ and friends
#
# Please note: There are a lot of pages on this site, and there are
# some misbehaved spiders out there that go _way_ too fast. If you're
# irresponsible, your access to the site may be blocked.
#

# Observed spamming large amounts of https://en.wikipedia.org/?curid=NNNNNN
# and ignoring 429 ratelimit responses, claims to respect robots:
# http://mj12bot.com/
User-agent: MJ12bot
Disallow: /

# advertising-related bots:
User-agent: Mediapartners-Google*
Disallow: /

# Wikipedia work bots:
User-agent: IsraBot
Disallow:

User-agent: Orthogaffe
Disallow:
```

Open Data

- https://github.com/ropensci/opendata
- https://www.opendatanetwork.com/
- https://www.data.gov/
- http://dataportals.org/
- https://toolbox.google.com/datasetsearch

Software Resources

- Tools and tutorials
  - https://selectorgadget.com/
  - https://www.regular-expressions.info/
  - http://www.txt2re.com/
  - https://regexr.com/
- Resources for R
  - https://cran.r-project.org/web/views/WebTechnologies.html
  - https://ropensci.org/packages/
  - https://www.datacamp.com/community/tutorials/r-web-scraping-rvest

References

Callegaro M. and Yang Y. (2018). The Role of Surveys in the Era of "Big Data". In: Vannette D., Krosnick J. (eds). The Palgrave Handbook of Survey Research. Palgrave Macmillan, Cham.

Japec, L., Kreuter, F., Berg, M., Biemer, P., Decker, P., Lampe, C., Lane, J., O'Neil, C., and Usher, A. (2015). Big data in survey research. Public Opinion Quarterly 79(4). 839–880.

Lazer, D. and Radford, J. (2017). Data ex Machina: Introduction to Big Data. Annual Review of Sociology 43(1). 19–39.

Qiu, L., Chan, S. H. M. and Chan, D. (2018). Big data in social and psychological science: theoretical and methodological issues. Journal of Computational Social Science 1(1). 59–66.

References

Munzert, S., Rubba, C. , Meißner, P., and Nyhuis, D. (2015). Automated data collection with R: A practical guide to web scraping and text mining. Chichester, West Sussex: John Wiley & Sons.