Web Data Collection with R

Introduction

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

Motivation

Why even bother with the Web?

old data

- now is presented/archived on the Web

new data

- now usually gets presented in the Web

Why even bother . . .

new types of data emerge(d)

- data that did not exist before, data that was not accessible before, data that was not combinable before
- such data is now available because so much is happening in the web
  - **search engines** (What does the Web offer? What are people looking for? What are hot topics?)
  - **Wikipedia** (What is . . . ? What is it related to? Is it a hot topic?)
  - **Twitter**, **Facebook**, **LinkedIn**, **Xing**, . . . (Who is connected to whom? Who listens to whom? Who talks to whom? What are hot topics?)
  - **Newspapers** (What are they talking about? How do they report? How do readers think about? . . .)
  - **Homepages** (What information do people/organisations present? To whom they are connected? . . .)

Why should we use R?

Reproducibility

- using programming languages and scripts makes your data gathering
  - explicit
  - sharable
  - reproducable
  - amendable

Efficiency

- using programming languages and scripts makes your data gathering more efficient
  - whenever you have to repeat yourself

Why should we use R?

All your research in one hand

- using R as your programming language allows for using just one environment for . . .
  - data gathering
  - data cleansing
  - data management
  - data analytics
  - data visualization
  - data reporting
- R is THE social science (and beyond) data tool, so
  - a lot of people might help with your code
  - a lot of people might understand your code
  - a lot of people might use your code
  - you have a lot of packages, active community, bug-fixes . . .

Who am I?

Who am I?

Peter Meißner

- ~~poilitical scientist~~
- ~~researcher at University of Konstanz (IDEP)~~
- freelance computational social scientist
-

Automated Data Collection With R (Munzert et al.)
- wikipediatrend
- diffr
- robotstxt
- hellno
- Uni-Konstanz-Mensa-Twitter-Bot

Who am I?

Who are you?

- What is your affiliation?
- Why did you join the course?

Building stones of the Web

Documents and Data

HTML

The single most important (non-data) format for us. HTML files are plain text and interpreted by browsers.

XML and JSON

XML and JSON are the two most important data formats in the Web. Both formats are plain text. XML actually is a whole family of formats – HTML can be sought as XML, Google Earth documents, Word, Excel, ...

CSS

HTML's helpful companion that defines how things defined by HTML should look like.

Selection and extraction

Regular Expressions

A formal scheme to express text patterns for text detection, extraction and manipulation.

XPath

A language for querying HTML and XML documents by selecting nodes and node sets and extracting their content or attributes.

CSS-selectors

A formal scheme for selecting parts (nodes, attributes, content) of HTML documents.

Communication

URI (URL)

The web adress identifying where to find and how to access a resource in the Web, e.g.:
`http:`
`//`www.ecpr.eu/Events/PanelDetails.aspx?PanelID=3681&EventID=97

HTTP

A (the most important one) standard for requesting and delivering content in the Web – others are POP, IMAP, FTP, ...

Cookies

HTTP's helpful companion that makes HTTP remembering information.

Browser and Online Tools

Developer tools

- tools to be found in all major browsers that are thought to help Web developers (e.g. Cntr-Shift-I in Chrome)
- they e.g. provide information on the
  - structure of the page (HTML/XML nodes, attributes, ...)
  - network traffic (HTTP requests and responses, cookies)
  - further resources used
  - a JS console

Selector Gadget

- a nice little tool that helps with generating CSS-selectors and XPath expressions to extract information from HTML

Scripting languages

Perl, PHP, Python, Ruby, . . .

- scripting languages for Web servers that allow for Web applications, Web shops, . . . we will never see them directly but might have to cope with their output

JavaScript (JS)

- scripting language executed within your browser

The Web in R

R-Packages: Web connections and data retrieval

RCurl

Uses C's libcurl library to make R speak HTTP (and HTTPS, FTP, FTPS). RCurl lays out the bases for webscraping with R. No RCurl, (nearly) no webscraping. Thank Duncan Temple Lang for this.

httr / curl

A package building on curl building on C's libcurl library and aiming on making things more convenient – Hadley Wickham and Jeroen Ooms did it.

R-Packages: Data Extraction (1)

stringr / stringi

Provide consistent and convenient string (text, a sequence of characters) handling (detection, extraction, replacement, . . .) with Regular Expressions. Marek Gagolewski and Hadley Wickham did it.

XML

A package to handle XML data based on C's lbxml library. Most importantly we can use it to query XML with XPath statements. Duncan Temple Lang's deed.

xml2

A package to handle XML data based on the C's libxml2 library Hadley Wickham and Jeroen Ooms again.

R-Packages: Data Extraction (2)

jsonlite

A package for reading and writing JSON data. Jeroen Ooms did it.

rvest

~~Very young~~ Well established package by Hadley Wickham building on curl, xml2, selectr and httr – it makes 85% of scraping the web with R a delicious piece of cake. It provides a neat workflow for most scraping task and accepts XPath as well as CSS-selectors for data queries.

R-Packages: API usage

twitteR

Package for using the Twitter API from within R.

wikipediatrend

A package to connect to stats.grok.se and import data on Wikipedia page access statistics

. . . and many many more

. . .

All the Web in R

CRAN Task View: Web Technologies and Services

- Collects and describes packages that have to do with Web technologies (extraction, creation, ...)
- http://cran.r-project.org/web/views/WebTechnologies.html

Rules of procedure

Our best friends – tools and procedures used

Make sure you have installed the packages below

```r
# packages from CRAN
p_needed     <- c(
  "RCurl", "XML", "xml2", "httpuv", "stringr",
  "jsonlite", "httr", "rvest", "devtools",
  "ggmap", "wikipediatrend", "d3Network",
  "RSelenium", "sp"
  )

packages     <- rownames(installed.packages())
p_to_install <- p_needed[ !(p_needed %in% packages ) ]
if ( length(p_to_install) > 0 ) {
  install.packages(
    p_to_install,
    repos="https://cran.rstudio.com/"
  )
}
```

Our best friends – tools and procedures used

Make sure you have installed the packages below

```
# packages from GitHub
p_to_install <- !("twitteR" %in% packages )
if ( p_to_install > 0 ){
  devtools::install_github("geoffjentry/twitteR")
}
```

Our best friends – tools and procedures used

Use Chrome (and or Mozilla) as browser for now

- use it because I use it and therefore things on your screen look like the things on my screen
- other browsers (Safari, most likely also Opera, perhaps also IE/edge) have the same functions but might differ in their particular implementation (google or bing for "developer tools in . . . ")

Our best friends – tools and procedures used

Use RStudio as R-frontend

- use it because I use it . . .
- use it because it is powerful, makes your live so much easier, has more colors than plain RGui, . . .

Our best friends – tools and procedures used

Follow the slides - .Rmd

- by opening the respective .Rmd file in RStudio
    - things like the one below are R-code snippets you might want to passe to the RStudio console (**Cntr-ENTER** on Windows) to replicate the examples:

```
` ` `{r, ...}
      dings <- 1+1
      dings
` ` `
```

Our best friends – tools and procedures used

Follow the slides - .Rmd

- the rest is just various types of text that you might use to make notes
    - see `http://rmarkdown.rstudio.com/authoring_basics.html` for an intro to markdown
    - later on, use the **Knit ...** button to knit your own HTML, PDF, ... presentation / output

Follow the slides - .pdf

- feel free but discouraged to follow the PDF versions of the slides to follow along

What to expect within the next sessions?

How do we proceed?

How do we proceed?

8. Xpath
9. Xpath Case Study

How do we proceed?

10. JSON
11. APIs
12. Simple APIs Case Study
13. Oauth APIs Case Study

How do we proceed?

14. HTML Forms
15. HTML Forms GET Case Study
16. HTML Forms POST 1 Case Study
17. HTML Forms POST 2 Case Study
18. JavaScript
19. Browserautomation

How do we proceed?

19. XHR
20. XHR Case Study
21. Managing Projects