Quantitative Content Analysis: Lecture 11

Matthias Haber

26 April 2017

Today's outline

- Text data sources and scripts
- Wordfish

Online text data sources

- **web pages** (e.g. `http://example.com`)
- **web formats** (XML, HTML, JSON, . . .)
- **web frameworks** (HTTP, URL, APIs, . . .)
- **social media** (Twitter, Facebook, LinkedIn, Snapchat, Tumbler, . . .)
- **data in the web** (speeches, laws, policy reports, news, . . .)
- **web data** (page views, page ranks, IP-addresses, . . .)

The Problems

phase	problems	examples
download	protocols	HTTP, HTTPS, POST, GET, ...
	procedures	cookies, authentication, forms, ...
extraction	parsing	translating HTML (XML, JSON, ...) into R
	extraction	getting the relevant parts
	cleansing	cleaning up, restructure, combine

Before scraping, do some googling!

- If the resource is well-known, someone else has probably built a tool which solves the problem for you.
- ropensci has a ton of R packages providing easy-to-use interfaces to open data.
- The Web Technologies and Services CRAN Task View is a great overview of various tools for working with data that lives on the web in R.

Example

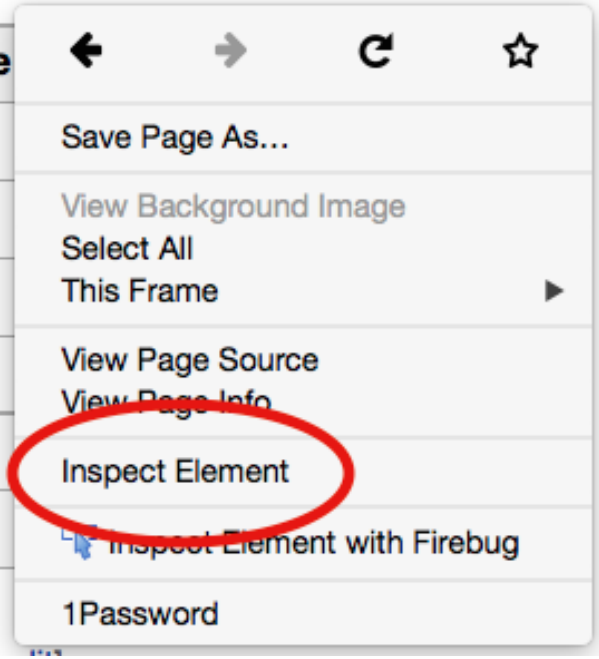
Inspecting elements

Simple table [edit]

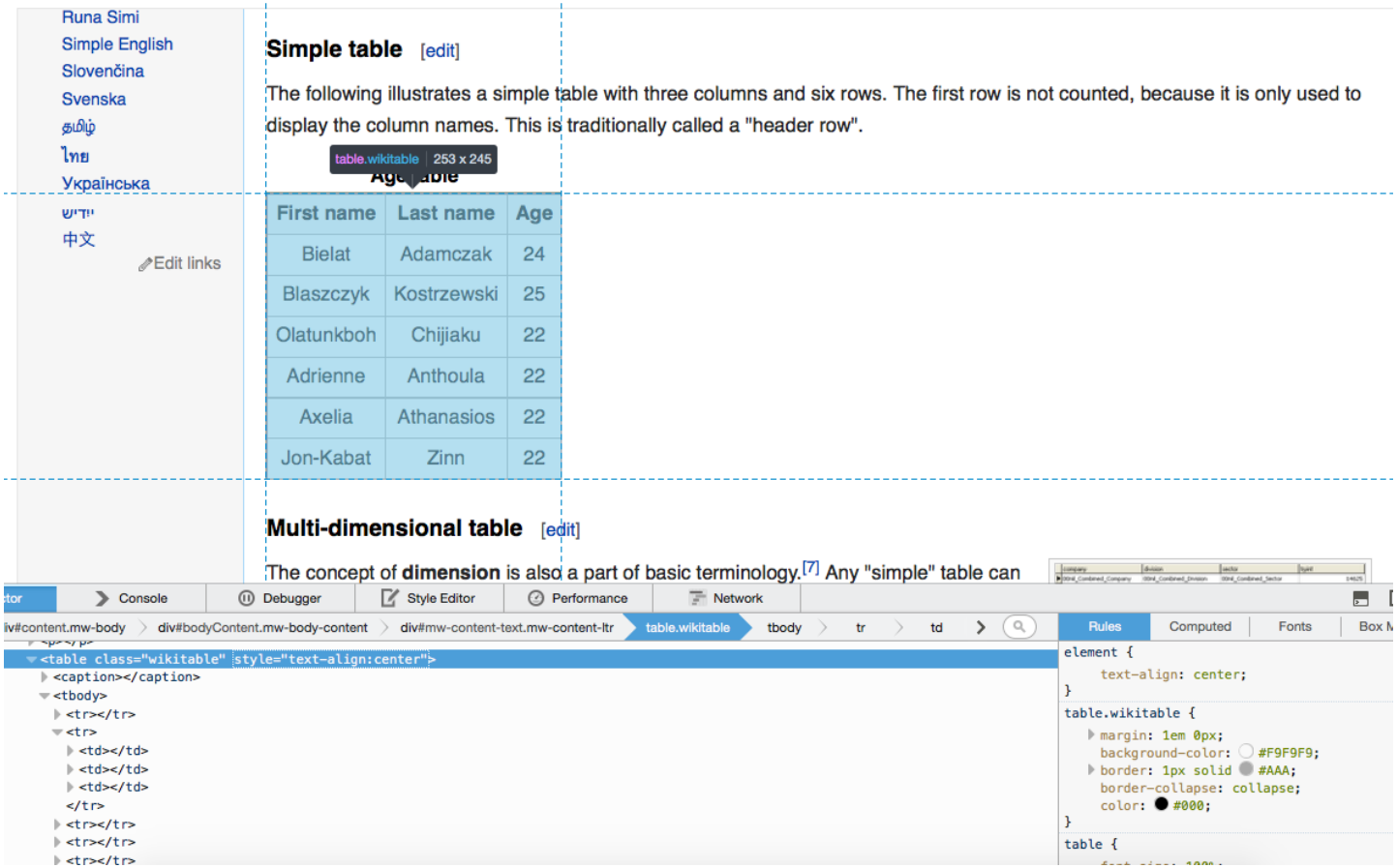
The following illustrates a simple table with three columns and six rows. display the column names. This is traditionally called a "header row".

Age table

First name	Last name	Age
Bielat	Adamczak	24
Blaszczyk	Kostrzewski	25
Olatunkboh	Chijiaku	22
Adrienne	Anthoula	22
Axelia	Athanasios	22
Jon-Kabat	Zinn	22



Hover to find desired elements



Wikitable

```
library(rvest)
src <- html("http://en.wikipedia.org/wiki/Table_(information)")
node <- html_node(src, css = ".wikitable")
```

- ".wikitable" is a CSS selector which says: "grab nodes (aka elements) with a class of wikitable".
- `html_table()` converts a single `<table>` node to a data frame.

```
html_table(node)
##   First name   Last name Age
## 1       Tinu    Elejogun  14
## 2  Blaszczyk Kostrzewski  25
## 3       Lily   McGarrett  16
## 4 Olatunkboh    Chijiaku  22
## 5   Adrienne    Anthoula  22
## 6     Axelia  Athanasios  22
## 7  Jon-Kabat        Zinn  22
```

Pipeable!

```
html("http://en.wikipedia.org/wiki/Table_(information)") %>%
  html_node(".wikitable") %>% html_table()
##   First name   Last name Age
## 1       Tinu    Elejogun  14
## 2  Blaszczyk Kostrzewski  25
## 3       Lily   McGarrett  16
## 4 Olatunkboh    Chijiaku  22
## 5   Adrienne    Anthoula  22
## 6     Axelia  Athanasios  22
## 7  Jon-Kabat        Zinn  22
```

Rvest

rvest is a nice R package for web-scraping by (you guessed it) Hadley Wickham.

```
library(dplyr)
library(rvest)
library(magrittr)
# First, grab the page source
html("http://en.wikipedia.org/wiki/Table_(information)") %>%
  # then extract the first node with class of wikitable
  html_node(".wikitable") %>%
  # then convert the HTML table into a data frame
  html_table()
##   First name   Last name Age
## 1       Tinu    Elejogun  14
## 2  Blaszczyk Kostrzewski  25
## 3       Lily   McGarrett  16
## 4 Olatunkboh    Chijiaku  22
## 5   Adrienne    Anthoula  22
## 6     Axelia  Athanasios  22
## 7  Jon-Kabat        Zinn  22
```

HTML / XML with rvest

```
rpack_html <-
"http://cran.r-project.org/web/packages" %>%
html()
rpack_html %>% class()
## [1] "xml_document" "xml_node"
```

HTML / XML with rvest (II)

```
rpack_html %>% xml_structure(indent = 2)
## <html [xmlns]>
##   <head>
##     <title>
##       {text}
##     <link [rel, type, href]>
##     <meta [http-equiv, content]>
##   <body>
##     {text}
##     <h1>
##       {text}
##     {text}
##     <h3 [id]>
##       {text}
##     {text}
##     <p>
##       {text}
##     {text}
##     <p>
##       {text}
```

HTML / XML with rvest (III)

```
rpack_html %>% html_text() %>% cat()
## CRAN - Contributed Packages
## Contributed Packages
##
## Available Packages
##  Currently, the CRAN package repository features 10472 available package
##    Table of available packages, sorted by date of publication
##    Table of available packages, sorted by name
## Installation of Packages
##
## Please type
##   help("INSTALL")
## or
##   help("install.packages")
## in R for information on how to install packages from this
## repository.  The manual
##
## R Installation and Administration
## (also contained in the R base sources)
## explains the process in detail.
```

Technologies and Packages

- Regular Expressions / String Handling
  - **stringr**, stringi
- HTML / XML / XPAth / CSS Selectors
  - **rvest**, xml2, XML
- JSON
  - **jsonlite**, RJSONIO, rjson
- HTTP / HTTPS
  - **httr**, curl, Rcurl
- Javascript / Browser Automation
  - **RSelenium**
- URL
  - **urltools**

Readings

- Basics on HTML, XML, JSON, HTTP, RegEx, XPath
  - Munzert et al. (2014): Automated Data Collection with R. Wiley. `http://www.r-datacollection.com/`
- curl / libcurl
  - `http://curl.haxx.se/libcurl/c/curl_easy_setopt.html`
- CSS Selectors
  - W3Schools: `http://www.w3schools.com/cssref/css_selectors.asp`
- Packages: httr, rvest, jsonlite, xml2, curl
  - Readmes, demos and vignettes accompanying the packages
- Packages: RCurl and XML
  - Munzert et al. (2014): Automated Data Collection with R. Wiley. Nolan and Temple-Lang (2013): XML and Web Technologies for Data Science with R. Springer

Twitter

Twitter has two types of APIs

- REST APIs –> reading/writing/following/etc.
- Streaming APIs –> low latency access to 1% of global stream - public, user and site streams
- authentication via OAuth
- documentation at `https://dev.twitter.com/overview/documentation`

Accessing the twitter APIs

To access the REST and streaming APIs, you will need to create a twitter application, and generate authentication credentials associated with this application. To do this you will first need to have a twitter account. You will also need to install at least the following R packages: `twitteR`,

```
install.packages(c('twitteR', 'streamR', 'RCurl', 'ROAuth', 'httr'))
```

Create a twitter application

To register a twitter application and get your consumer keys:

1. Go to `https://apps.twitter.com` in a web browser.
2. Click on 'create new app'.
3. Give your app a unique name, a description, any relevant web address, and agree to the terms and conditions. Set the callback URL to `http://127.0.0.1:1410`.
4. Go to the keys and access section of the app page, and copy your consumer key and consumer secret to the code below.
5. (optional): For actions requiring write permissions, generate an access token and access secret.

Use twitter in R

```
library(twitteR)
library(streamR)
library(ROAuth)

consumerKey <- 'your key here'
consumerSecret <- 'your secret here'

# Try this first, to use twitteR
setup_twitter_oauth(consumerKey, consumerSecret)
results <- searchTwitter('#Trump')
df <- as.data.frame(t(sapply(results, as.data.frame)))
```

Then try these instructions, to use streamR:
https://github.com/pablobarbera/streamR#installation-and-authentication

Media data from LexisNexis

Nexis includes a large selection of international newspapers updated daily. Among them The Daily Telegraph, International New York Times, The Observer, Le Figaro, Le Monde, Corriere della Sera, taz, die tageszeitung, Die ZEIT.

You can access Nexis through the Hertie Library:
`https://www.hertie-school.org/en/library/resources/#c6741`
(scroll down till you find the Nexis link).

Parse data from Nexis into R

```
library(tm)
library(tm.plugin.lexisnexis)
library(quanteda)

ln <- LexisNexisSource("lexisnexis.HTML")
tmCorpus <- VCorpus(ln)
myCorpus <- corpus(tmCorpus)
mydfm <- dfm(myCorpus)
```

Next Session

- Topic modelling