Data Management With R: Webscraping with rvest

Matthias Haber

06 November 2017

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

Not logged in Talk Contributions Create account Log in

Article Talk Read Edit View history Search Wikipedia

WIKIPEDIA
The Free Encyclopedia

Main page
Contents
Featured content
Current events
Random article
Donate to Wikipedia
Wikipedia store

Interaction

Help
About Wikipedia
Community portal
Recent changes
Contact page

Tools

Table (information)

From Wikipedia, the free encyclopedia

"Tabular" redirects here. For the typewriter key, see tab key.
For sortable tables in Wikipedia, see Help:Sorting

This article may **require cleanup** to meet Wikipedia's quality standards. No cleanup reason has been specified. Please help improve this article if you can. *(October 2009)* *(Learn how and when to remove this template message)*

A **table** is an arrangement of data in rows and columns, or possibly in a more complex structure. Tables are widely used in communication, research, and data analysis. Tables appear in print media, handwritten notes, computer software, architectural ornamentation, traffic signs, and many other places. The precise conventions and terminology for describing tables vary depending on the context. Further, tables differ significantly in variety, structure, flexibility, notation, representation and use.[1][2][3][4][5] In books and technical articles, tables are typically presented apart from the main text in numbered and captioned floating blocks.

Purchased Equipments (June, 2006)			
Item Num#	Item Picture	Item Description Shipping Handling, Installation, etc	Price Expense
1.		IBM Clone Computer. Shipping Handling, Installation, etc	$ 400.00 $ 20.00
2.		1GB RAM Module for Computer. Shipping Handling, Installation, etc	$ 50.00 $ 14.00
Purchased Equipments (June, 2006)			

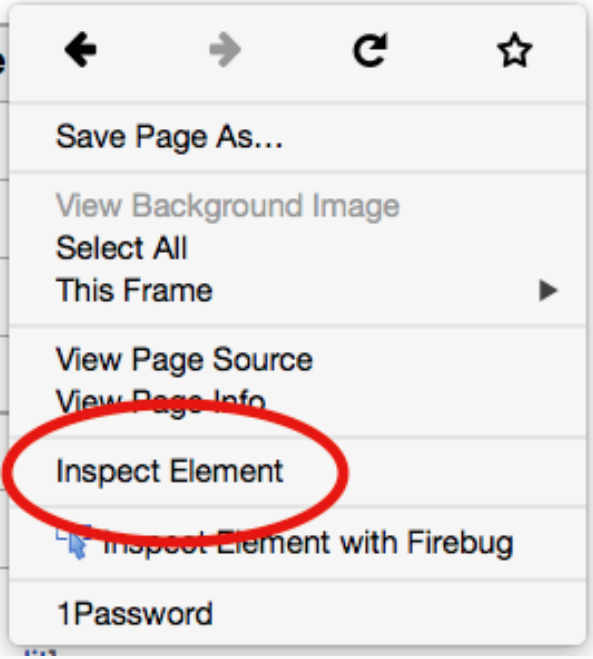
Inspecting elements

Simple table [edit]

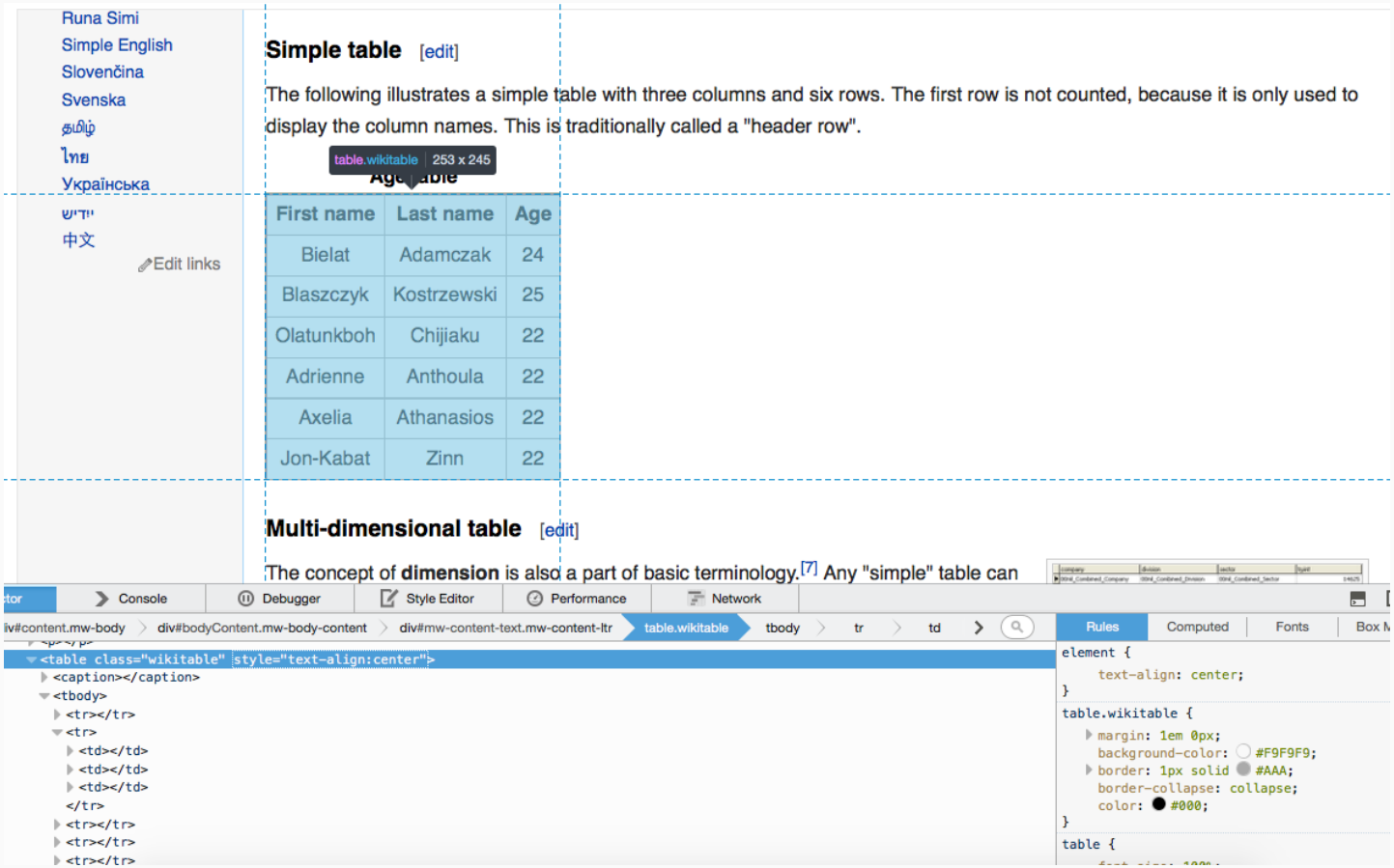
The following illustrates a simple table with three columns and six rows. display the column names. This is traditionally called a "header row".

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



Rvest

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

- see also: `https://github.com/hadley/rvest`
- convenient package to scrape information from web pages
- builds on other packages, such as xml2 and httr
- provides very intuitive functions to import and process webpages

Basic workflow of scraping with rvest

```
library(rvest)
```

```
## Loading required package: xml2
```

```
library(magrittr)
```

```
## Warning: package 'magrittr' was built under R version 3.
```

```
# 1. specify URL
"http://en.wikipedia.org/wiki/Table_(information)" %>%

# 2. download static HTML behind the URL and parse it into
read_html() %>%

# 3. extract specific nodes with CSS (or XPath)
```

Task 1

Navigate to *this page* and try the following:

Easy: Grab the table at the bottom of the page (hint: instead of grabbing a node by class with `html_node(".class")`, you can grab by id with `html_node("#id")`)

Medium: Grab the actual mean, max, and min temperature.

Hard: Grab the weather history graph and write the figure to disk (`download.file()` may be helpful here).

Task 1 (solution)

```
library(rvest)
src <- read_html(paste0("http://www.wunderground.com/histor
"airport/KVAY/2015/2/17/DailyHistory.html?",
"req_city=Cherry+Hill&req_state=NJ&",
"req_statename=New+Jersey&reqdb.zip=08002",
"&reqdb.magic=1&reqdb.wmo=99999&MR=1"))

# easy solution
tab1 <- src %>% html_node("#obsTable") %>% html_table()
# medium solution
tab2 <- src %>% html_node("#historyTable") %>% html_table()
  .[2:4, "Actual"]
# hard solution
link <- src %>% html_node("#history-graph-image img") %>% h
download.file(paste0("http://www.wunderground.com", link),
```

Selectorgadget + rvest to the rescue!

- Selectorgadget is a Chrome browser extension for quickly extracting desired parts of an HTML page.
- With some user feedback, the gadget find out the CSS selector that returns the highlighted page elements.
- Let's try it out on this page

Extracting links to download reports

```
domain <- "http://www.sec.gov"
susp <- paste0(domain, "/litigation/suspensions.shtml")
hrefs <- read_html(susp) %>% html_nodes("tr+ tr a:nth-chil
  html_attr(name = "href")

# download all the pdfs!
hrefs <- hrefs[!is.na(hrefs)]
pdfs <- paste0(domain, hrefs)
mapply(download.file, pdfs, basename(pdfs))
```

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'
```

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)
```

```
## Warning: package 'tm' was built under R version 3.3.3
```

```
## Loading required package: NLP
```

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

```
## Warning: package 'tm.plugin.lexisnexis' was built under
```

```
library(quanteda)
```

```
## quanteda version 0.9.9.48
```

```
## Using 7 of 8 cores for parallel computing
```

Homework Exercises

Homework Exercises

For this week's homework exersises go to Moodle and answer the Quiz posted in the Relational Data section.

Deadline: Sunday, November 12 before midnight.

That's it for today. Questions?