# Welcome to the Social Science Data Lab!

# The Social Science Data Lab

## The idea

- informal forum to exchange ideas, knowledge and skills
- focus on data and methods
- interactivity
- interdisciplinarity

# The Social Science Data Lab

## The idea

- informal forum to exchange ideas, knowledge and skills
- focus on data and methods
- interactivity
- interdisciplinarity
- also: *there ain't no such thing as a free lunch*—except for today!

# The Social Science Data Lab

Thanks to the Lorenz-von-Stein-Gesellschaft and Shirin Tumenbaeva!

*"There's no such thing as a free lunch."*

# The Social Science Data Lab

## How you can contribute to the Lab

- present your work in progress with a particular emphasis on data and methods issues
- introduce a new data base you created
- offer a tutorial on tools of data collection and analysis
- visit the Lab and interact with others

# Upcoming events

- **June 29, 2016**: The 'Queripedia' project: identifying entities for political events (Laura Dietz and Federico Nanni)
- **July 13, 2016**: Comparable and complete? On UNHCR Refugee Data (Moritz Marbach)

Schedule for next semester in development.
Any suggestions and/or contributions welcome!

# Three easy-to-learn tools to scrape data from the Web with R

Simon Munzert
r-datacollection.com
@RDataCollection
#rstats #webscraping

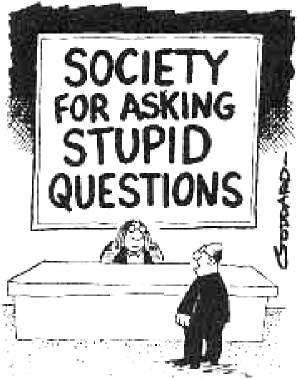
June 15, 2016
MZES | Social Science Data Lab

Materials available at
https://github.com/simonmunzert/rscrapingSSDL2016

# Outline

Welcome!
Why Web Scraping?
Regular Expressions
XPath
APIs
AJAX and Selenium
Good Practice

# First: ask questions! No matter what. . .



“Excuse me, but is this The Society for Asking Stupid Questions?”

# Why Web Scraping?

# Why Web Scraping?

## Web scraping

*A.k.a. screen scraping, crawling, web harvesting;* computer-aided collection of predominantly unstructured data (e.g., from HTML code)

The World Wide Web is full of various kinds of new data, e.g.:

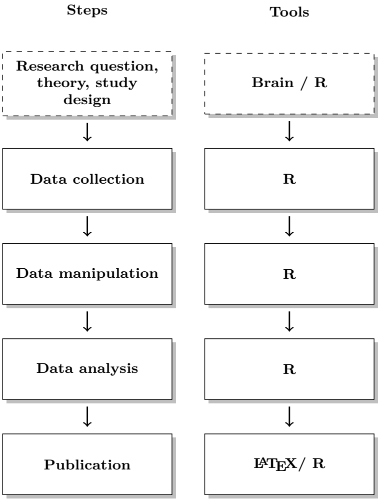
- open government data
- search engine data
- services that track social behavior

Practical arguments

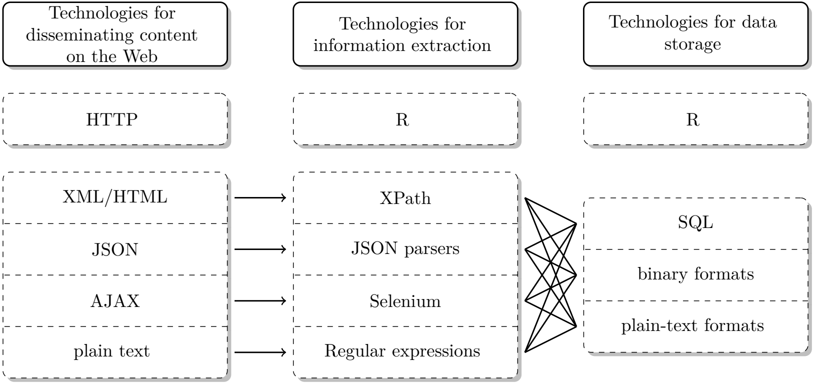
- financial resources are sparse
- . . . and so is our time
- reproducibility

# Why R?

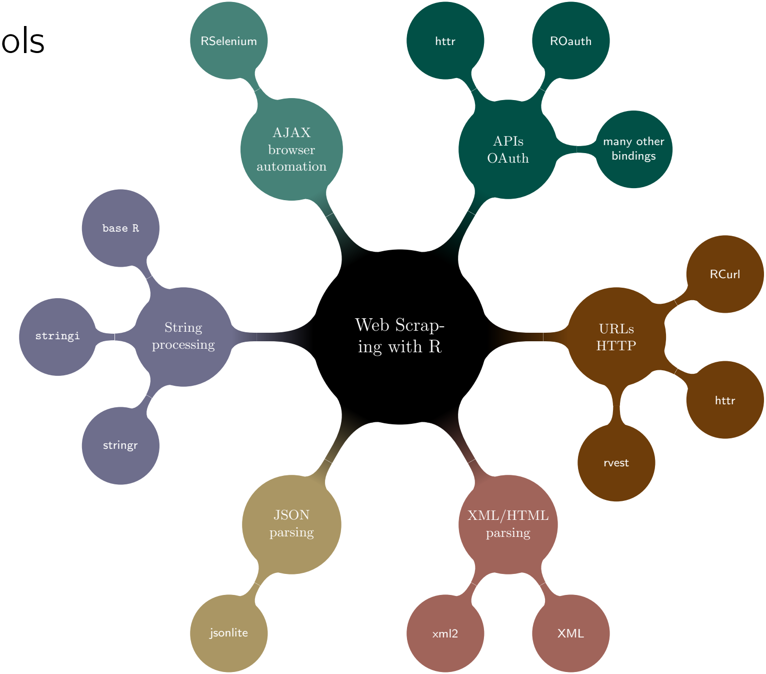
- free
- open source
- large community
- powerful tools for statistical analysis
- powerful tools for visualization
- flexible in processing all kinds of data/languages
- useful in every step of the workflow


Steps
Tools
Research question, theory, study design
Brain / R
Data collection
R
Data manipulation
R
Data analysis
R
Publication
LaTeX / R

# Technologies of the World Wide Web


Technologies for disseminating content on the Web
Technologies for information extraction
Technologies for data storage
HTTP
R
R
XML/HTML
JSON
AJAX
plain text
XPath
JSON parsers
Selenium
Regular expressions
SQL
binary formats
plain-text formats

# R tools

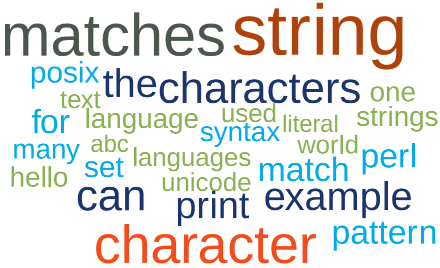

Web Scraping with R
AJAX browser automation
RSelenium
APIs OAuth
httr
ROauth
many other bindings
URLs HTTP
RCurl
httr
rvest
XML/HTML parsing
xml2
XML
JSON parsing
jsonlite
String processing
base R
stringi
stringr

# Technical Setup

1. make sure that the newest version of R (currently 3.3.0; available here) is installed on your computer
2. install the newest stable version of *RStudio* (available here)
3. install the following packages:

   ```
   pkgs <- c('RCurl', 'XML', 'stringr', 'jsonlite',
   'httr', 'rvest', 'pdftools', 'devtools', 'RSelenium',
   'plyr', 'dpylr', 'wikipediatrend', 'twitteR',
   'streamR', 'd3Network')
   ```
4. install the *Chrome* (from here) and *Firefox* browsers (from here)
5. install *Java* (from here)

# Regular Expressions



# What are regular expressions?

## Definition

- a.k.a. *Regex* or *RegExp*
- origins in formal language theory
- sequences of characters that describe patterns in text
- implemented in many programming languages, including R

## Why are regular expressions useful for web scraping?

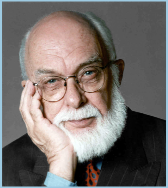
- information on the Web can often be described by patterns (email addresses, numbers, cells in HTML tables, ...)
- if the data of interest follow specific patterns, we can match and extract them—regardless of page layout and HTML overhead
- whenever the information of interest is (stored in) text, regular expressions are usful for extraction and tidying purposes

# Example: mapping locations of AJPS reviewers

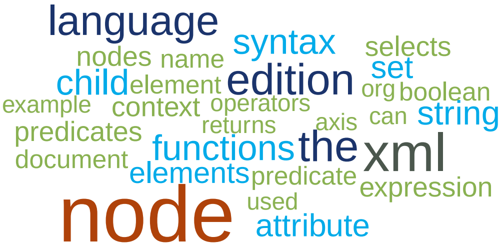
**Goal:** geolocate AJPS reviewers

**Tasks:**

- download PDF files from http://ajps.org/list-of-reviewers/
- import them into R (as plain text)
- extract information via regular expressions
- geocoding



# XPath


language
syntax
selects
nodes
name
set
child
element
edition
org
boolean
example
context
operators
string
can
predicates
returns
axis
functions
the
xml
document
elements
predicate
expression
node
used
attribute

# What’s XPath?

## Definition

- XML Path language, a W3C standard
- Query language for XML-based documents (i.e., for HTML as well)
- access node sets and extract content

## Why XPath for web scraping?

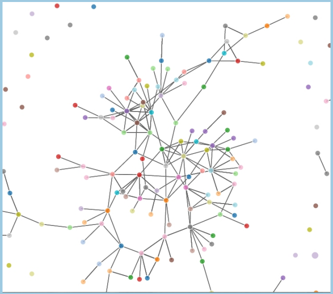
- Source code of webpages structures both layout and content
- not only content, but context matters
- enables us to extract content based on its location in the document, but (usually) regardless of its shape

# Example: a Wikipedia-based network of political scientists

**Goal:** build a network of political scientists

**Tasks:**

- gather list of political scientists
- fetch Wikipedia entries
- identify links
- construct connectivity matrix
- visualize network



# APIs

response
resource
data
request
web
tcp
the
client
message
rfc
html
example
connection
can
get
status
header
user
information
servers
line
content
also
method
may
methods
server
protocol
requests

# What are APIs?

## Definition

- **A**pplication **P**rogramming **I**nterface
- many web services provide APIs to access their data and services (Twitter, Google, Facebook, Wikipedia, . . . )
- common data formats: XML, JSON

## APIs in the context of web scraping

- instant access to clean data
- frees us from building manual scrapers
- forces us to understand the API architecture

# Data gathering with APIs

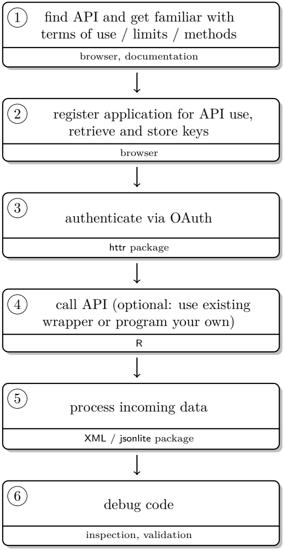
## Advantages

- pure data collection without ‘layout waste’
- standardized data access
- de facto automatic agreement of data owner
- robustness of calls

## Disadvantages

- requires knowledge of API architecture
- dependent upon API suppliers
- not always for free

# Data gathering with APIs



1 find API and get familiar with terms of use / limits / methods

browser, documentation

2 register application for API use, retrieve and store keys

browser

3 authenticate via OAuth

httr package

4 call API (optional: use existing wrapper or program your own)

R

5 process incoming data

XML / jsonlite package

6 debug code

inspection, validation

# Finding APIs on the Web

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

rOpenSci: Collection of R-API interfaces:
http://ropensci.org/

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

# Social media mining with R

## Why social media mining?

- network data
- communication data
- preference data

## Existing R bindings

- twitteR
- streamR
- Rfacebook
- Rlinkedin
- SocialMediaMineR
- tumblR
- ...

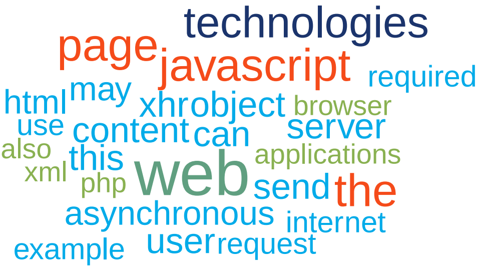
# Example: exploring Twitter's services

**Goal:** tap Twitter's REST and Streaming APIs

**Tasks:**

- register app
- manage authorization process
- get to know the twitteR and the streamR packages

# AJAX and Selenium



# What's AJAX?

- HTML/HTTP are used for static display of content
- 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
- **A**synchronous **J**avaScript **a**nd **X**ML' is a set of technologies that serve these purposes
- massively used in modern webpage design and architecture
- makes classical screen scraping more difficult

Example: https://twitter.com/regsprecher

# Selenium

## 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, XML, `download.file()`, etc.)

## 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?

- http://www.seleniumhq.org
- free software environment for automated web application testing
- several modules for different tasks; most important for our purposes: Selenium WebDriver
- Selenium WebDiver starts a server instance (as proxy) and passes commands (posed in R in our case) to the browser
- automated browsing via scripts

# Selenium and R

## Software requirements

- Java, https://www.java.com/de/download/
- Selenium server, http://selenium-release.storage.googleapis.com/2.45/selenium-server-standalone-2.45.0.jar or via RSelenium and `checkForServer()`
- Firefox browser, https://www.mozilla.org/en-US/firefox/new/
- RSelenium package

# Example: tapping the IEA Global Renewable Energy database

**Goal:** fetch policy data from IEA database

**Tasks:**

- get Selenium running
- inspect HTML form on http://www.iea.org/policiesandmeasures/renewableenergy/
- access page with RSelenium
- download data output
- import data into R
- tidy data

# Good Practice

messaging
communication
usenet
phone
many
can
using
this
one
also
the
mail
social
used
online
public
people
new
phones
snapchat
mobile
rules
users
messages
technology

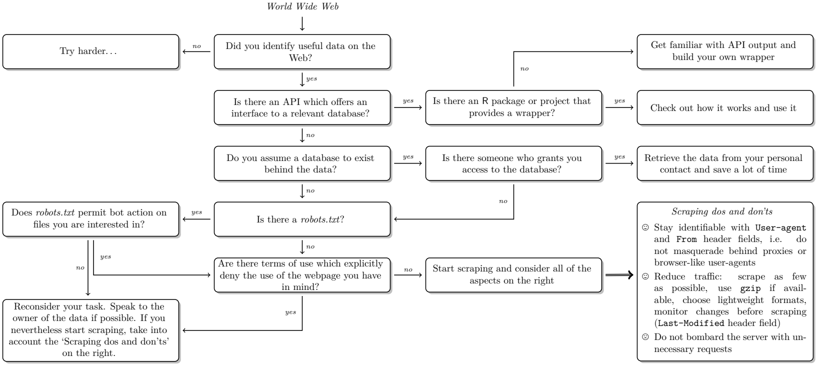
# Is web scraping legal?

- no unambiguous yes or no in any country according to current jurisdiction
- so far, court cases (especially in the US) often (but not always) dealt with commercial interest and often (but not always) huge masses of data
  - eBay vs. Bidder's Edge
  - AP vs. Meltwater
  - Facebook vs. Pete Warden
  - United States vs. Aaron Swartz

# A (not very useful) recommentation 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 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
5. consult current jurisdiction, e.g. on http://blawgsearch.justia.com or from a laywer specialized on internet law

# Scraping etiquette



*World Wide Web*

Did you identify useful data on the Web?

Try harder...

Is there an API which offers an interface to a relevant database?

Is there an R package or project that provides a wrapper?

Get familiar with API output and build your own wrapper

Check out how it works and use it

Do you assume a database to exist behind the data?

Is there someone who grants you access to the database?

Retrieve the data from your personal contact and save a lot of time

Is there a *robots.txt*?

Does *robots.txt* permit bot action on files you are interested in?

Are there terms of use which explicitly deny the use of the webpage you have in mind?

Start scraping and consider all of the aspects on the right

Reconsider your task. Speak to the owner of the data if possible. If you nevertheless start scraping, take into account the 'Scraping dos and don'ts' on the right.

*Scraping dos and don'ts*

- ☺ Stay identifiable with `User-agent` and `From` header fields, i.e. do not masquerade behind proxies or browser-like user-agents
- ☺ Reduce traffic: scrape as few as possible, use `gzip` if available, choose lightweight formats, monitor changes before scraping (`Last-Modified` header field)
- ☹ Do not bombard the server with unnecessary requests

# Thank you for your attention!

Simon Munzert
simon.munzert@mzes.uni-mannheim.de
@simonsaysnothin
@RDataCollection