Introduction to Data Science

Session 5: Web scraping and APIs

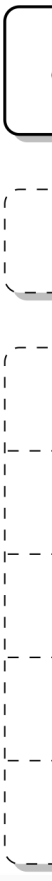
Simon Munzert
Hertie School | GRAD-C11/E1339

Table of contents

Scraping static webpa

Technologies of the world wide web

- To fully unlock the potential of web data for data science, we draw on certain web technologies.
- Importantly, often a basic understanding of these technologies is sufficient as the focus is on web data collection, not web development.
- Specifically, we have to understand
  - How our machine/browser/R communicates with web servers (→ **HTTP/S**)
  - How websites are built (→ **HTML**, **CSS**, basics of **JavaScript**)
  - How content in webpages can be effectively located (→ **XPath**, **CSS selectors**)
  - How dynamic web applications are executed and tapped (→ **AJAX**, **Selenium**)
  - How data by web services is distributed and processed (→ **APIs**, **JSON**, **XML**)



Web scraping

What is web scraping?

1. Pulling (unstructured) data from websites (HTMLs)
2. Bringing it into shape (into an analysis-ready format)

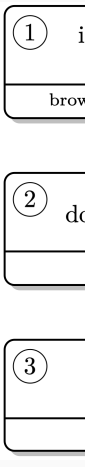
The philosophy of scraping with R

- No point-and-click procedure
- Script the entire process from start to finish
- Automate
    - The downloading of files
    - The scraping of information from web sites
    - Tapping APIs
    - Parsing of web content
    - Data tidying, text data processing
- Easily scale up scraping procedures
- Scheduling of scraping tasks

The scraping workflow

Key tools for scraping static webpages

1. You are able to inspect HTML pages in your browser using the web developer tools.
2. You are able to parse HTML into R with `rvest`.
3. You are able to speak XPath (or CSS selectors).
4. You are able to apply XPath expressions with `rvest`.
5. You are able to tidy web data with R/ `dplyr` / `regex`.



The big picture

- Every scraping project is different, but the coding pipeline is fundamentally similar.
- The (technically) hardest steps are location (XPath, CSS selectors) and extraction (clean-up), sometimes the scaling (from one to multiple sources).

Web scraping with rvest

`rvest` is a suite of scraping tools. It is part of the tidyverse and has m
scraping with R much more convenient.

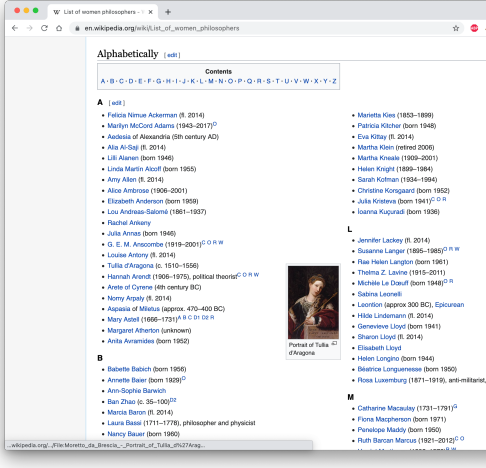
There are three key `rvest` verbs that you need to learn.[1]

1. `read_html()`: Read (parsing) an HTML resource.
2. `html_elements()`: Find elements that match a CSS selector or XPa
   expression.
3. `html_text2()`: Extract the text/value inside the node set.

[1] There is more in `rvest` than what we can cover today. Have a glimpse at th
(unofficial) cheat sheet.

Web scraping with rvest: example

- We are going to scrape a information from a Wikipedia article on women philosophers available at https://en.wikipedia.org/wiki/List_of_women_philosophers.
- The article provides two types of lists - one by period and one sorted alphabetically. We want the alphabetical list.
- The information we are actually interested in - names - is stored in unordered list elements.



Scraping with rvest: example (cont.)

Step 1: Parse the page

```
R> url_p ← read_html("https://en.wikipedia.org/wiki/List_of_wo
```

Step 2: Develop an XPath expression (or multiple) that select the info

```
R> elements_set ← html_elements(url_p, xpath = "//h2[text()='A
```

The XPath expression reads:

- `//h2`: Look for `h2` elements anywhere in the document.
- `[text()='Alphabetically']`: Look for `h2` elements with the conte
- `//following::li`: In the DOM tree following that element (at any
- `/a[1]` within these elements look for the first `a` element you ca

Scraping with rvest: example (cont.)

Step 3: Extract information and clean it up

```
R> phil_names ← elements_set %>% html_text2()
R> phil_names[c(1:2, 101:102)]
```

```
## [1] "A"                    "B"                    "Elisabeth
## [4] "Dorothy Emmet"
```

Step 4: Clean up (here: select the subset of links we care about)

```
R> names_iffer ←
+   seq_along(phil_names) ⩾  seq_along(phil_names)[str_detect(
+   seq_along(phil_names) ⩽   seq_along(phil_names)[str_detect
R> philosopher_names_clean ← phil_names[names_iffer]
R> length(philosopher_names_clean)
```

```
## [1] 267
```

```
R> philosopher_names_clean[1:5]
```

```
## [1] "Felicia Nimue Ackerman" "Marilyn McCord Adams"   "Aedesi
## [4] "Alia Al-Saji"           "Lilli Alanen"
```

Quick-n-dirty static webscraping with Se

The hassle with XPath

- The most cumbersome part of web scraping (data tidying aside) is the construction of XPath expressions that match the components of a page you want to extract.
- It will take a couple of scraping projects until you'll truly have mastered XPath.

A much-appreciated helper

- SelectorGadget is a JavaScript browser plugin that constructs XPath statements (or CSS selectors) via a point-and-click approach.
- It is available here: http://selectorgadget.com/ (there is also a Chrome extension).
- The tool is magic and you will love it.

What

- Y
- s
- I
- g
- y

Unde

- c
- T
- c
- c

SelectorGadget: example

SelectorGadget: example (cont.)

```
R> library(rvest)
R> url_p ← read_html("https://www.nytimes.com")
R> # xpath: paste the expression from Selectorgadget!
R> # note: we use single quotation marks here (' instead of ")
R> xpath ← '//*[contains(concat( " ", @class, " " ), concat( "
R> headlines ← html_elements(url_p, xpath = xpath)
R> headlines_raw ← html_text(headlines)
R> length(headlines_raw)
R> head(headlines_raw)
```

```
## [1] 29
```

```
## [1] "Retailers’ Latest Headache: Shutdowns at Their Vietnames
## [2] "With virus restrictions waning, it’s becoming clear: Bri
## [3] "Business updates: U.S. stock futures signaled a rebound
## [4] "Republicans at Odds Over Infrastructure Bill as Vote App
## [5] "Liberals Dig In Against Infrastructure Bill as Party Div
## [6] "Successful programs from around the world could guide Co
```

SelectorGadget: when to use and not to u

Having learned about a semi-automated approach to generating XPat

Why bother with learning XPath at all?

Well...

- SelectorGadget is not perfect. Sometimes, the algorithm will fail.
- Starting from a different element sometimes (but not always!) he
- Often the generated expressions are unnecessarily complex and
- In my experience, SelectorGadget works 50-60% of the times whe
- You are also prepared for the remaining 40-50%!

Scraping HTML tables

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

Built	Building	C
1870	Equitable Life Building	New
1889	Auditorium Building	Chic
1890	New York World Building	New
1894	Philadelphia City Hall	Phila
1908	Singer Building	
1909	Met Life Tower	
1913	Woolworth Building	
1930	40 Wall Street	New
1930	Chrysler Building	
1931	Empire State Building	
1972	World Trade Center (North Tower)	
1974	Willis Tower (formerly Sears Tower)	Chic
1996	Petronas Towers	Kual
2004	Taipei 101	Taipe
2010	Burj Khalifa	Duba

DATES	POLLSTER	GRADE	SA
• DEC. 28-30	Gallup	B-	1,
• DEC. 26-28	Rasmussen Reports/Pulse Opinion Research	C+	1,
• DEC. 24-28	Ipsos	A-	1,
• DEC. 23-27	Gallup	B-	1,
• DEC. 24-26	YouGov	B	1,

Scraping HTML tables

- HTML tables are everywhere.
- They are easy to spot in the wild - just look for `<table>` tags!
- Exactly because scraping tables is an easy and repetitive task, the
  `html_table()`.

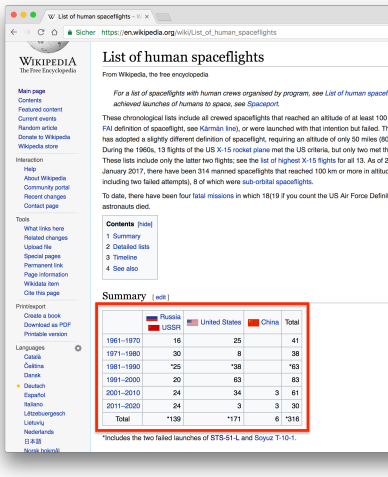
Function definition

```
R> html_table(x,
+     header = NA,
+     trim = TRUE,
+     dec = ".",
+     na.strings = "NA",
+     convert = TRUE
+  )
```

Argument	
`x`	Document (from `read_`
`header`	Use first row as header
`trim`	Remove leading and tr
`dec`	The character used as
`na.strings`	Character vector of val
`convert`	If `TRUE`, will run `type.`

Scraping HTML tables: example

- We are going to scrape a small table from the Wikipedia page https://en.wikipedia.org/wiki/List_of_human_spaceflights.
- (Note that we're actually using an old version of the page (dating back to May 1, 2018), which is accessible here. Wikipedia pages change, but this old revision and associated link won't.))
- The table is not entirely clean: There are some empty cells, but also images and links.
- The HTML code looks straightforward though.



Scraping HTML tables: example (cont.)

```
R> library(rvest)
R> url ← "https://en.wikipedia.org/wiki/List_of_human_spacefli
R> url_p ← read_html(url)
R> tables ← html_table(url_p, header = TRUE)
R> spaceflights ← tables[[1]]
R> spaceflights
```

```
## # A tibble: 7 × 5
##   ``        `Russia  Soviet Union` `United States` China Tota
##   <chr>     <chr>                  <chr>           <int> <chr
## 1 1961–1970 16                     25                 NA 41
## 2 1971–1980 30                     8                  NA 38
## 3 1981–1990 *25                    *38                NA *63
## 4 1991–2000 20                     63                 NA 83
## 5 2001–2010 24                     34                  3 61
## 6 2011–2020 24                     3                   3 30
## 7 Total     *139                   *171                6 *316
```

Web scraping: good

Scraping: the rules of the game

1. You take all the responsibility for your web scraping work.
2. Think about the nature of the data. Does it entail sensitive inform
   permission.
3. Take all copyrights of a country's jurisdiction into account. If you
4. If possible, stay identifiable. Stay polite. Stay friendly. Obey the sc
5. If in doubt, ask the author/creator/provider of data for permissio
   aren't bad that you get data.

Consult robots.txt

What's robots.txt?

- "Robots exclusion standard", informal protocol to prohibit web robots from crawling content
- Located in the root directory of a website (e.g., google.com/robots.txt)
- Documents which bot is allowed to crawl which resources (and which not)
- Not a technical barrier, but a sign that asks for compliance

What's robots.txt?

- Not an official W3C standard
- Rules listed bot by bot
- General rule listed under `User-agent: *` (most interesting entry for R-based crawlers)
- Directories folders listed separately

Exam

```
User
Disa
```

Univ

```
User
Disa
```

Allow

```
User
Disa
Allo
```

Craw

```
User
Craw
```

Downloading HTML files

Stay modest when accessing lots of data

- Content on the web is publicly available.
- But accessing the data causes server traffic.
- Stay polite by querying resources as sparsely as possible.

Two easy-to-implement practices

1. Do not bombard the server with requests - and if you have to, do so at modest pace.
2. Store web data on your local drive first, then parse.

Loop

```
R> f
+
+
+
+
+
+
+ }
```

-
-
-

Staying identifiable

Don't be a phantom

- Downloading massive amounts of data may arouse attention from server administrators.
- Assuming that you've got nothing to hide, you should stay identifiable beyond your IP address.

Two easy-to-implement practices

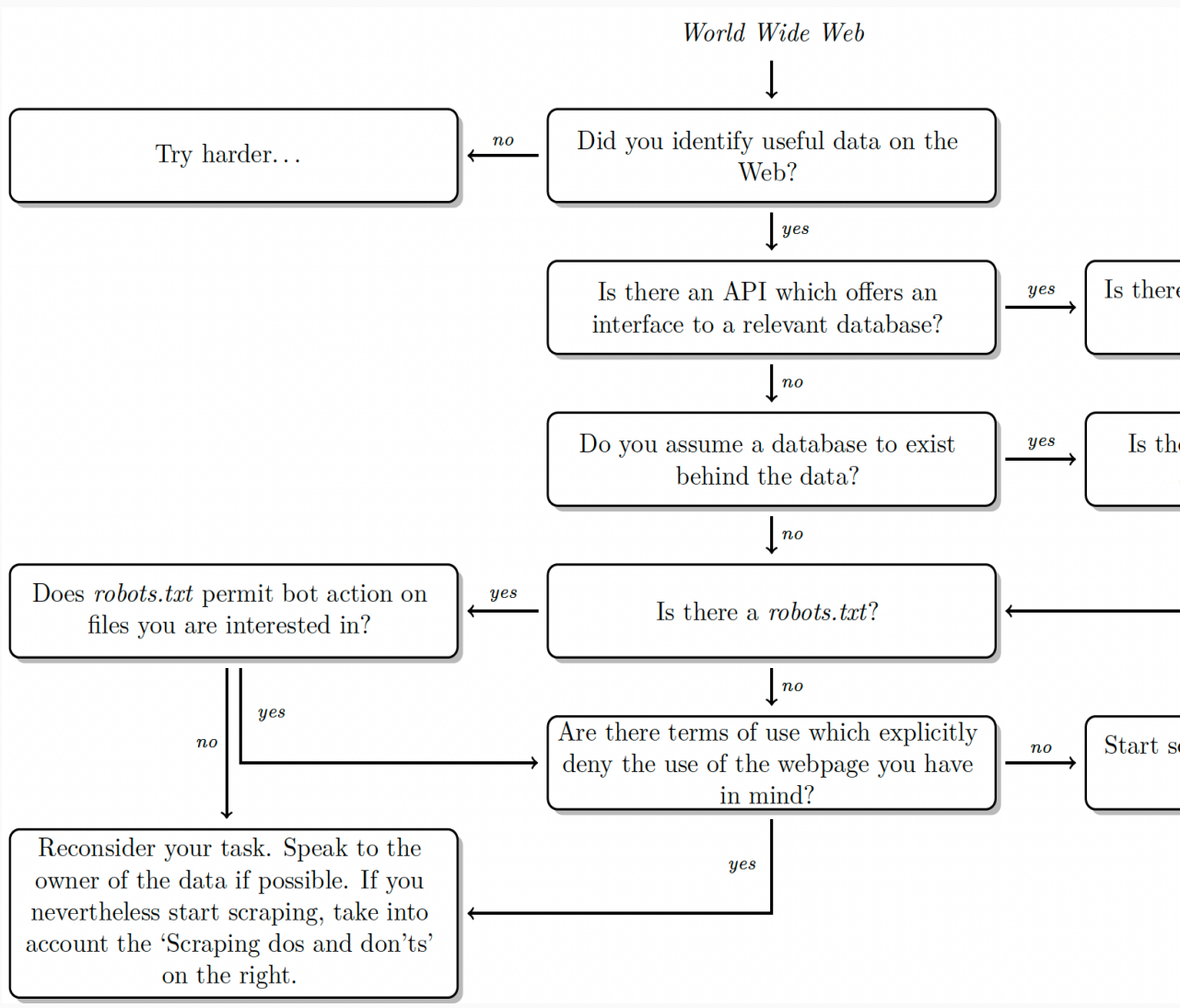
1. Get in touch with website administrators / data owners.
2. Use HTTP header fields `From` and `User-Agent` to provide information about yourself (by passing these to `add_headers()` from the `httr` library).

Stayi

```
R> u
R> r
+  a
+
+
+
+ )
R> h
+
+
```

-
-
- T

Scraping etiquette (cont.)

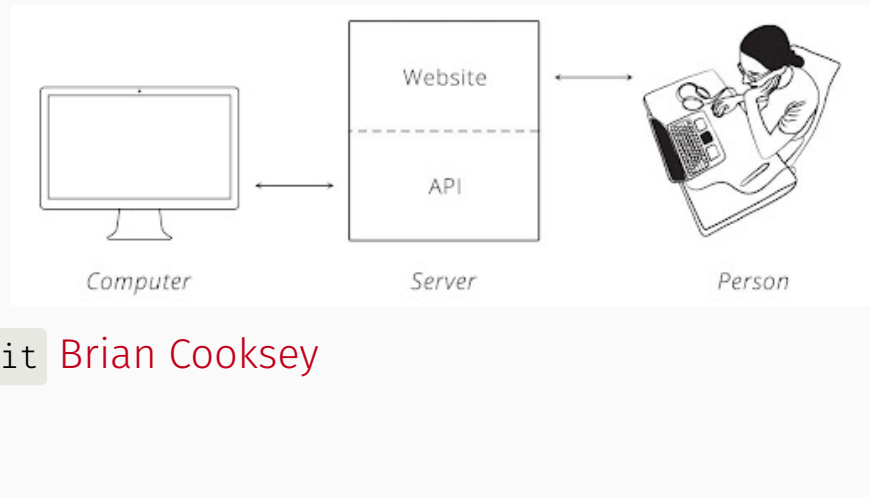


Web APIs: the b

What are web APIs?

Definition

- A web **A**pplication **P**rogramming **I**nterface lets you/your program query a provider for specific data.
- Think of web APIs as "data search engines": You pose a request, the API answers with a bulk of data.
- Many popular web services provide APIs (Google, Twitter, Wikipedia, ...).
- Often, APIs provide data in JSON, XML (can be any format though).



`Credit` Brian Cooksey

Key

What do APIs do?

Key perks from a web scraping perspective

- APIs provide instance access to clean data.
- They free us from building manual scrapers.
- They make it easier for a computer to interact with data on server.
- API usage implies mutual agreement about data collection.

Rest

[1] Kitc

Why do organizations have APIs?

Scalability and regulation of access

- Imagine if everyone decided to retrieve data from a server in an
  The amount of energy that this would consume would be very hi
- Also, access to data would be largely unregulated.
- With APIs, organizations can provide a regulated and organized w
  retrieve a large amount of data, without overwhelming or crashin
  violating their Terms of Use.

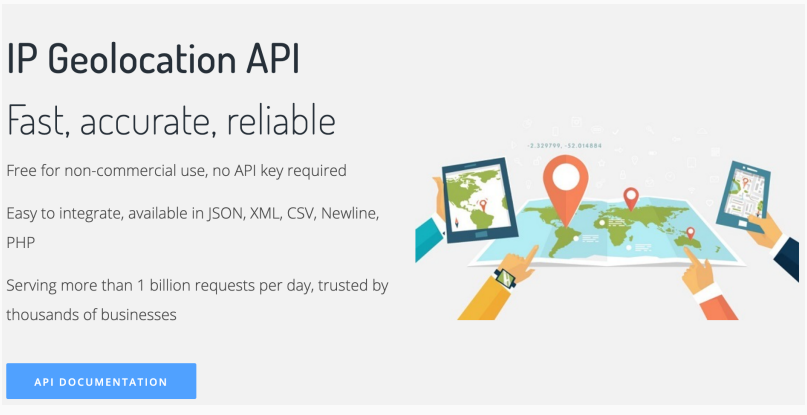
More reasons

- Monetization: Data can be turned into a sellable product.
- Innovation: Clients have access to data and develop their own pr
  solutions.
- Expansion: APIs can help companies move to different markets o
  other companies.

Example: Working with the IP API

The IP API

- The IP API at https://ip-api.com/ takes IP addresses and provides geolocation data (latitude, longitude, but also country state, city, etc.) in return.
- This is useful if you want to, e.g., map IP address data (although this is not perfectly accurate).
- The API is free to use and requires no registration. (There's also a pro service for commercial use though.)



An ex

{
}

Example: Documentation

Overview

JSON

Overview

This documentation is intended for developers who want to write applications that can query IP-API.
We serve our data in multiple formats via a simple URL-based interface over HTTP, which enables you to use our data directly from a user's browser or from your server.

Geolocation API

Response formats

JSON

XML

CSV

Newline

PHP

Batch API

Query multiple IP addresses in one HTTP request.

Batch JSON

Example: Calling the API with R

First, we specify the endpoint. We use the JSON endpoint and start with an empty query field.

```
R> endpoint ← "http://ip-api.com/json"
```

Next, we call the API with `httr`'s `GET()` method.

```
R> endpoint ← "http://ip-api.com/json"
R> response ← httr::GET(endpoint)
R> response
```

```
## Response [http://ip-api.com/json]
##   Date: 2025-10-02 15:38
##   Status: 200
##   Content-Type: application/json; charset=utf-8
##   Size: 315 B
```

Hoo
in re
func

```
R> r
R> r
```

```
## $
## [
##
## $
## [
##
## $
## [
##
## $
## [
##
## $
## [
##
```

Example: Calling the API with R *cont.*

This looks like an R list, which is useful. `httr::content()` automatically parsed the JSON file into an R list, which is useful. We could have also kept the raw (here: text) content with `httr::content(as = "text")`.

For more convenience (and flexibility), we can also use the powerful `jsonlite` package and its parser:

```
R> response_parsed ← jsonlite::fromJSON(endpoint)
R> response_parsed
```

```
## $status
## [1] "success"
##
## $country
## [1] "Germany"
##
## $countryCode
## [1] "DE"
##
## $region
```

Given
easil

```
R> a
```

```
##
## 1
##
## 1
##
## 1
##
## 1
##
## 1
##
## 1
##
## 1
##
## 1
##
## 1
##
## 1
```

Example: Calling the API with R *cont.*

We can easily modify the call to retrieve more data:

```
R> endpoint ← "http://ip-api.com/json/91.198.174.1"
R> response_parsed ← jsonlite::fromJSON(endpoint)
R> response_parsed
```

```
## $status
## [1] "success"
##
## $country
## [1] "The Netherlands"
##
## $countryCode
## [1] "NL"
##
## $region
## [1] "NH"
##
## $regionName
## [1] "North Holland"
##
## $city
```

The A
the a
requ
indiv
or do

```
R> e
R> r
+
R> h
+
+
```

```
##
## 1
## 2
##
## 1
## 2
##
## 1
```

Accessing APIs with R

Is there a pre-built API client for R?

1. **Yes.** Great. Use it if it provides the functionality you need (if not, see option 2).
2. **No.** Build your own API client.

API c

-
-
-
-
-

Build

-
-
-

API clients: example

The `ipapi` package by Bob Rudis provides high-level access to the IP
with the `geolocate()` function provided by the package to call the AP

```
R> # devtools::install_github("hrbrmstr/ipapi") # uncomment to
R> library(ipapi)
R> ip_df ← geolocate(c("", "10.0.1.1", "www.spiegel.de"), .pro
R> ip_df
```

```
##     status country countryCode region      regionName
##     <char>  <char>      <char> <char>          <char>
## 1: success Germany          DE     BE State of Berlin
## 2:    fail    <NA>        <NA>   <NA>            <NA>
## 3: success Germany          DE     HE           Hesse Frankfu
##        lat     lon      timezone                       isp
##      <num>   <num>        <char>                    <char>
## 1: 52.5203 13.3849 Europe/Berlin Vodafone Kabel Deutschland
## 2:      NA      NA          <NA>                      <NA>
## 3: 50.1103  8.7147 Europe/Berlin                Link11 GmbH
##                               org                      as
##                            <char>                  <char>
## 1: Vodafone Kabel Deutschland GmbH AS3209 Vodafone GmbH   95.
## 2:                            <NA>                    <NA>
```

Restricted API access

Why API access can be restricted

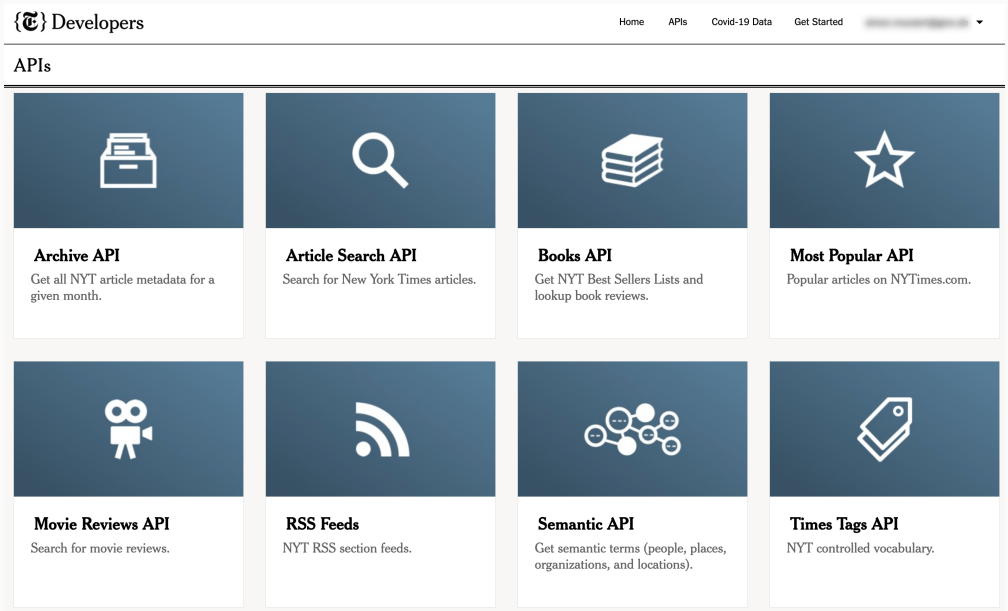
- The service provider wants to know who uses their API.
- Hosting APIs is costly. API usage limits can help control costs.
- The API hoster has a commercial interest: You pay for access (sor
  only).

Access tokens

- Access tokens serve as keys to an API.
- They usually come in form of a randomly generated string, such a
- Obtaining a token requires registration; sometimes payment. Som
- Once you have the token, you pass it along with your regular API

Restricted API access: example

The *New York Times* provides several APIs for developers at https://developer.nytimes.com/. In order to use them, we have to register as a developer (for free) and register our app. Then, we can use that key to call one of the APIs.



Restricted API access: example *cont.*

First, we load the key that we stored separately as a string (here: `nyti`

```
R> load("/Users/s.munzert/rkeys.RDa")
```

Next we specify the API endpoint using the API key:

```
R> endpoint ← "https://api.nytimes.com/svc/mostpopular/v2/viewe
R> url ← paste0(endpoint, "api-key=", nytimes_apikey)
```

Finally, we call the API and inspect the results:

```
R> nytimes_most_popular ← jsonlite::fromJSON(url)
R> nytimes_most_popular$results$title[1:3]
```

```
## [1] "How Social Security Will Be Affected by a Government Shu
## [2] "Trump to Withhold $18 Billion for New York-Area Transit
## [3] "White House Uses Shutdown to Maximize Pain and Punish Po
```

Restricted API access: some advice

NEVER hard-code your personal API key (or any personal information

Instead, use one of the following options:

1. Store your API keys in a separate file that you store somewhere e (see previous example).
2. Store your API keys in environment variables.

For the second option, you can use `Sys.setenv()` and `Sys.getenv()`

```
R> ## Set new environment variable called MY_API_KEY. Current s
R> Sys.setenv(MY_API_KEY="abcdefghijklmnopqrstuvwxyz0123456789"
R>
R> ## Assign the environment variable to an R object and pass i
R> my_api_key = Sys.getenv("MY_API_KEY")
```

The downside of this approach is that this environment variable will (on your system), you should modify the `.Renvion` file. Check out the

Recap: tapping APIs with R

1. Figure out whether an API is available that serves your needs.
2. Figure out whether an up-to-date and fully functional R client fo
   Terms of Use and client functionality, then use it.
3. If no client is available, build your own.

   a. Dive into the documentation.

   b. Use `httr` package to construct requests.

   c. Use `jsonlite` package to parse the JSON response (or other pa

   d. Address API error handling, user agent, authentication, paginat

   e. Write useful high-level functions that wrap your API calls and p

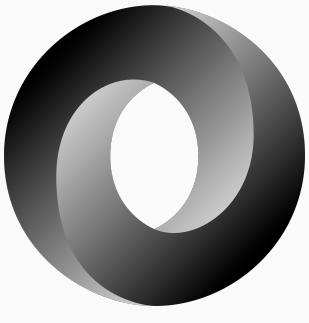
   f. Consider publishing your R API client as a package.

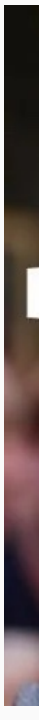
JSON

What's JSON?

Quick facts

- **J**ava**S**cript **O**bject **N**otation
- Popular data exchange format for web services / APIs
- "the fat-free alternative to XML"
- JSON ≠ Java, but a subset of JavaScript
- However, very flexible and not dependent upon any programming language
- Import of JSON data into R is relatively straightforward with the `jsonlite` package





Example

```
[
   {
      "name" : "van Pelt, Lucy",
      "sex" : "female",
      "age" : 32
   },
   {
      "name" : "Peppermint, Patty",
      "sex" : "female",
      "age" : null
   },
   {
      "name" : "Brown, Charlie",
      "sex" : "male",
      "age" : 27
   }
]
```

Basic syntax

Types of brackets

1. Curly brackets, `{` and `}`, embrace objects. Objects work similar to elements in XML/HTML and can contain other objects, key-value pairs or arrays.
2. Square brackets, `[` and `]`, embrace arrays. An array is an ordered sequence of objects or values.

Data structure

JSON can map complex data structures (objects nested within objects etc.), which can make it difficult to convert JSON into flattened R data structures (e.g., a data frame).

Also, there is no ultimate JSON-to-R converting function.

Luckily, JSON files returned by web services are usually not very complex. And `jsonlite` simplifies matters a lot.

Key-

Keys
cont

```
"nam
"age
```

Keys

```
"age
```

Key-

```
{"na
```

Valu

```
["va
```

JSON and R

Parsing JSON with R

- There are different packages available for JSON parsing with R.
- Choose `jsonlite` by Jeroen Ooms: It's well maintained and provides convincing mapping rules.

Key functions

There are two key functions in `jsonlite`:

- `fromJSON()`: converts input from JSON data into R objects following a set of conventions.
- `toJSON()`: converts input from R objects into JSON data

Get started with the package following this vignette.

Conv

```
R> T
R> x
R> t
```

```
## [
```

```
R> x
R> t
```

```
## [
```

```
R> x
R> t
```

```
## [
```

Summary

More on web scraping

Until now, the toy scraping examples were limited to single HTML pag
multiple pages. You might think of newspaper articles, Wikipedia pag
automating the scraping process becomes really powerful. Also, princ

In other cases, you might be confronted with

- forms,
- authentication,
- dynamic (JavaScript-enriched) content, or want to
- automatically navigate through pages interactively.

There's only so much we can cover in one session. Check out more m
solutions to some of these problems.

More on web APIs

Collecting data from the web using APIs provided by the data owner r
It allows for pure data collection without "HTML layout waste", standa
collection by the data owner, and robustness and scalability of data c

On the other hand, the rise of API architectures is not without issues.
creates dependencies on API suppliers. While APIs have the potential
also add to the siloing of information.

If you want to learn more about APIs in depth, check out this introdu
to APIs in R by Leo Glowacki.

There are many resources that give an overview of existing public API
Another useful resource is APIs for social scientists - a collaborative r

Finally, if you plan to write an R client for a web API, check out this gu

Coming up

Assignment

Assignment 2 is about to go online. Check it out and start scraping th

Next lecture

Databases! Bring coffee.