Web Data – Scraping and APIs

Scraping the web

Scraping the web: what?

An increasing amount of data is available on the web:

- Speeches, sentences, biographical information...

Scraping the web: what?

An increasing amount of data is available on the web:

- Speeches, sentences, biographical information...
- Social media data, newspaper articles, press releases...

Scraping the web: what?

An increasing amount of data is available on the web:

- Speeches, sentences, biographical information...
- Social media data, newspaper articles, press releases...
- Geographic information, conflict data...

Scraping the web: what?

An increasing amount of data is available on the web:

- Speeches, sentences, biographical information...
- Social media data, newspaper articles, press releases...
- Geographic information, conflict data...

These datasets are often provided in an unstructured format.

Scraping the web: what?

An increasing amount of data is available on the web:

- Speeches, sentences, biographical information...
- Social media data, newspaper articles, press releases...
- Geographic information, conflict data...

These datasets are often provided in an unstructured format.

Web scraping is the process of extracting this information automatically and transforming it into a structured dataset.

Scraping the web: why?

- Copy & pasting is time-consuming, boring, prone to errors, and impractical for large datasets

Scraping the web: why?

- Copy & pasting is time-consuming, boring, prone to errors, and impractical for large datasets
- In contrast, automated web scraping:

Scraping the web: why?

- Copy & pasting is time-consuming, boring, prone to errors, and impractical for large datasets
- In contrast, automated web scraping:
  1. Scales well for large datasets

Scraping the web: why?

- Copy & pasting is time-consuming, boring, prone to errors, and impractical for large datasets
- In contrast, automated web scraping:
  1. Scales well for large datasets
  2. Is reproducible

Scraping the web: why?

- Copy & pasting is time-consuming, boring, prone to errors, and impractical for large datasets
- In contrast, automated web scraping:
  1. Scales well for large datasets
  2. Is reproducible
  3. Involved adaptable techniques

Scraping the web: why?

- Copy & pasting is time-consuming, boring, prone to errors, and impractical for large datasets
- In contrast, automated web scraping:
  1. Scales well for large datasets
  2. Is reproducible
  3. Involved adaptable techniques
  4. Facilitates detecting and fixing errors

Scraping the web: why?

- Copy & pasting is time-consuming, boring, prone to errors, and impractical for large datasets
- In contrast, automated web scraping:
    1. Scales well for large datasets
    2. Is reproducible
    3. Involved adaptable techniques
    4. Facilitates detecting and fixing errors
- When to scrape?

Scraping the web: why?

- Copy & pasting is time-consuming, boring, prone to errors, and impractical for large datasets
- In contrast, automated web scraping:
  1. Scales well for large datasets
  2. Is reproducible
  3. Involved adaptable techniques
  4. Facilitates detecting and fixing errors
- When to scrape?
  1. Trade-off between your time today and your time in the future. Invest in your future self!

Scraping the web: why?

- Copy & pasting is time-consuming, boring, prone to errors, and impractical for large datasets
- In contrast, automated web scraping:
    1. Scales well for large datasets
    2. Is reproducible
    3. Involved adaptable techniques
    4. Facilitates detecting and fixing errors
- When to scrape?
    1. Trade-off between your time today and your time in the future. Invest in your future self!
    2. Computer time is cheap; human time is expensive

Scraping the web: two approaches

Two different approaches:

1. Screen scraping: extract data from source code of website, with html parser and/or regular expressions

Scraping the web: two approaches

Two different approaches:

1. Screen scraping: extract data from source code of website, with html parser and/or regular expressions
   - `rvest` package in R

Scraping the web: two approaches

Two different approaches:

1. Screen scraping: extract data from source code of website, with html parser and/or regular expressions
   - `rvest` package in R
2. Web APIs (application programming interfaces): a set of structured http requests that return JSON or XML data

Scraping the web: two approaches

Two different approaches:

1. Screen scraping: extract data from source code of website, with html parser and/or regular expressions
   - `rvest` package in R
2. Web APIs (application programming interfaces): a set of structured http requests that return JSON or XML data
   - `httr` package to construct API requests

Scraping the web: two approaches

Two different approaches:

1. Screen scraping: extract data from source code of website, with html parser and/or regular expressions
   - `rvest` package in R
2. Web APIs (application programming interfaces): a set of structured http requests that return JSON or XML data
   - `httr` package to construct API requests
   - Packages specific to each API: weatherData, WDI, Rfacebook... Check CRAN Task View on Web Technologies and Services for examples

The rules of the game

1. Respect the hosting site's wishes:

The rules of the game

1. Respect the hosting site's wishes:
   - Check if an API exists or if data are available for download

The rules of the game

1. Respect the hosting site's wishes:
   - Check if an API exists or if data are available for download
   - Keep in mind where data comes from and give credit (and respect copyright if you want to republish the data!)

The rules of the game

1. Respect the hosting site's wishes:
   - Check if an API exists or if data are available for download
   - Keep in mind where data comes from and give credit (and respect copyright if you want to republish the data!)
   - Some websites *disallow* scrapers on `robots.txt` file

The rules of the game

1. Respect the hosting site's wishes:
   - Check if an API exists or if data are available for download
   - Keep in mind where data comes from and give credit (and respect copyright if you want to republish the data!)
   - Some websites *disallow* scrapers on `robots.txt` file
2. Limit your bandwidth use:

The rules of the game

1. Respect the hosting site's wishes:
   - Check if an API exists or if data are available for download
   - Keep in mind where data comes from and give credit (and respect copyright if you want to republish the data!)
   - Some websites *disallow* scrapers on `robots.txt` file
2. Limit your bandwidth use:
   - Wait one or two seconds after each hit

The rules of the game

1. Respect the hosting site's wishes:
   - Check if an API exists or if data are available for download
   - Keep in mind where data comes from and give credit (and respect copyright if you want to republish the data!)
   - Some websites *disallow* scrapers on `robots.txt` file
2. Limit your bandwidth use:
   - Wait one or two seconds after each hit
   - Scrape only what you need, and just once (e.g. store the html file in disk, and then parse it)

The rules of the game

1. Respect the hosting site's wishes:
    - Check if an API exists or if data are available for download
    - Keep in mind where data comes from and give credit (and respect copyright if you want to republish the data!)
    - Some websites *disallow* scrapers on `robots.txt` file
2. Limit your bandwidth use:
    - Wait one or two seconds after each hit
    - Scrape only what you need, and just once (e.g. store the html file in disk, and then parse it)
3. When using APIs, read documentation

The rules of the game

1. Respect the hosting site's wishes:
    - Check if an API exists or if data are available for download
    - Keep in mind where data comes from and give credit (and respect copyright if you want to republish the data!)
    - Some websites *disallow* scrapers on `robots.txt` file
2. Limit your bandwidth use:
    - Wait one or two seconds after each hit
    - Scrape only what you need, and just once (e.g. store the html file in disk, and then parse it)
3. When using APIs, read documentation
    - Is there a batch download option?

The rules of the game

1. Respect the hosting site's wishes:
   - Check if an API exists or if data are available for download
   - Keep in mind where data comes from and give credit (and respect copyright if you want to republish the data!)
   - Some websites *disallow* scrapers on `robots.txt` file
2. Limit your bandwidth use:
   - Wait one or two seconds after each hit
   - Scrape only what you need, and just once (e.g. store the html file in disk, and then parse it)
3. When using APIs, read documentation
   - Is there a batch download option?
   - Are there any rate limits?

The rules of the game

1. Respect the hosting site's wishes:
   - Check if an API exists or if data are available for download
   - Keep in mind where data comes from and give credit (and respect copyright if you want to republish the data!)
   - Some websites *disallow* scrapers on `robots.txt` file
2. Limit your bandwidth use:
   - Wait one or two seconds after each hit
   - Scrape only what you need, and just once (e.g. store the html file in disk, and then parse it)
3. When using APIs, read documentation
   - Is there a batch download option?
   - Are there any rate limits?
   - Can you share the data?

The art of web scraping

Workflow:

1. Learn about structure of website

The art of web scraping

Workflow:

1. Learn about structure of website
2. Choose your strategy

The art of web scraping

Workflow:

1. Learn about structure of website
2. Choose your strategy
3. Build prototype code: extract, prepare, validate

The art of web scraping

Workflow:

1. Learn about structure of website
2. Choose your strategy
3. Build prototype code: extract, prepare, validate
4. Generalize: functions, loops, debugging

The art of web scraping

Workflow:

1. Learn about structure of website
2. Choose your strategy
3. Build prototype code: extract, prepare, validate
4. Generalize: functions, loops, debugging
5. Data cleaning

The art of web scraping

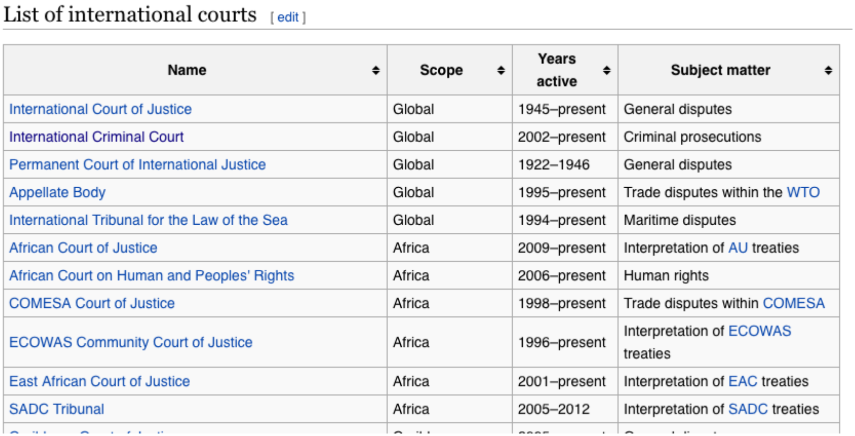
Three main scenarios

Three main scenarios

1. Data in table format

International court

From Wikipedia, the free encyclopedia

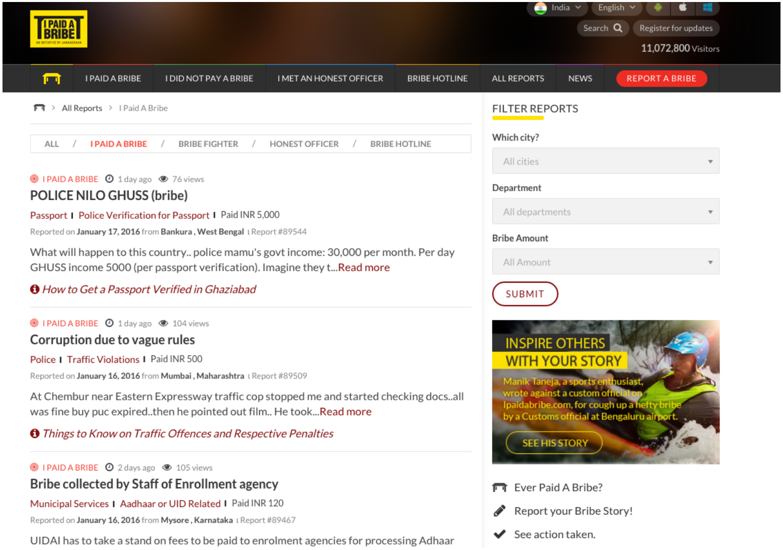


List of international courts [edit]

Name	Scope	Years active	Subject matter
International Court of Justice	Global	1945–present	General disputes
International Criminal Court	Global	2002–present	Criminal prosecutions
Permanent Court of International Justice	Global	1922–1946	General disputes
Appellate Body	Global	1995–present	Trade disputes within the WTO
International Tribunal for the Law of the Sea	Global	1994–present	Maritime disputes
African Court of Justice	Africa	2009–present	Interpretation of AU treaties
African Court on Human and Peoples' Rights	Africa	2006–present	Human rights
COMESA Court of Justice	Africa	1998–present	Trade disputes within COMESA
ECOWAS Community Court of Justice	Africa	1996–present	Interpretation of ECOWAS treaties
East African Court of Justice	Africa	2001–present	Interpretation of EAC treaties
SADC Tribunal	Africa	2005–2012	Interpretation of SADC treaties

Three main scenarios

2. Data in unstructured format



India English

Search Register for updates

11,072,800 Visitors

I PAID A BRIBE | I DID NOT PAY A BRIBE | I MET AN HONEST OFFICER | BRIBE HOTLINE | ALL REPORTS | NEWS | REPORT A BRIBE

All Reports > I Paid A Bribe

ALL / I PAID A BRIBE / BRIBE FIGHTER / HONEST OFFICER / BRIBE HOTLINE

I PAID A BRIBE 1 day ago 76 views

POLICE NILO GHUSS (bribe)

Passport | Police Verification for Passport | Paid INR 5,000

Reported on **January 17, 2016** from **Bankura , West Bengal** | Report #89544

What will happen to this country.. police mamu's govt income: 30,000 per month. Per day GHUSS income 5000 (per passport verification). Imagine they t...Read more

How to Get a Passport Verified in Ghaziabad

I PAID A BRIBE 1 day ago 104 views

Corruption due to vague rules

Police | Traffic Violations | Paid INR 500

Reported on **January 16, 2016** from **Mumbai , Maharashtra** | Report #89509

At Chembur near Eastern Expressway traffic cop stopped me and started checking docs..all was fine buy puc expired..then he pointed out film.. He took...Read more

Things to Know on Traffic Offences and Respective Penalties

I PAID A BRIBE 2 days ago 105 views

Bribe collected by Staff of Enrollment agency

Municipal Services | Aadhaar or UID Related | Paid INR 120

Reported on **January 16, 2016** from **Mysore , Karnataka** | Report #89467

UIDAI has to take a stand on fees to be paid to enrolment agencies for processing Adhaar

FILTER REPORTS

Which city?
All cities

Department
All departments

Bribe Amount
All Amount

SUBMIT

INSPIRE OTHERS WITH YOUR STORY

Manik Taneja, a sports enthusiast, wrote against a custom official on Ipaidabribe.com, for cough up a hefty bribe by a Customs official at Bengaluru airport.

SEE HIS STORY

Ever Paid A Bribe?

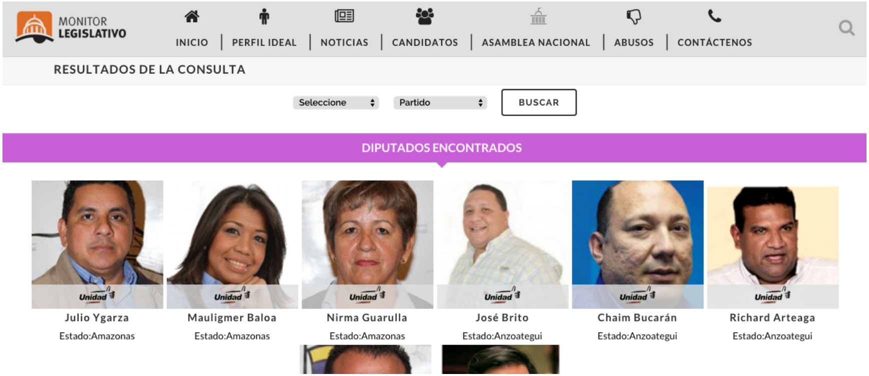
Report your Bribe Story!

See action taken.

www.ipaidabribe.com/reports/paid

Three main scenarios

3. Data hidden behind web forms


MONITOR LEGISLATIVO
INICIO | PERFIL IDEAL | NOTICIAS | CANDIDATOS | ASAMBLEA NACIONAL | ABUSOS | CONTÁCTENOS
RESULTADOS DE LA CONSULTA
Seleccione
Partido
BUSCAR
DIPUTADOS ENCONTRADOS

Julio Ygarza	Mauligmer Baloa	Nirma Guarulla	José Brito	Chaim Bucarán	Richard Arteaga
Estado:Amazonas	Estado:Amazonas	Estado:Amazonas	Estado:Anzoategui	Estado:Anzoategui	Estado:Anzoategui

Candidates on 2015 Venezuelan parliamentary election

Three main scenarios

1. Data in table format

Three main scenarios

1. Data in table format
    - Automatic extraction with `rvest`

Three main scenarios

1. Data in table format
   - Automatic extraction with `rvest`
2. Data in unstructured format

Three main scenarios

1. Data in table format
   - Automatic extraction with `rvest`
2. Data in unstructured format
   - Element identification with `selectorGadget`

Three main scenarios

1. Data in table format
   - Automatic extraction with `rvest`
2. Data in unstructured format
   - Element identification with `selectorGadget`
   - Automatic extraction with `rvest`

HTML: a primer

Hypertext Markup Language (HTML): hidden standard behind every website.

HTML: a primer

Hypertext Markup Language (HTML): hidden standard behind every website.

- HTML is text with marked-up structure, defined by tags:

-
  ```
  <!DOCTYPE html>
  <html>
  <body>

  <h1>My First Heading</h1>

  <p>My first paragraph.</p>

  </body>
  </html>
  ```
- What you see in your browser is an interpretation of the HTML document

HTML: a primer

- Some common tags:

HTML: a primer

- Some common tags:
  - Document elements: `<head>`, `<body>`, `<footer>`...

HTML: a primer

- Some common tags:
  - Document elements: `<head>`, `<body>`, `<footer>`...
  - Document components: `<title>`,`<h1>`, `<div>`...

HTML: a primer

- Some common tags:
  - Document elements: `<head>`, `<body>`, `<footer>`...
  - Document components: `<title>`,`<h1>`, `<div>`...
  - Text style: `<b>`, `<i>`, `<strong>`...

HTML: a primer

- Some common tags:
    - Document elements: `<head>`, `<body>`, `<footer>`...
    - Document components: `<title>`,`<h1>`, `<div>`...
    - Text style: `<b>`, `<i>`, `<strong>`...
    - Hyperlinks: `<a>`

HTML: a primer

- Some common tags:
  - Document elements: `<head>`, `<body>`, `<footer>`...
  - Document components: `<title>`,`<h1>`, `<div>`...
  - Text style: `<b>`, `<i>`, `<strong>`...
  - Hyperlinks: `<a>`
- An example: www.pablobarbera.com

Beyond HTML

- Cascading Style Sheets (CSS): describes formatting of HTML components (e.g. `<h1>`, `<div>`...), useful for us!

- Javascript: adds functionalities to the website (e.g. change content/structure after website has been loaded)

Parsing HTML code

First step in webscraping: read HTML code in R and parse it

Parsing HTML code

First step in webscraping: read HTML code in R and parse it

- Parsing = understanding structure

Parsing HTML code

First step in webscraping: read HTML code in R and parse it

- Parsing = understanding structure
- How? `rvest` package in R:

Parsing HTML code

First step in webscraping: read HTML code in R and parse it

- Parsing = understanding structure
- How? `rvest` package in R:
  - `read_html`: parse HTML code into R

Parsing HTML code

First step in webscraping: read HTML code in R and parse it

- Parsing = understanding structure
- How? `rvest` package in R:
  - `read_html`: parse HTML code into R
  - `html_text`: extract text from HTML code

Parsing HTML code

First step in webscraping: read HTML code in R and parse it

- Parsing = understanding structure
- How? `rvest` package in R:
    - `read_html`: parse HTML code into R
    - `html_text`: extract text from HTML code
    - `html_table`: extract tables in HTML code

Parsing HTML code

First step in webscraping: read HTML code in R and parse it

- Parsing = understanding structure
- How? `rvest` package in R:
  - `read_html`: parse HTML code into R
  - `html_text`: extract text from HTML code
  - `html_table`: extract tables in HTML code
  - `html_nodes`: extract components with CSS selector

Parsing HTML code

First step in webscraping: read HTML code in R and parse it

- Parsing = understanding structure
- How? `rvest` package in R:
  - `read_html`: parse HTML code into R
  - `html_text`: extract text from HTML code
  - `html_table`: extract tables in HTML code
  - `html_nodes`: extract components with CSS selector
  - `html_attrs`: extract attributes of nodes

Parsing HTML code

First step in webscraping: read HTML code in R and parse it

- Parsing = understanding structure
- How? `rvest` package in R:
  - `read_html`: parse HTML code into R
  - `html_text`: extract text from HTML code
  - `html_table`: extract tables in HTML code
  - `html_nodes`: extract components with CSS selector
  - `html_attrs`: extract attributes of nodes
- How to identify relevant CSS selectors? `selectorGadget` extension for Chrome and Firefox.

APIs

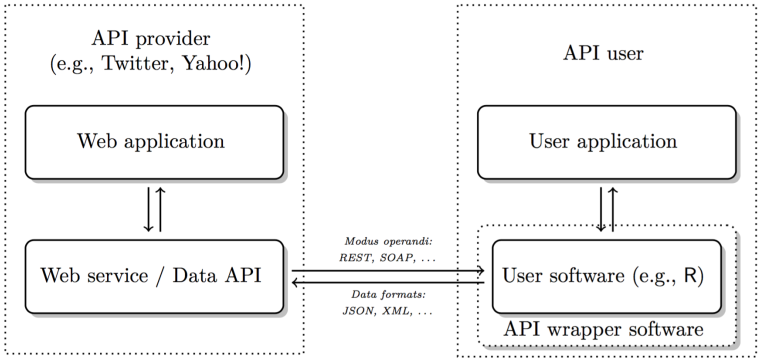
APIs

API = Application Programming Interface; a set of structured http requests that return data in a lightweight format.

APIs

API = Application Programming Interface; a set of structured http requests that return data in a lightweight format.

HTTP = Hypertext Transfer Protocol; how browsers and e-mail clients communicate with servers.



Source: Munzert et al, 2014, Figure 9.8

APIs

Types of APIs:

1. RESTful APIs: queries for static information at current moment (e.g. user profiles, posts, etc.)
2. Streaming APIs: changes in users' data in real time (e.g. new tweets, weather alerts...)

APIs

Types of APIs:

1. RESTful APIs: queries for static information at current moment (e.g. user profiles, posts, etc.)
2. Streaming APIs: changes in users' data in real time (e.g. new tweets, weather alerts...)

APIs generally have extensive documentation:

- Written for developers, so must be understandable for humans
- What to look for: endpoints and parameters.

APIs

Types of APIs:

1. RESTful APIs: queries for static information at current moment (e.g. user profiles, posts, etc.)
2. Streaming APIs: changes in users' data in real time (e.g. new tweets, weather alerts...)

APIs generally have extensive documentation:

- Written for developers, so must be understandable for humans
- What to look for: endpoints and parameters.

Most APIs are rate-limited:

- Restrictions on number of API calls by user/IP address and period of time.
- Commercial APIs may impose a monthly fee

Connecting with an API

Constructing a REST API call:

- Baseline URL endpoint: `https://maps.googleapis.com/maps/api/geocode/json`
- Parameters: `?address=budapest`
- Authentication token (optional): `&key=XXXXX`

From R, use `httr` package to make `GET` request:

```
library(httr)
r <- GET(
"https://maps.googleapis.com/maps/api/geocode/json",
query=list(address="budapest"))
```

If request was successful, returned code will be 200, where `4xx` indicates client errors and `5xx` indicates server errors.
If you need to attach data, use `POST` request.

JSON

Response is often in JSON format (Javascript Object Notation).

JSON

Response is often in JSON format (Javascript Object Notation).

- Type: `content(r, "text")`

JSON

Response is often in JSON format (Javascript Object Notation).

- Type: `content(r, "text")`
- Data stored in key-value pairs. Why? Lightweight, more flexible than traditional table format.

JSON

Response is often in JSON format (Javascript Object Notation).

- Type: `content(r, "text")`
- Data stored in key-value pairs. Why? Lightweight, more flexible than traditional table format.
- Curly brackets embrace objets; square brackets enclose arrays (vectors)

JSON

Response is often in JSON format (Javascript Object Notation).

- Type: `content(r, "text")`
- Data stored in key-value pairs. Why? Lightweight, more flexible than traditional table format.
- Curly brackets embrace objets; square brackets enclose arrays (vectors)
- Use `fromJSON` function from `jsonlite` package to read JSON data into R

JSON

Response is often in JSON format (Javascript Object Notation).

- Type: `content(r, "text")`
- Data stored in key-value pairs. Why? Lightweight, more flexible than traditional table format.
- Curly brackets embrace objets; square brackets enclose arrays (vectors)
- Use `fromJSON` function from `jsonlite` package to read JSON data into R
- But many packages have their own specific functions to read data in JSON format; `content(r, "parsed")`

Authentication

- Many APIs require an access key or token
- An alternative, open standard is called OAuth
- Connections without sharing username or password, only temporary tokens that can be refreshed
- `httr` package in R implements most cases (examples)

R packages

Before starting a new project, worth checking if there's already an R package for that API. Where to look?

- CRAN Web Technologies Task View (but only packages released in CRAN)
- GitHub (including unreleased packages and most recent versions of packages)
- rOpenSci Consortium

Also see this great list of APIs in case you need inspiration.

Why APIs?

Advantages:

- ‘Pure’ data collection: avoid malformed HTML, no legal issues, clear data structures, more trust in data collection...

Why APIs?

Advantages:

- ‘Pure’ data collection: avoid malformed HTML, no legal issues, clear data structures, more trust in data collection...
- Standardized data access procedures: transparency, replicability

Why APIs?

Advantages:

- ‘Pure’ data collection: avoid malformed HTML, no legal issues, clear data structures, more trust in data collection...
- Standardized data access procedures: transparency, replicability
- Robustness: benefits from ‘wisdom of the crowds’

Why APIs?

Advantages:

- 'Pure' data collection: avoid malformed HTML, no legal issues, clear data structures, more trust in data collection...
- Standardized data access procedures: transparency, replicability
- Robustness: benefits from 'wisdom of the crowds'

Disadvantages

Why APIs?

Advantages:

- ‘Pure’ data collection: avoid malformed HTML, no legal issues, clear data structures, more trust in data collection...
- Standardized data access procedures: transparency, replicability
- Robustness: benefits from ‘wisdom of the crowds’

Disadvantages

- They’re not too common (yet!)

Why APIs?

Advantages:

- ‘Pure’ data collection: avoid malformed HTML, no legal issues, clear data structures, more trust in data collection...
- Standardized data access procedures: transparency, replicability
- Robustness: benefits from ‘wisdom of the crowds’

Disadvantages

- They’re not too common (yet!)
- Dependency on API providers

Why APIs?

Advantages:

- ‘Pure’ data collection: avoid malformed HTML, no legal issues, clear data structures, more trust in data collection...
- Standardized data access procedures: transparency, replicability
- Robustness: benefits from ‘wisdom of the crowds’

Disadvantages

- They’re not too common (yet!)
- Dependency on API providers
- Lack of natural connection to R