



Intro to R

Basic Web-Scraping in R

Prof. Kevin Koehler

kevin.koehler@santannapisa.it

What is Web Scraping?

Web scraping

The process of **automatically extracting data from websites**



What is Web Scraping?

💡 Web scraping

The process of **automatically extracting data from websites**

Useful when:

No public [application programming interface \(API\)](#) is available

What is Web Scraping?

Web scraping


The process of **automatically extracting data from websites**

Useful when:

No public [application programming interface \(API\)](#) is available

Data is embedded in [HTML](#) pages

Legal and Ethical Considerations

 robots.txt

Respect robots.txt

Many websites have a `robots.txt` file which contains instructions on which parts of the site may be scraped and how. These are not legally binding, but your IP might be blocked if you don't follow the instructions.



Legal and Ethical Considerations

⚠ robots.txt

Respect robots.txt

Many websites have a `robots.txt` file which contains instructions on which parts of the site may be scraped and how. These are not legally binding, but your IP might be blocked if you don't follow the instructions.

You can inspect a site's instructions by adding `robots.txt` to the URL (e.g., <https://en.wikipedia.org/robots.txt>).



Legal and Ethical Considerations

There also is a useful R package for interacting with `robots.txt` (if there is one):

```
#install.packages("robotstxt")  
library(robotstxt)  
  
paths_allowed("https://en.wikipedia.org/")
```

This tells us that, in principle, we are allowed to scrape Wikipedia.

Legal and Ethical Considerations

 Avoid overloading servers

Many sites and servers monitor and rate-limit requests. **If you do not respect these limits, your IP might be blocked.**



Legal and Ethical Considerations

 Avoid overloading servers

Many sites and servers monitor and rate-limit requests. **If you do not respect these limits, your IP might be blocked.**

You can **artificially limit your request rate** with the `Sys.sleep()` function in R. Within the brackets you can specify the number of seconds the system should wait before proceeding with the next step.



Legal and Ethical Considerations

 Avoid overloading servers

Many sites and servers monitor and rate-limit requests. **If you do not respect these limits, your IP might be blocked.**

You can **artificially limit your request rate** with the `Sys.sleep()` function in R. Within the brackets you can specify the number of seconds the system should wait before proceeding with the next step.

There are also ways of programmatically varying your IP, but this is **illegal** if you do it to circumvent restrictions.



Legal and Ethical Considerations

 Personal data

Never scrape personal/private data without permission



Legal and Ethical Considerations

polite

The `polite` package provides functionality to automatically check `robots.txt` and enforce rate limits.



Static vs. dynamic

! Dynamic content

rvest is not good at scraping **dynamic content**, i.e., content rendered dynamically using Java script. We will focus on static websites in this session.



Structure

1. Parsing **HTML content** and identifying desired data

Structure

1. Parsing **HTML content** and identifying desired data
2. Using **rvest** to extract data

Structure

1. Parsing **HTML content** and identifying desired data
2. Using **rvest** to extract data
3. **Clean and structure** the data



Example: Scraping Wikipedia

For the purpose of this lesson, we will assume that we want to scrape results for Turkish parliamentary elections from Wikipedia.



Example: Scraping Wikipedia

For the purpose of this lesson, we will assume that we want to scrape [results for Turkish parliamentary elections](#) from Wikipedia.

Wikipedia allows scraping in principle:

```
paths_allowed("https://en.wikipedia.org/wiki/Parliamentary_elections_in_Turkey") returns TRUE
```



Example: Scraping Wikipedia

For the purpose of this lesson, we will assume that we want to scrape [results for Turkish parliamentary elections](#) from Wikipedia.

Wikipedia allows scraping in principle:

```
paths_allowed("https://en.wikipedia.org/wiki/Parliamentary_elections_in_Turkey") returns TRUE
```

Have a look at the page and see which part we need to scrape.

How can we identify the part we need to
scrape?



Some basic HTML

HTML stands for [Hyper Text Markup Language](#).



Some basic HTML

HTML stands for [Hyper Text Markup Language](#).

It is the standard language for creating **web pages**



Some basic HTML

HTML stands for **Hyper Text Markup Language**.

It is the standard language for creating **web pages**

HTML has a hierarchical structure formed by elements which consist of

- ▶ a **start tag** (e.g., `<table>`),
- ▶ optional **attributes** (`class='wikitable'`),
- ▶ an **end tag** (like `</table>`),
- ▶ and **contents** (everything in between the start and end tag).



Some basic HTML

HTML stands for **Hyper Text Markup Language**.

It is the standard language for creating **web pages**

HTML has a hierarchical structure formed by elements which consist of

- ▶ a **start tag** (e.g., `<table>`),
- ▶ optional **attributes** (`class='wikitable'`),
- ▶ an **end tag** (like `</table>`),
- ▶ and **contents** (everything in between the start and end tag).

This formal and hierarchical structure allows us to identify and scrape specific information.

Some basic HTML

```
<!DOCTYPE html>
<html>
  <head>
    <title>Page Title</title>
  </head>
  <body>
    <h1>My First Heading</h1>
    <p>My first paragraph.</p>
    <table class='mytable'>
      <tr><th>x</th>  <th>y</th></tr>
      <tr><td>1.5</td> <td>2.7</td></tr>
      <tr><td>4.9</td> <td>1.3</td></tr>
    </table>
  </body>
</html>
```



Inspecting HTML

Most browsers allow you to [inspect the HTML code of a website](#).



Inspecting HTML

Most browsers allow you to [inspect the HTML code of a website](#).

CSS selector gadget

If you use Google Chrome can also install and use the CSS selector gadget extension.



Introducing `rvest`

- ▶ Part of the **tidyverse** ecosystem
- ▶ Designed to make scraping easy and readable
- ▶ Inspired by Python's **BeautifulSoup**



Introducing `rvest`

- ▶ Part of the **tidyverse** ecosystem
- ▶ Designed to make scraping easy and readable
- ▶ Inspired by Python's **BeautifulSoup**

Core functions:

1. `read_html()` to read the website
2. `html_elements()` finds elements using structural features of the website
3. `html_attrs()` extracts attributes
4. `html_text()` and `html_text2()` extract text from elements
5. `html_table()` extracts tables and writes them in a data frame



Introducing rvest

Let's create basic HTML code:

```
example <- minimal_html("
  <h1>Headline</h1>
  <p id='first'>First paragraph</p>
  <p class='important'>Important paragraph</p>
  <table class='mytable'>
    <tr><th>x</th>  <th>y</th></tr>
    <tr><td>1.5</td> <td>2.7</td></tr>
    <tr><td>4.9</td> <td>1.3</td></tr>
    <tr><td>7.2</td> <td>8.1</td></tr>
  </table>
")
```

Headline

First paragraph

Important paragraph

x	y
1.5	2.7
4.9	1.3
7.2	8.1

And then extract parts with rvest

```
example %>% html_elements("p") %>%
  html_text2()
```

```
[1] "First paragraph"      "Important paragraph"
```

```
example %>% html_elements(".important") %>%
  html_text2()
```

```
[1] "Important paragraph"
```

```
example %>% html_element(".mytable") %>%
  html_table()
```

```
# A tibble: 3 x 2
```

```
      x     y
<dbl> <dbl>
1  1.5  2.7
2  4.9  1.3
3  7.2  8.1
```

Let's move over to R, scrape Wikipedia, and
produce a nice table

