Text Mining and Sentiment Analysis

Prof. Annamaria Bianchi
A.Y. 2024/2025

Lecture 20

19 May 2025

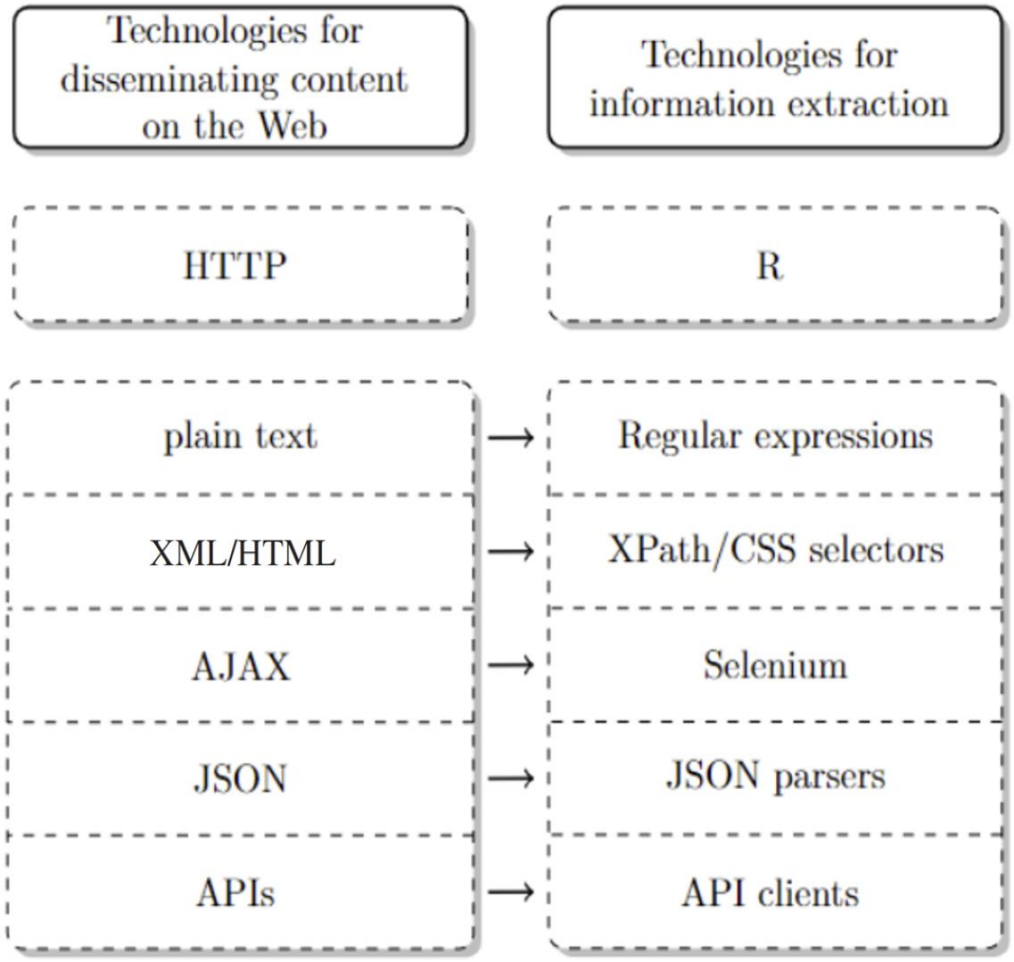
UNIVERSITÀ DEGLI STUDI DI BERGAMO | Dipartimento di Scienze Economiche

Outline

Introduction to Web Scraping with R

Introduction

- In the Big Data Era, the Web is a relevant source of information. In order to get such data, it is possible to implement Web Scraping or to use APIs.
- The figure shows the pillars of data distribution on the Web and data extraction (using R).



Adapted from Munzert et al. 2015

Introduction

HTPP is the standard for the communication on Web. It stands for Hypertext Transfer protocol.

Plain Text is obviously an important part of HTML,XML or JSON documents. You already know how to deal with textual data using **regular expressions** and R.

HTML/XML are two Markup Languages. The difference is that HTML is designed to display the data, while XML is designed for exchanging the data. You may encounter both format. The rvest functions work also with XML data. The main difference is that HTML has predefined tags, while XML has not. Once we obtain HTML or XLM documents, the relevant information are extracted using **XPath** queries or **CSS selectors**

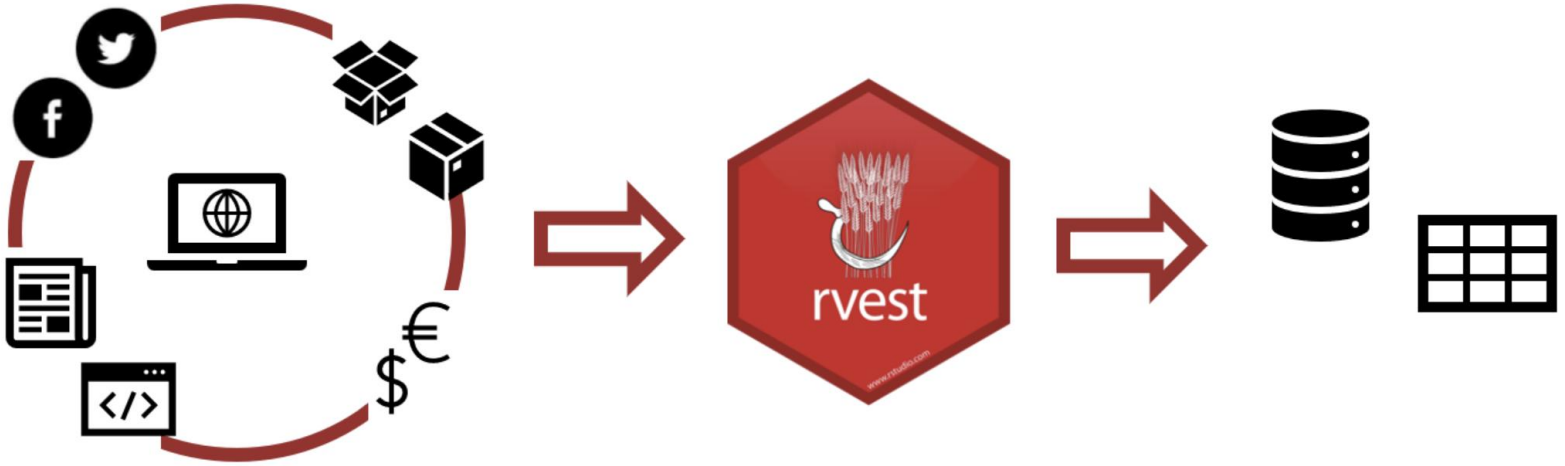
JSON (JavaScript Object Notation) is a standard for data storage and exchange format which is common to encounter on the Web. For examples, data collected through APIs are usually delivered in the JSON format. R allows to deal with JSON documents using **JSON parsers**.

AJAX is a combination of technologies in order to request and display data from the web server. It is very important whe scraping dynamic web pages. Basically, it allows to update a web page asynchronously, i.e., to update a web page, without reloading the whole page. Information from AJAX-eriched webpages are extracted using **Selenium**.

APIs (Application Programming Interface) are tools which allows programmers to directly ask for data. Here, we study how APIs can be used in order to get Twitter, Wikipedia, and others data. **API Clients** provide an interface to APIs (for example the rtweet package)

Web Scraping and good practices

Definition: Web scraping refers to the action of extracting data from websites.



For example, we can extract information about companies, product/services/films reviews or prices, social media posts, news articles and government websites.
In this part of the course, we will see how to scrape websites' data using the R package `rvest` which is based on the *tidyverse philosophy*. The workflow is presented in the figure.

This is not illegal per se but you should be aware that the data you are scraping may be subject to copyright or privacy regulations. So, before engaging in a project based on WS you should check regulations and permission for analyzing and publishing results about such data.

Legal aspects

Before engaging in a big Web Scraping project, I suggest to contact the legal office of your university/organization (especially if you work in a private company and you are not doing academic research). Whether it is legal or not is unclear in most cases. Page 278 of Munzert et al. 2015 offers an overview of legal cases about web scraping.

When doing web scraping, you should consider the following:

1. Take into account legal aspects and copyrights/internet laws of that jurisdiction;
2. Publish data is usually not allowed and you can commit copyright fraud;
3. Stay identifiable;
4. Comply with the website's Terms of Use and robot.txt file;
5. In case of doubt, contact the provider of data for permission (Usually for academic and scientific purposes it is not difficult to get data).

.

Terms of Use and robots.txt

The Terms of Use are usually linked in the footer. For example, we can inspect the terms of use of Twitter and Reddit:

X

Reddit

Terms of Use and robots.txt

The Terms of Use are usually linked in the footer. For example, we can inspect the terms of use of Twitter and Reddit:

X	access or search or attempt to access or search the Services by any means (automated or otherwise) other than through our currently available, published interfaces that are provided by us (and only pursuant to the applicable terms and conditions), unless you have been specifically allowed to do so in a separate agreement with us (NOTE: crawling or scraping the Services in any form, for any purpose without our prior written consent is expressly prohibited);
Reddit	Access, search, or collect data from the Services by any means (automated or otherwise) except as permitted in these Terms or in a separate agreement with Reddit (we conditionally grant permission to crawl the Services in accordance with the parameters set forth in our robots.txt file, but scraping the Services without Reddit's prior written consent is prohibited);

Terms of Use and robots.txt

In both cases there is a robots.txt file and scraping/crawling is not always allowed.

Robots.txt files are used by websites' maintainers in order to *prohibit* the scraping of specific data. However, is not clear if it is legally binding or not. The "Robots Exclusion Protocol" state which information may be scraped on the web by robots. It does not follow a specific grammar. However, some **basic rules** apply.

Common fields are the User-Agent, Disallow, Allow and Crawl-delay. Two symbols are important: the **asterisk** (*) which allows to generalize the rules and the **slash** (/) which can be used to encompasses the entire website.

Terms of Use and robots.txt

The general ban (no scraping at all) can be formulated as follows:

```
User-agent: *
Disallow: /
```

If you want to prohibit the “WebBOT” to scrape your private folder you can specify:

```
User-agent: WebBOT
Disallow: /private/
```

If you want to allows only some sub-directories at a delay of 3 seconds, you can specify:

```
User-agent: WebBOT
Disallow: /private/
Allow: /private/shared/
Crawl-delay: 3
```

Terms of Use and robots.txt

Let us inspect the robots.txt files of Twitter, Reddit and Facebook:

X: https://twitter.com/robots.txt

Reddit: https://www.reddit.com/robots.txt

Facebook: https://www.facebook.com/robots.txt

Terms of Use and robots.txt

`get_robotstxt()` downloading robots.txt file

`get_robotstxt(domain,….)`

`domain` domain from which to download robots.txt file

`parse_robotstxt()` function parsing robots.txt

`parse_robotstxt(txt)`

`txt` content of the robots.txt file

The function returns a named list with useragents, comments, permissions, sitemap

Terms of Use and robots.txt

```
> library(robotstxt)
> rtxt_text <- get_robotstxt("http://www.facebook.com/") #to download the file in
text

> rtxt_parase <- parse_robotstxt(rtxt_text)
> names(rtxt_parase)
[1] "useragents" "comments" "permissions" "crawl_delay" "sitemap"
[6] "host" "other"

> table(rtxt_parase$permissions$useragent, rtxt_parase$permissions$field)
```

Terms of Use and robots.txt

```
> table(rtxt_parase$permissions$useragent, rtxt_parase$permissions$field)
                      Allow Disallow
*                         0        1
Amazonbot                 0        1
Applebot                  4       32
Applebot-Extended         0        1
Bingbot                   4       32
ClaudeBot                 0        1
Discordbot                4       32
DuckDuckBot               4       32
facebookexternalhit       4       32
Google-Extended           0        1
Google-InspectionTool     4       32
Googlebot                 4       32
Googlebot-Image           4       34
Googlebot-News            4       32
Googlebot-Video           4       32
GPTBot                    0        1
LinkedInBot               4       32
msnbot                    4       32
PerplexityBot             0        1
PetalBot                  0        1
```

Terms of Use and robots.txt

`robotstxt()` Generate a representations of a robots.txt file. The function generates a list that entails data resulting from parsing a robots.txt file as well as a function called check that enables to ask the representation if bot (or particular bots) are allowed to access a resource on the domain

```
robotstxt( domain,…)
```

Terms of Use and robots.txt

```
> rtxt_list <- robotstxt("http://www.facebook.com/")
> rtxt_list$bots
  [1] "Amazonbot"               "Applebot-Extended"       "ClaudeBot"
  [4] "Google-Extended"         "GPTBot"                  "PerplexityBot"
  [7] "PetalBot"                "uptimerobot"             "viberbot"
 [10] "YaK"                     "Yandex"                  "Yeti"
 [13] "Applebot"                "Bingbot"                 "Discordbot"
 [16] "DuckDuckBot"             "facebookexternalhit"     "Googlebot"
 [19] "Googlebot-Image"         "Googlebot-Video"         "Googlebot-News"
 [22] "Google-InspectionTool"   "LinkedInBot"             "msnbot"
 [25] "Pinterestbot"            "ScreamingFrogSEOSpider"  "seznambot"
 [28] "Slurp"                   "teoma"                   "TelegramBot"
 [31] "Twitterbot"              "*"

> rtxt_list$permissions
```

Terms of Use and robots.txt

```
> rtxt_list$permissions
```

	field	useragent	value
1	Disallow	Amazonbot	/
2	Disallow	Applebot-Extended	/
3	Disallow	ClaudeBot	/
4	Disallow	Google-Extended	/
5	Disallow	GPTBot	/
6	Disallow	PerplexityBot	/
7	Disallow	PetalBot	/
8	Disallow	uptimerobot	/
9	Disallow	viberbot	/
10	Disallow	YaK	/
11	Disallow	Yandex	/
12	Disallow	Yeti	/
13	Disallow	Applebot	/*/plugins/*
14	Disallow	Applebot	/?*next=
15	Disallow	Applebot	/a/bz?
16	Disallow	Applebot	/ajax/
17	Disallow	Applebot	/album.php
18	Disallow	Applebot	/checkpoint/
19	Disallow	Applebot	/contact_importer/

Terms of Use and robots.txt

```
> rtxt_list$check(paths = "/", bot = "*")
[1] FALSE

> # OR:
> paths_allowed("/", "https://facebook.com/", bot = "*")
https://facebook.com/
[1] FALSE
```

HTML basics

Definition: HTML (HyperText Markup Language) is not a programming language. It is a markup language which defines the structure of a web page.

```
<html>
  <head>
    <title>Page title</title>
  </head>
  <body>
    <h1>This is a heading</h1>
    <p>This is a paragraph.</p>
    <p>This is another paragraph.</p>
  </body>
</html>
```

Source:https://www.w3schools.com/html/html_intro.asp

HTML basics

It is used in combination with other instruments such as CSS or JavaScript which regulate the style and the functionality of a web page respectively.

Let's analyse the **HTML name**:
HyperText refers to the possibility of linking the content of multiple web pages.
Markup refers to the annotation of the content displayed.

In order to understand the structure of an HTML page, let's consider the following example:

HTML basics

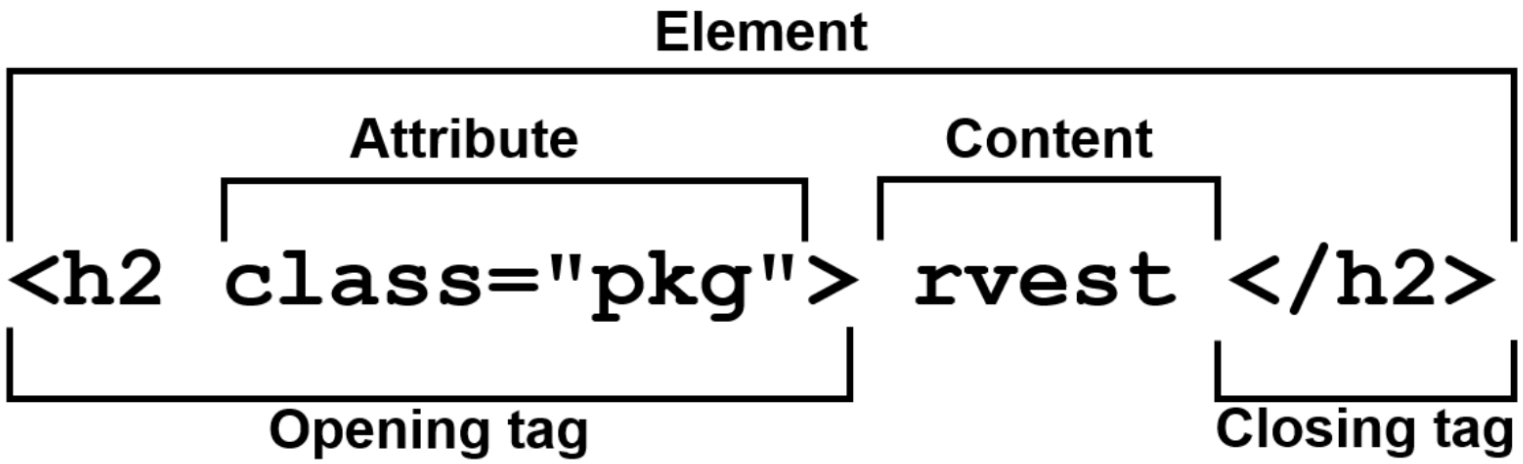
```
<!DOCTYPE html>
<html>

  <head>
    <title>Page title</title>
    <style>
     .myDiv {
        border: 5px outset red;
        background-color: lightblue;
        text-align: center;
      }
    </style>
  </head>

  <body>

    <div class="myDiv">
    <!-- ID is unique -->
      <h1 id="first">TM Packages</h1>
      <p>Some text &amp;amp; <b>some bold text.</b></p>
    </div>

    <div class="myDiv">
    <!-- Multiple elements with same class -->
      <h2 class="pkg">tidyverse</h2>
      <p> <b>tidyverse</b> is beautiful</p>

      <h2 class="pkg">rvest</h2>
      <p> I use <b>rvest</b> for <i>web scraping</i> </p>
    </div>

    </body>
</html>
```

As you can see, the structure of HTML is **hierarchical** (tree-structure).

First, we find the indication of the document type <!DOCTYPE html>. Then, everything is inside the <html> ... </html> element which is defined as the root element. The <head> ...</head> element contains the main information about the page (keywords, description, styles, etc.). Here you can also find the <title> ...</title> element which defines the title of the page. Then, all the rest is contained in the <body> ...</body> element. Here you can find headings, paragraphs and, basically, the content of a web page.

HTML basics

Thus, an HTML page is made up by several elements. Each element is composed by an **opening tag** which defines the name of the element (h for headings, p for paragraph etc.) and can contain optional *attributes* such as `class`, then there is the **content** and finally the **closing tag** which has the same name as the opening one as in the following figure.



Note that, the tags are *case-insensitive*, so they can be written in upper/lower case or in a mixed way.

HTML basics

HTML has several tags and a good resource for searching their meaning is the following website. The main tags we will consider in this course are:

- `<div> ...</div>`: which defines a section (division). It groups the content over lines and works with CSS;
- `<span> ...</span>`: It groups the content within lines and works with CSS. For example:

```
CSS definition:

div.redtext { color:red;
 font-family:"Times New Roman"}

span.redtext { color:red;
 font-family:"Times New Roman"}

In HTML:

<div class="redtext"><p>All these text  is  styled</p></div>
This is not styled <span class="redtext">but this is</span>
```

HTML basics

- `<p> ...</p>`: which indicates a paragraph;
- `<h#> ...</h#>`: which indicates an heading. There are 6 levels for headings, from the most important `<h1>` to the least one `<h6>`;
- `<a href=".."> ...</a>`: it is the anchor tag which links to other webpages or locations. In the case of a link to other webpages we have:

```
<a href="en.wikipedia.org/some_path"> Link to another webpage </a>
```

If the link points to a point in the same webpage, we have:

```
# Set the reference point:
<a id="intro"> Introduction </a>

# Link to the reference point

See the <a href=#intro> Introduction </a>
```

HTML basics

- `<ul>` and `<ol>`: are listing tags for unnumbered and numeric lists respectively. The list element is indicated with `<li>`. For example:

```
<p> This is a numeric list:</p>
   <ol>
    <li>Book1</li>
    <li>Book2</li>
    <li>Book3</li>
   </ol>
```

- `<table> ...</table>`: for indicating tables. For example this is a table with two columns and two rows.

```
<table>
  <tr>
    <th>Book</th>
    <th>Pages</th>
  </tr>
  <tr>
    <td>R4DS</td>
    <td>375</td>
  </tr>
</table>
```

Here, `<tr>` defines a table row, `<th>` the header and `<td>` the cell.

HTML basics

HTML has a large number of **attributes**. We will focus on the `class` a global attribute that specifies the class name of one or more elements and on the `id` global attribute which is unique for each element. These two attributes are used in combination with CSS. A comprehensive list of attributes can be found here.

Another attribute that we will encounter is the `data-hook` which allows compatibility with JavaScript and it can be used also as a CSS selector.

Then, in the example above we can also recognize the presence of **children**. For example, considering

```
<p> I use <b>rvest</b> for <i>web scraping</i> </p>,
```

you can see that the `<p>` element has two children, `<b>` and `<i>` which have content. They stand for bold and italics respectively.

HTML basics

When we browse webpages, the browser creates a Document Object Model (DOM) of he page. It is also called the DOM Tree due to its structure.
In order to access the DOM with the HTML code we are interested in, we can click with the right mouse button on the web page and then on “Inspect”. Let’s inspect the following page:

https://www.unibg.it/ateneo/chi-siamo/storia-e-identita

Exercise for you

HTML source code inspection.
(a) Open three webpages you frequently use in your browser.
(b) Have a look at the source code of all three.
(c) Inspect various elements with the Inspect Elements tool of your browser.