LAB # 2

Considering the huge amount of data available on the Web it seems to be essential to integrate **data sources**: **data banks**, **databases** and **Web sources**.

The Web as Data Source

The Web as a data source is getting more and more important for data scientists, even for official statistics. *An example of application is to collect prices related to goods and services contained in enterprises' websites using Web scraping techniques.*

Obtaining data from the Web implies to set up an extraction process, which typically is composed by four main different phases [1]:

Figure 1: **Web Information Extraction Process (Source [1])**

- **Crawling**: a Web crawler (also called Web spider or ant or robot) is a software program that systematically browses the Web **starting from an Internet address** (or a set of Internet addresses) and some pre-defined conditions (e.g., how many links navigate, the **depth**, types of files to ignore, etc.).
- **Scraping**: a scraper takes Web resources (documents, images, etc.), and engages a process for extracting data from those resources, finalized to data storage **for subsequent elaboration purposes**.
- **Indexing/Searching**: searching operations on a huge amount of data can be very slow, so it is necessary (through crawler) to index contents.
- Once these phases are completed it is possible to analyze the indexed contents in order to extract relevant information from the collected data and realize the analysis.

Crawling

The explosive growth of data available on the World Wide Web has made this the largest publicly accessible data-bank in the world, implying that web page content collection processes have been set up, the so called **Web crawling** [2].

1. Web crawlers are programs that automatically **browse** and **download** web pages by following **hyperlinks** in a methodical and **automated** manner.
2. It is the aim of the following experience with **RCrawling** library to implement the Web Crawling process. **RCrawler** is R based, **domain-specific**, and **multi-threaded**.
3. **The process**:
   (a) The crawler begins from a given website URL, provided by the user, and progressively fetches this and extracts new URLs (outlinks).
   (b) These in turn are added to the frontier list to be processed.
   (c) The crawling process stops when all URLs in the frontier are processed.
   (d) Crawling is performed by multiple worker **threads**.
   (e) The process of a crawling operation is performed by several concurrent processes or nodes in parallel.

```
library(Rcrawler)
ws <- "https://www.eataly.net/it_it/spesa-online/panetteria"
# Check current working directory
getwd()
# 4 cores and 4 parallel requests
# The crawler stops when it reaches the first level.

Rcrawler(Website=ws, no_cores=4, no_conn=4, MaxDepth=1)
```

Figure 2: An example of "crawling session"

Web Scraping

There is valuable data that is publicly available online, but seems to be locked away in web pages that are not amenable to data analysis [3].

Researchers are increasingly turning to these useful data sources. Nonetheless, they represent valuable data for important social science questions, marketing research, etc. **if we have a way to put them into a structured format**.

Fortunately, there are many tools available for translating unruly HTML into more structured data file.

The Process of Web scraping

There are essentially six steps to extracting text-based data from a web site:

1. Identify information on the Internet that you want to use.
2. If this information is stored on more than one web page, figure out how to automatically navigate to the web pages.
3. Locate the features on the web site that flag the data you want to extract. This means looking at the underlying HTML to find the elements you want and/or identifying some sort of pattern in the web site's text that you can exploit.
4. Write a script to extract, format, and save the data you want using the flags you identified.
5. Loop through all the web sites from step (2), applying the script to each of them.
6. Do some awesome analysis on your newly unlocked data!

The goal is to provide an introduction to the philosophy and basic implementation of Web scraping process using the open-source statistical programming language R, specifically **rvest** library.

One frequent task consists in collecting **food prices** from the Web, that is take data kept in HTML pages.

The main goal is to compare prices of regional foods, hence the process to set up consists in crawling e-commerce sites of selected sellers, scrape displayed data and store them in a structured format. Aiming to provide an example, the process starts from the web site:

`https://www.eataly.net/it_it/spesa-online/pasta-riso/pasta-fresca`,

which offers a list of few "pasta fresca" foods.

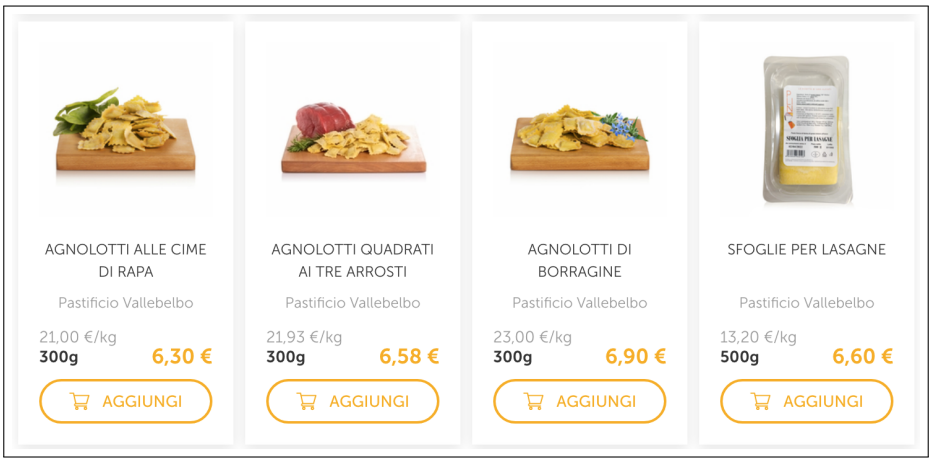

Figure 3: A sample of pasta fresca food products: `www.eataly.net`

The goal is to extract **product name**, **manufacturer**, **weight**, and **price** from the Web page, then store data into a structured **data set**.

---

1. Through the **Browser** show the underlying HTML document (Source View). It appears a very complex HTML page, and with a bit of patience and lucky **locate HTML part** that describe data you need. **Alternative you can use inspection tools offered by most of the browser applications.**

2. Typically requested data is enclosed in `<div>` or `<p>` tags, which represent a section in a document styled with CSS. A style is assigned to these sections by `class=` or `id` attributes. **The class is used to point to a class name in a style sheet.**

```
<div data-test-e2e="product-card-product-name" class="product-card-name" data-v-722596ee>
   <span data-v-722596ee>
       Agnolotti alle Cime di Rapa
   </span>
</div>
```

Figure 4: **A fragment of the Web Page**

3. Knowing that, it is an advantage to locate needed data by means of the `style class name` assigned to the `<div>` or `<p>` section.

Scraping process has been implemented in R environment [`scrap-pasta-script.R`].

```
page <- "https://www.eataly.net/it_it/spesa-online/pasta-riso/pasta-fresca"
library(rvest)
pasta.offer <- read_html(page)
```

Figure 5: **Locate the Web page and read it**

```
# Name
c.name <- c()
cards <- html_nodes(pasta.offer, ".product-card-name")
# for each card
for (c in cards){
   card.name <- html_text(c)
   card.name <- trimws(card.name, whitespace = "[ \n]")
   c.name <- c(c.name, card.name)
}
print(c.name)
```

Figure 6: **Locate the node `.product-card-name`, remove blanks and new line**

```r
# Manufacturer
c.manufacturer <- c()
cards <- html_nodes(pasta.offer, ".product-card-manufacturer")
# for each card
for (c in cards){
  card.manufacturer <- html_text(c)
  card.manufacturer <- trimws(card.manufacturer, whitespace = "[ \n]")
  c.manufacturer <- c(c.manufacturer, card.manufacturer)
}
print(c.manufacturer)
```

Figure 7: **Locate node `.product-card-manufacturer`, remove blanks and new line**

```r
# Price
c.price <- c()
cards <- html_nodes(pasta.offer, ".final-price")
# for each card
for (c in cards){
  card.price <- html_text(c)
  card.price <- gsub("[ €]", "", card.price)
  card.price <- gsub(",", ".", card.price)
  card.price <- as.numeric(card.price)
  c.price <- c(c.price, card.price)
}
print(c.price)
```

Figure 8: **Locate node `.final-price`, remove, replace characters, modify type**

```
# Info
c.info <- c()
cards <- html_nodes(pasta.offer, ".item-info")
# for each card
for (c in cards){
  card.info <- html_text(c)
  card.info <- strsplit(card.info, split=" ")
  c.info <- c(c.info, card.info[[1]][3])
}
print(c.info)
```

Figure 9: **Locate node `.item-info`, split the string, take the third item**

```
# The Data SET
df <- data.frame(product=c.name, weight=c.info, price=c.price, manufacturer=c.manufacturer)
print(df)
write.csv(df, file="pasta.csv")
```

Figure 10: **Arrange data into a data set, save it**

API as Data Source

OECD data (XML)

"Waste reduction has great benefits for the environment". According to OECD Environment at a Glance [2020] we compare the main indicators of treatment municipal waste: **% Landfilling**, **% Recycling**, **Total quantity of waste (Kg/hab)** in **each year**. Data is captured in a semi-structured data model (XML Language).

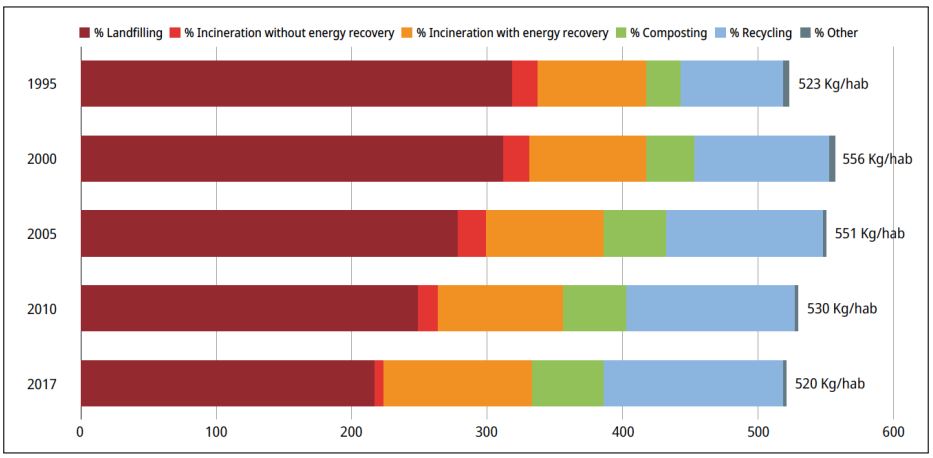

Figure 11: source: **Municipal Waste - OECD**

```
<municipal_waste>
   <time year="1995">
      <total unit="Kg/hab"> 523 </total>
      <percentages unit="%">
         <landfilling> 65 </landfilling>
         <recycling> 18 </recycling>
      </percentages>
   </time>
   <time year="2000">
      <total unit="Kg/hab"> 556 </total>
      <percentages unit="%">
         <landfilling> 62 </landfilling>
         <recycling> 23 </recycling>
      </percentages>
   </time>
</municipal_waste>
```

Use of OECD API

The Organization for Economic Cooperation and Development (OECD) exposes an Application Programming Interfaces (API) on its web site, that provides access to datasets in the **catalogue of OECD data section**.

The API allows you to query the data in several ways, using parameters to specify your request so that you can create innovative software applications which use OECD datasets. The API is available in **JSON** and **XML formats**.

R software includes the package 'OECD' which offers few functions (commands) to search and extract data through the OECD's API, exposed at the URL `https://stats.oecd.org`.

- Get a data frame with information on all available datasets:

  ```
  get_datasets()
  ```

  Returns a data frame with two variables: **id** and **description**.

- Search codes and descriptions of available OECD series:

  ```
  search_dataset(string, data=get_datasets(), ignore.case=TRUE)
  ```

  Returns a data frame containing the series codes and descriptions for the OECD series which match the given criteria [**string=a regular expression string to search for**].

- Browse the metadata related to a series:

  ```
  browse_metadata(dataset)
  ```

  Opens a web page in the default web browser.

- Download OECD data sets:

  ```
  get_dataset(
      dataset,
      filter=NULL,
      start_time=NULL,
      end_time=NULL,
      pre_formatted=FALSE)
      ...
  ```

  Returns a data frame with the requested data, downloaded through the OECD's API.

R Session [`API-script.R`]

1. Search for datasets over the topic "waste";
2. In the returned list identify the dataset "food waste";
3. Display the metadata of the dataset "food waste';
4. Get data: a selection of observations from the dataset "food waste". Use of the filter on **LOCATION** and on **years 2005-2010**;
5. Get the whole dataset "food waste". Use of `subset()` command to refine data: selection of observations on **LOCATION** and on **years 2005-2010**, projection on the variables **LOCATION**, **ObsValue**, **UNIT**, **Time**, **ACT**.
6. Get the whole data set "food waste". Use of **SQL SELECT** statement to refine data (selection and projection).

References

This Lab is mainly based on:

(1) Barcaroli G., Scannapieco M., Summa D. “On the use of Internet as a Data Source for official statistics: a strategy for identifying enterprises on the web” - 2016

(2) Khalil S., Fakir M. “An R package for parallel web crawling and scraping” - Elsevier, 2017

(3) Marble W. “Web Scraping with R” - 2016

(3) R packages: CRAN - “Package OECD” - 2022