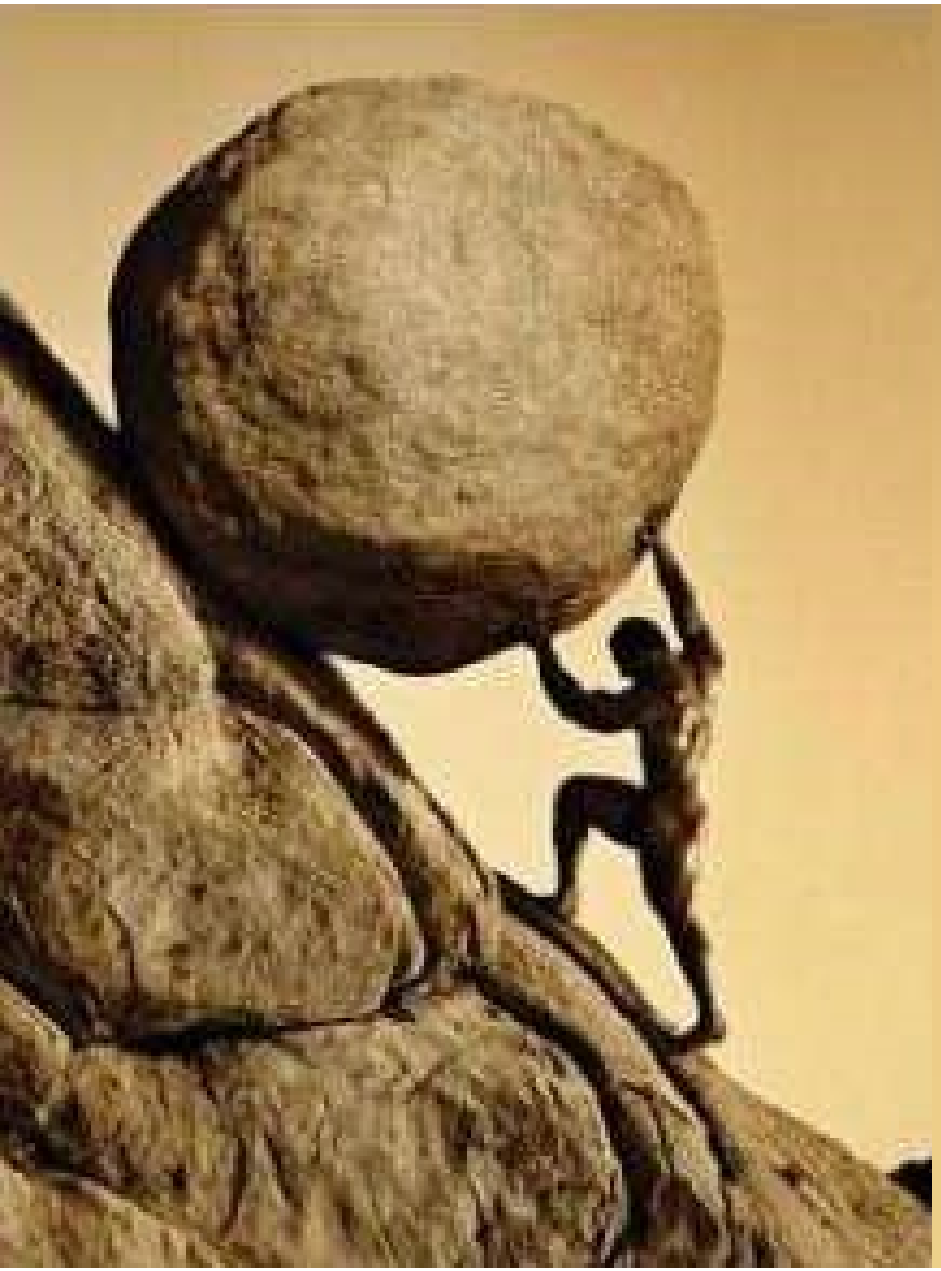


PROGRAMA DE PÓS-GRADUAÇÃO EM INFORMÁTICA
UNIVERSIDADE FEDERAL DO RIO DE JANEIRO

Introduction to Data Science

MODULE II – PART I

Data Scraping

Prof Sergio Serra e Jorge Zavaleta

What? The Data Science Process

Ask an interesting question & learn reproducibility

Get the Data

Explore the Data

Model the Data

Communicate/Visualize the Results

How were the data sampled?

Which data are relevant?

Are there privacy issues?

Module II

Where do data come from?

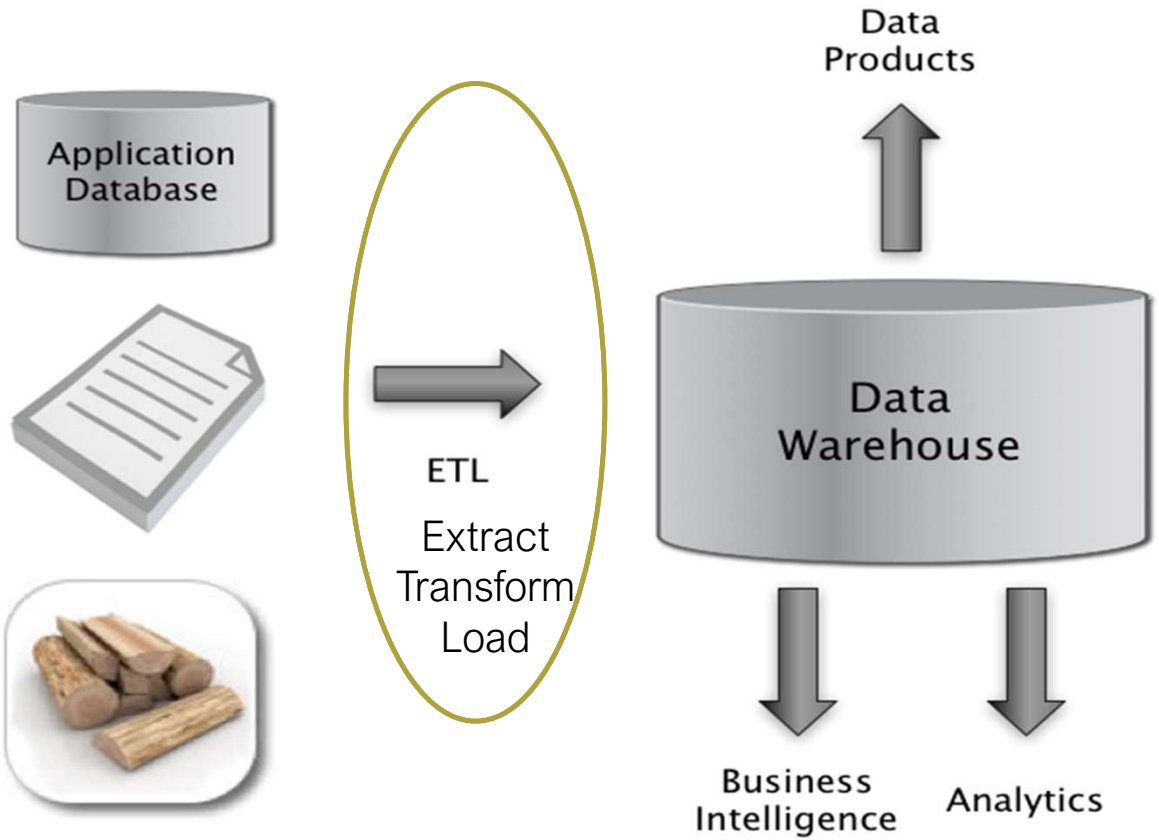
- Internal sources: already collected by or is part of the overall data collection of your organization.
  - For example: business-centric data available in the organization DB to record day to day operations; scientific or experimental data get from an essay.
- Existing External Sources: available in ready to read format from an outside source for free or for a fee.
  - For example: public government databases, stock market data, sports, COVID-19.
- External Sources Requiring Collection Efforts: available from external source but acquisition requires special processing.
  - For example: data appearing only in print form, or data on websites.

Ways to gather online data

How to get data generated, published or hosted online?

- API (Application Programming Interface): Using a pre-builtin set of functions developed by a company to access their services. Often pay to use.
  - For example: Google Map API, Facebook API, Twitter API
- RSS (Rich Site Summary): summarizes frequently updated online content in standard format. Free to read if the site has one.
  - For example: news-related sites, blogs
- Web scraping (crawling): using software, scripts or by-hand extracting data from what is displayed on a page or what is contained in the HTML file (often in tables).

Good old days…



Web scraping

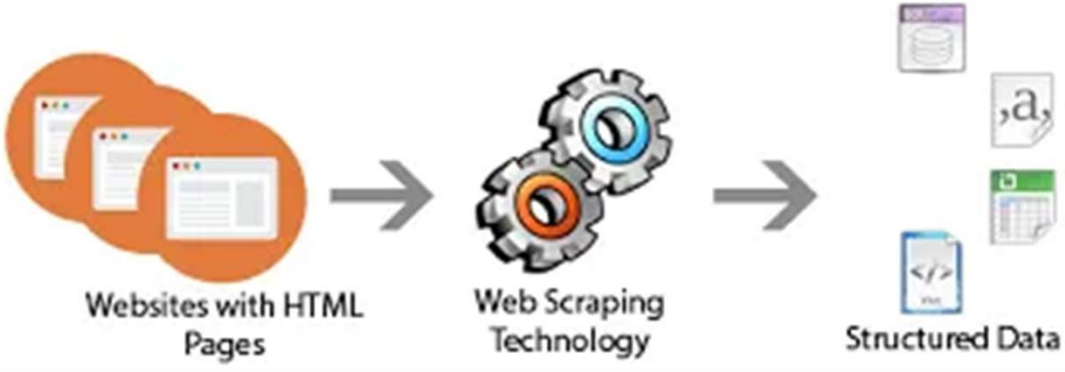
- Why do it?
  - Older government or smaller news sites might not have APIs for accessing data
  - Publish RSS feeds or have databases for download.
  - You don't have $$ to pay to use the API or API provided is rate limited.
  - Monitor a news site for trending new stories on a particular topic of interest
  - Do social network analytics using profile data found on a web forum.

Web scraping

- Who should you do it?
  - You just want to explore:
    - Are you violating their terms of service?
    - Privacy concerns for website and their clients?
  - You want to publish your analysis or product:
    - Do they have an API or fee that you are bypassing?
    - Are they willing to share this data?
    - Are you violating their terms of service?
    - Are there privacy concerns?
- How do you do it?

Web scraping

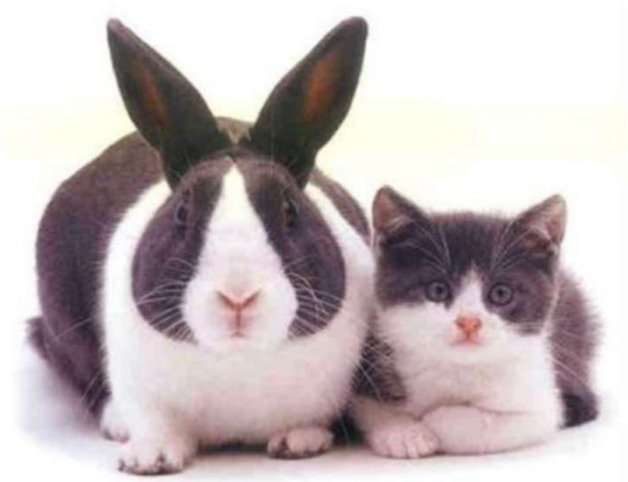
- Using python (or R) programs to get data from online
- Often much faster than manually copying data!
- Transfer the data into a form that is compatible with your code
- Legal and moral issues



Warning: Web scraping

Tips:

- Vast source of information; can combine with multiple datasets
- Automate repetitive tasks
- Keep up with sites / real-time data
- Be careful and polite (don't hit a server too often)
- Be Robust and immune to spider traps and other malicious behavior from web servers
- Give proper credit!
- Care about media law / obey licenses / privacy
- Do not be evil (no spam, overloading sites, etc)
- Do not forget data provenance!



It is Fun!

Obtaining Data: Web scraping

Robots.txt

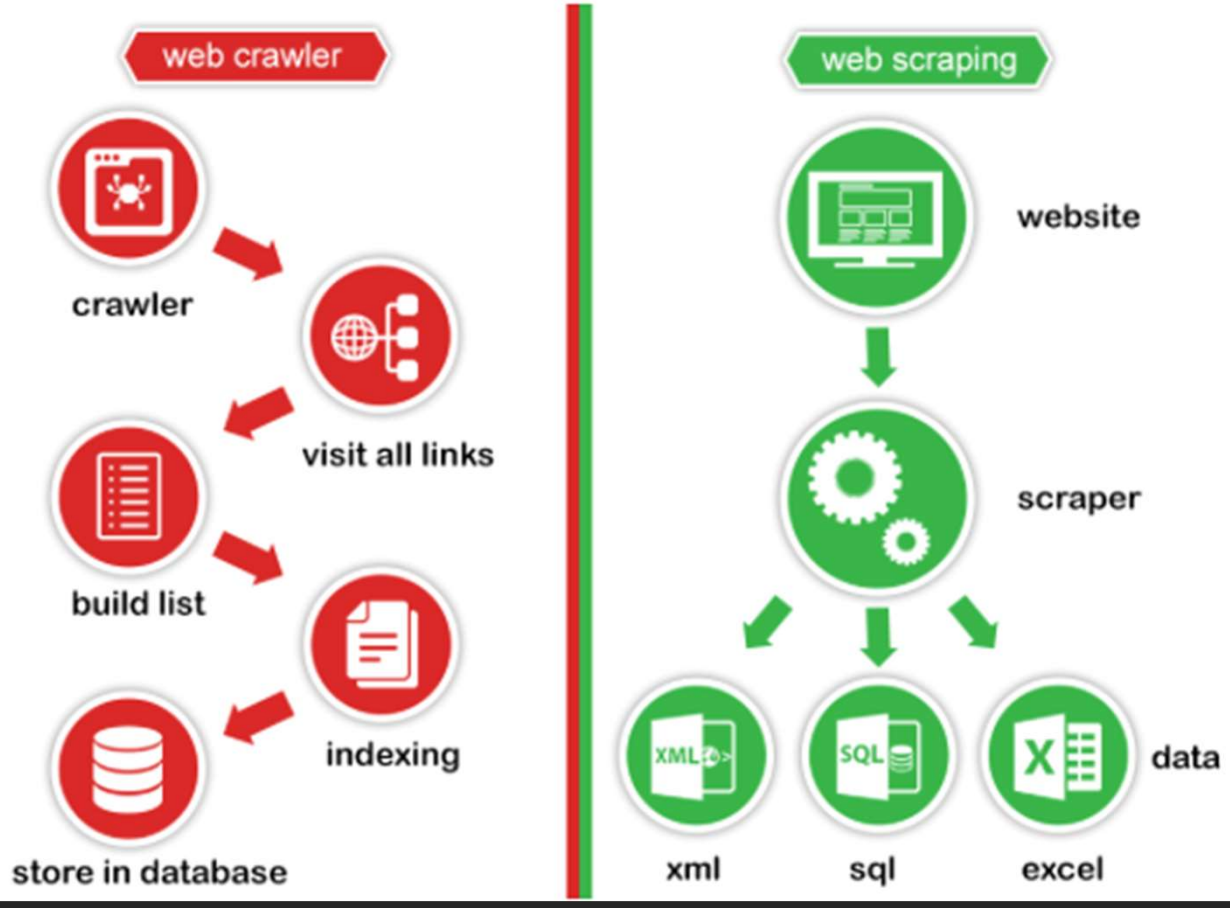
Protocol for giving spiders (“robots”) limited access to a website, originally from 1994
www.robotstxt.org/wc/norobots.html

Website announces its request on what can(not) be crawled

- Specified (access restrictions) by web site owner
- Gives instructions to web robots (e.g., your code)
- Located at the top-level directory of the web server
- E.g., http://google.com/robots.txt

Master class about → Spiders, crawlers, harvesters, bots
https://clgiles.ist.psu.edu/IST441/materials/powerpoint/crawl-week4/crawlers.pptx

Web Crawler x web scraper



Hands on...

NOTEBOOK:

BEAUTIFULSOAP

Remember….Web Servers

- A server maintains a long-running process (also called a daemon), which listens on a pre-specified port
- It responds to requests, which is sent using a protocol called HTTP (HTTPS is secure)
- Our browser sends these requests and downloads the content, then displays it.
    - Examples: request was successful, client error, often `page not found`; server error (often that your request was incorrectly formed)

Adaptado de:
Harvard IACS
CS109A
Pavlos Protopapas, Kevin Rader, and Chris Tanner

Obtaining Data: Web scraping

Remember…. HTML

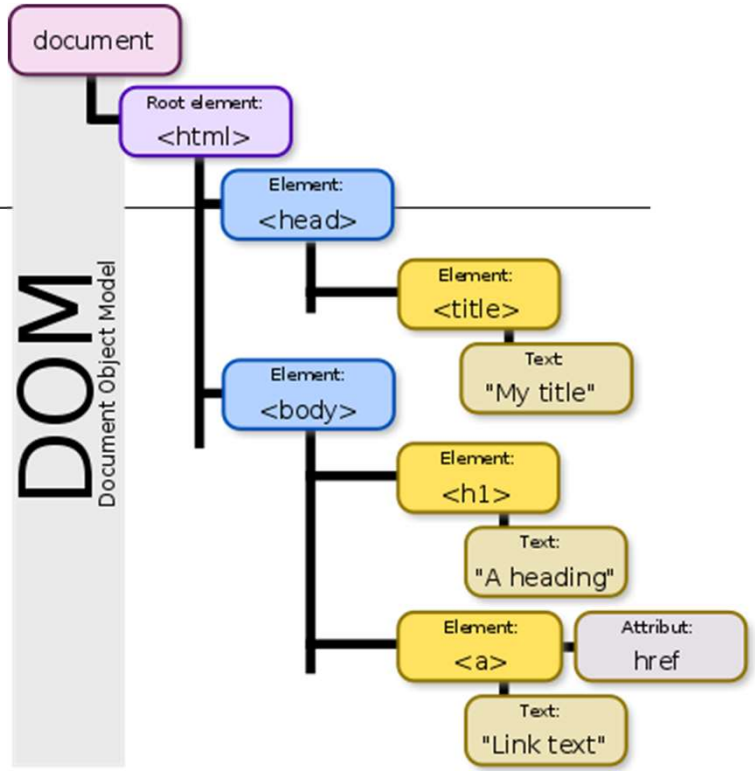
- Tags are denoted by angled brackets
- Almost all tags are in pairs e.g., Hello
- Some tags do not have a closing tag e.g.,

Example

```
<!DOCTYPE html>
<html>
  <head>
    <title>Ttle</title>
  </head>
  <body>
    <h1>Body Title</h1>
    <p>Body Content</p>
  </body>
</html>
```

Adaptado de:
Harvard IACS
CS109A
Pavlos Protopapas, Kevin Rader, and Chris Tanner

Remember…. HTML



- **<html>,** indicates the start of an html page
- **<body>,** contains the items on the actual webpage (text, links, images, etc)
- **<p>,** the paragraph tag. Can contain text and links
- **<a>,** the link tag. Contains a link url, and possibly a description of the link
- **<input>,** a form input tag. Used for text boxes, and other user input
- **<form>,** a form start tag, to indicate the start of a form
- **<img>,** an image tag containing the link to an image

Adaptado de:
Harvard IACS
CS109A
Pavlos Protopapas, Kevin Rader, and Chris Tanner

Obtaining Data: Web scraping

How to Web scrape:

1. **Get** the webpage content

   - Requests (Python library) **gets** a webpage for you

2. **Parse** the webpage content

   - (e.g., find all the text or all the links on a page)
   - BeautifulSoup (Python library) helps you **parse** the webpage.
   - Documentation: http://crummy.com/software/BeautifulSoup

Adaptado de:
Harvard IACS
CS109A
Pavlos Protopapas, Kevin Rader, and Chris Tanner

The Big Picture Recap

Data Sources	Files, APIs, Webpages (via `Requests`)
Data Parsing	Regular Expressions, Beautiful Soup
Data Structures/Storage	Traditional lists/dictionaries, PANDAS
Models	Linear Regression, Logistic Regression, kNN, etc

BeautifulSoup only concerns webpage data

Obtaining Data: Web scraping

1. **Get** the webpage content

Requests (Python library) **gets** a webpage for you

```
page = requests.get(url)
page.status_code
page.content
```

Adaptado de:
Harvard IACS
CS109A
Pavlos Protopapas, Kevin Rader, and Chris Tanner

Obtaining Data: Web scraping

1. **Get** the webpage content

Requests (Python library) **gets** a webpage for you

```
page = requests.get(url)
page.status_code
page.content
```

Gets the status from the webpage request.

200 means success.

404 means page not found.

Obtaining Data: Web scraping

1. **Get** the webpage content

Requests (Python library) **gets** a webpage for you!

```
page = requests.get(url)
page.status_code
page.content
```

Returns the content of the response, in bytes.

Adaptado de:
Harvard IACS
CS109A
Pavlos Protopapas, Kevin Rader, and Chris Tanner

2. **Parse** the webpage content

BeautifulSoup (Python library) helps you **parse a webpage** and, makes messy HTML digestible

```
soup = BeautifulSoup(page.content, "html.parser")
soup.title
soup.title.text
```

Adaptado de:
Harvard IACS
CS109A
Pavlos Protopapas, Kevin Rader, and Chris Tanner

2. **Parse** the webpage content

BeautifulSoup (Python library) helps you **parse** a webpage

```
soup = BeautifulSoup(page.content, "html.parser")
soup.title
soup.title.text
```

Returns the full context, including the title tag.

e.g.,

```
<title> UFRJ </title>
```

2. **Parse** the webpage content

BeautifulSoup (Python library) helps you **parse** a webpage

```
soup = BeautifulSoup(page.content, "html.parser")
soup.title
soup.title.text
```

Returns the text part of the title tag. e.g.,

UFRJ

Adaptado de:
Harvard IACS
CS109A
Pavlos Protopapas, Kevin Rader, and Chris Tanner

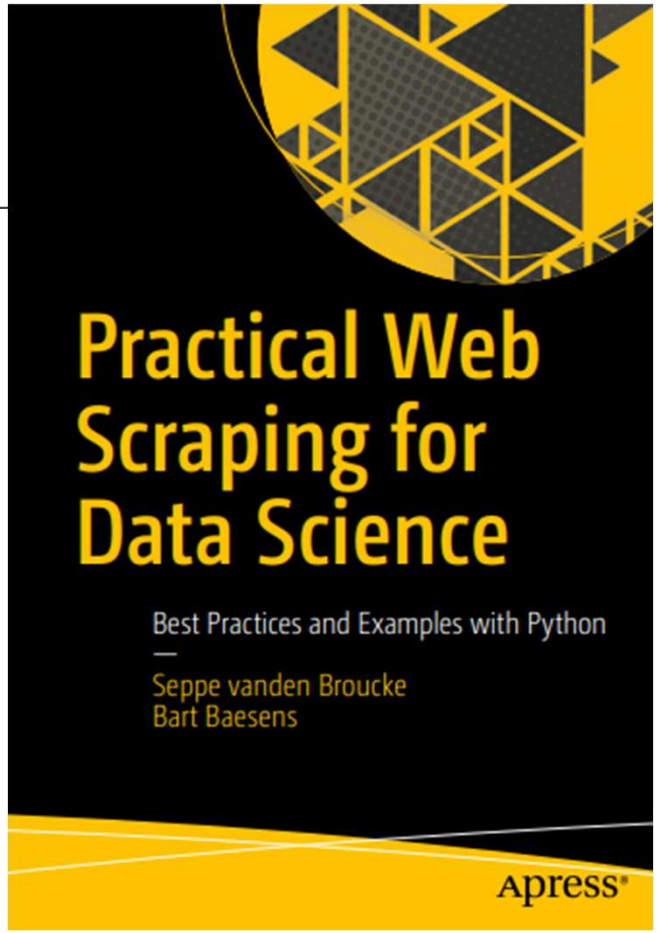
BeautifulSoup

- Provides functions for quickly accessing certain sections of HTML content

Example

```
import bs4
## get bs4 object
soup = bs4.BeautifulSoup(source)
## all a tags
soup.findAll('a')
## first a
soup.find('a')
## get all links in the page
link_list = [l.get('href') for l in soup.findAll('a')]
```

References



Beautiful Soup Documentation — Beautiful Soup 4.9.0 documentation (crummy.com)