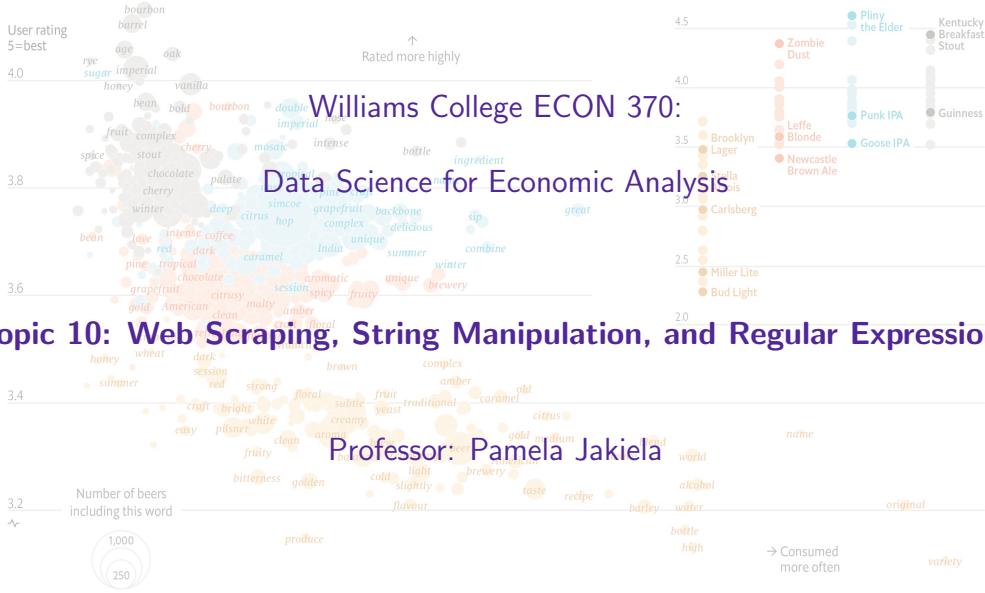


Williams College ECON 370:

Data Science for Economic Analysis

Topic 10: Web Scraping, String Manipulation, and Regular Expressions

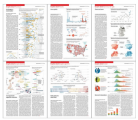
Professor: Pamela Jakiela

Outline

- Web scraping
- String variables and regular expressions
- Scraping the Williams website

Web Pages Are Built in **H**yper**T**ext **M**arkup **L**anguage (HTML)

ECON 370



source: The Economist

Instructor:
Pamela Jakiela

home
syllabus
schedule

Data Science for Economics

This is the website for Professor Pamela Jakiela's ECON 370 course at Williams College, Data Science for Economic Analysis.

Course Description:

This course provides a hands-on introduction to data science tools most relevant for economic analysis including data visualization, machine learning, and text analysis. Economists and other social scientists tend to use these data science tools differently than many researchers in statistics and computer science – conducting empirical analysis that is explicitly grounded in economic theory, and focusing on causal inference rather than prediction. Through a combination of lectures, hands-on labs, and group projects, students will develop the theoretical and practical skills needed to analyze economic data using modern data science techniques in R or Python.

Course Information:

Syllabus

Schedule

Textbooks

Projects:

Data Visualization Project

```
<!DOCTYPE html>
<html lang="en-US">
  <head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1">

<!-- Begin Jekyll SEO tag v2.8.0 -->
<title>Data Science for Economics | ECON 370</title>
<meta name="generator" content="Jekyll v3.10.0" />
<meta property="og:title" content="Data Science for Economics" />
<meta property="og:locale" content="en_US" />
<meta name="description" content="Course materials for data science for economic analysis" />
<meta property="og:description" content="Course materials for data science for economic analysis" />
<link rel="canonical" href="https://pjakiela.github.io/ECON370/" />
<meta property="og:url" content="https://pjakiela.github.io/ECON370/" />
<meta property="og:site_name" content="ECON 370" />
<meta property="og:type" content="website" />
<meta name="twitter:card" content="summary" />
<meta property="twitter:title" content="Data Science for Economics" />
<script type="application/ld+json">
{"@context":"https://schema.org","@type":"WebSite","description":"Course materials for data science for economic
analysis","headline":"Data Science for Economics","name":"ECON 370","publisher":{"@type":"Organization","logo":
{"@type":"ImageObject","url":"https://pjakiela.github.io/ECON370/economist-data-crop-v2.jpg"}},"url":"https://
pjakiela.github.io/ECON370/"}</script>
<!-- End Jekyll SEO tag -->

    <link rel="stylesheet" href="/ECON370/assets/css/style.css?v=a3f172edcf2af85e5418b475cb5c99920484cafd">
    <!--[if lt IE 9]>
    <script src="https://cdnjs.cloudflare.com/ajax/libs/html5shiv/3.7.3/html5shiv.min.js"></script>
    <![endif]-->
  </head>
  <body>
    <div class="wrapper">
      <header>
        <h1><div style="text-align: right"><a href="https://pjakiela.github.io/ECON370/">ECON 370</a></div></h1>


          <img src="/ECON370/economist-data-crop-v2.jpg" alt="Logo" />


        <div style="text-align: right"><small>source:  The Economist </small></div>

        <p><div style="text-align: right">Instructor: </div>
        <div style="text-align: right"><a href="https://pjakiela.github.io/">Pamela Jakiela </a></div>
        </p>

        <p><div style="text-align: right"><a href="https://pjakiela.github.io/ECON370/">home </a></div>
        <div style="text-align: right"><a href="https://pjakiela.github.io/ECON370/ECON370-
syllabus-2024-09-11.pdf">syllabus </a></div>
```

Elements and Attributes

A web page is made up of **elements** that are arranged in a hierarchical structure

- Elements start and end with a **tag**: for example, `<title>ECON 370</title>`
  - Every html page contains an html element (`<html>...</html>`)
  - The html element contains the elements (children) **head** and **body**
- Many elements appear multiple times within a page: for example, `<p>a paragraph</p>`
- Tags can contain **attributes** that encode element-specific information, for example:
  - `<h1 id="data-science-for-economics">Data Science for Economics</h1>`
  - `<p><a href="https://pjakiela.github.io/syllabus.pdf">Syllabus</a></p>`
- Sometimes data is stored as a table within the page: `<table>...</table>`

When to Scrape

Simple web scraping tools are useful when:

- Data is stored in a table
- A page contains many repetitions of the same structure/sequence of elements, and you want to combine those elements into a data frame for analysis

  Example: `<h4>ECON 105(F) SEM Gender in the Global Economy</h4><h4>ECON 107 SEM Inequality in a Classless Society: The Soviet Experiment and its Aftermath</h4>`
- A sequence of (ideally numerically indexed) pages use the same structure and elements, and you want to combine the information from multiple pages to build a data set
  - **Example:** building a data set of all recent NBER working papers

How to Scrape: Get the HTML, Extract Text and Attributes

1. Do the actual scraping: load HTML into R or Python
2. Extract elements and the text and attributes they contain:
   - Extract elements with tag "a": `html_elements(mypage, "a")` or `mypage.select("a")`
   - Extract elements with `class` equal to a with `(".a")`
   - Extract elements with `id` equal to a with `("#a")`
   - Extraction tasks can be sequenced (first extract parents, then specific children)
     - SelectorGadget can (sometimes) make this process easier
   - Final step is to extract text (printed on web page) or the value of an attribute

How to Scrape: Example

`https://ephsports.williams.edu/sports/womens-soccer/roster`

When Not to Scrape

Many websites cannot be scraped easily with simple tools (can you do it?)

- Policy advice: don't try to scrape websites that don't want to be scraped

Web scraping raises a range of ethical issues (should you do it?)

- Some personally-identifiable information posted on the web should not be used for research
  - In general, work products and things shared under creative comments licenses are in-bounds, personal (non-professional) content posted with some expectation of privacy is out-of bounds
  - When collecting personally identifiable information for research, check with your IRB
  - Collecting personally-identifiable information is often unnecessary
- Also: don't violate a websites terms of service (if you are bound by them) or copyright law
- Finally, be polite: to avoid over-burdening a server, always build in pauses between queries

String Variables

A **string variable** is a variable that is stored as a sequence of characters

- Categorical variables can be stored as strings or as labeled/indexed numeric variables
  - Relatively small range of possible valid values
  - Multiple observations with the same value
- Text data that does not represent a (small-ish) set of mutually exclusive categories
  - Examples: names, phone numbers, open-ended responses

The distinction between text variables and categorical variables can be subtle, context-specific

- "What is your name?" vs. "Who is your favorite Beatle?"
- "Pamela Jakiela" vs. "Ann Miller"

String Manipulation

- Split them into multiple strings
  - Split `name` into `firstname` and `lastname` using the **delimiter** ", "
    - "Harrison, George" → "Harrison" + "George"
- Combine multiple strings
  - Combine `lastname` and `firstname` into `fullname` using the **delimiter** " "
    - "Harrison" + "George" → "George Harrison"
- Detect or count (substring) matches
  - Does a phone number contain the area code 413?
  - How many items did a customer purchase?
- Replace substrings

String Variables

How might we answer: does a phone number contain the area code 413?

- Split `phone_number` into area code, the rest of the phone number
  - A good approach when generating a categorical variable for metropolitan area of residence, but maybe not if you just want to know if someone is from western Massachusetts
- Check whether `phone_number` contains the substring "413"
  - Consider the phone number: 510-408-1413

We actually want to know if `phone_number` starts with 413

Regular Expressions

A **regular expression** or **regex** is a string combining (normal) characters and **metacharacters** that can be parsed and then used to identify and manipulate pattern matches in text data

- Example: "^413" means starts with 413

Regular expression tools are useful for:

- Expanding the scope of pattern matching
  - "`[Pp]rofessor [Jj]akiela`", "`[0-9]`", "`[0-9]*`", "`[0-9]+`"
- Restricting the scope of pattern matching
  - "`^Professor`", "`Jakiela$`", "`[a-z]{2,}ing`"

To match a character that is also a metacharacter (e.g. "$"), use a slash: "\$"

- To match a "\", you need to use "\\"

Lab #10

Lab #10 can be completed in groups of up to 4 people, one submission per group

- Your group needs to write a script that scrapes data from Williams websites and processes it in order to answer (up to) 4 questions about sports teams and academic departments

The `ECON370-web-williams-soccer-example` scripts illustrate how to scrape a data table or other specific elements from simple web pages in either R (`rvest`) or Python (`BeautifulSoup`)

- Adapt the script to scrape and process data using these libraries
- Feel free to use SelectorGadget to identify the appropriate selectors