Tutorial 7: Indian PMs Scraping Example*

INF312: Worlds Become Data - Prof. Rohan Alexander

Luca Carnegie

February 27, 2024

Please redo the web scraping example, but for one of: Australia, Canada, India, or New Zealand. Use Quarto, and include an appropriate title, author, date, link to a GitHub repo, and citations. Submit a PDF.

My Choice - India.

1 Simulate Data

Our goal is a table that looks similar to this:

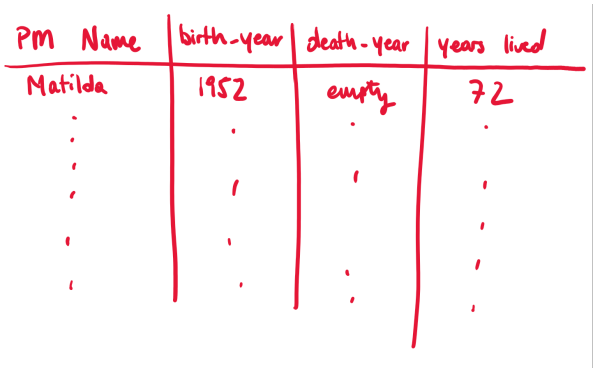

Figure 1: The Goal

Moving into the R environment, we are aiming for a table containing the Prime Minister's name, birth year, death year (if they are dead), as well as lifespan (in years). If they are not dead, the death year is empty. To simulate our desired outcome, we use similar methods to the textbook but adapting them to the different structure of the wikipedia table.

*Code available at: https://github.com/lcarnegie/INF312/tree/main/W7%20-%20Gather%20Data

This is the identical goalpost that was aimed for when looking at UK PMs in the textbook.

2 The Table

Table 1: Lifespan Chart of Indian Prime Ministers

Name	Born	Died	Age at Death
Jawaharlal Nehru	1889	1964	75
Gulzarilal Nanda	1898	1998	100
Lal Bahadur Shastri	1904	1966	62
Morarji Desai	1896	1995	99
Charan Singh	1902	1987	85
Indira Gandhi	1917	1984	67
Rajiv Gandhi	1944	1991	47
Vishwanath Pratap Singh	1931	2008	77
Chandra Shekhar	1927	2007	80
P. V. Narasimha Rao	1921	2004	83
Atal Bihari Vajpayee	1924	2018	94
H. D. Deve Gowda	1933	NA	NA
Inder Kumar Gujral	1919	2012	93
Manmohan Singh	1932	NA	NA
Narendra Modi	1950	NA	NA

All data was entirely scraped from Wikipedia's list of Indian Prime Ministers. The original table is quite extensive, not only listing personal information about each PM but also outlining their term, mandate, ministerial offices held and political parties. However, for our goal much of this information is irrelevant.

Once the table was scraped, the data of focus was parsed and cleaned using a similar approach to the one used in the textbook. The main difference in this exercise was the different regex patterns that needed to be changed in order to extract the correct data. After parsing and cleaning, the data were put into a kable table, displayed above.

3 Reflections

Understanding the new scraping function and exactly what each of them did individually was probably the most challenging part. There were a lot of functions from libraries that were not explicitly mentioned in the textbook (or perhaps not mentioned in the British PM example and still used) that needed to have their libraries retroactively installed.

The most satisfying part of this web scraping project was definitely getting the raw data from Wikipedia properly parsed and structured before generating my analysis. Scraping the Wikipedia pages for each Indian Prime Minister to extract details like their birth and death years was tedious, but necessary to construct the timeline. Cleaning the scraped data required debugging to handle inconsistencies - some pages listed precise dates while others just had years, which made writing regex patterns quite complex and cumbersome. After spending a full day writing custom parsers to extract and format the name, birth year, death year, and computed lifespan into a clean pandas DataFrame, it was incredibly rewarding to see the table filled with neatly organized data.

If I were to do a similar data scraping and cleaning project again in the future, there are three main things I would do differently next time to streamline the process:

1. Learn how to properly use the RStudio debugger. Having the ability to step through my code line-by-line in a debugger when errors occur would save so much time troubleshooting bugs compared to just using print statements. Proper use of breakpoints would also help identify issues faster.

2. Make even more use of large language models like GPT-4 for explanations. Whenever I was confused about R function arguments or returned objects, I could have simply asked GPT-4 for a plain English definition which would likely be faster than digging through documentation. This would accelerate my learning and cut down research time.

3. Brush up on regular expressions and use GPT-4 to help construct the right regex patterns. I spent way too long trying out slightly different regex to extract substrings from the messy HTML. With a stronger understanding of regex syntax, and GPT-4's help composing the expressions, I could have parsed the data much more efficiently.

I now know what areas I need to shore up on for scraping projects ahead. The work of getting usable data is challenging but so valuable once the data is clean.