CS636 - Data Analytics with R

Project 1

Group Members - Rahul Patel, Will Aponte, Satyapriya Yellamelli

Topic: DNA research from Oxford Academic

File Names:

1) Rcrawler.r
2) Plaintext.r
3) get_features.r
4) get_year_links.r

Rcrawler.r **: WARNING: DO NOT WEBSCRAPE MORE THAN 2 YEARS PER HOUR**

In this file, we created an encapsulated version of our web scraper to extract the content from the webpages on https://academic.oup.com/journals and retrieve the underlying html pages we needed. The libraries utilized here are:

1) rvest
2) stringr

The functions we created in order to do so are:

1) getYearLinks :
    a) Extracts the publication years 1994:2021 from the source and attaches the associated article URLs to each year. We also extract the necessary html pages. Here is where the user will input the year of publication and if the year is not within the range 1994:2021, an error will be thrown, stating, “Year out of Bound”.

This the function where we begin to extract information from the underlying html pages. Whatever data we extracted, and in this case are DOI, Title, Authors, Author Affiliations, Correspondence Author, Correspondence Author's Email, Publishing Date, Abstract, Keywords, Full Paper (Text format). The data is then stored into a dataframe ogDf, where it's subsets are returned output.

Note: Our sources did not include some information. We were unable to extract Author Affiliation information, Correspondence Author, Correspondence Author email due to its lack of existence on the website itself. These columns will be returning "NO" in response to this.

2) Functions within for loop:
    a) getTitle
        i) Here, in this function, the input taken here is the html_page and we extract the raw title from the corresponding html nodes, trim the extracted text and return the title of the articles.
    b) getAuthors
        i) Here, this function will input the html_page and we extract the Author names. The html node used here is "a.linked-name".
    c) getPubDate
        i) Here, this function will input the html_page and we extract the publication dates for the articles in the respective year. The html node used here is "div.citation-date".
    d) getDOI

i) Here, this function will input the html_page and we extract the DOI or digital object identifier, which is a string of numbers, letters and symbols used to permanently identify an article or document. Here it is used for citations in the extracted articles.

e) getKeywords
   i) Here, this function will input the html_page, to extract the identifiable keywords from each article. The html node utilized is “div.kwd-group” and returns the text for the article’s keywords.

f) getAbstract
   i) Here the function will input the html_page, and use the node “p.chapter-para”, to extract the abstract from each article.

g) getFullText
   i) Here, this function will input the html_page, to extract the entirety of the text for the articles, given year.

h) getYear
   i) This is the new master function after web scraping is complete and retrieve data locally. Otherwise it uses getYearLinks to scrape the data in case that it is not present locally.

PlainText.r

Here, we take the extracted CSV files for the articles and convert them into .txt files.

get_features.r

This is a related r script which was encapsulated in Rcrawler. It includes the multiple functions, mentioned above, associated with the related fields. It will extract the information based on the indicated link of the publication/article.

This is a related r script which was also encapsulated in Rcrawler. It will take the input from the user as getYearLinks("year") and extract the urls mentioned about the article of the corresponding year. It will return n number of urls based on the count of the available links.This is a related r script which was encapsulated in Rcrawler. It includes the multiple functions, mentioned above, associated with the related fields. It will extract the information based on the indicated link of the publication/article.

Issues:

1) It was mentioned to use the Rcrawler library, so our first attempt to collect all HTML pages from https://academic.oup.com/journals crashed the server and it took a very long time to scrape. We realized that we could not use this package in a function to get all the html pages, therefore we decided to use rvest instead.
2) Once we began using rvest we still ran into issues when web scraping. Our initial issue was that we were requesting an html page for each time we wanted to get article data. For example, we'd make our first request to the URL to get the title, then another request to get the Authors, then another for the Publication Date, etc. We resolved this issue by simply requesting the URL once and storing it as an object using read_html(url). Then using rvest functions and piping, we created our own functions for each specified datapoint to find the data in the HTML object using CSS selectors.(From William: I did not know what CSS selectors were prior to this project but my group partners were able

to explain it to me to the point I was able to create the functions needed to get the desired datapoints in extractURL function)

3) While developing these functions we of course ran into errors regarding typos, incorrect CSS selectors, crashing the server, and roadblocks in developing functions, but as a group we worked together to resolve these issues.
4) We also ran into issues with division of labor. Our schedules were all different than one another with work and school. We were only able to really collaborate on weekends and we'd all try our best to get some work done during the week. In our next project we'd like to improve the division of labor and working more as a team
5) The website related to our topic, has multiple issues for each year, we initially ran into a problem looping through all of the issues and concatenating it to the database for each year.
6) We kept on getting restricted by the website's server whenever, all 3 of us were running the code or sending requests. With that, the cooldown period for the server isn't mentioned anywhere and to see when any of us would regain access caused many delays.

How to Run RCrawler.r:

1) Go to the bottom of code and change year for getYearLinks to web scrape the articles for that year. Change getYear after scrapping.

Contributions:

1) Rahul- Worked on the function, *getYearLinks* to extract the required URLs for each article based on the year which is input by the user. Contributed on getting the crawled

web pages corresponding to each year. I helped debug any issues on the functions related to this project.

2) William- Worked on functions (`getTitle, getAuthors,getPubDate, getDOI, getKeywords`). As well as proof read, added content to the PDF file, and helped debug
3) Priya- Worked on getYearLinks to go through each issue for each year and extract the articles along with the required fields. Worked on the functions getAbstract and getFullText and tested the code to make sure it met the requirements that were asked. Solved any minor issues with code and debug. Created and wrote the team pdf document.