Data 607 Week 7

Jean Jimenez

2023-10-10

Importing Different Data Formats

This week, we were asked to create 3 files (json, html, xml) containing a table with information about books. Then, we were asked to import each of the tables into data frames.

```r
library(dplyr)
```

```
##
## Attaching package: 'dplyr'

## The following objects are masked from 'package:stats':
##
##     filter, lag

## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```r
library(tidyverse)
```

```
## -- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
## v forcats   1.0.0     v readr     2.1.4
## v ggplot2   3.4.3     v stringr   1.5.0
## v lubridate 1.9.3     v tibble    3.2.1
## v purrr     1.0.2     v tidyr     1.3.0

## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
## i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become error
```

html

To import the html data, I first created and uploaded the html file of books to github. I used the rvest package. I passed the url through read_html() function. The function gets the raw html data from the github page and stores the html data in a xml_document as a xml_node. Afterwards, I used dplyr to find the table in the html data by using html_nodes(). The table argument will look for the html

. After extracting the data from the table, I place it in the data frame html_books.

```
library(rvest)
```

```
##
## Attaching package: 'rvest'

## The following object is masked from 'package:readr':
##
##     guess_encoding
```

```
#url to html file

html_url="https://raw.githubusercontent.com/sleepysloth12/data607_wk7/main/books.html"


#read html
git_html= read_html(html_url)
class(git_html)
```

```
## [1] "xml_document" "xml_node"
```

```
#extract first table into dataframe

html_books = git_html %>%
  html_nodes("table") %>%
  .[[1]] %>%
  html_table()

html_books
```

```
## # A tibble: 3 x 6
##   Title        `Author(s)` `Original Language` `Year Published` `City Published`
##   <chr>        <chr>       <chr>                          <int> <chr>
## 1 The Communi~ Friedrich ~ German                          1848 London
## 2 The Social ~ Jean-Jacqu~ French                          1762 Geneva
## 3 Existential~ Jean-Paul ~ French                          1946 New Haven
## # i 1 more variable: `Number of Words` <int>
```

json

To import the json data,I used both the package jsonlite and httr.I used httr's GET() which retrieves the data from a web page. I stored the webpage data in git_json (which stores it as a response class). Afterwards, I use httr's content() to extract the information from the web response and store it as a character vector. Then, I used jsonlite's fromJSON() function to retrieve the information from the json file and store it as list. The flatten=TRUE value will make a wide data frame. Finally, I converted that list to a data frame.

```
library(jsonlite)
```

```
##
## Attaching package: 'jsonlite'
## The following object is masked from 'package:purrr':
##
##     flatten
```

```r
library(httr)

#json url
json_url="https://raw.githubusercontent.com/sleepysloth12/data607_wk7/main/books.json"

#getting json info from github page
git_json=GET(json_url)
class(git_json)
```

```
## [1] "response"
```

```r
json_content=content(git_json, "text")
class(json_content)
```

```
## [1] "character"
```

```r
#get json data
json_data = fromJSON(json_content, flatten = TRUE)
class(json_data)
```

```
## [1] "list"
```

```r
json_books= as.data.frame(json_data)

json_books
```

```
##                      Books.Title               Books.Authors
## 1      The Communist Manifesto Friedrich Engels, Karl Marx
## 2          The Social Contract       Jean-Jacques Rousseau
## 3 Existentialism Is a Humanism            Jean-Paul Sartre
##   Books.OriginalLanguage Books.YearPublished Books.CityPublished
## 1                 German                1848              London
## 2                 French                1762              Geneva
## 3                 French                1946           New Haven
##   Books.NumberOfWords
## 1               24000
## 2               42750
## 3               17852
```

xml

To import the xml data, I used the package xml12. I used its function read_xml() to read and get the xml data from the url.This stores the data as an xml_document in xml_nodes. Afterwards, we convert that xml data to a list then a dataframe.

```r
library(xml2)

#xml_url
xml_url="https://raw.githubusercontent.com/sleepysloth12/data607_wk7/main/books.xml"

#read the xml data from webpage

books_xml_dat=read_xml(xml_url)
class(books_xml_dat)
```

```
## [1] "xml_document" "xml_node"
```

```r
#turn xml node to list
books_xml_lst=as_list(books_xml_dat)

xml_books=as.data.frame(books_xml_lst)

xml_books
```

```
##   Books.Book..The.Communist.Manifesto. Books.Book..Friedrich.Engels..Karl.Marx.
## 1              The Communist Manifesto              Friedrich Engels, Karl Marx
##   Books.Book..German. Books.Book..1848. Books.Book..London. Books.Book..24000.
## 1              German              1848              London              24000
##   Books.Book..The.Social.Contract. Books.Book..Jean.Jacques.Rousseau.
## 1              The Social Contract              Jean-Jacques Rousseau
##   Books.Book..French. Books.Book..1762. Books.Book..Geneva. Books.Book..42750.
## 1              French              1762              Geneva              42750
##   Books.Book..Existentialism.Is.a.Humanism. Books.Book..Jean.Paul.Sartre.
## 1              Existentialism Is a Humanism              Jean-Paul Sartre
##   Books.Book..French..1 Books.Book..1946. Books.Book..New.Haven.
## 1                French              1946              New Haven
##   Books.Book..17852.
## 1              17852
```

Conclusion

The data frames are not identical. They are each unique because the structures of html, json, and xml are different. The html data frame was the closest one to the actual data. html has no attributes to store the data in and there is no nested structure. The json file type can handles nested objects and arrays. In my json data frame, you can see the object 'Books' and its different attributes ('Books.NumberOfWords', 'Books.OriginalLanguage' ,etc.). The xml file was the most nested, with books being the list of all books, and book being each book. Each data point had its own column and it is a wide wide table. If I had to choose between the three, I would choose json. json also maintains the class of the data (ie. year being int).