DATA607 7th Week Assignment

Koohyar Pooladvand <- K00hPy

2024-03-10

Introduction

The goal of this week's assignment is to work with HTML, XML, and JSON files. In this process, I have selected three of my favorite books from Amazon and created different files for each. For each book, I have included the title, authors, language, version, publisher, links, and summary.

I have created these files myself while learning from various online resources. I have generated five files in formats such as HTML, XML, CSV, XLS, and JSON. I used NotePad++ to created these three files.

Next, I will write R code using RStudio to load the information from each of the five files into separate R data frames.

Afterwards, I will compare the loaded data to see if they are identical or not. I have uploaded all these files to GitHub to ensure reproducibility.

Code Initiation

You can also embed plots, for example:

```
## [1] "All required packages are installed"
```

Read the information

In this section of the code, we use different methods and packages to load the five files individually to different dataframes in R. We start by defining the paths that contain the files and then we use RStudio to load them.

```
#local addresses
GitHub_raw <- "https://raw.githubusercontent.com/koohpi/DATA607_7TH_ASSGNMNT/main/"
XML_path <- "Data/Book_data.xml"
XLSX_path <- "Data/Book_data.xlsx"
HTML_path <- "Data/Book_data.html"
JSON_path <- "Data/Book_data.json"
CSV_path <- "Data/Book_data.csv"

#GitHub Addresses added to the local for reproducibility
XML_path <- paste0(GitHub_raw , XML_path)
XLSX_path <- paste0(GitHub_raw , XLSX_path)
HTML_path <- paste0(GitHub_raw , HTML_path)
JSON_path <- paste0(GitHub_raw , JSON_path)
CSV_path <- paste0(GitHub_raw , CSV_path)
```

```r
#"https://raw.githubusercontent.com/koohpi/DATA607_7TH_ASSGNMNT/main/Data/Book_data.csv"

# Read the XML file and parse it into an XML document object
#Read_XLM <- XML::xmlTreeParse(XML_path, useInternalNodes = TRUE)
# Read XML file
Read_XML <- tryCatch(
  xml2::read_xml(XML_path),
  error = function(e) {
    message("Error loading XML file:", conditionMessage(e))
    NULL
  }
)
# Check if XML file is loaded
if (!is.null(Read_XML)) {
  message("XML file loaded successfully!")
}
```

```
## XML file loaded successfully!
```

```r
# Read HTML file
Read_html <- tryCatch(
  rvest::read_html(HTML_path),
  error = function(e) {
    message("Error loading HTML file:", conditionMessage(e))
    NULL
  }
)

# Check if XML file is loaded
if (!is.null(Read_XML)) {
  message("HTML file loaded successfully!")
}
```

```
## HTML file loaded successfully!
```

```r
#read HTML file from the path to RStudio
#Read_html <- rvest::read_html(HTML_path)

# Read HTML file
Read_json <- tryCatch(
  jsonlite::fromJSON(JSON_path),
  error = function(e) {
    message("Error loading JSON file:", conditionMessage(e))
    NULL
  }
)

# Check if XML file is loaded
if (!is.null(Read_json)) {
  message("JSON file loaded successfully!")
}
```

```
## JSON file loaded successfully!
```

```
#read JSON file from the path to RStudio
#Read_json <- jsonlite::fromJSON(JSON_path)

#read CSV file to the path to RStudio
Read_csv <- read.csv(CSV_path,check.names = TRUE,
                     na.strings = "", dec = ".", quote = "\"")
# Read the xlsx and the sheet1 from the file as text with exact column name
#Read_xlsx <- readxl::read_xlsx(XLSX_path, sheet = "Sheet1",
#                               col_names = TRUE,
#                               col_types = "text", na = "" )
Read_xlsx <- openxlsx::read.xlsx(XLSX_path, sheet = "Sheet1",
                                 colNames=TRUE, check.names = TRUE,
                                 na.strings = "",
                                 detectDates = FALSE)
print("CSV and XLSX files also loaded as references")
```

```
## [1] "CSV and XLSX files also loaded as references"
```

Analyzing the loaded files

Comparing the all read file into the Rstudio show CSV, JSON, and XLSX was read as imported into the RStudio. However the data from html and XLM need further analyses to become a DF, JSON imported file is complete.

```
# use kable to show the loaded data and then compare them to see how they differ if any

# Display the dataframe using kable, since there are some column with a lot of information, I want to o

Read_xlsx_DF <- data.frame(Read_xlsx)
Read_csv_DF <- data.frame(Read_csv)


# Select specific columns using dplyr's select function
selected_columns <- dplyr::select(Read_csv_DF, 1:7)

# Display the selected columns using kable
print("This is how the data loaded from the CSV looks like")
```

```
## [1] "This is how the data loaded from the CSV looks like"
```

```
knitr::kable(selected_columns)
```

Title	Authors	Release.Date	Language	ASIN	Version	Publisher
Build the Life You Want: The Art and Science of Getting Happier	Arthur C. Brooks, Oprah Winfrey, et al.	12-Sep-23	English	B0C4V8[illegible]	[illegible]	Penguin Audio
Harry Potter and the Sorcerer's Stone, Book 1	J.K. Rowling	20-Nov-15	English	B017V4[illegible]	[illegible]	Pottermore Publishing

Title	Authors	Release.Date	Language	ASIN	Version	Publisher
The Handmaid's Tale	Margaret Atwood	1-Sep-12	English	B008X6SZ0K	Unabridged	Audible Studios

```
# Select specific columns using dplyr's select function
Read_json_DF <- data.frame(Read_json)
selected_columns <- dplyr::select(Read_json_DF, 1:7)

# Display the selected columns using kable
print("This is how the data loaded from the JSON file looks like w/o tidying")
```

```
## [1] "This is how the data loaded from the JSON file looks like w/o tidying"
```

```
knitr::kable(selected_columns)
```

Title	Authors	Release.Date	Language	ASIN	Version	Publisher
Build the Life You Want: The Art and Science of Getting Happier	Arthur C. Brooks, Oprah Winfrey, et al.	12-Sep-23	English	B0C4V8H4CX	Unabridged	Penguin Audio
Harry Potter and the Sorcerer's Stone, Book 1	J.K. Rowling	20-Nov-15	English	B017V4IM1G	Unabridged	Pottermore Publishing
The Handmaid's Tale	Margaret Atwood	1-Sep-12	English	B008X6SZ0K	Unabridged	Audible Studios

```
# Select specific columns using dplyr's select function
selected_columns <- dplyr::select(Read_xlsx_DF, 1:7)

# Display the selected columns using kable
print("This is how the data loaded from XLSX file looks like")
```

```
## [1] "This is how the data loaded from XLSX file looks like"
```

```
knitr::kable(selected_columns)
```

Title	Authors	Release.Date	Language	ASIN	Version	Publisher
Build the Life You Want: The Art and Science of Getting Happier	Arthur C. Brooks, Oprah Winfrey, et al.	12-Sep-23	English	B0C4V8H4CX	Unabridged	Penguin Audio
Harry Potter and the Sorcerer's Stone, Book 1	J.K. Rowling	20-Nov-15	English	B017V4IM1G	Unabridged	Pottermore Publishing
The Handmaid's Tale	Margaret Atwood	1-Sep-12	English	B008X6SZ0K	Unabridged	Audible Studios

```
#Let's compare the three loaded dataframe
#since data has two text heavy columns, I decided to only compare columns 1:7 and ignore 8 and 9.
print("Let's compare loaded xlsx and json")
```

```
## [1] "Let's compare loaded xlsx and json"
```

```
if (identical(Read_xlsx_DF[,1:7], Read_json_DF[,1:7])){
  print("The loaded xlsx and json are the same")
}else{
  # Find rows in dataframe1 but not in dataframe2
  print("Loaded json and xlsx files are not identical and the differeces are saved in json_vs_csv")
  json_vs_csv <- setdiff(Read_json_DF[,1:7], Read_xlsx_DF[,1:7])
}
```

```
## [1] "The loaded xlsx and json are the same"
```

Convert XML file to DF

As seen previously, reading XML and HTML files into RStudio requires extra attention compared to JSON and CSV files.

In this section, we address the loaded XML file. First, we examine the structure of the XML file. It is loaded using the xml2 package as a list. We then use xml_find_all to identify how the data is structured, using "book" as the decoder for the set of information. Subsequently, we iterate through each "book" list in the loaded list, parsing the data in each set and separately storing them in a list. Later, we will collapse this list into a dataframe using the same name as expected.

In the remaining code, we remove the firt row that has the header, and compare it with the XLSX file as a reference.

```
#Use the pointer to parse data from XML file. we use XML library to do so here.
# Point locations


library(xml2)
#use xml2 and find_all and text function to structure the dataframe, I know how book is structured and

XML_data <- xml_find_all(Read_XML, "//book")

# Initialize empty lists to store parsed data, this follows the structure of the XML file
Title <- list()
Authors <- list()
Release_date <- list()
Language <- list()
ASIN <- list()
Version <- list()
Publisher <- list()
Summary <- list()
Link <- list()

# Loop through each book element and parse its child elements, it follows the known format of the data
for (book in XML_data) {
  Title <- c(Title, xml_text(xml_find_all(book, ".//title")))
  Authors <- c(Authors, xml_text(xml_find_all(book, ".//author")))
  Release_date <- c(Release_date, xml_text(xml_find_all(book, ".//Release_date")))
  language <- c(Language, xml_text(xml_find_all(book, ".//Language")))
  ASIN <- c(ASIN, xml_text(xml_find_all(book, ".//ASIN")))
```

```
  Version <- c(Version, xml_text(xml_find_all(book, ".//Version")))
  Publisher <- c(Publisher, xml_text(xml_find_all(book, ".//Publisher")))
  Summary <- c(Summary, xml_text(xml_find_all(book, ".//Summary")))
  Link <- c(Link, xml_text(xml_find_all(book, ".//Link")))
}

# Create a dataframe from the parsed data
Read_XML_DF <- data.frame(
  Title = unlist(Title),
  Authors = unlist(Authors),
  `Release.Date` = unlist(Release_date),
  Language = unlist(language),
  ASIN = unlist(ASIN),
  Version = unlist(Version),
  Publisher = unlist(Publisher),
  Summary = unlist(Summary),
  Link = unlist(Link)
)

#remove the first row since it is
Read_XML_DF <- Read_XML_DF[-1,]

# View the dataframe
# Select specific columns using dplyr's select function
selected_columns <- dplyr::select(Read_XML_DF, 1:7)

# Display the selected columns using kable
print("This is the data loaded from XML file to RStudio")
```

```
## [1] "This is the data loaded from XML file to RStudio"
```

```
knitr::kable(selected_columns)
```

	Title	Authors	Release.Date	Language	ASIN	Version	Publisher
2	Build the Life You Want: The Art and Science of Getting Happier	Arthur C. Brooks, Oprah Winfrey, et al.	12-Sep-23	English	B0C4V8H4GX	Unabridged	Penguin Audio
3	Harry Potter and the Sorcerer's Stone, Book 1	J.K. Rowling	20-Nov-15	English	B017V4IM1G	Unabridged	Pottermore Publishing
4	The Handmaid's Tale	Margaret Atwood	1-Sep-12	English	B008X6SZ0K	Unabridged	Audible Studios

```
#Let's compare the loaded dataframe
#since data has two text heavy columns, I decided to only compare columns 1:7 and ignore 8 and 9.
print("Let's compare loaded xlsx and xml")
```

```
## [1] "Let's compare loaded xlsx and xml"
```

```
if (identical(Read_xlsx_DF[,1:7], Read_XML_DF[,1:7])){
  print("The loaded xlsx and XML are the same")
}else{
  # Find rows in dataframe1 but not in dataframe2
  print("Loaded xlsx and XML files are not identical and the differeces are saved in XML_vs_xlsx")
  XML_vs_xlsx <- setdiff(Read_XML_DF[,1:7], Read_xlsx_DF[,1:7])
}
```

```
## [1] "Loaded xlsx and XML files are not identical and the differeces are saved in XML_vs_xlsx"
```

Convert HTML file to DF

The HTML is also when it is loaded has created a list, but the list is actually a pointer. We use html_nodes and html_table to convert the data to the format that can be later be transformed to a DF.

When the data is loaded to `html_data` then we use a simple line of code to transformed it to a data.frame called `temp` and a more time consuming path of going thru each column of the table using `tr td: ...`, in this process individual column are separated and later combined to a single datafarme called `Read_HTML_DF`.

```
#read the table from the html file
library(rvest)
# Read the 1t table i nth html since it is our table of interest
html_data <- html_table(html_nodes(Read_html, "table")[1])

# now that the data is loaded in list, we can see the first object of the list to the entire dataframe,
temp <- data.frame(html_data[[1]])

#another way is to use the table expression and load the children column by column, this is more flexib

HTML_Title <- list()
HTML_Authors <- list()
HTML_Release_date <- list()
HTML_Language <- list()
HTML_ASIN <- list()
HTML_Version <- list()
HTML_Publisher <- list()
HTML_Summary <- list()
HTML_Link <- list()

#go thru the 9 columns one at a time
HTML_Title <- html_text(html_nodes(Read_html, "table tr td:nth-child(1)"))
HTML_Authors <- html_text(html_nodes(Read_html, "table tr td:nth-child(2)"))
HTML_Release_data <- html_text(html_nodes(Read_html, "table tr td:nth-child(3)"))
HTML_Language <- html_text(html_nodes(Read_html, "table tr td:nth-child(4)"))
HTML_ASIN <- html_text(html_nodes(Read_html, "table tr td:nth-child(5)"))
HTML_Version <- html_text(html_nodes(Read_html, "table tr td:nth-child(6)"))
HTML_Publisher <- html_text(html_nodes(Read_html, "table tr td:nth-child(7)"))
HTML_Summary <- html_text(html_nodes(Read_html, "table tr td:nth-child(8)"))
HTML_Link <- html_text(html_nodes(Read_html, "table tr td:nth-child(9)"))


# Create a dataframe from the parsed data
Read_HTML_DF <- data.frame(
```

```
  Title = unlist(HTML_Title),
  Authors = unlist(HTML_Authors),
  `Release.Date` = unlist(HTML_Release_data),
  Language = unlist(HTML_Language),
  ASIN = unlist(HTML_ASIN),
  Version = unlist(HTML_Version),
  Publisher = unlist(HTML_Publisher),
  Summary = unlist(HTML_Summary),
  Link = unlist(HTML_Link)
)


# View the dataframe
# Select specific columns using dplyr's select function
selected_columns <- dplyr::select(Read_HTML_DF, 1:7)

# Display the selected columns using kable
print("This is the data loaded from HTML file to RStudio")
```

```
## [1] "This is the data loaded from HTML file to RStudio"
```

```
knitr::kable(selected_columns)
```

Title	Authors	Release.Date	Language	ASIN	Version	Publisher
Build the Life You Want: The Art and Science of Getting Happier	Arthur C. Brooks, Oprah Winfrey, et al.	12-Sep-23	English	B0C4V8H4CX	Unabridged	Penguin Audio
Harry Potter and the Sorcerer's Stone, Book 1	J.K. Rowling	20-Nov-15	English	B017V4IM1G	Unabridged	Pottermore Publishing
The Handmaid's Tale	Margaret Atwood	1-Sep-12	English	B008X6SZ0K	Unabridged	Audible Studios

```
#Let's compare the loaded dataframe
#since data has two text heavy columns, I decided to only compare columns 1:7 and ignore 8 and 9.
print("Let's compare loaded xlsx and html")
```

```
## [1] "Let's compare loaded xlsx and html"
```

```
if (identical(Read_xlsx_DF[,1:7], Read_HTML_DF[,1:7])){
  print("The loaded xlsx and HTML are the same")
}else{
  # Find rows in dataframe1 but not in dataframe2
  print("Loaded xlsx and XML files are not identical and the differeces are saved in XML_vs_xlsx DF")

  HTML_vs_xlsx <- setdiff(Read_HTML_DF[,1:7], Read_xlsx_DF[,1:7])
}
```

```
## [1] "Loaded xlsx and XML files are not identical and the differeces are saved in XML_vs_xlsx DF"
```

Conclusion

In this week's assignment, we created three different files for a same sets of data from scratch in HTML, XML, and JSON, and loaded them into data frames (DFs) in RStudio. I also create csv and xlsx for my reference. HTML as the easiest one, nd I was able to load it by simple line of code, however, I chose to use a more complex method, in case I am dealing with a messy HTML file in the near future. aide from HTML, It has been shown that JSON is the easiest and most convenient format for structured original tables containing data about the three different books. Among them XML and second HTML methods parsing data column to column, were a little trickier and required a few more steps to load the data. There are libraries that are loaded to help us with different formats, and some of them were surprisingly use to sue and helpful. Overall, success depends on how well we understand the structure of the presented data specifically JSON and XML files, which can be challenging in practice.

Additionally, we conducted a simple comparison between the loaded DFs, and currently, none of them are exactly the same. The difference lies in the data column. I will attempt to standardize them by changing the date column type to a consistent format in the future.

End