# DATA607 Assignment 5: Working with XML and JSON in R

*Albert Gilharry*

*15 March 2018*

## Contents

### Intro

**For this assignment, we were tasked with creating HTML, XML, and JSON files of 3 or our favourite books on one of our favorite topics. At least one of the books must have more than one author. Each of the different file structures should be loaded into R data frames. This is a primer for further work with these structures in the semseter.**

### Load Libraries

```r
library("tidyverse")
```

```
## -- Attaching packages ------------------------------------------------------------------------------

## v ggplot2 2.2.1     v purrr   0.2.4
## v tibble  1.4.2     v dplyr   0.7.4
## v tidyr   0.8.0     v stringr 1.2.0
## v readr   1.1.1     v forcats 0.2.0

## -- Conflicts ---------------------------------------------------------------------------------------
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```r
library("rvest")
```

```
## Loading required package: xml2

##
## Attaching package: 'rvest'
## The following object is masked from 'package:purrr':
##
##     pluck

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

```
library("XML")
```

```
##
## Attaching package: 'XML'

## The following object is masked from 'package:rvest':
##
##     xml
```

```
library("methods")
library("jsonlite")
```

```
##
## Attaching package: 'jsonlite'

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

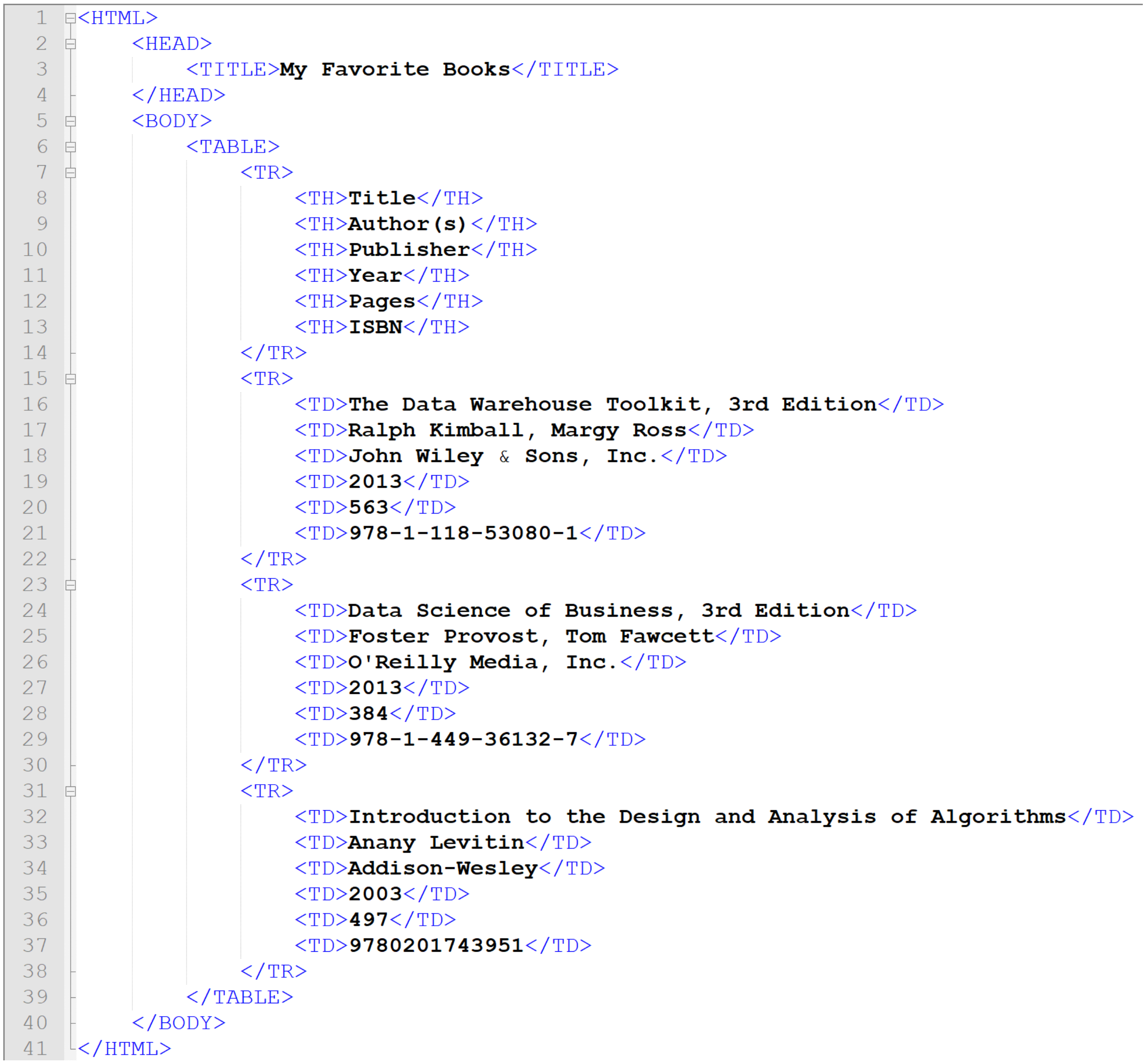
```
library("stringr")
library("DT")
library ("RCurl")
```

```
## Loading required package: bitops

##
## Attaching package: 'RCurl'

## The following object is masked from 'package:tidyr':
##
##     complete
```

## Preview HTML File

## Load HTML Data & Create Data Frame

```
# load HTML data into data frame
url <- "https://raw.githubusercontent.com/albert-gilharry/DATA607-Assignment-5/master/data/books.html"
htmlBooks <- url %>%
  read_html() %>%
  html_nodes("table") %>%
  html_table()

htmlBooks <- htmlBooks[[1]]
```



```html
<HTML>
    <HEAD>
        <TITLE>My Favorite Books</TITLE>
    </HEAD>
    <BODY>
        <TABLE>
            <TR>
                <TH>Title</TH>
                <TH>Author(s)</TH>
                <TH>Publisher</TH>
                <TH>Year</TH>
                <TH>Pages</TH>
                <TH>ISBN</TH>
            </TR>
            <TR>
                <TD>The Data Warehouse Toolkit, 3rd Edition</TD>
                <TD>Ralph Kimball, Margy Ross</TD>
                <TD>John Wiley & Sons, Inc.</TD>
                <TD>2013</TD>
                <TD>563</TD>
                <TD>978-1-118-53080-1</TD>
            </TR>
            <TR>
                <TD>Data Science of Business, 3rd Edition</TD>
                <TD>Foster Provost, Tom Fawcett</TD>
                <TD>O'Reilly Media, Inc.</TD>
                <TD>2013</TD>
                <TD>384</TD>
                <TD>978-1-449-36132-7</TD>
            </TR>
            <TR>
                <TD>Introduction to the Design and Analysis of Algorithms</TD>
                <TD>Anany Levitin</TD>
                <TD>Addison-Wesley</TD>
                <TD>2003</TD>
                <TD>497</TD>
                <TD>9780201743951</TD>
            </TR>
        </TABLE>
    </BODY>
</HTML>
```

Figure 1: HTML FIle

### View HTML Data Frame

```
datatable(htmlBooks, options = list(filter = FALSE), filter = "none")
```

```
## PhantomJS not found. You can install it with webshot::install_phantomjs(). If it is installed, pleas
```

### Preview XML File

### Load XML Data & Create Data Frame

The books with multiple authors posed a problem because the built in functionality to convert XML to a data frame concatenates the author nodes without a delimiter. For this reason, I looped through the data to format the authors' names properly. This may not be most efficient way of doing so, but this is a very small data set, so it is fine.

```
url <- getURL("https://raw.githubusercontent.com/albert-gilharry/DATA607-Assignment-5/master/data/books
doc <- xmlParse(url)

data <- xpathSApply(doc, "//BOOKS/BOOK/AUTHORS",xmlChildren, simplify = TRUE)
authors = c()
for(i in 1:length(data)){
  c <- c()
  for(j in 1:length(data[[i]])){
   c <-  append( c, xmlValue(data[[i]][[j]]) )
  }
  authors <- append(authors, paste(unlist(c),collapse = ", "))
}

# use the built in function to create the data frame
xmlBooks <- xmlToDataFrame(url,stringsAsFactors = FALSE)

# fix the authors
xmlBooks$AUTHORS <- authors
```

### View XML Data Table

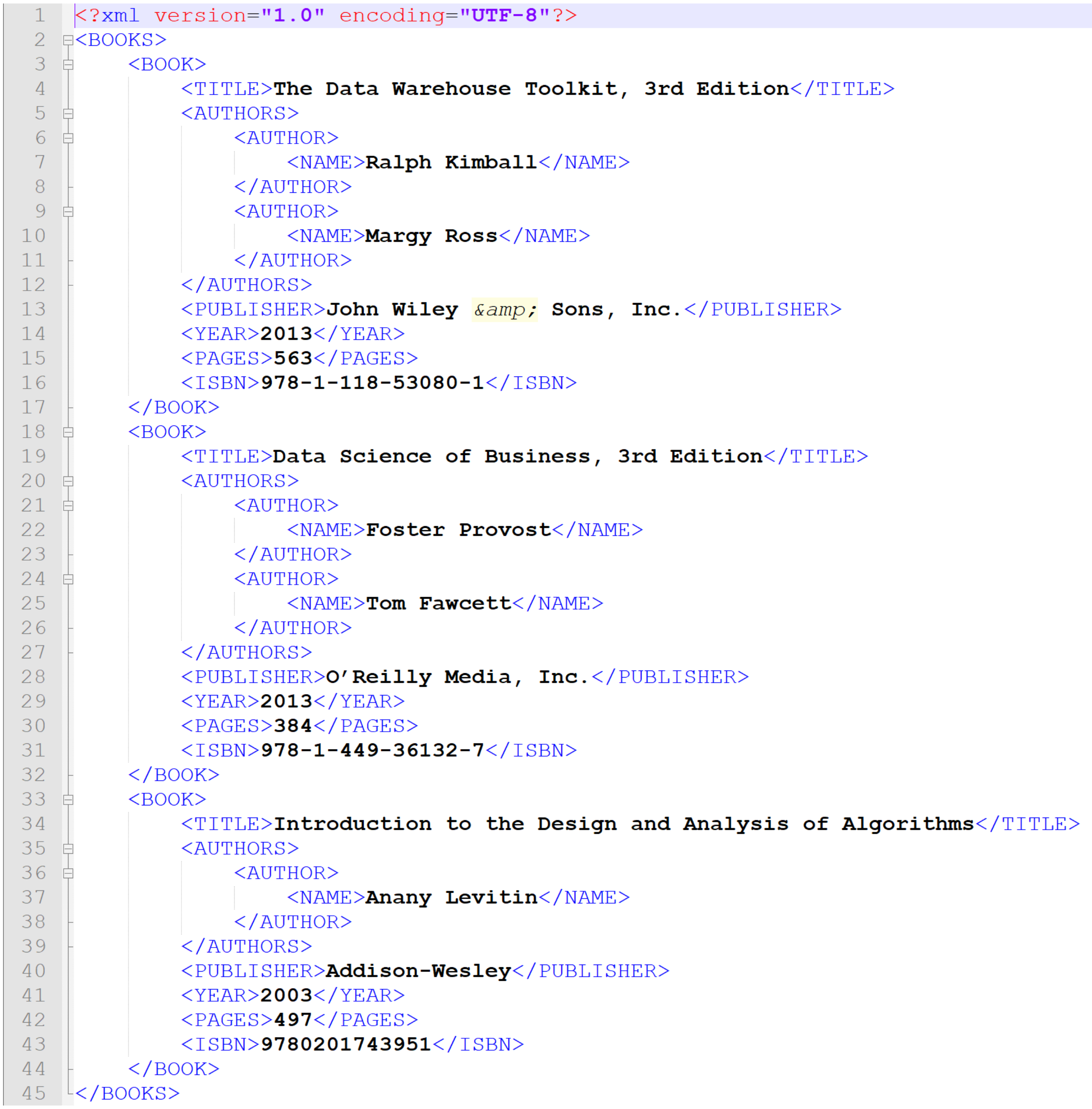
```
datatable(xmlBooks, options = list(filter = FALSE),filter="none")
```

### Preview JSON File

### Load JSON Data & Create Data Frame

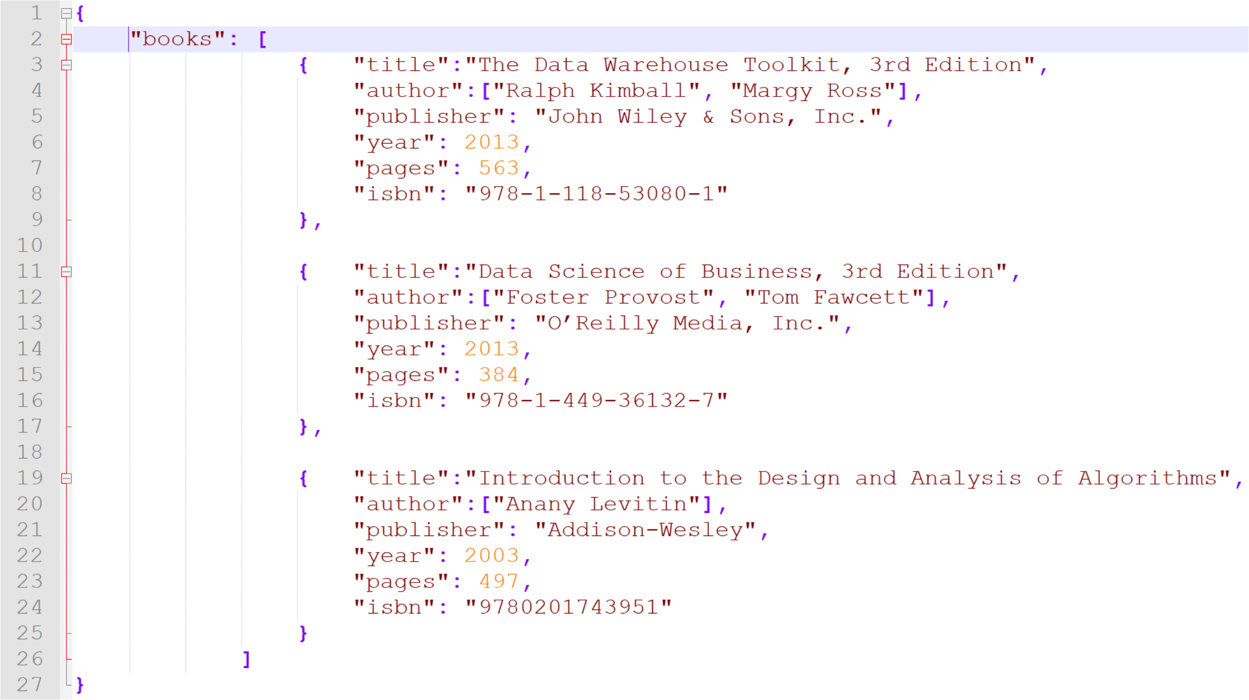
The books with multiple authors posed a problem again because the built in functionality to create a data frame from JSON data attaches an list for the authors. I looped through the data to format the authors. This may not be the most efficient way of doing so, but this is a very small data set, so it is fine.

```
# load JSON data into data frame
url <- getURL("https://raw.githubusercontent.com/albert-gilharry/DATA607-Assignment-5/master/data/books
jsonBooks <- fromJSON(url)
authors = c()
jsonBooks <- jsonBooks$books
```



```xml
<?xml version="1.0" encoding="UTF-8"?>
<BOOKS>
    <BOOK>
        <TITLE>The Data Warehouse Toolkit, 3rd Edition</TITLE>
        <AUTHORS>
            <AUTHOR>
                <NAME>Ralph Kimball</NAME>
            </AUTHOR>
            <AUTHOR>
                <NAME>Margy Ross</NAME>
            </AUTHOR>
        </AUTHORS>
        <PUBLISHER>John Wiley &amp; Sons, Inc.</PUBLISHER>
        <YEAR>2013</YEAR>
        <PAGES>563</PAGES>
        <ISBN>978-1-118-53080-1</ISBN>
    </BOOK>
    <BOOK>
        <TITLE>Data Science of Business, 3rd Edition</TITLE>
        <AUTHORS>
            <AUTHOR>
                <NAME>Foster Provost</NAME>
            </AUTHOR>
            <AUTHOR>
                <NAME>Tom Fawcett</NAME>
            </AUTHOR>
        </AUTHORS>
        <PUBLISHER>O'Reilly Media, Inc.</PUBLISHER>
        <YEAR>2013</YEAR>
        <PAGES>384</PAGES>
        <ISBN>978-1-449-36132-7</ISBN>
    </BOOK>
    <BOOK>
        <TITLE>Introduction to the Design and Analysis of Algorithms</TITLE>
        <AUTHORS>
            <AUTHOR>
                <NAME>Anany Levitin</NAME>
            </AUTHOR>
        </AUTHORS>
        <PUBLISHER>Addison-Wesley</PUBLISHER>
        <YEAR>2003</YEAR>
        <PAGES>497</PAGES>
        <ISBN>9780201743951</ISBN>
    </BOOK>
</BOOKS>
```

Figure 2: HTML FIle



```
{
    "books": [
                {   "title":"The Data Warehouse Toolkit, 3rd Edition",
                    "author":["Ralph Kimball", "Margy Ross"],
                    "publisher": "John Wiley & Sons, Inc.",
                    "year": 2013,
                    "pages": 563,
                    "isbn": "978-1-118-53080-1"
                },

                {   "title":"Data Science of Business, 3rd Edition",
                    "author":["Foster Provost", "Tom Fawcett"],
                    "publisher": "O’Reilly Media, Inc.",
                    "year": 2013,
                    "pages": 384,
                    "isbn": "978-1-449-36132-7"
                },

                {   "title":"Introduction to the Design and Analysis of Algorithms",
                    "author":["Anany Levitin"],
                    "publisher": "Addison-Wesley",
                    "year": 2003,
                    "pages": 497,
                    "isbn": "9780201743951"
                }
            ]
}
```

Figure 3: HTML FIle

```r
# create a comma separated list for authors of each book
for(i in 1:nrow(jsonBooks)){
 authors <- append(authors, paste(unlist( jsonBooks$author[i] ),collapse = ", "))
}

# update authors
jsonBooks$author <- authors
```

## View JSON Data Table

```r
datatable(jsonBooks, options = list(filter = FALSE),filter="none")
```

## Conclusion

In conclusion, R packages make it relatively easy parse and load HTML, XML, and JSON data into data frames. These data frames were not identical due to how the packeges handle one to many relationships in JSON, and XML. The processing that I did eventlually led to all data frames being identical.