Northeastern University

CS6020: Collecting, Storing, and Retrieving Information

Data Import

Data Import

IMPORTING DATA FROM XML

Lesson Objectives

- After completing this lesson, you are able to:
  - create and read an XML document
  - appreciate the use and importance of XML
  - read XML data into an R data frame
  - parse the elements of an XML document

Importing Data From XML

XML DOCUMENT FORMAT

What is XML?

- XML stands for Extensible Markup Language.
- It was designed to describe data in a human readable format that is simple to parse.
- The purpose of XML is as a software and hardware independent encoding format for carrying information.
- The data is described within XML in form of tree.

XML Document Format

- An XML document consists of matching nested tags describing data.
- The tags are free-form and require the sender and receiver of the document to agree.
- There are numerous industry standard XML schemas.

```xml
<?xml version="1.0"
encoding="UTF-8"?>
<Student>
  <FName>John</FName>
  <LName>Doe</LName>
  <Mark>60.0</Mark>
  <Grade>A </Grade>
</Student>
```

XML Declaration

- XML documents must begin with a declaration that specifies information needed by the parser.
- The general declaration looks like this:

```
<?xml version="1.0" encoding="UTF-8"?>
```

Tags

- The tag is a markup construct that begins with < and ends with >.
- Tags come in three flavors:
  - *start tag*: `<Student>`
  - *end tag*: `</Student>`
  - *empty-element tag*: `<isActive />`
- Every start tag must have a matching end tag or the document will not parse.

Parent and Child Elements

- The characters between the start- and end-tags, if any, are the element's *content*.
- Tags nested within other tags are referred to as *child elements*.

```
<course crn="3387">
  <title>Programming in R</title>
  <program>Data Science</program>
  <instructor>
    <name>Martin Schedlbauer, Ph.D.</name>
    <email>m.schedlbauer@neu.edu</email>
  </instructor>
</course>
```

Attributes

- Many XML elements have attributes:

```
<course crn="3387">
  <title>Programming in R</title>
  <program>Data Science</program>
  <instructor>
    <name>Martin Schedlbauer, Ph.D.</name>
    <email>m.schedlbauer@neu.edu</email>
  </instructor>
</course>
```

Importing Data From XML

LOADING XML DOCUMENTS

Reading XML Documents in R

- To read and parse XML in R requires the external package "***XML***".
- The ***XML*** package provides the functions necessary to load an XML file and parse its document tree:
  - `xmlParse()`
  - `xmlToDataFrame()`

Parsing into Object

- To load an XML document into an internal document object use
  - `xmlParse()`
    - parses the XML data and creates a document object of class *`XMLInternalDocument`*

```
xmlobj <- xmlParse("pubmed_sample.xml")
```

- Once the files has been successfully parsed, R stores the XML document in the internal object `xmlobj` .

Parsing into Data Frame

- To load an XML document into a data frame use
  - `xmlToDataFrame()`
    - parses the XML data and generates an R data frame representing the data in the document

```
xmldf <- xmlToDataFrame("pubmed_sample.xml")
```

- Once the files has been successfully parsed, R stores the XML document in the data frame `xmldf`.

Parsing via HTTP

- An XML document can also be loading through via HTTP through its URL using either `xmlParse()` or `xmlToDataFrame()`.

```
> url <- "http://www.statistics.life.ku.dk/primer/mydata.xml"
> data <- xmlToDataFrame(url)
> head(data)
  Girth Height Volume
1   8.3     70   10.3
2   8.6     65   10.3
3   8.8     63   10.2
4  10.5     72   16.4
5  10.7     81   18.8
6  10.8     83   19.7
```

The XML Document

- Below is an excerpt of the XML document:

```
<?xml version="1.0"?>
<document>
  <row>
    <Girth>8.3</Girth>
    <Height>70</Height>
    <Volume>10.3</Volume>
  </row>
  <row>
    <Girth>8.6</Girth>
    <Height>65</Height>
    <Volume>10.3</Volume>
  </row>
…
</document>
```

```
data <- xmlToDataFrame(url)
```

```
Girth Height Volume
1    8.3        70     10.3
2    8.6        65     10.3
…
```

Importing Data From XML

NAVIGATING THE XML TREE

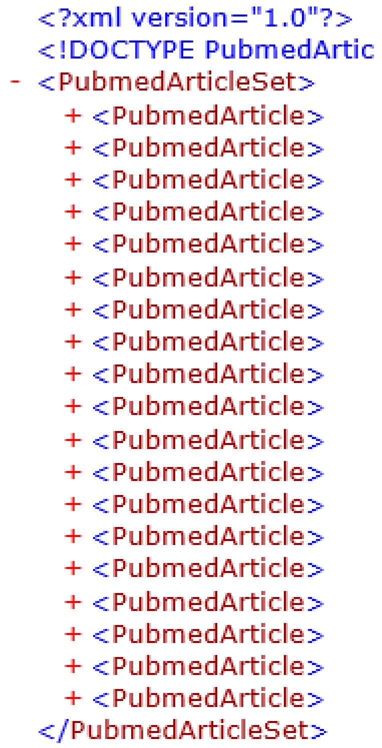
Parsing into a Tree

- To navigate the document object, R requires parsing with `xmlTreeParse()` followed by retrieving the root node object using `xmlRoot()`.

```
> xmlobj <- xmlTreeParse("pubmed_sample.xml")
> r <- xmlRoot(xmlobj)
```

- `r` is of class *XMLNode*.

Getting Name and Size



```
<?xml version="1.0"?>
<!DOCTYPE PubmedArtic
- <PubmedArticleSet>
  + <PubmedArticle>
  + <PubmedArticle>
  + <PubmedArticle>
  + <PubmedArticle>
  + <PubmedArticle>
  + <PubmedArticle>
  + <PubmedArticle>
  + <PubmedArticle>
  + <PubmedArticle>
  + <PubmedArticle>
  + <PubmedArticle>
  + <PubmedArticle>
  + <PubmedArticle>
  + <PubmedArticle>
  + <PubmedArticle>
  + <PubmedArticle>
  + <PubmedArticle>
  + <PubmedArticle>
  + <PubmedArticle>
</PubmedArticleSet>
```

```
> xmlName(r)
[1] "PubmedArticleSet"
> xmlSize(r)
[1] 19
```

- `xmlName()`
  - get the name of the root element
- `xmlSize()`
  - number of root children

Accessing Child Nodes

- Each child node is accessible through subscripting through its parent node.
- For example, to retrieve the first PubMed article, use:

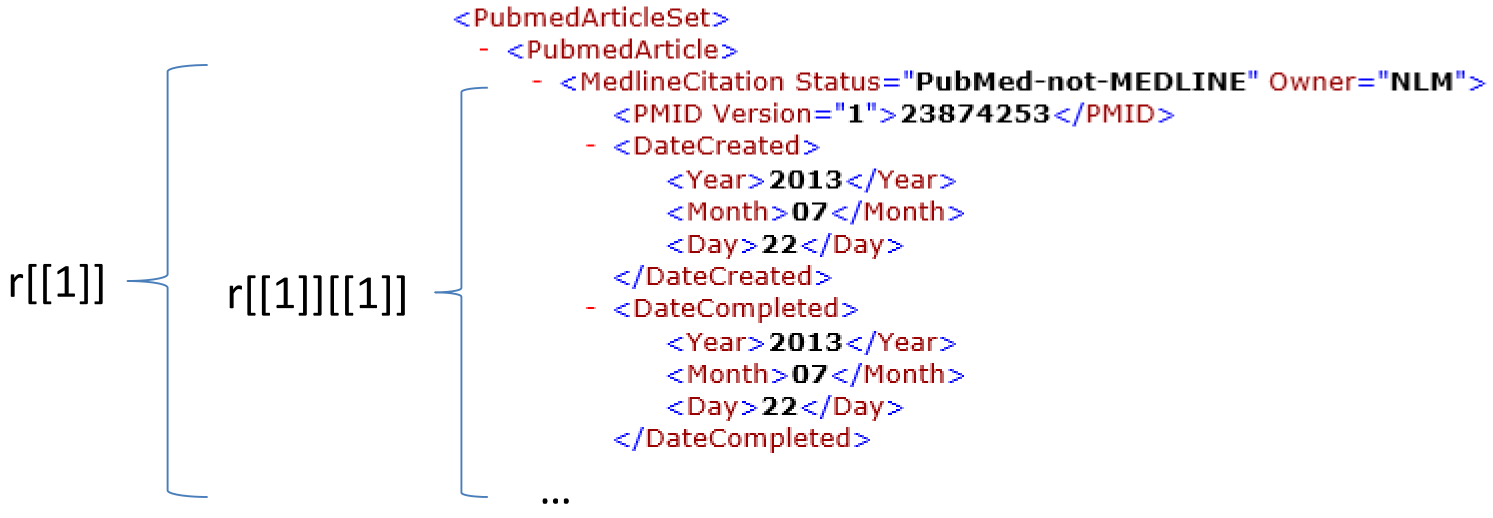
```
> r[[1]]
<PubmedArticle>
 <MedlineCitation Owner="NLM" Status="PubMed-not-MEDLINE">
  <PMID Version="1">23874253</PMID>
  <DateCreated>
  ...
> xmlName(r[[1]])
[1] "PubmedArticle"
```

Navigating the Tree

- Use repeated subscripting to travel the node tree.

```
> r[[1]][[2]][[1]][[1]]
<PubMedPubDate PubStatus="received">
 <Year>2012</Year>
 <Month>1</Month>
 <Day>15</Day>
</PubMedPubDate>
```

Navigating Nodes



```
<PubmedArticleSet>
 - <PubmedArticle>
   - <MedlineCitation Status="PubMed-not-MEDLINE" Owner="NLM">
       <PMID Version="1">23874253</PMID>
     - <DateCreated>
         <Year>2013</Year>
         <Month>07</Month>
         <Day>22</Day>
       </DateCreated>
     - <DateCompleted>
         <Year>2013</Year>
         <Month>07</Month>
         <Day>22</Day>
       </DateCompleted>
   …
```

The root of the tree is *<PubmedArticleSet>*, therefore:

- `r[[n]]` is the `n`th child of *<PubmedArticleSet>*
- `r[[n]][[k]]` is the `k`th child of the `n`th node under *<PubmedArticleSet>*
- and so on…

Node Attributes

- Attributes are name/value pairs attached to a start tag.

```
<Article PubModel="Print-Electronic">
   <Journal>
    <ISSN IssnType="Print">1556-3316</ISSN>
    <JournalIssue CitedMedium="Print">
     <Volume>8</Volume>
     <Issue>2</Issue>
     <PubDate>
      <Year>2012</Year>
      <Month>Jul</Month>
     </PubDate>
    </JournalIssue>
    …
```

Accessing Attributes

- Retrieve the attributes as a vector using `xmlAttrs()` then access individual attributes using subscripting.

```
> r[[1]][[2]][[1]][[1]]
<PubMedPubDate PubStatus="received">
 <Year>2012</Year>
 <Month>1</Month>
 <Day>15</Day>
</PubMedPubDate>
> attrs <- xmlAttrs(r[[1]][[2]][[1]][[1]])
> attrs
 PubStatus
"received"
```

The `sapply()` Function

- `sapply()` applies a function over a list or vector.
- It essentially implements the Visitor pattern.
- The example applies the `xmlName()` function to each child node.

```
> r[[1]][[1]][[2]]
<DateCreated>
 <Year>2013</Year>
 <Month>07</Month>
 <Day>22</Day>
</DateCreated>
```

Reading Values

- Once we have discovered the correct node, then the `xmlValue()` function is used to read the value between the tags.

```
> r[[1]][[1]][[2]]
<DateCreated>
 <Year>2013</Year>
 <Month>07</Month>
 <Day>22</Day>
</DateCreated>
> r[[1]][[1]][[2]][[1]]
<Year>2013</Year>
> xmlValue(r[[1]][[1]][[2]][[1]])
[1] "2013"
```

Reading a Node List

- To get all of the child nodes as a list, use subsetting.

```
> r[[1]][[1]][[2]][1:3]
$Year
<Year>2013</Year>

$Month
<Month>07</Month>

$Day
<Day>22</Day>

attr(,"class")
[1] "XMLNodeList"
> r[[1]][[1]][[2]][1:3]$Year
<Year>2013</Year>
```

Summary

- In this lesson, you learned that:
  - XML is an important data interchange format
  - XML documents consist of nested tags representing a tree structure
  - R has support for reading, parsing, and navigating local and web XML documents

Summary, Review, & Questions…