Working with XML files

What is XML?

- XML stands for eXtensible Markup Language.
- XML is a specification, for writing documents that store both the data and the description of the data.
- XML arrange data in a tree-like format.
- The technology was developed specifically for data storage and easy exchange over the internet.

What will we learn?

In this course, we learn:

- to understand and read XML documents.
- to process XML datasets using the R package XML.

Writing XML

A XML sample:

```
<?xml version="1.0" ?>
<GRADES>
    <STAT406 Term = "Fall 2010">
        <Instructor> Gary </Instructor>
        <STUDENT>
            <NAME> TOM SAWYER </NAME>
            <TEST1> 66</TEST1>
            <TEST2> 80</TEST2>
            <FINAL>90</FINAL>
        </STUDENT>
        <STUDENT>
            <NAME> BECKY THATCHER </NAME>
            <TEST1> 86</TEST1>
            <TEST2>90</TEST2>
            <FINAL>70</FINAL>
        </STUDENT>
    </STAT406>
```

Writing XML

```
<STAT426 Term = "Winter 2009">
    <Instructor> Wes </Instructor>
    <STUDENT>
        <NAME> TOM SAWYER </NAME>
        <TEST1> 56</TEST1>
        <TEST2> 81</TEST2>
        <FINAL>96</FINAL>
    </STUDENT>
    <STUDENT>
        <NAME> BECKY THATCHER </NAME>
        <TEST1> 80</TEST1>
        <TEST2>78</TEST2>
        <FINAL>89</FINAL>
    </STUDENT>
</STAT426>
</GRADES>
```

Writing XML

- The first line is the XML declaration.
- The second line begins the data part. It declares the root element named *GRADES*. In XML there is only one root element.
- Next, the document declares the child element named STAT406 which represents one particular course. The children element of STAT406 are the tags Instructor, and STUDENT which contain the grades of the students.
- Finally, the document end with the closing tag of the root element *GRADES*.
- The element *STAT406* contains one attribute named "Term" which specifies the semester.

Writing XML

Rules for writing XML documents:

- Each document must contain one and only one root element.
- Each element is declared using tags and each opening tag must have a closing tag (except for an empty element).

  `<NAME> TOM SAWYER </NAME>`
- The elements must be properly nested: If you start element A and then element B, then you must close B first before closing A.
- XML is case-sensitive.
- We can add comments to the XML document as in

  `<!-- This is a comment element -->`

Writing XML

Rules for writing XML documents:

- Attribute values must be enclosed in matching single or double quotations.
- Attributes are used typically to add contextual information to an element. As with STAT406 element above where the attribute *Term* specifies the semester of the course. Attribute values are always within quotes.
- There is one exception to the closing tag rule. If an element is empty it can be written as

```
<sourceinfo newspaperid="21"></sourceinfo>
```

or

```
<sourceinfo newspaperid="21"/>
```

Writing XML

- Because the characters $<, >, \&, ', "$ all have special meaning in XML, we need special characters to define them in any other meaning.
- We use **&** for &, **<** for $<$, **>** for $>$, **"** for the double quotes " and **'** for '.
- Note that the closing semi-colons are part of the definition.
- Here is an example

```
<athlete>
      <name> Athlete1 </name>
      <weight units="Kg"> &gt; 80 </weight>
</athlete>
```

Processing XML document in R

- Generally speaking, in order to process a XML file, we need to understand the data contained in the file and what we intend to do with it (once again).
- The R package **XML** extends R's capability in processing XML files.

Processing XML document in R

- As we now know, we can load the package XML by calling *library*.
- If not already install use

```
install.packages('XML',dep=T)
```

```
library('XML')
```

- As an example let us process the file 'NSFExample.xml' available in ctools.

Processing XML document in R

- We can read the XML file into R using *xmlTreeParse* as in

  `doc=xmlTreeParse('NSFExample.xml')`
- xmlTreeParse parses the xml file and reads the entire file into memory.
- It returns an object of class XMLDocument.

Processing XML document in R

- The object XMLDocument returned by xmlTreeParse is a fairly complicated object. Essentially, it returns the tree underpinning the XML document.
- We will use special functions contained in the `XML` package to work with XMLDocument objects.
- To access the root element of the tree, we use *xmlRoot* as in

  `root=xmlRoot(doc)`
- Recall that the root is the main element of the tree, the one that contains all the other elements.

Processing XML document in R

- The root of the tree is similar to a list. We can access its children the same way we access components of a list, using $[[\cdot]]$. For example:

  `root[[1]]; root[[2]]`

- The same holds for each node of the tree. For instance `root[[1]][[1]]` gives the first child of the first child of `root` (assuming it exists).
- As we have seen above, each element or node in a XML document has a **name**, possibly some **attributes**, **children** (unless it is a leaf) and possibly a **value**. The library provides functions to retrieve all this information.

Processing XML document in R

- The name of a node is provided by the function **xmlName**.
- The data value held by a node can be pulled using **xmlValue**.
- The list of attributes can be pulled using **xmlAttrs**. The value on a specific attribute can be pulled using **xmlGetAttr**.
- The list of all children nodes can be pulled used **xmlChildren**.

Processing XML document in R

- Get the name of "node1"

```
node1=root[[1]]
xmlName(node1)
[1] "Award"
```

- We can get the first child of "node1".

```
node11=node1[[1]]
xmlName(node11)
[1] "AwardTitle"
xmlValue(node11)
[1] "Kent State Informal Analysis Seminar"
```

Processing XML document in R

- We can also get all the children of "node1" using xmlChildren. This returns a list where each component is an object of class xmlNode.

```
xmlChildren(node1) #returns a list of children nodes
$AwardTitle
<AwardTitle>Kent State Informal Analysis Seminar</Award

$AwardEffectiveDate
<AwardEffectiveDate>01/01/2014</AwardEffectiveDate>
...
```

Processing XML document in R

- Since **xmlChildren** returns a list, we can also access specific children by name.

```
node2 = xmlChildren(node1)$AwardTitle
xmlValue(node2)
```

Processing XML document in R

- Once we understand the structure of the document, we need to process it. That is, pull out the information we need.
- Typically we want to go through each children node of the root element and process it.
- For the NSF example, suppose we want the title, Amount, dates, PI's name and the University.

Processing XML document in R

```
lst_ch=xmlChildren(root[[1]]) # we get the children of
    the single child of root
V1=xmlValue(lst_ch$AwardTitle);
V2=xmlValue(lst_ch$AwardEffectiveDate)
V3=xmlValue(lst_ch$AwardExpirationDate)
V4=xmlValue(lst_ch$AwardAmount)
V5=paste(xmlValue(xmlChildren(lst_ch$Investigator)$FirstName)
      ,xmlValue(xmlChildren(lst_ch$Investigator)$LastName));
V6=xmlValue(xmlChildren(lst_ch$Institution)$Name)
dt=as.data.frame(matrix(NA,ncol=6,nrow=1))
dt[1,]=c(V1,V2,V3,V4,V5,V6)
edit(dt)
```

Processing XML document in R

- National Science Foundation makes these award xml file available in big folders organized by year where each award is a file in a folder.
- Suppose we wish to recover the same information above but from all the 2014 Awards.
- You can download this folder from

  `http://www.nsf.gov/awardsearch/download.jsp`

Processing XML document in R

We can write a function that can be applied automatically to each such file.

```
xmlGetNeededInfo=function(root){
   #root is the root of a given file
   lst_ch=xmlChildren(root[[1]])
   V1=xmlValue(lst_ch$AwardTitle);
   V2=xmlValue(lst_ch$AwardEffectiveDate)
   V3=xmlValue(lst_ch$AwardExpirationDate)
   V4=xmlValue(lst_ch$AwardAmount)
   V5=paste(xmlValue(xmlChildren(lst_ch$Investigator)$FirstName)
       xmlValue(xmlChildren(lst_ch$Investigator)$LastName));
   V6=xmlValue(xmlChildren(lst_ch$Institution)$Name)
   return(c(V1,V2,V3,V4,V5,V6))
}
```

Processing XML document in R

- We get the list of all such files.

```
Files=system('ls',intern=T)
#On windows machines  use the following
#Files = list.files(" ")
L=length(Files)
```

- We iterate through each one of them and apply the function xmlGetNeededInfo.
- For the code below to work make sure the folder *2014* (or whichever year you have downloaded) is placed in your current working directory.

Processing XML document in R

```
Files=system('ls ./2014',intern=T)
#On windows machines  use the following
#Files = list.files("./2014")
L=length(Files)
dt=as.data.frame(matrix(NA,ncol=6,nrow=L))
names(dt)=c('Title','Start','End','Amount','PI',
     'University')
for( jj in 1:L){
   filename=paste("./2014/",Files[jj],sep="")
   doc=xmlTreeParse(filename)
   rt=xmlRoot(doc);
   val=xmlGetNeededInfo(rt)
   dt[jj,]=val
}
```

Conclusion

- We have covered a very basic introduction to XML, focusing on how to process such file in R using the package XML.
- We have used the function xmlTreeParse followed by xmlChildren, xmlName, xmlValue, xmlAttrs, xmlGetAttr of the package XML to parse XML document in memory and retrieve data.
- These notes were prepared using in part the book "XML" (2nd edition) by Kevin H. Goldberg, where you can learn more.
- In particular I would also recommend learning XPath (the SQL of XML), which is very useful to deal with large XML file.