Lab5

October 11, 2016

This is an R Markdown document. Markdown is a simple formatting syntax for authoring HTML, PDF, and MS Word documents. For more details on using R Markdown see http://rmarkdown.rstudio.com.

Some background

What is XML?

XML is a markup language that defines a set of rules for encoding documents in a format that is both human-readable and machine-readable. The W3C's XML 1.0 Specification and several other related specifications (all of them free open standards) define XML.

Why XML?

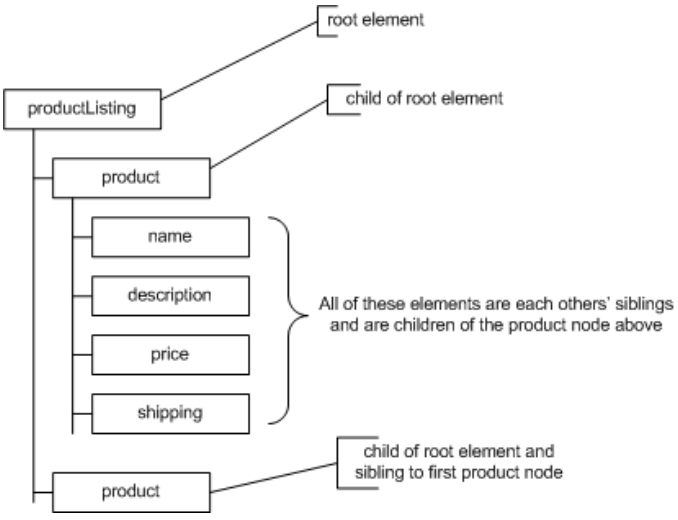
1. Plain text, platform independent.
2. Extensible. Old parsers can work on newly extended XML files.
3. Both human- and machine- readable.

What to expect from this lab?

1. Read and understand the structure of XML files.
2. How to parse XML files in R.

Structures and components of XML files

Tree structures and the root tag (figure from this XML introduction by Tom Myer)



- XML files comprise of *nodes* and *parent-children* relationships.
- An XML file is a tree, not a forest – only one root node.

Components of an XML file

Example:

```
<?xml version="1.0" encoding="UTF-8"?>
<students>
  <person SocialID="123" SchoolID="92031482">
    <name>Jim Brown</name>
    <gender>Male</gender>
    <major>Mathematics</major>
    <minor> Statistics </minor>
  </person>
  <person SocialID="234" SchoolID="91405720">
    <name>Susan Leicester</name>
    <gender>Female</gender>
    <major>Chemistry</major>
  </person>
</students>
```

The declaration

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

- Is not required, but if present must occupy the first line.
- Carries encoding information (what special symbols to expect, French, Swedish, Greek symbols, etc).

Tags and the root tag

- Here, `<students>`, `<person>`, `<name>`, `<gender>`, ... are all tags.
- Each opening/closing tag pair encapsulates a *node*.
- `<students>` is the root tag. It is *unique*. Every time we read an XML file, we query the root. Starting from the root, we access all other components.
- Apart from its uniqueness, the root tag is no different from other tags.
- **Every tag must be closed. No open tag allowed.**
- Tags names may not contain spaces and are **case-sensitive**.
- Duplication in tag names is allowed.

Tag Attributes

- Here, `SocialID` and `SchoolID` are tag attributes.
- Attributes are defined inside tags.
- Attributes must always be quoted.

Elements

- Here, Jim Brown, Male, Mathematics, ... are all elements.
- under a tag, elements can co-exist with subtags.

Example:

```
<?xml version="1.0" encoding="UTF-8"?>
  <person StudentID="123">
  He was a nice and genuine person, despite his appearance.
    <name>Bruce Lee</name>
    <occupation>Terminator</occupation>
  </person>
```

Elements vs attributes (Optional)

- In the previous example, we can put `StudentID` either as an attribute or as the element of a child tag.
- There are many guidelines you can find on Internet, like this one: http://www.ibm.com/developerworks/library/x-eleatt/
- When to use elements: subject to updates in the future; contain further structures, especially child tags, ...
- When to use attributes: marker of the entire tag (ID, type, category, ...)

Parsing XML files in R

1. Install and load the package, read the file and get to the root.

```
## Install and load the package
if (!require(XML)) {
  install.packages("XML", dep = TRUE)
  require(XML)
}
## Read the file
doc = xmlTreeParse("studentdata.xml")
## Get to the root
root = xmlRoot(doc)
root
```

2. Preview and exploratory queries

- Treat "root" as a list.
- `xmlSize(root)` gives the number of children of root.
- `print(root[[1]])` prints the entire first child of root.

3. Navigate the XML tree structure

- `root[[1]]` is the sub-tree rooted at the first child tag.
- That is, the first `<person>` tag can be viewed as the root tag of `root[[1]]`.
- Use `root[[1]][[1]]` or so to access children down in further layers.

4. Access each component

- `xmlName(NodeTagName)` gets Node tag name.
- `xmlAttrs(NodeTagName)` gets all attributes.

- `xmlGetAttr(node=NodeTagName, name=AttributeName, defualt=NULL)` gets a specific attribute. The parameter default specifies the value to return if the queried attribute does not exist.
- (Now we set "student1" to be the first child of root.)

```
student1 = root[[1]]
student1
```

Then `student1[[1]]` and `student1[["name""]]` both access its first child.

- `xmlValue(NodeTagName)` gets the element and/or all further children of a tag.

```
# Display values in a Node with 4 Sub-Nodes
xmlValue(student1)

# Display values in a Node without going into its Sub-Nodes
xmlValue(root[[4]], recursive = FALSE)
```

- `xmlSApply(NodeTagName, FunctionName)`: recall that an XML tag is like a list in R (but not exactly...)
- When you write `xmlSApply(NodeTagName, function(x) WhatToReturn)`, just imagine looping x over all children of `NodeTagName`. This helps you write the `WhatToReturn` part.
- `xmlChildren(NodeTagName)` lists all children in a parsed format. Its text entry captures the element of this tag. (In this course) you can only query duplicated tag names (except for the first one) by index, not by names.

Once we know how to access any component in an XML file, we can combine them in whatever desired fashion to facilitate further analysis in R.

5. reading multiple files

If you need to read multiple XML files, it is helpful to loop over all files under the directory.

```
# List all files under the current working directory. Specify the extension to filter other files .
list.files(pattern="*.xml")
```