Reading XML

Jeffrey Leek
Johns Hopkins Bloomberg School of Public Health

XML

- Extensible markup language
- Frequently used to store structured data
- Particularly widely used in internet applications
- Extracting XML is the basis for most web scraping
- Components
    - Markup - labels that give the text structure
    - Content - the actual text of the document

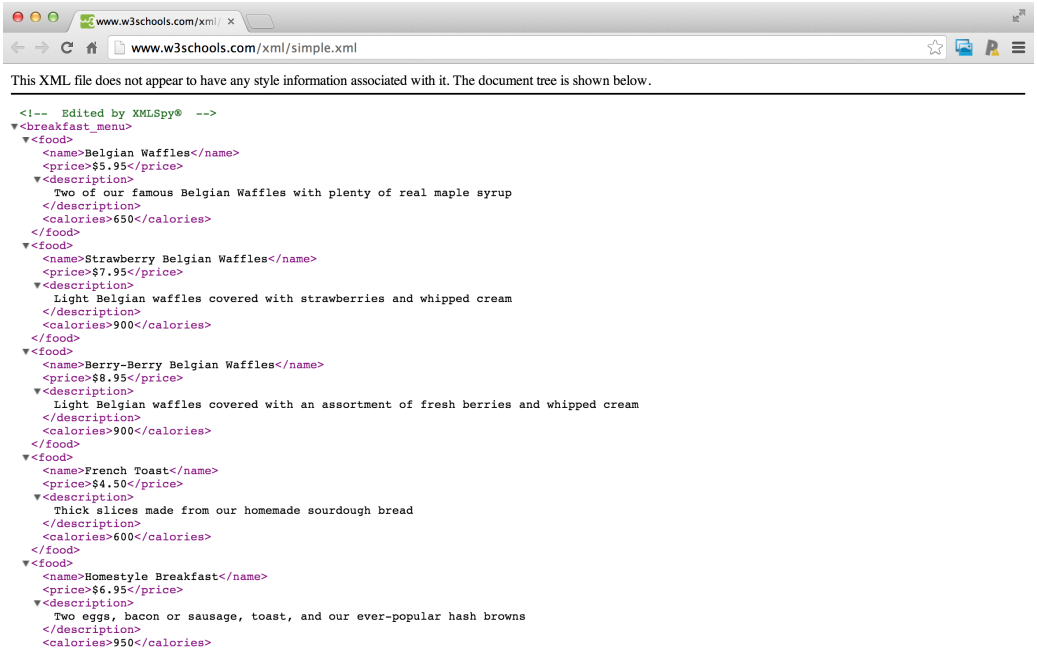
http://en.wikipedia.org/wiki/XML

Tags, elements and attributes

- Tags correspond to general labels
  - Start tags `<section>`
  - End tags `</section>`
  - Empty tags `<line-break />`
- Elements are specific examples of tags
  - `<Greeting> Hello, world </Greeting>`
- Attributes are components of the label
  - `<img src="jeff.jpg" alt="instructor"/>`
  - `<step number="3"> Connect A to B. </step>`

http://en.wikipedia.org/wiki/XML

Example XML file



http://www.w3schools.com/xml/simple.xml

Read the file into R

```
library(XML)
fileUrl <- "http://www.w3schools.com/xml/simple.xml"
doc <- xmlTreeParse(fileUrl,useInternal=TRUE)
rootNode <- xmlRoot(doc)
xmlName(rootNode)
```

```
[1] "breakfast_menu"
```

```
names(rootNode)
```

```
  food   food   food   food   food
"food" "food" "food" "food" "food"
```

Directly access parts of the XML document

```
rootNode[[1]]
```

```
<food>
  <name>Belgian Waffles</name>
  <price>$5.95</price>
  <description>Two of our famous Belgian Waffles with plenty of real maple syrup</description>
  <calories>650</calories>
</food>
```

```
rootNode[[1]][[1]]
```

```
<name>Belgian Waffles</name>
```

Programatically extract parts of the file

```
xmlSApply(rootNode,xmlValue)
```

```
                                "Belgian Waffles$5.95Two of our famous Belgian Waffles with plenty of rea

                          "Strawberry Belgian Waffles$7.95Light Belgian waffles covered with strawberries and

"Berry-Berry Belgian Waffles$8.95Light Belgian waffles covered with an assortment of fresh berries and

                                                    "French Toast$4.50Thick slices made from our homemade so

                          "Homestyle Breakfast$6.95Two eggs, bacon or sausage, toast, and our ever-popula
```

Programatically extract parts of the file

```
xmlSApply(rootNode,xmlValue)
```

```
                                "Belgian Waffles$5.95Two of our famous Belgian Waffles with plenty of rea

                          "Strawberry Belgian Waffles$7.95Light Belgian waffles covered with strawberries and

"Berry-Berry Belgian Waffles$8.95Light Belgian waffles covered with an assortment of fresh berries and

                                                        "French Toast$4.50Thick slices made from our homemade so

                           "Homestyle Breakfast$6.95Two eggs, bacon or sausage, toast, and our ever-popula
```

XPath

- */node* Top level node
- *//node* Node at any level
- *node[@attr-name]* Node with an attribute name
- *node[@attr-name='bob']* Node with attribute name attr-name='bob'

Information from: http://www.stat.berkeley.edu/~statcur/Workshop2/Presentations/XML.pdf 

Get the items on the menu and prices

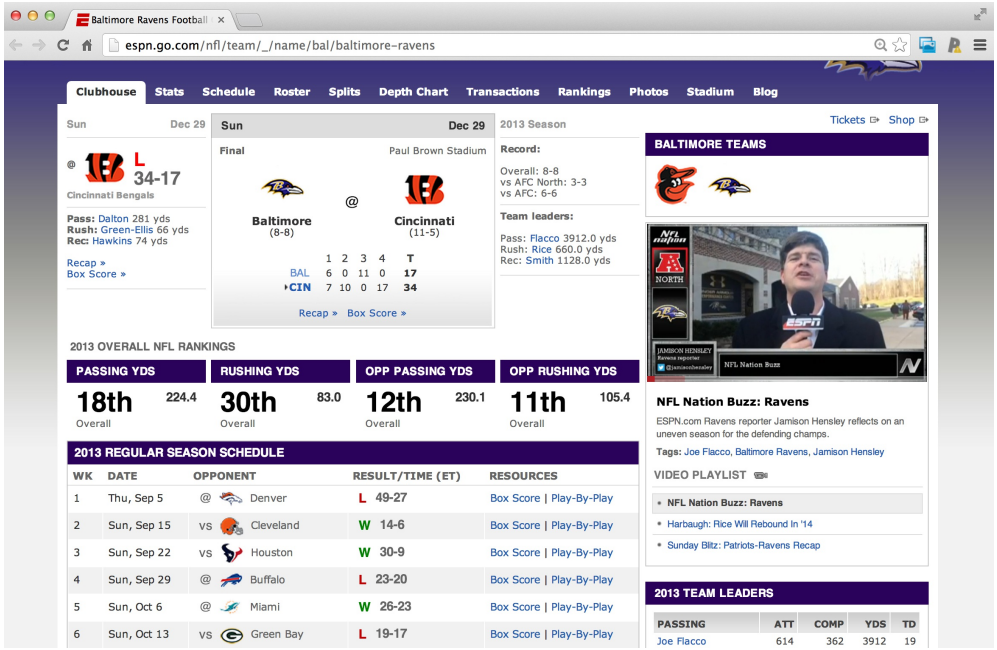
```
xpathSApply(rootNode,"//name",xmlValue)
```

```
[1] "Belgian Waffles"             "Strawberry Belgian Waffles"  "Berry-Berry Belgian Waffles"
[4] "French Toast"                "Homestyle Breakfast"
```

```
xpathSApply(rootNode,"//price",xmlValue)
```

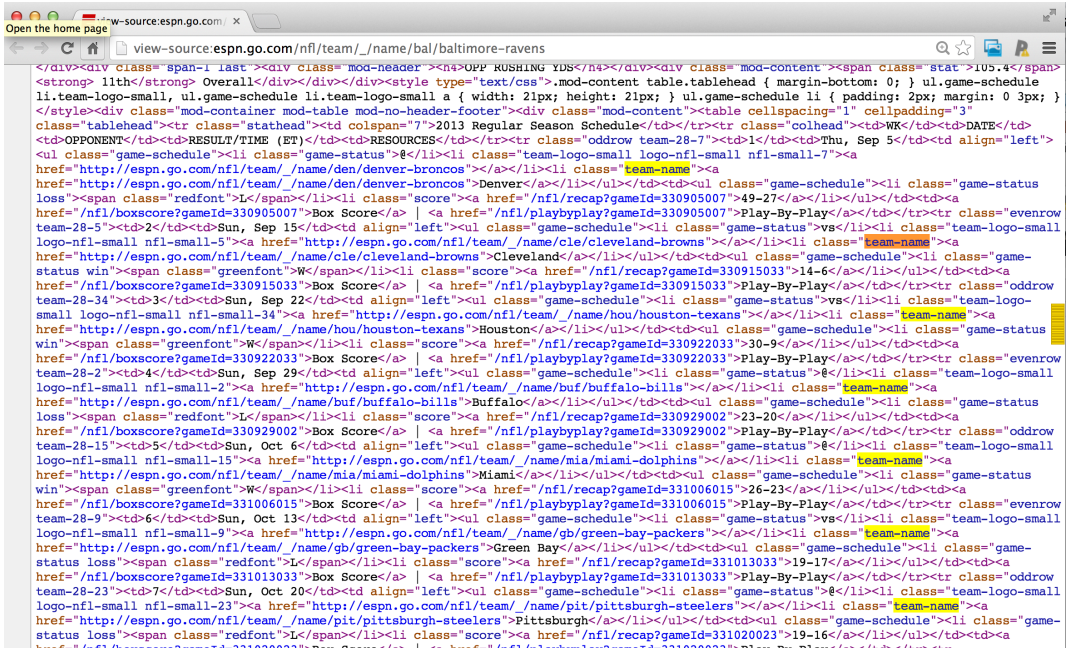
```
[1] "$5.95" "$7.95" "$8.95" "$4.50" "$6.95"
```

Another example



http://espn.go.com/nfl/team/_/name/bal/baltimore-ravens

Viewing the source



http://espn.go.com/nfl/team/_/name/bal/baltimore-ravens

Extract content by attributes

```
fileUrl <- "http://espn.go.com/nfl/team/_/name/bal/baltimore-ravens"
doc <- htmlTreeParse(fileUrl,useInternal=TRUE)
scores <- xpathSApply(doc,"//li[@class='score']",xmlValue)
teams <- xpathSApply(doc,"//li[@class='team-name']",xmlValue)
scores
```

```
 [1] "49-27"    "14-6"     "30-9"     "23-20"    "26-23"    "19-17"    "19-16"    "24-18"
 [9] "20-17 OT" "23-20 OT" "19-3"     "22-20"    "29-26"    "18-16"    "41-7"     "34-17"
```

```
teams
```

```
 [1] "Denver"       "Cleveland"    "Houston"      "Buffalo"      "Miami"        "Green Bay"
 [7] "Pittsburgh"   "Cleveland"    "Cincinnati"   "Chicago"      "New York"     "Pittsburgh"
[13] "Minnesota"    "Detroit"      "New England"  "Cincinnati"
```

Notes and further resources

- Official XML tutorials short, long
- An outstanding guide to the XML package