



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 / API / open data websites
- Components
 - Markup - labels that give the text structure
 - Content - the actual text of the document

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

↳ Web scraping - A computer software technique of extracting information from websites

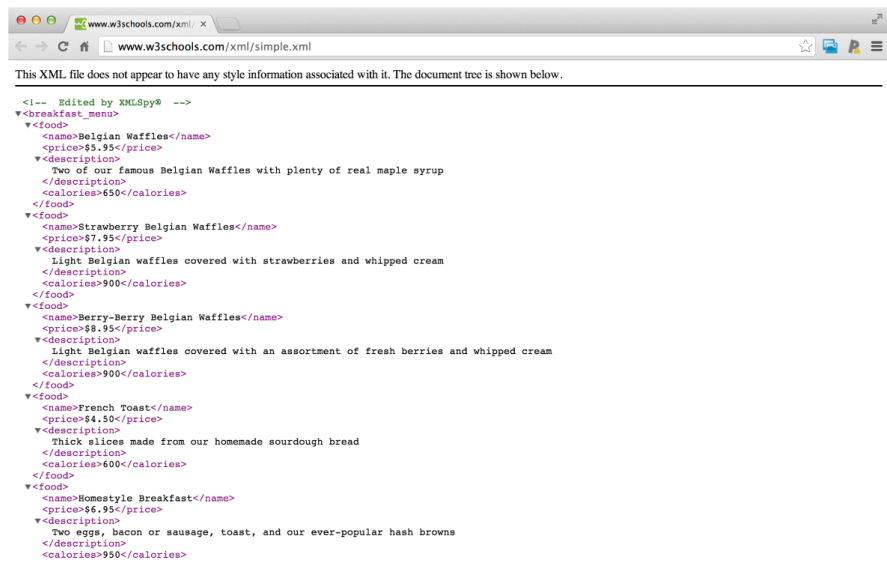
Tags, elements and attributes

- 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
 - ``
 - `<step number="3"> Connect A to B. </step>`

one element

<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"
```

↳ xmlTreeParse parses the XML file

↳ xmlRoot returns the outermost tag, the name of which can be viewed w/ xmlName

→ First nested level

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>
```

Access elements like lists in R
^ subsequent double brackets
will dig deeper

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

5 This will return everything
contained in the given node

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 sc
"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>

Better intro

XPath is language needed to
learn to work w/ XML
↳ Extract specific
components of the XML

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"
```

Gets all of the "name" nodes

Another example

Baltimore Ravens Football

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

ClubhouseStatsScheduleRosterSplitsDepth ChartTransactionsRankingsPhotosStadiumBlog

Sun

Dec 29

34-17

Cincinnati Bengals

Pass: Dalton 281 yds
Rush: Green-Ellis 66 yds
Rec: Hawkins 74 yds

Recap >
Box Score >

Sun

Dec 29

Final

Paul Brown Stadium

Baltimore

(8-8)

Cincinnati

(11-5)

	1	2	3	4	T
BAL	6	0	11	0	17
CTN	7	10	0	17	34

Recap >
Box Score >

2013 Season

Record:

Overall: 8-8
vs AFC North: 3-3
vs AFC: 6-6

Team leaders:

Pass: Flacco 3912.0 yds
Rush: Rice 560.0 yds
Rec: Smith 1128.0 yds

BALTIMORE TEAMS

NFL Nation Buzz: Ravens

ESPN.com Ravens reporter Jamison Hensley reflects on an uneven season for the defending champs.

Tags: Joe Flacco, Baltimore Ravens, Jamison Hensley

VIDEO PLAYLIST

NFL Nation Buzz: Ravens

Harbaugh: Rice Will Rebound In '14

Sunday Blitz: Patriots-Ravens Recap

2013 TEAM LEADERS

PASSING	ATT	COMP	YDS	TD
Joe Flacco	614	362	3912	19

2013 OVERALL NFL RANKINGS

PASSING YDS	RUSHING YDS	OPP PASSING YDS	OPP RUSHING YDS
18th	30th	12th	11th
Overall	Overall	Overall	Overall
224.4	83.0	230.1	105.4

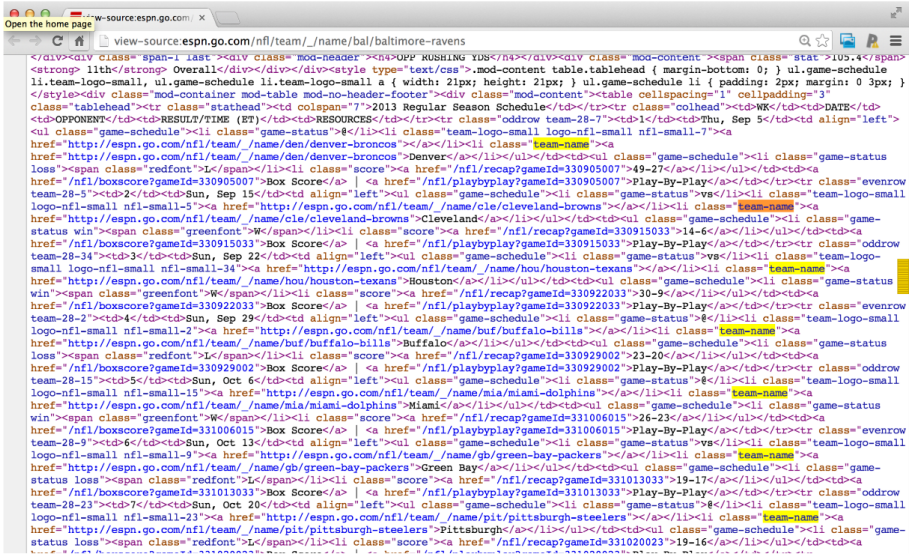
2013 REGULAR SEASON SCHEDULE

WK	DATE	OPPONENT	RESULT/TIME (ET)	RESOURCES
1	Thu, Sep 5	@ Denver	L 49-27	Box Score Play-By-Play
2	Sun, Sep 15	vs Cleveland	W 14-6	Box Score Play-By-Play
3	Sun, Sep 22	vs Houston	W 30-9	Box Score Play-By-Play
4	Sun, Sep 29	@ Buffalo	L 23-20	Box Score Play-By-Play
5	Sun, Oct 6	@ Miami	W 26-23	Box Score Play-By-Play
6	Sun, Oct 13	vs Green Bay	L 19-17	Box Score Play-By-Play

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

11/14

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"
```

↳ Use `htmlTreeParse` b/c not pure XML file
↳ They must've updated their site had to use "`//div[...]`" instead of '`li`', and '`game-info`' instead of '`team-name`'

Notes and further resources

- Official XML tutorials [short](#), [long](#)
- [An outstanding guide to the XML package](#)