

Outline

Background

E4X

The XML package

Create XML

Create XML with R4X brewing distilling Adding content Import

Manipulate XML

Manipulate XML with R4X XPath-like Adding content

Examples

RSS Reader Tag cloud

References

The R4X package

Romain François

Background

The XML packa

Create XML

Create XML with R4) brewing distilling

Import

Manipulate XI

Manipulate XML with R4X XPath-like

Adding cont

RSS Reader

lag cloud

```
F4X
```

Ecmascript (Javacript) for XML. Example from Wikipedia: http://en.wikipedia.org/wiki/E4X

The R4X package

Romain François

Background

E4X

The XML packa

Create XML
Create XML with

orewing distilling

Import

Manipulate XML

Manipulate XML with R4X XPath-like

Adding cor

RSS Reader

ag cloud

This package provides facilities for the S language to

- parse XML files, URLs and strings, using either the DOM (Document Object Model)/tree-based approach, or the event-driven SAX (Simple API for XML) mechanism;
- parse HTML documents,
- perform XPath queries on a document,
- generate XML content to buffers, files, URLs, and internal XML trees;
- ▶ read DTDs as S objects.

The R4X package

Romain François

Background F4X

The XML package

Create XML
Create XML with R4X
brewing

Import

Manipulate XML

Manipulate XML with R4>

Adding con

RSS Reader

ag cloud

The XML Package in 3 slides

Creating XML content

```
> x <- xmlNode( "test",
     xmlNode( "bar", attrs = c( fruit = "mango") ),
     xmlNode( "bar", attrs = c( fruit = "apple" )),
  attrs = c(type="foo"))
> x
<test type="foo">
 <bar fruit="mango"/>
 <bar fruit="apple"/>
</test>
> class(x)
[1] "XMLNode"
```

The R4X package

Romain François

Background

The XML package

Create XML
Create XML with R4

brewing distilling

Import

Manipulate XML

Manipulate XML with R4X XPath-like

Evamples

RSS Reader

The XML package in 3 slides

Append content to an XML structure, the addChildren function

```
> x
<test type="foo">
 <bar fruit="mango"/>
 <bar fruit="apple"/>
</test>
> addChildren(x,
    xmlNode( "bar", attrs =
      c(fruit = "pineapple")))
<test type="foo">
 <bar fruit="mango"/>
 <bar fruit="apple"/>
 <bar fruit="pineapple"/>
</test>
```

The R4X package

Romain François

Background

The XML package

Create XML with R4X brewing distilling

Import

Manipulate XML

XPath-like

Examples

Tag cloud

The XML package in 3 slides

Query content of an XML structure

```
> # The "fruit" attribute of the first child of x
> xmlAttrs(xmlChildren(x)[[1]], "fruit")
  fruit
"mango"
> # The "fruit" attribute of each child of x
> xmlApply( x, xmlAttrs, "fruit" )
$bar
  fruit
"mango"
$bar
  fruit
```

The R4X package

Romain François

Background

The XML package

Create XML with Pd

brewing

Adding co

Import

Manipulate XML

XPath-like

RSS Reader

-

References

"apple"

The xml generic function

The default method tries to convert strings into XML nodes, including nested nodes. Remember: Strings can be multiline in R.

```
> y <- xml( '<test><foo blah="1"/><bar/></test>')
> y <- xml( '
    <test>
       <foo blah="1"/>
       <bar/>
    </test>
> y
<test>
<foo blah="1"/>
<bar/>
</test>
> class( y )
[1] "XMLNode"
```

The R4X package

Romain François

Background

The XML package

-

Create XML with R4X

rewing istilling

Adding cont

Import

Manipulate XMI

Manipulate XML with R4X

Adding con

Examples

SS Reader

Dynamic content with brew

The brew package provides a psp-like templating framework for R. The $\mbox{\ensuremath{^{\prime\prime}}}=$ operator is used by R4X to add dynamic content without having to use paste or sprintf.

```
> f <- c("mango", "apple", "strawberry" )</pre>
> x <- xml( '
    <fruits>
      <fruit><%= f[1] %></fruit>
      <fruit><%= f[2] %></fruit>
      <fruit><%= f[3] %></fruit>
    </fruits>
  1)
> x
<fruits>
<fruit>mango</fruit>
 <fruit>apple</fruit>
 <fruit>strawberry</fruit>
</fruits>
```

The R4X package

Romain François

Background

The VMI and

Create XIVIL

eate XML with F

brewing

Adding cont

Import

Manipulate XI

Manipulate XML with R4

Adding cor

Examples

ag cloud

More brewing

The full rfunbrew syntax can be used as well as just the <%=, but it can become quickly difficult to manage. See the distilling feature for stronger taste...

```
> x <- xml( '
    <fruits>
      <%for( i in f) {%>
        <fruit><%= i %></fruit>
      <%}%>
    </fruits>
> x
<fruits>
 <fruit>mango</fruit>
 <fruit>apple</fruit>
 <fruit>strawberry</fruit>
</fruits>
```

The R4X package

Romain François

Background

The XMI nacka

Create XML

Create XML with I

istilling

Import

Manipulate XML

Manipulate XML with R4X

Adding cor

Examples

RSS Reader

For stronger taste, distill rather than brew

Distilling consists of generating some brew templates and let brew do the hard work. The simplest way to use the distill feature is by wrapping R code to curly braces 1 .

```
> x <- xm1( txt <- '
    <fruits>
      <fruit>{f[1]}</fruit>
    </fruits>
> x
<fruits>
 <fruit>mango</fruit>
</fruits>
> cat( distill( txt ) )
 <fruits>
    <fruit><%=f[1]%></fruit>
 </fruits>
```

Romain François

Background

The XML packa

Create XML with R4X

distilling

Adding content

Manipulate XML

XPath-like

Examples

SS Reader ag cloud

The R4X package

¹Obviously this means that the code can't contain any curly braces, but we can live without surely.

loop generators

distill can also generate brew code to loop a variable from 1 to another variable using the special <0> tag. Because brew is so well-designed, the loop generators can be nested with no problem. See xml.matrix for an example.

Distilling Tag	Corresponding brew code	
<@i n>	<% for(i in 1:n){ %>	
<@i~x>	<% for(i in x){ %>	
<@i?y>	<pre><% for(i in seq(along=y)){ %></pre>	
@	<%}%>	

The R4X package

Romain François

Background

E4X

reate XML

rewing

distilling

Adding conto

Import

Manipulate XML

Manipulate XML with R4X

Adding cont

kamples

ng cloud

The XML package

Create XML
Create XML with R4>

brewing

distilling Adding content

Adding content Import

Manipulate

Manipulate XML with R42

Adding cont

RSS Reader

```
> cat( txt )
 <fruits>
    <@i|length(f)>
      <fruit>{f[i]}</fruit>
    </@>
  </fruits>
> cat( distill( txt ) )
 <fruits>
    <% for( i in 1:length(f)){%>
      <fruit><%=f[i]%></fruit>
    <%}%>
  </fruits>
> ( x <- xml( txt ) )</pre>
<fruits>
 <fruit>mango</fruit>
 <fruit>apple</fruit>
 <fruit>strawberry</fruit>
</fruits>
  solutions
```

<fruits>
<fruit>mango</fruit>
<fruit>apple</fruit>
<fruit>strawberry</fruit>
<fruit>blueberry</fruit>
<fruit>raspberry</fruit>
</fruit>raspberry</fruit></fruits></fruits></fruits></fruits></fruits></fruits></fruits></fruits></fruits></fruits></fruits></fruits></fruits></fruits></fruits></fruits></fruits></fruits></fruits></fruits></fruits></fruits></fruits></fruits></fruits></fruits></fruits></fruits></fruits></fruits></fruits></fruits></fruits></fruits></fruits></fruits></fruits></fruits></fruits></fruits></fruits></fruits></fruits></fruits></fruits></fruits></fruits></fruits></fruits></fruits></fruits></fruits></fruits></fruits></fruits></fr>

solutions

4 D > 4 B > 4 E > 4 E > 9 Q P

The R4X package

Romain François

Background

The XML packag

Create XML with R4

distilling

Adding content

Import

Manipulat

Manipulate XML with R42

Adding co

Example:

Tag cloud

Importing XML from connections

Most importantly files or url connections, but why not a pipe, ...

```
> cat(readLines("test.xml"), sep = "\n")
<test type="foo">
 <bar fruit="mango"/>
 <bar fruit="apple"/>
 <bar fruit="pineapple"/>
</test>
> x <- xml(file("test.xml"))</pre>
> x
<5/>
> xml(url("http://www.example.com/test.xml"))
```

The R4X package

Romain François

Background

The XML packag

reate XML Create XML with R4X

ewing stilling

Import

Manipulate

Manipulate XML with R4X

Adding con

RSS Reader

Tag cloud

Example XML Structure

We will use this simple XML structure to demonstrate the slicing of objects of class XMLNode.

```
<root>
 <child id="1">
 <subchild id="sub1">foo</subchild>
 <subchild id="sub2">bar</subchild>
 </child>
 <child id="2">
 <subchild id="a">blah</subchild>
 <subchild id="b">bob</subchild>
 <something id="c"/>
 </child>
 <fruits>
 <fruit>banana</fruit>
 <fruit>mango</fruit>
 </fruits>
</root>
```

The R4X package

Romain François

ackground

The XML package

Create XML

Create XML with R4X

rewing

Adding co

Import

Manipulate XML

Manipulate XML with R4X

Adding content

Examples

RSS Reader Tag cloud

XPATH-like syntax

R4X defines an XPAT $\check{\mathsf{H}}$ -like syntax to manipulate XML structures with the usual R extractors [and [[

	-	
path expression		LL
"child"	list	XMLNode
"child/subchild"	list	XMLNode
"child/subchild/#"	vector	vector
"child/subchild/#n"	numeric vector	numeric vector
"child/@id"	vector	vector
"child//@id"	vector	vector
"child/~sub.*"	list	XMLNode
"fruits"	XMLNode	XMLNode

Table: Classes of result for various path expressions.

The R4X package

Romain François

Background

The XML packag

Create XML

Create XML with R4X brewing distilling

Import

Manipulate XML

XPath-like

Adding content

SS Reader

slicing with [[

The double square bracket [[behaves similarly as for lists, it gives back a single object

```
> x[[ "child" ]] # XMLNode, the first one
<child id="1">
 <subchild id="sub1">foo</subchild>
 <subchild id="sub2">bar</subchild>
</child>
> # XMLNode, first <subchild> of first <child>
> x[[ "child/subchild" ]]
NULL
> x[[ "child/subchild/#" ]] # character vector
NUIT.I.
> x[[ "child/subchild/@id" ]] # character vector
```

The R4X package

Romain François

Background

The XML packag

reate XML

rewing

Import

Manipulate XML

Manipulate XML with R4>
XPath-like

Adding conten

Adding conten

SS Reader

References

NUIT.I.

slicing with [

The *single* square bracket [gives an XMLNode or a list of XMLNode if the path matches more than one node

```
> x[ "child" ] # mutiple <child> : list of XMLNode
$child
<child id="1">
 <subchild id="sub1">foo</subchild>
 <subchild id="sub2">bar</subchild>
</child>
> x[ "fruits" ] # single <fruits> : XMLNode
$fruits
<fruits>
 <fruit>banana</fruit>
 <fruit>mango</fruit>
</fruits>
```

The R4X package

Romain François

Background

The XML package

Create XML
Create XML with R42

prewing

Adding co

Manipulate XML

Manipulate XML with R4

XPath-like

Adding conte

RSS Reader

Tag cloud

Appending content with [[<-.XMLNode

The [[extractor also works to add content to an XML structure using the XPath-like expressions.

```
> y <- xml( '<test><foo/></test>' )
<test>
 <foo/>
</test>
> type <- "foo-bar"
> y[[ "foo/bar/test" ]] <- '<test type="{type}" />'
> y
<test>
 <foo/>
[1] "<test type=\"{type}\" />"
</test>
```

The R4X package

Romain François

Background

The XML packag

reate XML Create XML with R4X

orewing

Adding co

Import

Manipulate

Manipulate XML with R4X

Adding content

Example

Fag cloud

Appending content with [<-.XMLNode

This is not implemented yet. Appending content is only implemented for the double square bracket extractor at the moment.

```
> result <- try(
    y[ "foo/blah" ] <- '<blah />' , silent = TRUE )
> class(result)
[1] "character"
> strwrap( result, 40 )
[1] "<blah />"
```

The R4X package

Romain François

Background

The XML packag

Create XML with R4.
brewing
distilling

Import

Manipulate XML

Manipulate XMI with R4X

XPath-like

Adding content

RSS Reader

Tag cloud

RSS: R Site Summary

Definition of RSS from the w3c. See http://www.w3schools.com/rss for more information.

- RSS stands for Really Simple Syndication
- RSS allows you to syndicate your site content
- RSS defines an easy way to share and view headlines and content
- RSS files can be automatically updated
- RSS allows personalized views for different sites
- ▶ RSS is written in XML

The R4X package

Romain François

Background

The XML package

Create XML

brewing distilling

Import

Manipulate XML

Manipulate XML with R4X XPath-like

Adding conte

Examples

RSS Reader

ag cloud

RSS: Example

Example RSS feed from http://www.w3schools.com/rss.

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<rss version="2.0">
<channel>
 <title>W3Schools Home Page</title>
 <link>http://www.w3schools.com</link>
 <description>
     Free web building tutorials
 </description>
 <item>
    <title>RSS Tutorial</title>
    <link>http://www.w3schools.com/rss</link>
    <description>
        New RSS tutorial on W3Schools
    </description>
 </item>
</channel>
</rss>
```

The R4X package

Romain François

Background

The XML package

Create XML

rewing

Adding co

Import

Manipulate XML

Manipulate XML with R4

Adding conte

Examples

RSS Reader

cloud

Fetching data from Planet R

planet R (http://planetr.stderr.org/) aggreggates information from several R related sites as an RSS feed.

```
> planetr <- xml(
    url( "http://planetr.stderr.org/rss10.xml ") )
> titles <- planetr[ "item/title/#" ]
> cat( strwrap( head( titles, 4 ), 50, exdent = 4 ),
    sep = "\n" )
```

The R4X package

Romain François

ackground

The XML package

Create XML

rewing istilling

mport

Manipulate XML

Manipulate XML with R4X XPath-like

Adding cont

examples

RSS Reader

ag cloud

References

NULL

Tag cloud

◆□ → ◆周 → ◆三 → ◆三 → ◆9 ◆

Generating a simple tag cloud. See the operators package for details. Generated with the following script from the words used in all descriptions of R packages.

```
> all <- casefold( readLines( "descriptions.txt" ) )</pre>
> all <- all %s~% "/[^\\w\\s]//pg" %/~% "\\s+"
> all <- all %without% commonWords
> tab <- rev( sort( table( all ) ) )[1:250]
> words <- names(tab)
> for( word in words ){
   if( ( plural <- sprintf("%ss", word) ) %in% words ) {
     tab[word] <- tab[word] + tab[plural]
     tab[plural] <- 0
> tab <- tab[ tab != 0 ]
> tab <- tab[ sort(names(tab)) ]</pre>
> ncuts <- 8
> sizes <- as.numeric( cut ( tab, ncuts ) )</pre>
> refs <- round( seq( 10,24, length=ncuts) )</pre>
> words <- names(tab)
```

Tag cloud

Generating a simple tag cloud. R4X code to write the html page.

```
> tags <- xml( '
    <html>
      <head>
      <style type="text/css">
      <@ilncuts>
            .cl{i}{
               font-size:{refs[i]}pt;
      </@>
      </style>
      </head>
      <body>
        <@i|length(tab)>
          <span class="cl{sizes[i]}">{words[i]}</span>
        </@>
      </body>
    </html>')
    tags %>% "tags.html"
```

The R4X package

Romain François

Background

The XML package

Create XML

Create XML with R4X

brewing

Adding co

Import

Manipulate

Manipulate XML with R4X

Adding cor

Examples

Tag cloud

ag ciouu

1.2 al algorithm allows analyses analySIS applications applied approach arbitrary association available basic bayesian binary book bootstrap c calculate calculation carlo censored chain class classes classification cluster clustering code collection common components computation computational compute computing conditional confidence control correlation count covariates create currently curves data database datasets density described design designed detection different discrete display distance distribution eg either engineering environment error estimate estimation estimator et etc exact examples experiments features file finance financial first fit fitting framework function functionality gaussian gene general generalized genetic graph graphical graphics group gui hazard hierarchical if implementation implemented implements include included including independent inference information interface intervals its kernel large level library likelihood linear local logistic main manipulating map markov matrices matrix maximum may mean measures method microarray missing mixture model modeling modelling monte most multiple multivariate network nonlinear nonparametric normal number object observations order output package parameter parametric perform plot plotting point population possible power probability problems procedure process processes program programming proportional provide provided quantitative r random regression related response results risk robust routines s sample sampling selection series set simple simulation single smoothing so software spatial specified splus squares standard statistical statistics structure support survival system teaching test testing theory through time tools trees univariate useful user uses USINg utilities utility value variable variance various vector version very wavelet weighted without written



References

XML references from W3C:

- ► E4X: http://www.w3schools.com/e4x/default.asp
- RSS: http://www.w3schools.com/rss/default.asp

R References

- \blacktriangleright XML $(\hat{\Omega})$. http://www.omegahat.org/RSXML/
- brew: http://www.rforge.net/brew/
- ▶ operators: http://www.mango-solutions.com

The R4X package

Romain François

