R2X – R to XML bridge

Johannes Willkomm

May 10, 2020

Contents

1 Introduction

```
library(r2x)
```

Passing a simple named list with all scalar values results in a plain XML document structure:

```
read_xml(r2x(list(a=1,b=2,c='test')))
```

```
{xml_document}
<r2x>
[1] <a>1</a>
[2] <b>2</b>
[3] <c>test</c>
```

Attributes on R values are mapped to XML attributes:

```
struct <- list(a=structure(1, aa='f'),b=2,c='test',
   d=structure('', n=1,m=2))
doc <- read_xml(r2x(struct))
doc
```

```
{xml_document}
<r2x>
[1] <a aa="f">1</a>
[2] <b>2</b>
[3] <c>test</c>
[4] <d n="1" m="2"/>
```

The inverse operation is to convert an XML document back to an R structure

```
writeLines(deparse(x2r(doc)))
```

```
list(a = structure(1, aa = "f"), b = 2, c = "test", d = structure("", n = 1, m = 2))
```

This deparsed code looks somewhat unwieldy at first, in particular because the attributes are listed after the element content in this code. The code that x2r generates internally is designed to appear somewhat more readable. This code is available via the function r2x_deparse or via the deparse method overload for XML documents provided by r2x:

```
writeLines(deparse(doc))
```

The resulting code is

```
r2x <- list(
    ‘a‘ = element(‘aa‘ = 'f',
                  '1'),
    ‘b‘ = '2',
    ‘c‘ = 'test',
    ‘d‘ = element(‘n‘ = '1', ‘m‘ = '2')
)
```

This code uses the helper function element to create a value from the last argument with attributes listed as the named preceding arguments. This results in a notation somewhat more resembling the XML code.

As a small example what could be done from R with the r2x transformation, how about defining a HTML5 document in R:

```
  htmldef <- list(
      head = list(
          title = 'Test',
          style = '
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣h2␣{␣color:␣blue;␣}
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣'
      ),
      body = list(
          div = list(
              h2 = 'First␣Section',
              p = 'First␣paragraph.'
          )
      )
  )

html <- r2x(htmldef,
            name = 'html',
            namespace = 'http://www.w3.org/1999/xhtml')

as.html.file <- function(html) {
    writeLines(html, con = (outfile <- tempfile('rep', fileext =
       '.html')))
    outfile
}
```

```
viewer <- getOption('viewer')
viewer(as.html.file(html))
```

The way is also paved to define XSLT transformations and possibly even entire XSLT pipelines in R. As an example consider the following R script which defines both the XML document and the XSLT stylesheet and performs the XSLT transformation

```
copy_xsl <- element(
    version = '1.0',
    val = list(
        `xsl:output` = element(
            method = 'xml'
        ),
        `xsl:template` = element(
            match = '/',
            val = list(
                `xsl:apply-templates` = element(
                    select = 'node()'
                )
            )
        ),
        `xsl:template` = element(
            match = '@*|node()',
            val = list(
                `xsl:copy` = list(
                    `xsl:apply-templates` = element(
                        select = '@*|node()'
                    )
                )
            )
        )
    )
)

as.xslt <- function(xsldef) {
    read_xml(r2x(xsldef,
                 name = 'xsl:stylesheet',
                 namespaces = list(xsl =
                     'http://www.w3.org/1999/XSL/Transform')))
}

example_xml <- element(a=1,b=2,c=3,
                       val=list(
                           e1 = element(a=2,b=3,c=4,
                                        val=list(e2 =
                                            element(a=2,b=3,c=4)))))

xslt_doc <- as.xslt(copy_xsl)
```

```
xml_doc <- read_xml(r2x(example_xml))
result <- xml_xslt(xml_doc, xslt_doc)

identical(r2x_deparse(xml_doc),
          r2x_deparse(result))
```

This code produces as output the value of the last expression, which is TRUE, meaning the transformed structure is identical to the original one.

```
[1] TRUE
```

Both examples are arguably quite technical. It certainly is not recommendable to write XSL or HTML5 directly from R in this way. What R2X is potentially much more useful for, is for accessing information from XML documents and to generate the bits and pieces of XML data that are needed when generating dynamic documents from a template. There, you would have a template such as a HTML or OpenOffice document such as an invoice, plus a XSLT stylesheet that injects dynamic information into such a document. The dynamic info that the transformation needs is some adhoc XML format that can be easily generated from whatever DB script. Exactly this could be done with R and R2X too, and quite conveniently.

1.1 Limitations

Currently, R2X can only handle XML documents where any meaningful text node is a leaf in the tree. That is, text nodes that are siblings with elements are ignored by the `x2r` transformation. In our experience, XML documents often tend to have this leaf-text structure, which has the advantage that they can be reformatted, and pretty-printed with indentation, among others. As a counter example, HTML is not in leaf-text form in general.