

R2X – R to XML bridge

Johannes Willkomm

April 24, 2020

Contents

1	Introduction	1
----------	---------------------	----------

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))
```

While this code looks somewhat unwieldy, 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))
```

```
r2x <- list(
  'a' = element('aa' = 'f',
               val = '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 **val** with attributes listed as the named preceding arguments. This results in a notation somewhat more resembling the XML code.

The way is now 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
```

However, if the suggested notation is really more convenient to use than the XSLT syntax in XML format remains to be seen. While the use of the helper function `element` and careful code arrangement and indentation clearly goes a long way towards readability of the XSLT, it still looks a little bit more verbose than the XML version, it seems to me.