An OpenAPI pipeline for transformable documents

Ashley Noel Hinton
ahin017@aucklanduni.ac.nz

Paul Murrell
paul@stat.auckland.ac.nz

Department of Statistics, The University of Auckland

In A transformable markup document format the authors proposed that a markup `document` format written using XML would make a good source format for authoring documents to be transformed to various output formats, including HTML and PDF. We suggested OpenAPI pipelines could provide a means of managing multiple transformations on these documents. This reports describes how an OpenAPI pipeline provides a sensible means of managing document transformation, and describes two example transformation pipelines.

1 Using OpenAPI pipelines for transformation

The OpenAPI architecture helps to break tasks in data analysis down into small pieces making it easier for people to contribute to a data problem. The goal of the OpenAPI project is to make it easier for people to connect with data. Meaningful steps in a data workflow can be wrapped as modules. Modules can be arranged in pipelines, and shared to be recombined by other authors in their own pipelines. Pipelines and modules can be executed by OpenAPI glue system software. The whole project is open source, and open to contributions from anyone (Introducing OpenAPI, OpenAPI version 0.6).

It is OpenAPI's dividing of tasks into modules which makes it an ideal candidate for handling document transformation. In A transformable markup document format we described how the author of a transformable markup `document` author may wish to perform multiple discrete transformations. For example, an author wishing to produce an HTML document may wish to performm the following transformations:

1. Merge XML from external documents indicated by `<xi:include>` elements.

2. Convert the document to Knitr HTML.

3. Process the code chunks in the Knitr HTML to produce and HTML output.

Several technologies already exist for handling each of these steps. For example, the `xmllint` (`http://www.xmlsoft.org/`) command line tool can substitute XInclude code. The `xsltproc` command line tool (`http://www.xmlsoft.org/`) can be used to apply an XSL stylesheet to the `document` to produce Knitr HTML. The Knitr package in R can be used to execute chunks of R code and produce HTML output.

What an OpenAPI pipeline offers is the ability to wrap a transformation step in a module which takes a file as an input, and produces another file as an output. The output of one module can be passed as the intput of another module, thus building a pipeline which describes the entire transformation. Each module in an OpenAPI pipeline specifies its execution language, meaning an OpenAPI pipeline can have access to a wide variety of tools.

The following sections describes the transformable markup `document` used for the transformation pipelines described in this report. This report itself was authored using this `document` format, and the final output was produced by the example pipelines. Two example pipelines follow the description of the `document` format: the first transforms the source , the second transforms the source to PDF.

2 The `document` markup format

The transformable `document` format described in this report is largely the same as that described in A transformable markup document format. The source `documeent` is an XML file with `document` as the root element. This document has two child elements: `metadata` and `body`.

The `metadata` element contains the document metadata, with elements for the document `title` and `subtitle`, `author` information, `date` of publication, and a `description` section. An example `metadata` element follows:

```
<metadata>
  <title>Today should be a holiday</title>
  <author>
    <name>Ashley Noel Hinton</name>
    <email>ahin017@aucklanduni.ac.nz</email>
  </author>
  <date>25 December 2015</date>
</metadata>
```

The `body` element contains the document's main content. The following elements are used in the same way as they are used in HTML (`https://www.w3.org/TR/html-markup/elements.html`):

- `a` – hyperlink
- `code` – code fragment
- `figcaption` – figure caption
- `figure` – figure with optional caption
- `h1` – heading
- `h2` – heading
- `h3` – heading
- `img` – image
- `li` – list item
- `ol` – ordered list
- `p` – paragraph
- `pre` – preformatted text
- `q` – quoted text
- `section` – section
- `ul` – unordered list

The `<url>` element is introduced in the `document` format to indicate a hyperlink where the enclosed URL is both the href and the value. The following code block demonstrates the use of the `url` element:

```
<ul>
  <li>modular</li>
  <li>reusable</li>
  <li>shareable</li>
  <li><url>https://github.com/anhinton/conduit</url></li>
</ul>
```

The resulting output:

- modular
- reusable
- shareable
- `https://github.com/anhinton/conduit`

The `document` XML format uses `<code>` elements to indicate blocks of computer code, just as in HTML. Dynamic code chunks which are to be executed are marked using the `class` attribute to `code`. For example chunks of R code which are to be executed used the Knitr package are wrapped in a `<code>` element with `class="knitr"`. An author can also provide a `name` attribute for the knitr code chunk, as well as knitr `options`. A document author can also use `CDATA` sections to wrap code with reserved XML characters. The following code demonstrates how to include an R code chunk to be executed with Knitr:

```
<code class="knitr" name="knitrDemo" options="tidy=FALSE">
<![CDATA[x <- rnorm(n = 10)
mean(x)]]></code>
```

The following is the result of executing this R code chunk:

```
x <- rnorm(n = 10)
mean(x)
```

```
## [1] 0.1132167
```

The `document` format also makes use of the `include` element from XInclude (`http://www.w3.org/2001/XInclude`) namespace to include XML content from external files. This allows `document` authors to embed other documents which may be authored separately from the main document.

As the `document` format used in this report will be used to produce both HTML and PDF output, XML entities have been used to represent special characters which require specific representations in either output language. The following entities are defined in the doctype declaration of the source `document`. These entities are provided with values appropriate for HTML by default; these values are substituted for LaTeX values as part of the transformation to PDF:

- `mdash`— em dash, a typograhical dash character.
- `ndash`– en dash, a typograhical dash character.
- `pcnt`% percent sign.

The next sections describes some simple transformations which can be performed on the `document` markup format using freely available open source tools. This report was itself written in the `document` markup format—the source code is available at `report.xml`.

3 OpenAPI transformation pipelines

In this section we will descibe the two pipelines, and their modules, used to transform a tranformable `document` file to HTML output, and to PDF output, respectively. Both transformation pipelines include executing chunks of embedded R code. .

3.1 Transformation to HTML

The first pipeline example, `toHtml`, converts a `document` source file to an HTML file. The pipeline also executes the R code embedded in the source document. Figure 1 shows a graph of the `toHtml`, which consists of four modules (ellipse-shaped nodes). The source `document`, report.xml, is provided as in input to the pipeline, and the ouptut document, report.html, is produced as an output from the final module. The source XML for the `toHtml` pipeline can be found at `transform/toHtml/pipeline.xml`.

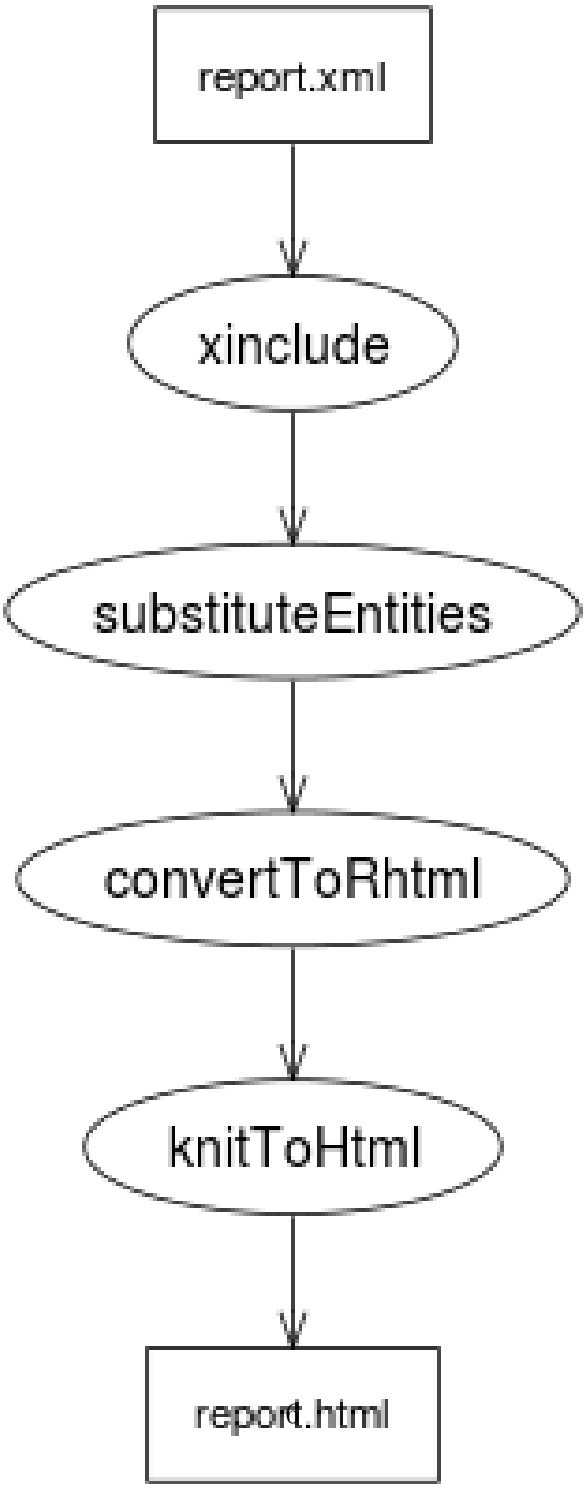
The following sections describe the details of the modules used in the `toHtml` pipeline.

3.1.1 Include referenced XML

The first module in the `toHtml` pipeline is the `xinclude` module, which processes all of the `<xi:include>` elements in the source document and produces a document with the reference XML documents inline. The `xinclude` module is a bash-language module, and wraps a bash script which calls the `xmllint` command-line tool to perform the transformation. This module requires four input objects: report which references the source document, `report.xml`; metadataExample which references the XML file `metadataExample.xml`; elementsExample which references the XML file `elementsExample.xml`; and knitrExample which references the XML file `knitrExample.xml`. The module produces a single output, report, which references the transformed XML file produced by the transformation. The XML source for the `xinclude` module can be found at `transform/toHtml/xinclude.xml`.

3.1.2 XML entities

The second transformation module in the pipeline is the `substituteEntities` module. This module replaces the XML entities in the source document with the values provided in the document's doctype declaration. The `substituteEntities` module is a bash-language module with one input, report, an XML file provided by the `xinclude` module. This module wraps a script which uses the `xmllint` command-line tool to replace the source document's XML entities with their values. The module produces one output, report, which references the transformed source XML file. The XML source for the `substituteEntities` module can be found at `transform/toHtml/substituteEntities.xml`.


report.xml
xinclude
substituteEntities
convertToRhtml
knitToHtml
report.html

3.1.3 Produce `.Rhtml` file

The third module in this pipeline is the `convertToRhtml` module. This module transforms the source document into an HTML document with Knitr R code chunks. This module is a bash-language module with two inputs: report, an XML file provided by the `substituteEntities` module; and toRhtml, which references an XSLT stylesheet file `xsl/toRhtml.xsl`, which describes the transformation. The module wraps a script which uses the `xsltproc` command-line tool to transform the source document into a Knitr HTML file. The module produces one output, report, which references the Knitr HTML file produced in the transformation. The XML source for the `convertToRhtml` module can be found at `transform/toHtml/convertToRhtml.xml`.

3.1.4 Produce `.html` file

The fourth module in the `toHtml` pipeline is the `knitToHtml` module. This module executes the R code chunks in a Knitr HTML file and returns the resulting HTML file. This module is an R-language module with one input, report, a Knitr HTML file provided by the `convertToRhtml` module. The module wraps an R script which calls the `knit` function from the Knitr package to execute the R code chunks, and returns an HTML file. The module produces three outputs: report, which references the HTML file produced by the module source script; toHtmlGraph, a PNG image file; and toPdfGraph, a PNG image file. The XML source for the `knitToHtml` module can be found at `transform/toHtml/knitToHtml.xml`.

3.2 Transformation to PDF

The second pipeline example, `toPdf`, converts a `document` source file to a LaTeX file. This pipeline also converts the source document to an HTML file, as in the toHtml. In fact, the `toPdf` pipeline is an extension of the `toHtml` pipeline. Figure 2 shows a graph of the `toPdf` pipeline. The ellipse-shaped nodes show the same modules as in the `toHtml` pipeline. The hexagon-shaped nodes show the six new modules which are used to produce PDF output. This pipeline will produce the output document report.pdf as a module output, as well as report.html as before. The XML source for the `toPdf` pipeline can be found at `transform/toPdf/pipeline.xml`.

The following sections describe the details of the modules used in the `toPdf` pipeline.

3.2.1 Substitute LaTeX entity values

The first new module in the `toPdf` pipeline is the `latexChars` module. This module replaces the entity definitions in the source `document` doctype

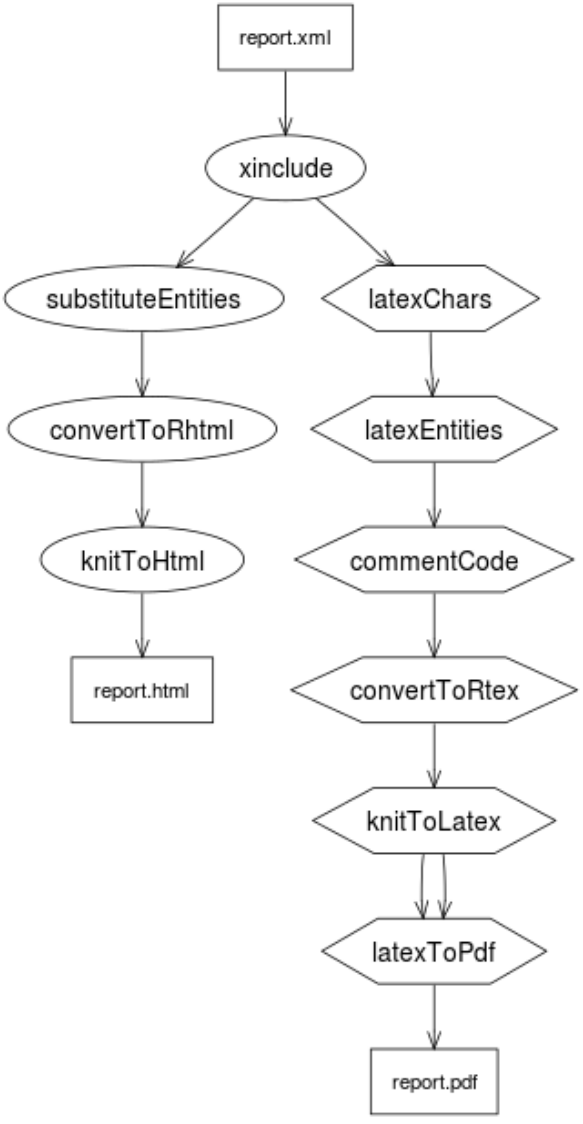


Figure 2: Figure 2: the toPdf pipeline

declaration with values appropriate to LaTeX character typesetting—the source document entity values are appropriate to HTML. This module is a bash-language module with one input, report, an XML file provided by the `xinclude` module. The module wraps a bash script which substitutes the entity values using the `sed` command-line tool. The module produces one output, report, which references the transformed XML file. The XML source for the `latexChars.xml` module can be found at `transform/toPdf/latexChars.xml`.

3.2.2 XML entities

The `latexEntities` modules is identical to the `substituteEntities` module in the `toHtml` pipeline. It takes its input from the `texChars` modules, and produces the output report, which references the transformed source document. The source XML for `latexEntities` is the same as for `substituteEntities`, `transform/toHtml/substituteEntities.xml`.

3.2.3 R code comments

The `commentCode` module adds the LaTeX comment character, %, to the beginning of each line of R code identified by `<code class = "knitr">`, to conform to the Knitr packages standards for R code chunks in a Knitr LaTeX document. This module is an R-language module with one input, report, provided by the `latexEntities` module. The module wraps a source script which uses the XML package to perform the required transformation on the `code` elements with class set to knitr. The module produces one output, report, which references the transformed source document. The XML source for the `commentCode` module can be found at `transform/toPdf/commentCode.xml`.

3.2.4 Produce `.Rtex` file

The `convertToRtex` module transforms the source XML document into a LaTeX document with Knitr R code chunks. This module is a bash-language module with two inputs: report, which is provided by the `commentCode` module; and toRtex, which references the XSLT stylesheet `xsl/toRtex.xsl`. The module wraps a bash script which transforms the source document to a LaTeX document using the command-line tool `xsltproc`. The module produces one output, report, which references the Knitr LaTeX file resulting from the transformation. The XML source for the `convertToRtex` module can be found at `transform/toPdf/convertToRtex.xml`.

3.2.5 Produce `.tex` file

The `knitToLatex` module executes the chunks of R code in a Knitr LaTeX file and produces a LaTeX file. This is an R-language module with a single input, report, provided by the `convertToRtex` module. The module wraps an R script which call the `knit` function from the Knitr package to execute the R code chunks and returns a LaTeX file. The module produces three outputs: report, which references the HTML file produced by the module source script; toHtmlGraph, a PNG image file; and toPdfGraph, a PNG image file. The XML source for the `knitToLatex` module can be found at `transform/toPdf/knitToLatex.xml`.

3.2.6 Produce `.pdf` file

The `latexToPdf` module produces a PDF file from a LaTeX source file. This is a bash-language module with three inputs, report, toHtmlGraph, and toPdfGraph, which are provided by the `knitToLatex` module. The module wraps a bash script which produces a PDF file using the `pdflatex` command-line tool. The module produces one output, report, the PDF file produced from the LaTeX source file. The XML source for the `latexToPdf` module can be found at `transform/toPdf/latexToPdf.xml`.

4 Summary

5 Technical requirements

- Conduit version 0.6-3, a prototype OpenAPI glue system R package, was used to produce the final version of this report (`https://github.com/anhinton/conduit/releases/tag/v0.6-3`).
- Knitr version 1.12.3, an R package, was used for the transformations in this report (`http://yihui.name/knitr/`).
- `pdflatex` using pdfTeX version 3.14159265-2.6-1.40.16 (TeX Live 2015/Debian) and kpathsea version 6.2.1 was used for the transformations in this report (`http://pdftex.org`).
- R version 3.3.1 was used for the transformations in this report (`https://www.r-project.org/`).
- `sed` version 4.2.2 was used for the transformations in this report (`http://www.gnu.org/software/sed/`).
- All of the transformations described in this report were produced on a machine running Ubuntu 16.04 LTS 64-bit (`http://www.ubuntu.com/`).

- XML version 3.98-1.4, an R package, was used for the transformations in this report (`http://www.omegahat.net/RSXML`).
- `xmllint` using `libxml` version 20903 was used for the transformations in this report (`http://www.xmlsoft.org/`).
- `xsltproc` using `libxml` 20903, `libxslt` 10128 and `libexslt` 817 was used for the transformations in this report (`http://www.xmlsoft.org/`).

6 Resources

- The transformation to HTML pipeline uses the source document `report.xml`, and the input documents `metadataExample.xml`, `elementsExample.xml`, `knitrExample.xml`, and `xsl/toRhtml.xsl`. The source XML for the pipeline can be found at `transform/toHtml/pipeline.xml`, and the source XML for the modules can be found at `transform/toHtml/convertToRhtml.xml`, `transform/toHtml/knitToHtml.xml`, `transform/toHtml/substituteEntities.xml`, and `transform/toHtml/xinclude.xml`. The R script used to execute this pipeline can be found at `toHtml.R`.
- The transformation to PDF pipeline uses the source document `report.xml`, and the input documents `metadataExample.xml`, `elementsExample.xml`, `knitrExample.xml`, and `xsl/toRtex.xsl`. The source XML for the pipeline can be found at `transform/toPdf/pipeline.xml`, and the source XML for the modules can be found at `transform/toPdf/commentCode.xml`, `transform/toPdf/convertToRhtml.xml`, `transform/toPdf/convertToRtex.xml`, `transform/toPdf/knitToHtml.xml`, `transform/toPdf/knitToLatex.xml`, `transform/toPdf/latexChars.xml`, `transform/toPdf/latexToPdf.xml`, `transform/toPdf/substituteEntities.xml`, and `transform/toPdf/xinclude.xml`. The R script used to execute this pipeline can be found at `toPdf.R`.