Programmatically working with Word files in R

The ROOXML package

Duncan Temple Lang

Package URL: <http://www.omegahat.org/ROOXML>

We describe an R package - ROOXML - that permits R users to programmatically query and update the contents of a Word 2007 document. The functions provide convenient access to the individual files in the Open Packaging (ZIP) archive and allow the user to think of the archive as a collection of files. There are functions to create and insert images, tables, and verbatim output from R. Similar tools for older versions of Word can be developed based on DCOM (e.g. the [RDCOMClient](http://www.omegahat.org/RDCOMClient) or rcom packages for R).

Keywords: Open Office XML, Microsoft Word, document model

# Overview

The ROOXML package allows us to work with Microsoft Word files from within R. We can read their contents and their meta data. We can also modify the contents of the file, removing material or adding figures, tables, text, comments, and so on. We can create new files and build them up from within R. These tools even allow us to use Word as a literate data analysis or reproducible report generation tool containing R code. (See …)

Word documents are files on the file system. They are however ZIP archives and contain a collection of files arranged in different directories within the archive. So the representation in R of a Word .docx file is just a string identifying the file by (full) name and we create such an object in R with the function wordDoc(). For example, if we had a file name myDoc.docx in the directory docs/, we could create the R object to represent it as

ar = wordDoc(“docs/myDoc.docx”)  
ar = wordDoc(“docs/myDoc”)

In other words, we can omit the .docx extension if we want.

If there file doesn’t actually exist, the wordDoc() function copies an empty document from the ROOXML package to a file with the specified name. If this is not what is desired, test for the existence of the file with file.exists() before calling wordDoc() or alternatively, call wordDoc(, create = FALSE).

The resulting object is of (S3) class “WordArchive” and “ZipArchive”.

# Querying the Archive

We have defined a method for the names() function that operates on an archive and lists the names of the files it contains. For example,

names(ar)

[1] "[Content\_Types].xml" "\_rels/.rels"

[3] "word/\_rels/document.xml.rels" "word/document.xml"

[5] "word/theme/theme1.xml" "docProps/thumbnail.jpeg"

[7] "word/settings.xml" "word/fontTable.xml"

[9] "word/webSettings.xml" "docProps/core.xml"

[11] "word/styles.xml" "docProps/app.xml"

The listing of the files is the basic information. More interesting information is available about the document. For example, we can find out when the document was created, last modified, who the author is, keywords, revision number, and so on. The general properties of the document are available via the properties() function:

properties(ar)

$title

NULL

$subject

NULL

$creator

[1] "Duncan Temple Lang"

$keywords

NULL

$lastModifiedBy

[1] "Duncan Temple Lang"

$revision

[1] "7"

$created

$created$text

[1] "2008-10-29T23:51:00Z"

$created$.attrs

type

"dcterms:W3CDTF"

attr(,"namespaces")

[1] "xsi"

$modified

$modified$text

[1] "2008-10-30T01:04:00Z"

$modified$.attrs

type

"dcterms:W3CDTF"

attr(,"namespaces")

[1] "xsi"

There are also functions to access the individual elements in the properties and some other information within the meta-data of the document. These functions currently are:

|  |  |
| --- | --- |
| keywords | Either a single string containing the value of the keywords field, or if sep is specified, a vector of the individual keywords separated by the regular expression in sep. |
| images | A character vector giving the names of the files within the archive that are referenced as images within the document (ignoring images not referenced). The names of the vector are the relationship identifiers by which the images are referenced within the document. |
| comments(), commentTable() | The values |
| hyperlinks | A character vector giving the text of the hyperlinks and the names being the URLs/targets of the links. |

Created

keywords, images, comments

Add figures(), tables(), references(),

# Modifying the Archive

We can add files to the archive.

# Modifying the Document

While we may want to add content to the archive such as images and R code, the primary focus is on the contents of the document and the other files are typically supporting this content. The function addToDoc() allows us to insert content to the document. At present, it appends the content to the end of the document, i.e. after the last paragraph or to an empty paragraph. There are three basic elements we add: tables, plots and the output of R commands.

## Tables

If the R object being converted has a “dim” attribute of length 2, or is of class “table”, we represent this as a table with rows and columns.

## Plots/images

For objects of class “lattice”, we add these to the document as simple images. For regular R graphics plots, we provide some functionality that allow a plot to be identified as such, via the plot() function. If one calls this with no arguments, it copies the active graphics device to a PDF file and uses that as the plot. If one specifies a character string as the argument,

## Verbatim

The function toWordprocessML() is a generic function that takes an R object and constructs an XML node that represents that object in a manner that can be displayed in R.

# Dynamic Documents

We can use word to

# General Open Packaging Facilities

The ROOXML package provides some additional functionality that might be of use for others programming and developing additional high-level end-user functionality.

getPart:

getRelationships() & resolveRelationships():

getProperty() & setProperty()

## Functionality in Rcompression

The Rcompression package provides the functionality for working with ZIP archive files.