An introduction to RTF and the `rtfSweave` package

Stephen Weigand `weigand@mayo.edu`

April 13, 2015

Contents

1 Overview

In this document I try to provide some very basic information about RTF and show how to use the `rtfSweave` package with `Sweave` to make RTF statistical reports. For me, the key to understanding RTF was reading the excellent *RTF Pocket Guide* by Sean Burke[1]. It covers important aspects of RTF and only after reading it did I find that the RTF specifications published by Microsoft made some sense. The *Pocket Guide* can be downloaded as a PDF from the publisher's website (currently `http://shop.oreilly.com/product/9780596004750.do`).

Rich Text Format (RTF) is a mark-up language in that RTF files are plain text files containing document "content" such as text, tables, and figures along with commands indicating how these should be displayed. In this sense, RTF is somewhat similar to HTML or LaTeX. RTF was developed by Microsoft and is tightly integrated into Microsoft Word (MS Word), so much so that RTF is essentially a text-based interface to Word documents. If you are creating a statistical report for an MS Word user, there may be an advantage to writing the RTF directly rather than having to use a converter from LaTeXor other format. This should not be taken as arguing that RTF is a replacement for LaTeX.

One important way that RTF differs in spirit from HTML or LaTeX is that in the RTF specification there is much less separation between content and presentation.For example,

headings in an RTF document are really just short paragraphs with special formatting and not structural elements.

At this point, it may make sense to show an example of an RTF document. Here is the "Hello, world!" example from the *Pocket Guide*. Note that RTF has no built-in comments so I am using "//" to indicate "pseudo-comments" which annotate the file.

```
{\rtf1                  // RTF version 1, your only choice
\ansi                   // Document is in the ANSI character set

\deff0                  // Default font is font 0, defined below
\deflang1033            // Default language code 1033 = en-us
\plain \fs24            // Plain format has fontsize 24/2 = 12 points

{\fonttbl               // Font table
{\f0 Times New Roman;} // Font 0 defined as Times New Roman
{\f1 Arial;}            // Font 1 defined as Arial
}

{\pard \fs28 \b         // Paragraph set at 28/2 = 14 pts in bold
Greeting                //   to resemble a level-1 heading
\par}                   // End paragraph

{\pard                  // New paragraph set with "plain" formatting
Hello, World!
\par}                   // End paragraph
}
```

Figure 1: An annotated RTF version of "Hello, World!"

2 Document prolog

The *Guide* has information on the prolog. In it you define some document defaults, a font table, a color table (optional), and an info group containing metadata (optional).

3 Paragraphs

RTF has a paragraph entity, but no heading entities. This means that headings are just paragraphs with special formatting such as boldface type. Following the *P*ocket Guide[1], paragraphs are written using the following form.[1]

```
{\pard
```

[1]Note that there are no spaces at the end of these lines. If there were they would show up as extraneous spaces in the document. Blank lines don't matter in RTF but spaces do.

```
A short paragraph.
\par}
```

Formatting instructions for the paragraph can follow the `\pard` command on the same line. For many size-related commands, the units are in "twips" where one twip equals one twentieth of a point. (If 20 twips equals 1 point, and 72 points/inch, then there are $72 \times 20 = 1440$ twips per inch.[2]) One exception is that text sizes are specified in half points. To illustrate, the following paragraph has 120 twips (six points) of space before and after the paragraph and is set in font zero at 24 half points (12 points). A single space after the RTF commands is optional but helps readability.

```
{\pard \sb120 \sa120 \f0 \fs24
A short paragraph.
\par}
```

If the paragraph you are writing is more than one line, a helpful way to handle spaces is to indent everything after the first line with a space. That way, it will be easy to see that there are no inadvertent spaces at the end of a line. This is an example.

```
{\pard \sb120 \sa120 \f0 \fs24
A short paragraph that
 is written in multiple
 lines.
\par}
```

A heading paragraph is similar but with additional formatting instructions. The following creates the appearance of a heading by being displayed in font 1 (`\f1`), at 16 points (`\fs32`) and in bold (`\b`). The command `\keepn` keeps the headling-like paragraph on the same page as the paragraph that follows it.

```
{\pard \sb120 \sa120 \f1 \fs32 \b \keepn
Introduction
\par}
```

The mark-up is readable but verbose. One simple solution is to come up with some macro-like substitutions and write a function to "preprocess" the file to do search and replace. For example, one could write `\paragraph` in documentation chunks and use R to replace this pseudo command with valid RTF commands to specify, for example, $120/20 = 6$ points of space before and after the paragraph, and text set in font zero at 12 points.

```
> preprocess <- function(original.rtf, new.rtf){
+   doc <- readLines(original.rtf)
+   doc <-
+     sub("^{\\paragraph$",
+         "{\\pard \\sb120 \\sa120 \\f0 \\fs24",
+         doc)
+   writeLines(doc, new.rtf)
+ }
```

[2] My guess is that twips were used to allow RTF commands to be specified in integer units.

With these kind of substitutions specifying paragraphs, sections, subsections, etc, the document can be made more readable and much easier to write.

4 Figures

PNG, JPEG, or Windows enhanced metafile (EMF) images can be included in an RTF document by converting the image file to text using a hexdump and wrapping it in some RTF control commands. (See `http://en.wikipedia.org/wiki/Hex_dump` for details on a hexdump.)

Here is a partial example of RTF that would display a PNG file in a paragrap by itself.

```
{\pard{\pict\pngblip
8950 4e47 0d0a 1a0a 0000 000d 4948 4452
0000 0064 0000 0064 0803 0000 0047 3c65
6600 0000 0650 4c54 4500 0000 ffff ffa5
...
5771 c762 1cd6 f1ed 15bf 211b b221 2b21
3f2f a623 9b76 0cff 8d00 0000 0049 454e
44ae 4260 8200}\par}
```

This type of hexdump can be generated as follows.

```
> tmp <- tempfile()
> png(tmp, height = 2, width = 2, units = "in", res = 10)
> plot(1:10)
> dev.off()

null device
          1

> size <- file.info(tmp)$size
> hex <- readBin(tmp, what = "raw", size)
> cat(hex, fill = 60, sep = " ")

89 50 4e 47 0d 0a 1a 0a 00 00 00 0d 49 48 44 52 00 00 00 14
00 00 00 14 08 03 00 00 00 ba 57 ed 3f 00 00 00 60 50 4c 54
45 86 86 86 88 88 88 8f 8f 8f 90 90 90 92 92 92 95 95 95 97
97 97 99 99 99 9b 9b 9b 9f 9f 9f a5 a5 a5 a9 a9 a9 ab ab ab
ae ae ae b1 b1 b1 b4 b4 b4 b7 b7 b7 bc bc bc bf bf bf c8 c8
c8 d8 d8 d8 e0 e0 e0 e4 e4 e4 ea ea ea ec ec ec ee ee ee ef
ef ef f2 f2 f2 f3 f3 f3 fd fd fd fe fe fe ff ff ff 1b 4f fa
f6 00 00 00 09 70 48 59 73 00 00 01 89 00 00 01 89 01 9e 2e
11 35 00 00 00 56 49 44 41 54 18 95 63 90 c7 02 18 06 58 50
0a 9b 20 23 90 94 91 e7 e7 13 11 16 e0 e5 e1 e1 92 05 0b 0a
01 49 21 79 31 56 66 16 36 0e 6e 41 41 88 4a 19 88 16 09 51
```

```
71 79 71 71 39 71 0c 8b a4 31 6c 17 92 91 e7 91 94 97 61 94
a2 a6 8f d8 99 18 38 29 d0 8e 00 00 de af 2d ef 1f 70 8e 89
00 00 00 00 49 45 4e 44 ae 42 60 82
```

5 Tables

Tables in RTF are defined by a sequence of independent table rows defined by the `{\trowd ... \row}` construct. Each row consists of some number of "cells." Here is a simple two-row table where each row has three cells (which are displayed as columns).

```
{\trowd
\cellx1440\cellx2880\cellx4320
\pard\intbl Aligator \cell
\pard\intbl Bear \cell
\pard\intbl Cat \cell
\row}

{\trowd
\cellx1440\cellx2880\cellx4320
\pard\intbl Dog \cell
\pard\intbl Elephant \cell
\pard\intbl Fox \cell
\row}
```

The `\cellx` control word specifies where the right margin of the cell is placed relative to the page margin. In both rows above, the first cell's right edge is at 1440 twips (1 inch) from the page margin, the second cell's right edge is at 2880 twips (2 inches), and the third cell's right margin is at 4320 twips (3 inches).

6 Using `rtfSweave`

At present, the `rtfSweave` simply provides a driver so that `Sweave` can process an RTF file. Assuming that the file with text and code is `myreport.Rnw`, the commands to process the file are

```
require("rtfSweave")
Sweave("myreport.Rnw", driver = RweaveRtf(),
       syntax = SweaveSyntaxRtf)
```

Actually, the `syntax` argument is not necessary because `Sweave` is smart enough to find the right syntax object, `SweaveSyntaxRtf`.

This syntax object closely follows the `SweaveSyntaxNoweb` object and therefore, code chunks are defined by

```
<<optional label, various options>>=
Your code here

@
```

To include R code in text chunks, use `{\Sexr <code> }`.

6.1 File naming conventions

W

References

[1] Sean M Burke, *RTF Pocket Guide*. O'Reilly, 2003.