

PhUSE 2008

Paper PO03

%RTFparser

Duong Tran, Tranz Ltd, London, UK

ABSTRACT

Since the introduction of the Output Delivery System (ODS), SAS[®] programmers has been taking advantages of these richer alternative formats (XML, HTML, PDF, RTF etc) over the traditional plain text. Within the pharmaceuticals industry in clinical reporting there has been a surge of interest in using some of these alternative formats in recent years.

There are many published papers about exchanging data between SAS and MS Excel[®] and hardly any from Rich Text Format (RTF) to SAS, but as RTF is becoming more popular a format in clinical reporting, a utility to extract data from RTF documents is welcomed^[2].

This paper outline strategies to process RTF documents **generically**, introduces **%RTFParser** (a SAS utility macro that convert RTF document into SAS dataset) then highlight some of its potential applications and a brief discussion with some other related publications.

INTRODUCTION

RTF is a plain text format developed by Microsoft[®] for multi-platform document interchange. Most word processors are able to read and write RTF documents but it is a format that is not very pleasant for the eyes. Here are two RTF documents that will produce exactly same results when viewed in MS Word[®].

Example 1

```
{\rtf1\ansi\def0{\fonttbl {\f1 Arial;}}Hello!World!}
```

Example 2

```
{\rtf1\ansi\def0{\fonttbl {\f1 Arial;}}Hello!  
Wor  
ld!}
```

As you can see, even from this simple demonstration that **Example 2** is not so human-legible, but RTF is like any other formats have rules and rules can be decoded. This might sound unconvincing but to extract data from an RTF table you only need to understand a few rules. The main two are where table row begin and the end of each cell definition.

PROCESSING RTF TABLE DOCUMENT

As mentioned in the introduction that RTF is not pleasant to the eyes, especially when it is in free-flow form (that is when the text lines do not break at the eyes friendly place like in Example 2). So where do we begin to process an RTF document if we 'can not' see what its general pattern is? When something is in a form that is difficult to work with we typically transform it into something that is easier to deal with, as noted by Hagendoorn^[2]. So now the question is how do we do this? Well a table row is identified by the RTF control words **!trowd** and end with **!row**. That is we need to **break/join** text at an ideal point. This is the '**most**' critical requirement for programming purpose.

Example 3

Name	Height	Weight
Alfred	1.6	60
Alice	1.5	50

PhUSE 2008

This table could possibly be represented in RTF as below (excerpt only).

```
:  
\trowd\trkeep\trhd\trqc\trgaph60\clbrdrb\brdrs\brdrw15\brdrf1\clbrdrf1\brdrs\brdrw15\brdrf8\cltbl  
\clvertalb\clcbpat8\cellx1000\clbrdrb\brdrs\brdrw15\brdrf1\cltbl\clvertalb\clcbpat8\cellx2000\clbrdrb  
\brdrs\brdrw15\brdrf1\clbrdrf1\brdrs\brdrw15\brdrf8\cltbl\clvertalb\clcbpat8\cellx3000\pard\plain\intbl  
\keepn\sb20\sa20\qlf1\fs20\cf1{Alice\cell}\pard\plain\intbl\keepn\sb20\sa20\qlf1\fs20\cf1{1.6\cell}\pard  
\plain\intbl\keepn\sb20\sa20\qlf1\fs20\cf1{60\cell}{\row}\trowd\trkeep\trhd\trqc\trgaph60\clbrdrb\brdrs  
\brdrw15\brdrf1\clbrdrf1\brdrs\brdrw15\brdrf8\cltbl\clvertalb\clcbpat8\cellx1000\clbrdrb\brdrs\brdrw15  
\brdrf1\cltbl\clvertalb\clcbpat8\cellx2000\clbrdrb\brdrs\brdr1\brdrf1\clbrdrf1\brdrs\brdrw15\brdrf8  
\cltbl\clvertalb\clcbpat8\cellx3000\pard\plain\intbl\keepn\sb20\sa20\qlf1\fs20\cf1{Alice\cell}\pard\plain  
\intbl\keepn\sb20\sa20\qlf1\fs20\cf1{1.5\cell}\pard\plain\intbl\keepn\sb20\sa20\qlf1\fs20\cf1{50\cell}  
{\row}  
:
```

The first step is to do the transformation and this is not a difficult task. After the transformation the RTF table document should ideally be one record per row as shown here.

```
\trowd\trkeep\...etc..\cf1{Alfred\cell}\pard\...cf1{1.6\cell}\pard\....\cf1{60\cell}{\row}  
\trowd\trkeep\...etc..\cf1{Alice\cell}\pard\...cf1{1.5\cell}\pard\....\cf1{50\cell}{\row}
```

But now what about all the control words? How do we **remove/compress** them in a general way? The **'best'** solution to this in my opinion is via a regular expression. SAS now has a set of PRX CALL routines and Functions to handle Perl Regular Expression and here is the RTF regular expression (the **KEY** to parse RTF).

`^\[a-z]+(-?[0-9]+)? ?/`

source: Burke^[1]

Once we have applied the Perl Regular Expression we will get something like this.

```
Alfred ^^^ 1.6 ^^^ 60 ^^^  
Alice ^^^ 1.5 ^^^ 50 ^^^
```

We can now scan through this and store the data in three variables.

APPLICATIONS

The need to convert document format is a common task, but why would anyone want to convert RTF to SAS? In clinical reporting it is quite common to compare two instances of the same output table or to compare QC data to the actual output table. The **%RTFparser** utility can be used for these purposes. Note a utility such as the UNIX **diff** and alike are not appropriate for the purpose of comparing RTF tables because although the values in the tables may be the same, the font sizes, cells widths etc may be different. What we are interest in this case is to compare the corresponding cell value between the tables.

Comparing Instances of an RTF table

With the **%RTFparse** utility one can efficiently compare two instances of an RTF table. We can build a reporting tool such as the **%Rdiff**.

Example 3

```
%Rdiff(bfiles = /old_run/t_ae*.rtf  
      ,cfiles = /new_run/t_ae*.rtf  
      )
```

PhUSE 2008

Compare Report

Base	Date	Compare	Date	Match
t_ae1.rtf	20-Jun-08	t_ae1.rtf	21-Jun-08	Yes
t_ae2.rtf	20-Jun-08	t_ae2.rtf	20-Jun-08	No
etc	etc	etc	etc	etc

Comparing RTF table with QC data

As concluded in Hagendoorn^[2] this method increase the efficiency and accuracy in QC activities.

Example 4

```
%RTFParser(ifile = t_ae.rtf    /* the RTF file */
           ,odset = ae        /* output dataset */
           )

/* v_t_ae.sas */
Data qcae;
:
  data steps
:
Run;

Proc summary;
:
Run;
etc..

Proc compare base=ae compare=qcae;
Run;
```

DISCUSSION

Here are some published papers regarding comparing RTF tables. Each has their merits but also drawbacks.

Franklin^[4] - Uses Visual Basic Application (VBA) and this would be a good choice for comparing different instances of an RTF table. The advantage of this method is that MS Word[®] do the comparison, the drawback is of course we do not get the table in a SAS dataset.

Xu^[3] - Describes a more general method to compare documents and they can be different formats but has a drawback for RTF. That is, it needs to be converted to text first.

Hagendoorn^[2] – Convert the RTF table directly but stripping the RTF control words is a challenge without applying the RTF Regular Expression.

CONCLUSION

Converting RTF table to SAS is not as complex as first envisaged given that SAS v9.1.3 now has a set of PRX CALL Routines and Functions. Indeed I hope I have convinced you that this is in fact relatively simple and believe that the %RTFparser utility will prove an invaluable tool as RTF is becoming a popular format in clinical reporting.

REFERENCES

- [1] Burke S. M. "RTF Pocket Guide"
- [2] Michiel Hagendoorn, Jonathan Squire, Johnny Tai. "Save Those Eyes: A Quality-Control Utility for Checking RTF Output Immediately and Accurately". SUGI 31, paper 066-31.
- [3] Michelle Xu, Jay Zhou. "%DIFF: A SAS Macro to Compare Documents in Word or ASCII Format". PharmaSUG 2007, paper cc09.
- [4] David Franklin. "Comparing ODS RTF Output Files in Batch Using SAS (or, 500 ODS RTF Documents to Compare in 15 minutes!)". PharmaSUG 2007, paper tt02
- [5] Xuejing (Susan) Mao, Mario Widel. "An Efficient Report Checking". PharmaSUG 2007, paper ad04

PhUSE 2008

- [6] David L. Cassell. "The Basic of the PRX Functions". SAS Global Forum 2007, paper 223-2007
[7] Duong Tran. "%RiTEN". PhUSE 2007, paper po07

CONTACT INFORMATION

Your comments and questions are valued and encouraged. Contact the author at:

Duong Tran

Tranz Ltd

Phone: 07752326413

Email: trand000@aol.com

Web: www.tranz.co.uk

Brand and product names are trademarks of their respective companies.