

MORE CUSTOMIZATION? CREATING SYMBOLS IN RTF FILES USING ODS

Richard Rowell, Connetics Corporation, Palo Alto, CA
Jim Lenihan, Independent Consultant, South San Francisco, CA

ABSTRACT

If you are using SAS® ODS to produce RTF files you may run into problems if you need to put symbols in your tables or listings. This paper will show you how to put various (mathematical) symbols used in statistics on to your report. It will show some of the problems the authors have had in creating certain symbols and how you can "troubleshoot" if you have difficulty in putting out your symbol.

INTRODUCTION

Before the days of the SAS® Output Delivery System (ODS), it was difficult to place mathematical and other symbols in the RTF output with SAS® code. Complicated macros were capable of doing this, but usually these macros would have to be validated and/or modified. Most of the time the procedure was to give the RTF document to the publisher(s). But what if the data had to be rerun? This would involve yet another trip to the publisher(s), where he/she may not be as glad to see you as the first time. However, it is now possible to place symbols in your header, footer or data of your report with very little coding.

In this paper, we will demonstrate code to place mathematical and/or Greek symbols to an RTF document. We will also show you some trouble spots we encountered and alternative methods to display symbols.

Before discussing the code, please note that all output was created using SAS® 8.2 for Windows 2000. All programming was done with the PROC TEMPLATE procedure. Other versions of SAS®, other platforms, the non-use of PROC TEMPLATE, or other destinations for ODS may give different results.

CODE

Now let's suppose you want to print the Greek letter alpha in the footnote of your report in order to show the critical region of your statistical test. Usually, one might place an α to represent α . Now, with SAS® ODS, displaying the Greek letter alpha is as simple as Figure 1 below:

Figure 1

```
ods escapechar="^";  
%let symbol_alpha=%bquote(^R/RTF"u945\");
```

Notice the three digit number (945) towards the end of the macro variable is referred to as a Unicode decimal number. Since computers can only process numbers, Unicode was derived to provide a unique number for every

character; thereby, eliminating problems with conflicting standards. Unicode decimal numbers are represented by mostly three and four digit numbers, but Unicode can also be formatted as hexadecimal.

To find out all you want to know about Unicode including the Unicode decimal number visit the following web sites:

<http://www.alanwood.net/unicode/index.html>

<http://www.unicode.org/>

As you see in Figure 1, we have set the escape-character to '^'. You should be careful to choose an escape-character that does not appear in your code or output. For example, a person using Windows should not pick '\ as an escape character.

The \R signifies that raw text will be put into the output destination and /RTF is that destination for our symbol. The 945 is the unique Unicode decimal number for the alpha symbol. Note that a u is placed in front of the Unicode decimal number; this must always be present. When this macro is invoked an α appears.

Unicode decimal numbers provide a way to access numerous symbols for your RTF output. Mathematical symbols such as, beta, summation, delta, epsilon, mu, and sigma and commonly used symbols such as trademark, copyright, and register are right at our fingertips. Table 1 (below) shows a sample of symbols that may be used.

Table 1

name	symbol
SYMBOL_CENT	¢
SYMBOL_COPYRIGHT	©
SYMBOL_PLUSMINUS	±
SYMBOL_PI1	π
SYMBOL_ALPHA	α
SYMBOL_APPROX	≈
SYMBOL_TRADEMARK	™
SYMBOL_GE	≥
SYMBOL_DIVIDE	÷
SYMBOL_DELTA1	Δ
SYMBOL_DELTA2	Δ
SYMBOL_EPSILON	ε
SYMBOL_YHAT	ŷ

"MORE CUSTOMIZATION? : CREATING SYMBOLS IN RTF FILES USING ODS"

name	symbol
SYMBOL_INFINITY	∞
SYMBOL_NE	≠
SYMBOL_LE	≤
SYMBOL_MU	μ
SYMBOL_REGISTER	®
SYMBOL_SUMMATION1	Σ
SYMBOL_BETA	β
SYMBOL_SIGMA	σ
SYMBOL_SUMMATION2	Σ

Figure 2

```
(1) footer f1 ;
(2) define f1 ;
(3) text
(4)   " +Above Normal Range" "^1n"
(5)   "-Below Normal Range" "^1n"
(6)   "A reading &symbol_ge 40 for SGPT/ALT
is considered Above Normal" "^1n"
(7)   "A reading &symbol_le 30 for SGPT/ALT is
considered Below Normal" "^1n"
(8)   "An example of &symbol_alpha." "^2n "
(9)   "&font9.Output:" "
&tab.&outdir.&pgmonly..rtf" "^1n"
(10)  "Source:&tab.&source." ;
(11)  style={font_face=Arial font_size=2.5
font_weight=medium background=white} ;
(12)  just=left ;
(13)  end ;
```

OUTPUT

The following code (Figure 2) from PROC TEMPLATE produces the following footnote (Table 2).

Please note in lines 4-10 Figure 2 the sets of double quotes used in each line. All seven lines may have been placed within one pair of quotations, however, the code would have been very unreadable. Arranging the code (quotation marks) similar to Figure 2 makes this code much more readable and concise.

Table 2:

Treatment Group: Brand X							
Site-Subject	Visit	Collection Date	SGOT/AST (U/L)	SGPT/ALT (U/L)	Alkaline Phosphatase (U/L)	Total Bilirubin (mg/dL)	Direct Bilirubin (mg/dL)
0-00	BASELINE	06/12/2002	17	24	59	0.4	0.1
	WEEK 4	07/09/2002	23	27	72	0.4	0.1
0-02	BASELINE	06/24/2002	27	29	60	0.9	0.2
	WEEK 4	07/22/2002	25	24	59	1.1	0.2
0-06	BASELINE	07/25/2002	27	50 +	91	0.6	0.1
	WEEK 4	08/21/2002	24	48 +	77	0.6	0.1
0-07	BASELINE	08/06/2002	26	29	46	1.2	0.2
	WEEK 4	09/03/2002	22	22	47	1.3	0.2
0-08	BASELINE	10/28/2002	7	16	152 +	0.3	0.1
	WEEK 4	11/25/2002	7	14	149 +	0.4	0.1
0-09	BASELINE	11/05/2002	52 +	77 +	73	0.4	0.1

+Above Normal Range
-Below Normal Range
A reading ≥ 40 for SGPT/ALT is considered Above Normal
A reading ≤ 30 for SGPT/ALT is considered Below Normal
An example of α

Output: S:\clinical\kfd\002\analysis\statfiles\output\test\l_talk_in2.rtf
Source: S:\clinical\kfd\002\analysis\statfiles\programs\listings\test\l_talk_in2.sas (01AUG2003)

CAREFUL NOW

Unfortunately, footnotes similar to Figure 2 are not that simple. Little subtleties exist and were only solved through trial and error.

First, if your leading character is a symbol then leave a blank space between the open quote and the symbol. If you look at line (4) in Figure 2 you will notice the blank space. Failure to do this will result in the string being printed out as '^R/RTF'.

"MORE CUSTOMIZATION? : CREATING SYMBOLS IN RTF FILES USING ODS"

Second, to print a superscript in your footnote you may think that the macro variable code is:

```
%bquote(^R/RTF\^super 1').
```

Although that would print 1, your entire string thereafter would also be superscripted. The correct way would be:

```
%bquote(^R/RTF\^super 1\nosubsub\').
```

In addition to superscripting text, superscripting a symbol is definitely not trivial. After a few rounds of agony, the correct way to accomplish this is:

```
%bquote(^R/RTF\^super\u8482\~\nosubsub\').
```

The omission of the tilde will also superscript the entire string.

Third, to insert a tab the macro variable to use is:

```
%bquote(^R/RTF\tab\').
```

Although difficult to see, a space is present between the last backward slash and the single quote is important (You may also leave a space in your actual text). If the space is not present, you will get a string that reads 'R\RTF'.

And last, a space exists between '^2n' and the closed quotes as you see on line 8 in Figure 2. If the space is omitted, the characters on the next line will fail to translate correctly.

ALTERNATIVE METHODS

Besides the method of using Unicode numbers, there have been other methods used to display symbols.

(1) `"^S={font_face=Symbol}" '61'x "^S={}"`

(2) `"^S={font_face=Symbol} a ^S={}"`

The codes above are alternative methods for displaying an alpha symbol. In the first method, you must use the correct hexadecimal number and use the symbol font. However, it is quite difficult and perhaps impossible to use a macro variable with this method due to the necessity of a hexadecimal. In other words, the coder would have to write this cumbersome looking string perhaps countless times in his/her code.

The coder is able to use a macro variable for the second method above. However, this method is limited to displaying (mostly) Greek characters. Symbols, such as \geq , are unattainable with this method.

There is another more "open-ended method" that may be evolving. This method allows you to move to "specialized formatting effects" across destinations.

To see an example, look at the macro variable 'sup' below. This method however does not work well with symbols.

```
%let sup=%bquote(^{super 1 } ) ;
```

Unfortunately, we cannot at this time use this method to print out mathematical symbols. Maybe things will change in the future.

CONCLUSION

Symbols enhance reports and enhanced reports add value. By using this method you have saved yourself time, or the time of another person, from having to manually manipulate the symbols. Now, the report is under the complete control of the coder.

Although limitations exist, symbol generation via the Unicode method using in-line formatting is easy and flexible enough for users to provide custom output in RTF.

REFERENCES

http://support.sas.com/rnd/base/topics/templateFAQ/Template_rtf.html

PROC TEMPLATE FAQ and Concepts

RTF File Specifications, Version 1.5
Microsoft Corporation

Poppe, Frank "How To Do Horrible Things With Raw RTF Specifications and the SAS RTF ODS"

<http://www.alanwood.net/unicode/index.html>
Alan Wood's Unicode resources

<http://www.unicode.org/standard/WhatIsUnicode.htm>
What is Unicode?

ACKNOWLEDGEMENTS

SAS® and all other SAS® Institute Inc. product or service names are registered trademarks or trademarks of SAS® Institute in the USA and other countries. ® indicates USA registration.

CONTACT INFORMATION

Richard Rowell
Connetics Corporation
3290 West Bayshore Road
Palo Alto, CA 94303
USA
rrowell@connetics.com

Jim Lenihan
15 Moonlight Ct
South San Francisco, CA 94080
USA
(650) 742-0131
jameslenihan@sbcglobal.net