Paper 143-28

Why DATA _NULL_ When You Can RTF Faster? (Shattering the 2-Dimensional Paper Barrier)

Dante diTommaso, Fred Hutchinson Cancer Research Center, Seattle, WA

ABSTRACT

I have yet to work with a statistician or epidemiologist that enjoys reviewing pages of raw SAS® output. I hope I never meet a programmer willing to hand enter results into someone's spreadsheet or document. Therefore, results management, a data manipulation task particularly relevant to SAS report generation, is a common effort that too often results in dull fixed-font `DATA _NULL_`, `PROC REPORT` or similar tables. I present a macro I have developed that instead produces customized RTF tables with less effort than analogous fixed-font programming.

The macro and examples described below are available at the website listed in **CONTACT INFORMATION**.

INTRODUCTION

My primary programming objective has become to spend as little time as possible massaging SAS output in favor of data analysis. To minimize format programming, I have developed a highly flexible macro that generates rich text format (RTF) documents from any combination of SAS data sets. This paper describes the macro, provides examples of use, and suggests developing a set of reporting macros that utilize this or a similar approach to eliminate redundant aesthetics programming for standard reports.

DELIVERING KNOWLEDGE

SAS: The Power To Know. From a marketing perspective, SAS' official slogan is on target. Whatever your data, whatever your goal, the SAS System offers a powerful solution. However, from a programming perspective, more appropriate slogans come to mind. SAS: The Power to Generate lots of output that needs to be heavily filtered, manipulated and formatted before those seeking knowledge care to review the results. Not as snappy, definitely, but more indicative of the nature of SAS programming.

The basic approach to using the RTF writer is to assemble desired output into a data set (programming that is nearly impossible to avoid), quickly add desired formatting tags, and then combine output tables in a single or multiple RTF documents. The process handles ODS as easily as customized data sets.

HIGH-DIMENSIONAL PAPER

Even in the elusive paperless society the basic concept of paper will endure, however modified the details become. Increasingly, electronic pages supplant hard copies, but while e-paper retains basic structure it also presents typically untapped flexibility. This momentum affords opportunities to push beyond black and white 2-dimensional paper. Whether or not this is a great push forward, or a return to the origins of paper is open to debate.

Plummeting prices on high-speed color printers further reinforce the motivation to guide the eye and mind with high-dimensional paper. Modify fonts, colors and table structure to reveal information otherwise lost among the typical antiseptic sea of `Courier New` digits and characters. My RTF writer brings high-dimensional paper within comfortable reach.

MACRO PHILOSOPHY

Repeat after me: If you see a problem once, solve it once; if you see a problem multiple times, *solve it once*.

MINIMIZE REPEAT TASKS

Toward this end, it is essential to embrace differences between machines and humans. Unlike magicians, computers are terrific repeat performers. Programmers need to harness this trait by turning over familiar tasks to SAS at the earliest possible point along the path from raw data to final reports. Doing so leaves programmers plenty of time to address interesting challenges that require intuition, ingenuity, keen judgment and other defining human abilities.

The key to developing truly useful macros is minimizing the data processing required before the macro call. The RTF macro described below does not eliminate pre-processing. It does, however, reduce routine work and can be integrated into a powerful and efficient reporting system composed of standard report macros.

SIMPLE YET FLEXIBLE

Make the simple tasks simple and the complicated task possible. While not an original objective, it is one I embrace. I have tried to ensure that my RTF writer reflects this measure of usefulness. Invoking the macro is simple:

```
%WRITERTF(data set, filename)
```

This statement creates a visually pleasing RTF file *filename* containing the output in *data set*. Data set observations and table rows have a one-to-one correspondence.

If you need more control, optional parameters allow you to specify document structure (multiple tables; page margins, size and orientation; headers and footers; etc.) and table appearance (display columns; cell padding; pre-defined table styles; etc.).

Microsoft® Word® supports document properties such as title, author, company, and hyperlink. You can pass this information to the macro as a document information data set.

- The macro only processes character variables. Any numeric data must first be formatted for display as desired. You do not have to drop numeric variables from a data set, but the macro ignores any.

PREPARING TABLE CONTENT

The macro expects one non-display *format* variable in the data set the contents of which is not table content. This variable contains format instructions for each table cell (vertical alignment, background color, borders, etc.) and the text within (horizontal alignment, font name & color, etc.).

The macro does not require format instructions or even the *format* variable. Pre-defined table styles are available, which you can customize as desired. See **Pre-Defined Table Styles** below.

THE *Format* VARIABLE

The *format* variable contains #-delimited substrings. The full string specifies how to format each cell of the current observation (table row). Each #-delimited substring formats the corresponding cell of the current row. The last substring persists to all remaining cells (i.e., if the data set has 6 display columns and the *format* string has only 3 #-delimited substrings, all format instructions in the third and final substring apply to cells 3 through 6.

Format substrings are =-delimited keyword/value pairs. These format details pertain to a specific cell in the current row. See **Table 1** for keywords and valid options.

Paper 143-28

Table 1: Text And Cell Format Options

Keywords (no values)		
TITLE		Observation contains column titles (span right across blank cells & repeat across page breaks)
DEF		Referenced table cell gets DEFault formatting
<NULL>†		Table cell is blank (no **TITLE** spanning)
Table Cell options		
VA	=	Vertical Alignment (**T**op, **C**enter, or **B**ottom)
BGC	=	BackGround Color (*common color name*)
BT	=	Border color, Top (*common color name*)
BL	=	Border color, Left (*common color name*)
BB	=	Border color, Bottom (*common color name*)
BR	=	Border color, Right (*common color name*)
BA	=	Border color, All sides (*common color name*)
BSZ	=	Border SiZe (line width in points)
RH	=	Row Height (font size in points)
Text options		
HA	=	Horizontal Alignment (**L**eft, **C**enter, or **R**ight)
FN	=	Font Name (**T**imes, **A**rial, **C**ourier, or **S**ymbol)
FC	=	Font Color (*common color name*)
FST	=	Font STyle (combination of **B**old, **I**talic, **U**nderline, smallca**P**s, and super(**H**)- or sub(**L**)-script)
FSZ	=	Font SiZe (in points, overrides default size)

† The string "<NULL>" is the actual variable content; no cell-specific *format* string is necessary.

The format information also flags title rows (keyword **TITLE**), which repeat after any page break within a table. **TITLE** row content also spans adjacent empty cells to the right. A variable value of keyword **<NULL>** blocks a column header from spanning the blank cell. Note that you use the keyword **<NULL>** as table content rather than part of the *format* string.

The following example declares the current observation a title row with the first cell shaded gray, and the second and third cells underlined in red (remaining cells get the default format):

```
format = 'TITLE BGC=gray#BB=RED#BB=RED#DEF'
```

WITHIN CELL FORMATTING

Within a table cell, page header or page footer, you can turn basic formats ON or OFF relative to the default *format*. For example, you can **bold** individual characters in a cell. Do this by modifying the actual variable value (table cell content) prior to invoking `%WRITERTF()`. The syntax is to tag characters or words of interest with format instructions:

```
"<FON=opt>formats ON<FON> but no more"
"<FOF=opt>formats OFF<FOF> and back on"
```

This is a great feature for fine-tuning table content. For example, if a cell has no formatting by default (in the *format* variable) but contains a character you want bold and in symbol font (e.g., an alpha, beta, superscript, etc.) then modify table content to:

```
"<FON=BS>s<FON><FON=H>1<FON> = 2.5"
```

The resulting table content is: σ^1 = 2.5. Note that formats are added or subtracted from the formats specified in the format variable. Valid special formats are: bold, **B**, italic, **I**, underline, **U**, small caps, **P**, superscript, **H**, subscript, **L**, and basic fonts (Arial, **A**, Courier, **C**, Symbol, **S**, Times New Roman, **T**).

- See macro documentation and example code for details on the format variable and within-table formatting.

FEATURES

The macro supports a variety of features that allow customization of the RTF document as well as table content.

PRE-DEFINED TABLE STYLES

I have defined 4 table styles that can eliminate the need for any *format* variable programming. Additional styles are not difficult to add and once you have successfully added one, adding more is downright simple.

FONT SIZE AND COLUMN WIDTH

The macro calculates font size and column widths adequate, although not optimal, for accommodating the table content. Since these calculations are rough you may need to override defaults for certain dense or unbalanced tables. The macro provides a variety of options for fine-tuning table presentation, including explicit specification of both font size and column widths.

AUTO HEADER/FOOTER TEXT

The macro generates automatic page header (name of programmer, version of SAS) and page footer (page number, filename) text. Page numbers are reported as "Page *p* of *t*", the page number and total pages in the document.

TITLE ROWS

Initial title rows in a table will repeat across any page breaks necessary in the middle of a table. Any row in a table can be a **TITLE** row – for which content spans adjacent blank cells. However, only initial title rows will repeat after a page break.

SECTION BREAKS

Add a section-break before second or later tables in a document. A section-break allows you to change page setup as well as headers and footers.

BOOKMARKS

The macro includes a bookmark in the RTF document for each table. This allows the RTF reader to quickly cycle through document content.

16 OR 24 COLORS

Older versions of MS Word support 16 colors. Newer versions support a full spectrum. The macro allows you to choose between 16 and 24 colors. With some effort, adding custom colors is possible.

%PREP4RTF(*data set*, *formats*, *title_rows*)

The flexibility of `%WRITERTF()` is dramatically improved by this ancillary macro which prepares ODS data sets for the RTF writer. Pass in the *data set*, the display format for each variable, and the maximum number of *title rows*. The macro creates title rows from variable labels and returns a new data set.

Use the SAS Output Delivery System (ODS) to create data sets that you can write to an RTF document in two simple steps. For example, `PROC REG` produces an ODS data set of regression model parameter estimates. Assume I have created `ESTIMATES` from the ODS data set, keeping character variables `DEPENDENT` and `VARIABLE`, and numeric results `ESTIMATE`, `STDERR`, `TVALUE`, and `PROBT`. Preparing data set `ESTIMATES` for writing to an RTF document is just a matter of specifying appropriate formats for each variable:

```
%PREP4RTF(ESTIMATES,
          $8. $9. 5.2 5.3 COMMA8. PVALUE6.4,
          MAXTR=2)
```

The keyword parameter `MAXTR` stipulates that column titles (which `%PREP4RTF` generates from variable labels supplied by ODS) should take up no more than two rows. The macro generates data set `ESTIMATES_RTF` which you subsequently write to an RTF file:

```
%WRITERTF(ESTIMATES_RTF, C:\PROJECT\REPORT,
          STYLE=4, PORT=1, COL24=1);
```

Table 2: %WriteRTF() Keyword Parameters	
Data Set Details	
OMIT	= ***list*** of variables in the data set to ignore
DOCINFO	= ***data set*** of doc details supported by MS Word
Document Layout	
ACTION	= **0** (default) initialize & finalize file with 1 table **1** initialize RTF file and add first table **2** append to existing RTF file with section break **3** same as 2, plus finalize RTF file **4** append to existing RTF file, no section break **5** same as 4, plus finalize RTF file
LEGAL	= **1** for legal size paper (letter is default)
PORT	= **1** for portrait orientation (landscape is default)
LINCH	= ***inches*** for Left page margin
RINCH	= ***inches*** for Right page margin
HINCH	= ***inches*** between top of page and page Header
FINCH	= ***inches*** between Footer and bottom of page
HCUSH	= ***inches*** between Header and table content
FCUSH	= ***inches*** between table content and Footer
Page Headers and Footers	
H1 to H6	= *page header text* (may require macro quoting) H1: left align, top line H2: center, top line H3: right align, top line (defaults to current date) H4: left align, 2nd line (defaults to system info) H5: center, 2nd line H6: right align, 2nd line (defaults to user name)
F1 to F4	= *page footer text* (may require macro quoting) F1: left align, penultimate line F2: center, penultimate line F3: right align, penultimate line F4: center, last line (**report name** and **page number** appear on last line, left align and right align, respectively)
Table Format	
STYLE	= **1**, **2**, **3**, **4**, or **5** (current pre-defined table styles)
CELLPAD†	= ***twips*** (1440/inch) between cell border and text
FS	= ***pts*** to override automatic Font Size
COLWIDTH	= ***list*** of Column Widths to override automatic widths (relative widths, `COLWIDTH=1 1 2 4` specifies that columns 3 and 4 are twice and four times as wide, respectively, as the first two columns)
FSTEP†	= ***number*** of 0.5 pt *increments* to adjust default font size (values <0 decrease size)
FSFACT†	= ***multiple*** (relative to 1) to use when calculating column widths to accommodate larger font size for titles
COL24	= **1** for 24 colors (default is 16)
† Useful for fine tuning font size and column widths when content is wrapping in a cell or cells, alternatively specify both `FS` and `COLWIDTH`.	

Parameters `STYLE`, `PORT`, and `COL24` demonstrate the flexibility of the macro. See **Table 2** for a list of parameters and brief explanations of each. Detailed descriptions appear in the macro documentation.

SHORTCOMINGS

RTF is a Microsoft standard for displaying richly formatted text. Microsoft published RTF version 1.6 specifications on the MS Developers Network (http://msdn.microsoft.com). However, with the launch of Office 2000/XP, RTF version 1.6 is outdated.

More importantly, few RTF readers (including MS WordPad®) are sufficiently RTF compliant to handle customized tables. Without access to Microsoft Word (yourself and more importantly your client), `%WRITERTF()` can not generate useful documents.

If you have MS Word but your client does not, convert final RTF documents to alternative formats such as postscript or PDF.

Several features are implemented by turning control over to MS Word (eg, page numbering and page breaks). Such features are resistant to change.

Finally, as with any programming focused on aesthetics, `%WRITERTF()` programming still involves an iterative process to make results look just right. The number of iterations depends on your need for perfection. (Not to mention that ODS RTF in SAS Version 9 may offer flexibility that obviates the need for such macros … maybe.)

CONCLUSION

I have been using `%WRITERTF()` while it has evolved over two years. In its current form, the macro saves me untold hours of `DATA _NULL_`, `PROC REPORT` and similar display programming, and the superficiality of the resulting documents has produced goofy grins on the faces of statisticians receiving even unwelcome results (although the later usual register eventually). More importantly, I can produce these documents quickly.

Initially, the task of specifying display options in the format variable may seem onerous. However, techniques that combine informats with array processing render format programming barely more trouble than deciding how you want the table to appear. Consider the example "Listening to the Data" in the **Appendix**. Data-driven formatting not only looks striking, it adds incredible value to reports, shattering the 2-dimensional paper barrier with higher dimensions of color and style.

The files referenced below (see **CONTACT INFORMATION**) provide several examples of how to work with these macros. Examining the data sets that `%WRITERTF()` processes can be more helpful than the code that generates them since, as with virtually any SAS task, 100 programmers will manage to find at least 110 distinct paths to the same endpoint.

REFERENCES

Paul Hamilton, ODS to RTF: Tips and Tricks, PharmaSUG 2002 (http://www.pharmasug.org/psug2002/bp2002/psug2002_bp.html, "Best Paper" in *FDA Compliance: Electronic Submission & Validation*).

CONTACT INFORMATION

I value and encourage your comments and questions:

Dante diTommaso
Fred Hutchinson Cancer Research Center
1100 Fairview Ave N, MW-500
Seattle, Washington 98109
Phone: (206) 667-6470
Fax: (206) 667-4812
Email: dante@scharp.org
Files: http://dantegd.home.mindspring.com/sas/

TRADEMARK CITATION

Paper 143-28

```
*+-------------------------------------------------------------+*;
*|                      A P P E N D I X                        |*;
*---                                                          ---*;
*| "Listen To The Data" Example                                |*;
*| IDEA: USE INFORMATS IN COMBINATION WITH ARRAY PROGRAMMING   |*;
*|       TO APPLY DESIRED FORMATTING TO SPECIFIC RESULTS       |*;
*| PROGRAMMER: DANTE DITOMMASO (SUGI 28, PAPER 143, MARCH 2003)|*;
*+-------------------------------------------------------------+*;

* [ STEP 1: FAKE SOME DATA ] *;
DATA LISTEN (KEEP=VAR:);
  DO CNT = 1 TO 8;
    RAN1 = RANUNI(23581); RAN2 = RANUNI(49603); RAN3 = RANUNI(73805);
    VAR1 = 'factor ' !! TRIM(LEFT(PUT(CNT, 2.)));
    VAR2 = 7*RAN1; VAR3 = 3*RAN2;
    VAR4 = 5*RAN3; VAR5 = RAN1*RAN2*RAN3;
    OUTPUT;
  END;
RUN;
*  ==> [ SEE  IMAGE 1 ] <==  *;


* [ STEP 2: DECIDE BASIS FOR DISTINGUISHING RESULTS.    *;
*           USE INFORMATS TO DETERMINE WHICH RTF FORMAT *;
*           WILL BE APPLIED TO EACH TABLE RESULT. ]     *;
PROC FORMAT;
  INVALUE IDXA
  LOW - 3.5   = 1
  3.5 <- HIGH = 2;   * IE, VALUES >3.5 GET 2ND FORMAT *;

  INVALUE IDXB
  LOW - 1.5   = 1
  1.5 <- HIGH = 2;

  INVALUE IDXC
  LOW - 2.5   = 1
  2.5 <- HIGH = 2;

  INVALUE PVAL
  LOW - 0.05   = 1
  0.05 <- HIGH = 2;
RUN;
```

Paper 143-28

```
*+-----------------------------------------------------------+*;
*|                    A P P E N D I X (continued)            |*;
*---                                                        ---*;
*| "Listen To The Data" Example (continued)                  |*;
*+-----------------------------------------------------------+;

* [ STEP 3: PREP THE DATA SET FOR %WRITERTF() ] *;
DATA LISTEN (KEEP=COL: FORMAT);
  * ESTABLISH VARIABLE ORDER = TABLE COLUMN ORDER *;
  ATTRIB COL1-COL5 LENGTH=$9; ATTRIB FORMAT LENGTH=$75;

  * ADD COLUMN HEADERS PRIOR TO ACCESSING THE DATA *;
  IF _N_ EQ 1 THEN DO;
    COL1 = 'FACTOR'; COL2 = 'SITES';
    COL5 = 'P-VALUE'; FORMAT = 'TITLE FST=P';
    OUTPUT;
  END;

  * ESTABLISH FORMAT ARRAYS. INFORMATS CREATED ABOVE *;
  * WILL MAP VALUES TO A SPECIFIC DISPLAY FORMAT     *;
  ARRAY SITEA [2] $18 _TEMPORARY_ ('#BGC=TAN FC=MAR'
                                   '#BGC=LTBLU FC=NAVY');
  ARRAY SITEB [2] $18 _TEMPORARY_ ('# FC=BLUE' '# FC=RED');
  ARRAY SITEC [2] $18 _TEMPORARY_ ('#BGC=NAVY FC=LTBLU'
                                   '#BGC=OLI FC=LTGRE');
  ARRAY PVALU [2] $18 _TEMPORARY_ ('# FST=B' '# DEF');

  SET LISTEN;

  * FORMAT NUMERIC VARIABLES FOR DISPLAY *;
  COL1 = VAR1;
  COL2 = PUT(VAR2, 3.1); COL3 = PUT(VAR3, 3.1);
  COL4 = PUT(VAR4, 3.1); COL5 = PUT(VAR5, PVALUE5.);

  * LISTEN TO THE DATA!! LET THE DATA TELL YOU *;
  * THE CORRECT FORMAT! INFORMATS CONVERT THE  *;
  * DATA TO THE CORRECT ARRAY REFERENCE!       *;
  FORMAT = 'DEF' !!                                  /* COL1 */
           TRIM( SITEA[ INPUT(VAR2, IDXA.) ] ) !!    /* COL2 */
           TRIM( SITEB[ INPUT(VAR3, IDXB.) ] ) !!    /* COL3 */
           TRIM( SITEC[ INPUT(VAR4, IDXC.) ] ) !!    /* COL4 */
           TRIM( PVALU[ INPUT(VAR5, PVAL.) ] );      /* COL5 */
  OUTPUT;
RUN;
*  ==> [ SEE  IMAGE 2 ] <==  *;



* [ STEP 4: CREATE THE RTF DOCUMENT ] *;
%WRITERTF(LISTEN, C:\PROJECT\LISTEN, PORT=1, STYLE=2, COL24=1)
*  ==> [ SEE  IMAGE 3 ] <==
```

Paper 143-28

```
*+---------------------------------+*;
*|   A P P E N D I X (continued)   |*;
*---                             ---*;
*| "Listen To The Data" Images     |*;
*+---------------------------------+*;
```

IMAGE 1: RAW DATA SET

NB:

There is a 1-to-1 correspondence between data set observations and rows in the eventual RTF table.

VIEWTABLE: Work.Listen

	VAR1	VAR2	VAR3	VAR4	VAR5
1	factor 1	4.217	1.986	1.285	0.102
2	factor 2	6.156	0.931	2.646	0.144
3	factor 3	6.943	2.468	0.965	0.157
4	factor 4	4.941	1.054	1.954	0.097
5	factor 5	6.239	0.839	2.898	0.144
6	factor 6	0.916	1.376	1.153	0.014
7	factor 7	0.646	0.821	1.53	0.008
8	factor 8	2.693	0.794	0.336	0.007

IMAGE 2: FORMATTED DATA SET WITH DISPLAY INSTRUCTIONS

NB:

1) TITLE text "Sites" will span columns 2 - 4,
2) TITLE text will appear in small caps font style,
3) <COL1> text will appear in default format, and
4) formatting instructions for remaining columns were determined dynamic-ly using the INFORMATS specified above.

VIEWTABLE: Work.Listen

	COL1	COL2	COL3	COL4	COL5	FORMAT
1	Factor	Sites			P-Value	TITLE FST=P
2	factor 1	4.2	2.0	1.3	0.102	DEF#BGC=LTBLU FC=NAVY# FC=RED#BGC=NAVY FC=LTBLU# DEF
3	factor 2	6.2	0.9	2.6	0.144	DEF#BGC=LTBLU FC=NAVY# FC=BLUE#BGC=OLI FC=LTGRE# DEF
4	factor 3	6.9	2.5	1.0	0.157	DEF#BGC=LTBLU FC=NAVY# FC=RED#BGC=NAVY FC=LTBLU# DEF
5	factor 4	4.9	1.1	2.0	0.097	DEF#BGC=LTBLU FC=NAVY# FC=BLUE#BGC=NAVY FC=LTBLU# DEF
6	factor 5	6.2	0.8	2.9	0.144	DEF#BGC=LTBLU FC=NAVY# FC=BLUE#BGC=OLI FC=LTGRE# DEF
7	factor 6	0.9	1.4	1.2	0.014	DEF#BGC=TAN FC=MAR# FC=BLUE#BGC=NAVY FC=LTBLU# FST=B
8	factor 7	0.6	0.8	1.5	0.008	DEF#BGC=TAN FC=MAR# FC=BLUE#BGC=NAVY FC=LTBLU# FST=B
9	factor 8	2.7	0.8	0.3	0.007	DEF#BGC=TAN FC=MAR# FC=BLUE#BGC=NAVY FC=LTBLU# FST=B

IMAGE 3: RESULTING RTF DOCUMENT
NB: White space (margins) has been removed so image could fit on this page.

November 16, 2002
SAS v8.2 on WIN_98 GIACOMO

FACTOR		SITES		P-VALUE
factor 1	4.2	2.0	1.3	0.102
factor 2	6.2	0.9	2.6	0.144
factor 3	6.9	2.5	1.0	0.157
factor 4	4.9	1.1	2.0	0.097
factor 5	6.2	0.8	2.9	0.144
factor 6	0.9	1.4	1.2	**0.014**
factor 7	0.6	0.8	1.5	**0.008**
factor 8	2.7	0.8	0.3	**0.007**

listen.rtf Page 1 of 1