

Using Advanced Features of DDE to Customize Formatted Tables in Excel

Dennis J. Beal, Science Applications International Corporation, Oak Ridge, TN

ABSTRACT

Dynamic data exchange (DDE) is a powerful tool within SAS® that users can send data stored in SAS data sets directly into formatted Microsoft Excel tables. This process reduces human error and saves time since the user no longer needs to format the table manually. This paper presents advanced features of customized formatting within Excel using SAS code and macros. These advanced features include intermingling mathematical symbols with standard text font within the same cell; bold, superscript, underline or italicize specific characters within individual cells; load both numeric and character data into the same Excel column; loading data from different variables for specific rows within the body of the table; presenting numerical data in multiple formats within the same Excel column; and freezing the column header rows. The intended audience is advanced SAS users who want complete control of their formatted tables without using the Output Delivery System.

1. INTRODUCTION

The Output Delivery System (ODS) within SAS has many file formats into which SAS output can be exported. However, what if the SAS user wants a highly customized format for displaying data that ODS does not currently support? One answer is to use Dynamic Data Exchange (DDE). DDE is a communication protocol available on Windows that enables some applications such as Microsoft Access, Excel, and Word to communicate with each other in a client/server fashion. The client application initiates a conversation with the server application and asks the server to perform a specific task. SAS (version 6.08 and later) can act as a DDE-client, but not as a server. Nearly all things that can be done manually in Excel can be automated from within a base SAS program using DDE. DDE enables a SAS session to take control of the Excel application and tell it precisely what to do. With the use of macro variables, SAS programs become dynamic tools that can export customized formatted tables in Excel. This is an extension of a paper presented by the author at the 2004 Southeast SAS Users Group Conference (Beal 2004).

2. EXAMPLE DATA

In order to illustrate how flexible and powerful DDE can be used to create highly customized tables in Excel from within SAS, a fictitious data set was created. Environmental soil samples were collected at two locations for four quarters and analyzed for several types of chemical analysis: field measurements, inorganics, organic polychlorinated biphenyls (PCBs), organic volatiles and radionuclides. Environmental data are usually qualified by the laboratory analyzing the environmental samples with character qualifiers to determine whether the chemical was detected in the sample or not. Detected results are then compared to current environmental criteria. Most chemicals have a maximum criterion that results should not exceed. However, the field measurement pH has an allowable range (6.5 to 9 standard units). Some chemicals do not have a criterion to compare against the data.

The ratio of the mean to the criteria is calculated for each chemical other than field measurements with at least one detected result. These ratios are then summed across chemicals for an overall sum of ratios (SOR). The SOR should be less than one in order to meet waste acceptance criteria for the landfill to accept the soil for disposal.

The formatted summary table in Excel is shown in Table 1 in section 5. SAS code used to create Table 1 is discussed in sections 3 and 4. The frequency of detection for each chemical is shown in the table as "x / y" where x denotes the number of detected results and y denotes the total number of samples collected. The frequency that detected results exceed the environmental criteria is shown as "x / y" where x denotes the number of detected results that exceeds the criterion and y denotes the number of detected results for that chemical. Some quarters have missing data for some chemicals. Therefore, the frequency of detection may only show 3 results instead of 4. Note the ratio column is summed across chemicals within each location.

3. ASSIGN MACROS

Before the data can be sent directly to Excel using DDE, the data set as it will appear in Excel should be created. In addition, several macros need to be defined that will be used in the commands SAS will send to Excel. Sections 3.1 through 3.8 describe these macros with SAS code. The SAS code presented in sections 3.1 through 3.8 uses the SAS System for personal computers version 9.1.3 SP4 running on a Windows XP Professional platform. SAS/BASE is the only module required to run the SAS code in this paper. Beal (2004) shows SAS code that dynamically transforms the input summarized SAS data set into a SAS data set as it will appear in Excel with header rows.

3.1 INITIAL MACROS

The SAS program begins by defining initial macros that tell SAS the name and location of the input data set, the subdirectory where the Excel file will be saved and name of the Excel file.

```
%let PATHDAT=C:\Projects\data\;      ** subdirectory of input data set;
libname sesug "&PATHDAT";            ** create library name for directory;
%let PATHTAB=C:\Projects\tables\;    ** subdirectory for created Excel table;
%let INPDS = sesug.sesug_summary;    ** input summarized data set name;
%let OUTPDS = outpds;                ** SAS data set name sent to Excel;
%let FN = Summary tables.xls;        ** created Excel filename containing table;
```

The following macros store the number of Excel rows needed for the title (including the blank line after the title), the number of rows needed for the column headings, the maximum number of footnotes (including the blank line after the table and before the footnotes) needed for the table, the number of columns in the table and the estimated maximum number of pages the table will be.

```
%let TITLEHEAD    = 2 ;      ** rows needed for table title (including blank lines);
%let COLHEAD       = 2 ;      ** rows needed for column headings;
%let FOOTNOTE_OBS  = 10;      ** rows needed for footnotes;
%let NUM_COLUMNS   = 11;      ** total number of columns for the table;
%let MAX_PAGES     = 5;       ** estimated maximum number of pages for the table;
%let SHEETNAME     = Soil;    ** sheet name shown in tab in Excel;
```

The macro obsnvars is available in the SAS online documentation. A SAS data set is input and the macro assigns the number of variables and observations in the input data set to macro variables.

```
%macro obsnvars(ds);
  %global dset nvars nob;
  %let dset=&ds;
  %let dsid = %sysfunc(open(&dset));
  %if &dsid %then %do;
    %let nob = %sysfunc(attrn(&dsid,NOBS));
    %let nvars=%sysfunc(attrn(&dsid,NVARS));
    %let rc = %sysfunc(close(&dsid));
  %end;
  %else
    %put Open for data set &dset failed - %sysfunc(sysmsg());
  %mend obsnvars;
```

3.2 DEFINE MACRO VARIABLE FOR NUMBER OF STATIONS

The data set stations contains all the unique stations for soil. The number of unique stations is stored in the macro variable N_STATIONS. Each of the unique station names are stored in the macro variables STATION1, STATION2, etc.

```
proc sort data=&INPDS out=stations(keep=station) nodupkey; by station; run;

  ** store number of unique stations for soil **;
%obsnvars(stations); %let N_STATIONS = &nob; run;

data _null_;
  set stations;
  call symput('STATION'||left(_N_), trim(left(station))); run;
```

3.3 DETERMINE ROWS FOR EXCEL TABLE

The macro datarows stores the first and last rows where data will appear in the table for each station and analysis type in the macro variables STATION&i._MIN_ROW_ANATYPE&j. and STATION&i._MAX_ROW_ANATYPE&j. The Excel row where the analysis type head row will appear is stored in the macro variable STATION&i._ANATYPE_HEADROW&j. The data sets datarows&i._&j. (for varying values of *i* and *j*) contain non-header row data that will be later loaded into the Excel table. The data sets anagroup_rows&i._&j. (for varying values of *i* and *j*) contain the analysis type header row data. The data set headers_station contains the station header row data. The rows the station headers will appear in the Excel table are stored in the macro variables HEAD_STATION_ROW_GROUP&i.

```
%macro datarows;
  %do i = 1 %to &N_STATIONS;
    %global HEAD_STATION_ROW_GROUP&i. N_STATION&i._ANATYPES;

    data station&i.;
      set &OUTPDS; * &OUTPDS is the SAS data set with Excel row and column numbers;
      where station = "&STATION&i.";
      attrib _numeric_ format=best9. informat=best9.;

      proc sort data=station&i. out=anatypes(keep=media station anatype) nodupkey;
        by media station anatype; run; ** run is very important before %obsnvars();

      %obsnvars(anatypes); %let N_STATION&i._ANATYPES = &nobs;

      %do t = 1 %to &&N_STATION&i._ANATYPES;
        %global STATION&i._ANATYPE&t.;
        %end;

    data _null_;
      set anatypes;
      call symput("STATION&i._ANATYPE"||left(_N_), trim(left(anatype))); run;

      %do j = 1 %to &&N_STATION&i._ANATYPES;
        %global STATION&i._MIN_ROW_ANATYPE&j. STATION&i._MAX_ROW_ANATYPE&j.
          STATION&i._ANATYPE_HEADROW&j.;

    data datarows&i._&j.;
      set station&i.;
      where anatype="&STATION&i._ANATYPE&j." and bold=' ';

      proc sort data=datarows&i._&j.; by station anatype;
      proc summary data=datarows&i._&j.;
        var row;
        by station anatype;
        output out=rowmin&i._&j.(drop=_type_ _freq_) min=minrow max=maxrow; run;

    data assign_dat_rows&i._&j.;
      set rowmin&i._&j.;
      call symput("STATION&i._MIN_ROW_ANATYPE&j.", minrow);
      call symput("STATION&i._MAX_ROW_ANATYPE&j.", maxrow); run;

    data anagroup_rows&i._&j.;
      set station&i.;
      where chemical="&STATION&i._ANATYPE&j." and bold='N';
      call symput("STATION&i._ANATYPE_HEADROW&j.", row); run;
    %end; ** end j loop;
  %end; ** end i loop;

  data headers_station;
    set &OUTPDS;
    where bold='Y';
    call symput('HEAD_STATION_ROW_GROUP'||left(_N_), row);
    keep chemical bold row; run;
%mend datarows;
%datarows
```

3.4 BOLD CELLS THAT EXCEED CRITERIA

The macro bold bolds all the contents of a single cell. The inputs to the macro are the row and column numbers for the cell and the font size. The macro bold_all calls the bold macro for each cell that needs bolding. The inputs to the macro bold_all are the data set name that contains the rows and columns of cells that will be bolded and the desired font size of the text. The SAS functions %unquote and %str allow the use of macro variables in the X4ML commands. The data set bolds subsets the data for records that require bolding. Only the row and column variables are kept since this data set is used later in the bold_all macro call in section 4.14.

```
%macro bold(rnum, cnum, size);
data _null_;
  file sas2xl;    ** SAS2XL is defined in section 4.1;
  put %unquote(%str('%[select("r&rnum.c&cnum.")]%')); ** select cell to bold;
  put %unquote(%str('%[format.font("Arial",&size,true,false,false,false,1,
    false,false)]%')); ** third parameter is "true" for BOLD; run;
%mend bold;

%macro bold_all(dsname, size);
  %obsnvars(&dsname); %let TOTAL = &nobs;
  %if &TOTAL > 0 %then %do;
    data _null_;
      set &dsname;
      call symput('ROW' || left(_N_), row);
      call symput('COL' || left(_N_), col); run;
    %local i;
    %do i = 1 %to &TOTAL;
      %bold(&&ROW&i., &&COL&i., &size);
    %end;
  %end;
%mend bold_all;

data bolds;
  set &OUTPDS;
  if xcedsum > 0 then do;
    if (mind ^= . and mind <= criteria_min) or (mind >= criteria and criteria ^= .)
  then do; COL = 7; output; end; ** bold the minimum detected column;
    if (maxd ^= . and maxd <= criteria_min) or (maxd >= criteria and criteria ^= .)
  then do; COL = 8; output; end; ** bold the maximum detected column;
    COL = 9; output; ** bold the criteria column;
    COL = 10; output; ** bold the frequency > criteria column;
  end;
  keep row col; run; ** the variable ROW is defined in the data set &OUTPDS;

  %obsnvars(bolds); %let N_BOLDs = &nobs; run;
```

3.5 FORMAT IN SCIENTIFIC NOTATION

For very large or small numbers displayed in the table, scientific notation may be the preferred format. The data set scientific keeps the rows and columns of cells with numbers that round at least as large as 1.0E+6 or smaller than 1.0E-3. All numeric columns displayed in the table are checked to see if scientific format is required. The macro sci formats a single cell in scientific notation. Its only inputs are the row and column numbers for the cell. The format can be changed to any number of decimal places such as "0.00E+00". The macro sci_all calls the sci macro for each cell that needs to be in scientific notation. The input to the macro sci_all is the data set scientific. The macro sci_all is invoked in section 4.14.

```
data scientific;
  set &OUTPDS;
  if abs(ltmin) >= 999995 or 0 < abs(ltmin) < 0.00995 then do; COL = 4; output; end;
  if abs(ltmax) >= 999995 or 0 < abs(ltmax) < 0.00995 then do; COL = 5; output; end;
  if abs(mean) >= 999995 or 0 < abs(mean) < 0.00995 then do; COL = 6; output; end;
  if abs(mind) >= 999995 or 0 < abs(mind) < 0.00995 then do; COL = 7; output; end;
  if abs(maxd) >= 999995 or 0 < abs(maxd) < 0.00995 then do; COL = 8; output; end;
  if abs(criteria) >= 999995 or 0 < abs(criteria) < 0.00995 then do; COL = 9;
output; end;
```

```

keep row col; run;

%macro sci(rnum, cnum);
  data _null_;
    file sas2xl;
    put %unquote(%str('%[select("r&rnum.c&cnum.")]%')); ** select cell;
    put '[format.number("0.0E+00")]'; run;
%mend sci;

%macro sci_all(dsname);
  %obsnvars(&dsname); %let TOTAL = &nobs;
  %if &TOTAL > 0 %then %do;
    data _null_;
      set &dsname;
      call symput('ROW' || left(_N_), row);
      call symput('COL' || left(_N_), col); run;
    %local i;
    %do i = 1 %to &TOTAL;
      %sci(&&ROW&i., &&COL&i.);
    %end;
  %end;
%mend sci_all;

```

3.6 FORMAT WITH COMMAS

The data set commas contains rows and columns for cells with rounded numeric data between 10,000 and 1E+6 that will be displayed with commas. All numeric columns displayed in the table are checked to see if comma format is required. The macro comma formats a single cell with commas. The symbol '#' tells Excel to display a digit in that position if it exists, whereas "000,000" tells Excel to display leading zeros. The macro comma_all calls the comma macro for each cell requiring formats with commas and is invoked in section 4.14. The input to the macro comma_all is the data set commas.

```

data commas;
  set &OUTPDS;
  if 9995 <= abs(ltmin) < 999995 then do; COL = 4; output; end;
  if 9995 <= abs(ltmax) < 999995 then do; COL = 5; output; end;
  if 9995 <= abs(mean) < 999995 then do; COL = 6; output; end;
  if 9995 <= abs(min) < 999995 then do; COL = 7; output; end;
  if 9995 <= abs(maxd) < 999995 then do; COL = 8; output; end;
  if 9995 <= abs(criteria) < 999995 then do; COL = 9; output; end;
  keep row col;

%macro comma(rnum, cnum);
  data _null_;
    file sas2xl;
    put %unquote(%str('%[select("r&rnum.c&cnum.")]%'));
    put '[format.number("###,###")]'; run;
%mend comma;

%macro comma_all(dsname);
  %obsnvars(&dsname); %let TOTAL = &nobs;
  %if &TOTAL > 0 %then %do;
    data _null_;
      set &dsname;
      call symput('ROW' || left(_N_), row);
      call symput('COL' || left(_N_), col); run;
    %local i;
    %do i = 1 %to &TOTAL;
      %comma(&&ROW&i., &&COL&i.);
    %end;
  %end;
%mend comma_all;

```

3.7 FORMAT TO 3 SIGNIFICANT DIGITS

Usually the reader wants to see only a few significant digits in tables. The data set dec3 keeps all rows and columns with numbers rounded between 0.01 and 9.95. The macro dec3 formats an individual cell to 3 significant digits. The format "0.0#" tells Excel to always display one digit to the left and right of the decimal point, but only display the last digit if one exists. For example, the number 9.30067 is displayed 9.3, whereas the number 1.0176 is displayed 1.02. However, the number with all its precision (defined in the attrib varname format=best9. informat=best9.; SAS statement) is seen in the formula bar in Excel when the cell is clicked on. The macro dec3_all calls the macro dec3 for each cell requiring this format and is invoked in section 4.14. The data set dec3 is its only input. Other macros similar to dec3 can also be defined to display other significant digits. For example, numbers between 10 and 99.9 can be formatted with "##.0" that displays the last digit for non-integers.

```
data dec3;
  set &OUTPDS;
  if 0.00995 <= abs(ltmin) < 9.95 then do; COL = 4; output; end;
  if 0.00995 <= abs(ltmax) < 9.95 then do; COL = 5; output; end;
  if 0.00995 <= abs(mean) < 9.95 then do; COL = 6; output; end;
  if 0.00995 <= abs(mind) < 9.95 then do; COL = 7; output; end;
  if 0.00995 <= abs(maxd) < 9.95 then do; COL = 8; output; end;
  if 0.00995 <= abs(criteria) < 9.95 then do; COL = 9; output; end;
  keep row col; run;

%macro dec3(rnum, cnum);
  data _null_;
    file sas2xl;
    put %unquote(%str('%[select("r&rnum.c&cnum.")]%'));
    put '[format.number("0.0#")]' run;
%mend dec3;

%macro dec3_all(dsname);
  %obsnvars(&dsname); %let TOTAL = &nobs;
  %if &TOTAL > 0 %then %do;
    data _null_;
      set &dsname;
      call symput('ROW' || left(_N_), row);
      call symput('COL' || left(_N_), col); run;
    %local i;
    %do i = 1 %to &TOTAL;
      %dec3(&&ROW&i., &&COL&i.);
    %end;
  %end;
%mend dec3_all;
```

3.8 SUPERScript CELLS IN BODY OF TABLE

Specific characters within the body of the table can be superscripted. Every row for the parameter pH has a footnote "d" that should be superscripted. The following SAS code dynamically identifies which rows and columns are affected and the specific character that should be superscripted. The macro superscript_all is invoked in section 4.14.

```
data superscripts;
  set &OUTPDS;
  where chemical =: 'pH';
  COL = 1;
  LEN = length(chemical); ** the character superscripted is the last character;
  keep row col len;

%macro super(rnum, cnum, len);
  data _null_;
    file sas2xl;
    put %unquote(%str('%[select("r&rnum.c&cnum.")]%')); ** select cell;
    put %unquote(%str('%[font.properties("Arial","Italic",9,,true,,,,,&LEN,1)]%'));
    ** superscript 1 character in column &LEN; run;
%mend super;
```

```

%macro superscript_all(dsname);
  %obsnvars(&dsname); %let TOTAL = &nobs;
  %if &TOTAL > 0 %then %do;
    data _null_;
      set &dsname;
      call symput('LEN' || left(_N_), len);
      call symput('ROW' || left(_N_), row);
      call symput('COL' || left(_N_), col); run;
      %local i;
      %do i = 1 %to &TOTAL;
        %super(&ROW&i., &COL&i., &LEN&i.);
      %end;
    %end;
  %mend superscript_all;

```

4. SEND DATA TO EXCEL

Now that a data set has been created as it will appear in Excel and all the necessary formatting macros have been defined, we now focus on the specific commands used in SAS to send the formats and data to Excel. Our goal is to create from within SAS a high quality formatted table in Excel that is ready for publication or to be imported into Word. The server application is Microsoft Excel 2003. Excel version 4 Macro Language (X4ML) is used to communicate the specific commands from SAS to Excel. Features that were added to Excel after version 4 may not have an X4ML equivalent. For example, SAS code in sections 4.1 through 4.15 will do all Excel formatting to create Table 1 except for displaying the column headings at the top of every page when printed. This feature was added after Excel version 4. This is the only task that must be completed manually in Excel after the table is created by selecting File → Page Setup → Sheet → Rows to repeat at top → r1:r4 where 4 is the number of rows to be repeated on every page. All of the X4ML commands are located in the file “macrofun.exe” online at <http://www.microsoft.com/downloads/search.aspx?displaylang=en>. The executable file will create the “macrofun.hlp” help file that explains the full syntax of hundreds of X4ML functions, examples and cross-references to related commands and functions. A hard copy manual also exists for Excel 4 (Microsoft 1992). There is information that only appears in either the manual or in the help file, so having both should contain all the information needed for X4ML.

4.1 START EXCEL FROM SAS

The following SAS code launches Excel if it is not already running. If Excel is already running, the same SAS code will create a new file named in the macro variable FN defined in section 3.1. However, if an Excel file with the same name already exists in the macro variable PATHTAB subdirectory, this existing file will be overwritten as long as the file is not open. If the file is closed, it will be overwritten. If an Excel file with the same name as FN is open from a different subdirectory than PATHTAB, a DDE error will occur because Excel cannot have two files with the same name open at the same time, even if they are in separate subdirectories. The options statement uses the ‘@’ symbol for missing values because when the X4ML function formula.replace(“@”, “ ”) is executed, it searches for the ‘@’ symbol and replaces it with a blank. If the default missing symbol (a period “.”) is used, Excel will replace the decimal point of each number with the specified symbol (blank in this case). The symbol used can be any symbol that will not be displayed in the Excel table. The filename sas2xl dde “excel|system” statement is a DDE system doublet that is used throughout the remaining SAS program and tells SAS the server application receiving the commands. The first _null_ data step launches Excel. The macro variable TAB is the hexadecimal tab that is used to move between columns in Excel. The macro variable S is used to delete the one trailing blank from character variables. The second _null_ data step creates a new Excel file with the name specified in the macro variable FN with one worksheet named from the SHEETNUM macro variable. The error(false) command turns off the error checking within Excel and prevents Excel from displaying any message boxes that would stop the process until they are clicked away.

```

options noxsync noxwait xmin missing=@;
filename sas2xl dde 'excel|system';
data _null_;
  length fid rc start stop time 8;
  fid=fopen('sas2xl','s');
  if (fid le 0) then do;
    rc=system('start excel');
    start=datetime();
    stop=start+10;
    do while (fid le 0);

```

```

        fid=fopen('sas2xl','s');
        time=datetime();
        if (time ge stop) then fid=1;
        end;
    end;
    rc=fclose(fid); run;

%let tab='09'x;    * tab-character;
%let s+=(-1);     * use to delete trailing blank from character variables;

data _null_;
    file sas2xl;
    put '[new(1)]';          ** new workbook with 1 sheet;
    put '[error(false)]';   ** switch off Excel error messages;
                                ** save the file with the given file name;
    put %unquote(%str('%[Save.as("&pathtab.&FN")]%'));
%end; run;

```

4.2 MACROS FOR SELECTING ROWS AND CELLS

The sel_rows and sel_cells macros are simple but critically important macros that are used ubiquitously throughout the remaining SAS program. The sel_rows macro creates the DDE triplet filename reference RECRANGE for a specified range of rows and columns and tells Excel precisely where in the table to write data from a SAS data set. The notab option is essential because it will prevent the default conversion of blanks into delimiters within Excel.

```

%macro sel_rows(row_startno, col_startno, row_endno, col_endno);
    filename recrange dde "excel[&FN]sheet1!r&row_startno.c&col_startno.c
        &col_startno.:r&row_endno.c&col_endno." notab;
%mend sel_rows;

```

The sel_cells macro allows SAS macro variables to be used in selecting rows and columns for receiving formatting commands. For example, the macro call %sel_cells(3, 1, 4, &NUM_COLUMNS) selects rows 3 through 4 for columns 1 through &NUM_COLUMNS to receive the next X4ML formatting commands.

```

%macro sel_cells(row_startno, col_startno, row_endno, col_endno);
    put %unquote(%str('%[select("r&row_startno.c&col_startno.:
        r&row_endno.c&col_endno.")]%'));
%mend sel_cells;

```

4.3 MACRO VARIABLES FOR ASSIGNING ROWS

Key rows from the table must be stored into macro variables that will be used as inputs to the %sel_rows macro. The macro variables are self-explanatory by their names.

```

%macro rows;
    %local i;
    %global FIRST_ROW_COL_HEAD TITLE_PLUS_COLHEAD FIRST_ROW_DATA
        LAST_ROW_DATA FIRST_ROW_FOOTNOTES LAST_ROW_FOOTNOTES;
    %let FIRST_ROW_COL_HEAD = %eval(&TITLEHEAD + 1);
    %let TITLE_PLUS_COLHEAD = %eval(&TITLEHEAD + &COLHEAD);
    %let FIRST_ROW_DATA = %eval(&TITLE_PLUS_COLHEAD + 1);
    %let LAST_ROW_DATA = %eval(&TITLE_PLUS_COLHEAD + &N_DATASET);
    %let FIRST_ROW_FOOTNOTES = %eval(&LAST_ROW_DATA + 1);
    %let LAST_ROW_FOOTNOTES = %eval(&LAST_ROW_DATA + &FOOTNOTE_OBS);
    %do i = 1 %to &FOOTNOTE_OBS;
        %global ROW_&i._FOOTNOTES;
        %let ROW_&i._FOOTNOTES = %eval(&LAST_ROW_DATA + &i.);
    %end;
%mend rows;
%rows

```

The following SAS code assigns the rows where the sum of ratios (SOR) is displayed into SAS macro variables SOR_ROW1, SOR_ROW2, etc.

```

data sor;
  set &OUTPDS;
  where pxcde='SOR';
  call symput('SOR_ROW'||left(_N_), row); run;

%obsnvars(sor); %let N_SOR = &nobs;

```

4.4 OVERRIDING EXCEL DEFAULT FORMATS

Excel has its own default formatting rules when it displays data. Often we want to use the formats we specify in SAS and not use the default formats in Excel. For example, the criterion for pH in our data set has a range from 6.5 to 9 standard pH units. If we send the text string "6.5 – 9" to Excel, Excel may interpret the dash between the numbers as a minus sign and subtract 9 from 6.5. Similarly, Excel will interpret the text string "3 / 4" for the frequency of detection as the date 03/04/2006. The macro variable N_PH_ROWS contains the number of rows where the criteria must be presented as text.

```

data ph;
  set &OUTPDS;
  where criteria_txt ^= ' ';
  call symput('PH_ROW'||left(_N_), row); run;

%obsnvars(ph); %let N_PH_ROWS = &nobs;

```

The symbol '@' in the following data step `_null_` tells Excel to leave the formatted SAS values alone and not to use the default Excel formats for the first row of data through the last row of footnotes for columns 1 through 3. Only the '@' symbol can be used for overriding Excel default formats. Note no numeric data will appear in these first 3 columns. In addition, the tenth column showing the frequency of detected results exceeding the criteria also should not use the default Excel formats. Last, the code tells Excel to use SAS formats for the rows where the criteria displayed should be text (such as pH) in column 9.

```

data _null_;
  file sas2xl;
  %sel_cells(&FIRST_ROW_DATA, 1, &LAST_ROW_FOOTNOTES, 3);
  put '[format.number("@")]'; ** rid Excel formats and use SAS formats;
  %sel_cells(&FIRST_ROW_DATA, 10, &LAST_ROW_FOOTNOTES, 10);
  put '[format.number("@")]';
  %do i = 1 %to &N_PH_ROWS;
    %sel_cells(&PH_ROW&i., 9, &PH_ROW&i., 9);
    put '[format.number("@")]';
  %end; run;

```

4.5 TABLE TITLE AND COLUMN HEADINGS

The title of the table is stored in the data step `lengths`. The length of the title is also calculated because a footnote will be shown at the end of the title. The length of the table title and the title are then stored in the macro variables `TITLE_LENGTH` and `TITLE`.

```

data lengths;
  TITLE="Table 1. Summary of Quarterly &SHEETNAME Dataa";
  ltitle=length(title);
  call symput('TITLE_LENGTH', ltitle);
  call symput('TITLE', trim(left(title))); run;

```

The `_null_` data step writes the title in the first row and column headings on rows 3 and 4. The tab macro variable is used to tab across columns when writing the column headings. There is an extra space at the end of some column headings in the last put statement so the headings on this row will look even on the table due to the superscripts in columns 6, 9 and 10.

```

%sel_rows(1, 1, &TITLE_PLUS_COLHEAD, &NUM_COLUMNS);
data _null_;
  file rechange;
  put "&TITLE";
  put;

```

```

        put &tab &tab 'Frequency of' &tab 'Non-detects' &tab &tab &tab 'Detected' &tab
&tab &tab 'Frequency >';
        put 'Chemical ' &tab 'Units ' &tab 'Detection ' &tab 'Min ' &tab 'Max ' &tab
'Meanb' &tab 'Min ' &tab 'Max ' &tab 'RLUAc' &tab 'RLUAc' &tab 'Ratio ';
run;

```

4.6 STATION AND ANALYSIS TYPE HEADINGS

The macro headings places the formatted station and analysis type headings into the body of the table. The data sets and rows assigned to each heading were stored in macro variables in section 3.3. The X4ML function `border(,,,1,1)` puts a single thin line over and under each station header row. The third parameter of the `format.font` function toggles the bold on while the fourth parameter toggles the italics on. The `alignment(7)` function centers the station heading across all columns, while `alignment(2)` left justifies the analysis type heading.

```

%macro headings;
  %local i j;
  %do i = 1 %to &N_STATIONS;
    %sel_rows(&&HEAD_STATION_ROW_GROUP&i., 1, &&HEAD_STATION_ROW_GROUP&i., 1);
    data _null_;
      set headers_station;
      if _N_=&i.;
        file rechange;
        put CHEMICAL; run;
    data _null_;
      file sas2xl;
      %sel_cells(&&HEAD_STATION_ROW_GROUP&i., 1, &&HEAD_STATION_ROW_GROUP&i.,
&NUM_COLUMNS);
      put '[border(,,,1,1)]';
      put '[format.font("Arial",9,true,true,false,false,1,false,false)]';
      put '[alignment(7)]'; ** 7=center across selection; run;

  %do j = 1 %to &&N_STATION&i._ANATYPES;
    %sel_rows(&&STATION&i._ANATYPE_HEADROW&j., 1, &&STATION&i._ANATYPE_HEADROW&j., 1);
    data _null_;
      set title_station&i._anatype&j.;
      file rechange;
      put CHEMICAL; run;
    data _null_;
      file sas2xl;
      %sel_cells(&&STATION&i._ANATYPE_HEADROW&j., 1,
&STATION&i._ANATYPE_HEADROW&j., &NUM_COLUMNS);
      put '[format.font("Arial",9,false,true,false,false,1,false, false)]';
      put '[alignment(2)]'; ** 2=left justify across selection; run;
  %end;
  %end;
%mend headings;

```

4.7 AUTOFIT COLUMNS

The macro autofit reduces the column widths for all columns to the minimum width so all data are visible. This macro is invoked in section 4.14 after all data have been written to the table.

```

%macro autofit(startno, endno);
  data _null_;
    file sas2xl;
    put %unquote(%str('%[column.width(0,"c&startno.:c&endno.",false,3)]%'));
    run;
%mend autofit;

```

4.8 FORMAT DATA CELLS

The macro `borders_data` formats the data cells by replacing missing values with either blanks or dashes. The symbol in the formula `replace` function need not be '@', but it must agree with the latest options `missing=` statement. For this project blanks are preferred for missing data in columns 4, 5, 7 and 8 while dashes are preferred

for missing data in columns 9, 10 and 11. The data in the first column are left justified while data in the other columns are centered.

```
%macro borders_data(min_row, max_row);
  data _null_;
    file sas2xl;
    %sel_cells(&MIN_ROW, 4, &MAX_ROW, 5);
    put '[formula.replace("@", " ")]';
    %sel_cells(&MIN_ROW, 6, &MAX_ROW, 6);
    put '[formula.replace("@", "--")]';
    %sel_cells(&MIN_ROW, 7, &MAX_ROW, 8);
    put '[formula.replace("@", " ")]';
    %sel_cells(&MIN_ROW, 9, &MAX_ROW, &NUM_COLUMNS);
    put '[formula.replace("@", "--")]';
    %sel_cells(&MIN_ROW, 1, &MAX_ROW, 1);
    put '[format.font("Arial",9,false,false,false,false,1,false,false)]';
    put '[alignment(2)]'; ** 2=left justified;
    %sel_cells(&MIN_ROW, 2, &MAX_ROW, &NUM_COLUMNS);
    put '[format.font("Arial",9,false,false,false,false,1,false,false)]';
    put '[alignment(7)]'; ** 7=center across selection; run;
%mend borders_data;
```

4.9 LOAD DATA

The macro load_data loads the data from the SAS data sets into the appropriate rows and columns in the Excel table. Separate put statements are used depending on whether the criteria is text or numeric and for the separate sum of ratios line. Two spaces are inserted before the chemical name to help offset the chemical names from the analysis type headers. The macro TAB is used to tab between columns. The borders_data macro is called to format the data after it is written into the table.

```
%macro load_data;
  %local i j;
  %do i = 1 %to &N_STATIONS;
    %do j = 1 %to &&N_STATION&i._ANATYPES;
      %sel_rows(&&STATION&i._MIN_ROW_ANATYPE&j., 1,
        &&STATION&i._MAX_ROW_ANATYPE&j., &NUM_COLUMNS);
      data load&i._&j.;
        set datarows&i._&j.;
        file rechange;
        if criteria_txt = ' ' and pxcdd ^= 'S' then do;
          put ' ' CHEMICAL &s &tab UNITS &s &tab PDETC &s &tab LTMIN &tab LTMAX
            &tab MEAN &tab MIND &tab MAXD &tab CRITERIA &tab PXCED &tab SOR;
          end;
        else if criteria_txt ^= ' ' and pxcdd ^= 'S' then do;
          put ' ' CHEMICAL &s &tab UNITS &s &tab PDETC &s &tab LTMIN &tab LTMAX
            &tab MEAN &tab MIND &tab MAXD &tab CRITERIA_TXT &tab PXCED &tab SOR;
          end;
        else if pxcdd = 'S' then do;
          put &tab &tab &tab &tab &tab &tab &tab &tab PXCED &tab SOR;
          end; run;

      %borders_data(&&STATION&i._MIN_ROW_ANATYPE&j., &&STATION&i._MAX_ROW_ANATYPE&j.);
    %end;
  %end;
%mend load_data;
%load_data
```

4.10 FORMAT TITLE AND COLUMN HEADINGS

The following _null_ data step performs several formatting functions for the title and column headings. The title is centered across all columns in Arial 9 point bold type. All column headings are also in Arial 9 point bold type. Groups of individual cells are centered. For example, "r3c3" in the select function tells Excel to center the contents of the cell in row 3 column 3, while "r3c4:r3c5" tells Excel to center the contents of the cells ("Non-detects") in columns 4

and 5 in row 3. Note that this centering across individual cells cannot be performed manually in Excel without first merging the cells.

```
data _null_;
  file sas2xl;
  *** center title and specify fonts;
  %sel_cells(1, 1, &TITLEHEAD, &NUM_COLUMNS);
  put '[alignment(7)]'; ** 7=center across selection;
  put '[format.font("Arial",9,true,false,false,false,1,false,false)]';

  *** specify fonts and center column headings;
  %sel_cells(&FIRST_ROW_COL_HEAD, 1, &TITLE_PLUS_COLHEAD, &NUM_COLUMNS);
  put '[format.font("Arial",9,true,false,false,false,1,false,false)]';
  put '[select("r3c3, r3c4:r3c5, r3c7:r3c8, r3c10")]';
  put '[alignment(7)]';
  %sel_cells(&TITLE_PLUS_COLHEAD, 1, &TITLE_PLUS_COLHEAD, &NUM_COLUMNS);
  put '[alignment(7)]';
```

The font.properties function superscripts 1 character beginning at position &TITLE_LENGTH of the title in row 1 column 1 using Arial Bold Italic type font. Similarly, the last character in each of the column headings on row 4 is superscripted.

```
put '[select("r1c1")]';
put %unquote(%str(%[font.properties("Arial","Bold Italic",9,,true,,,,,,
  &TITLE_LENGTH,1)]%)));
** superscript last character in each column header;
%sel_cells(&TITLE_PLUS_COLHEAD, 1, &TITLE_PLUS_COLHEAD, 1);
put '[font.properties("Arial","Bold Italic",9,,true,,,,,,9,1)]';
%sel_cells(&TITLE_PLUS_COLHEAD, 2, &TITLE_PLUS_COLHEAD, 2);
put '[font.properties("Arial","Bold Italic",9,,true,,,,,,6,1)]';
%sel_cells(&TITLE_PLUS_COLHEAD, 3, &TITLE_PLUS_COLHEAD, 3);
put '[font.properties("Arial","Bold Italic",9,,true,,,,,,10,1)]';
%sel_cells(&TITLE_PLUS_COLHEAD, 4, &TITLE_PLUS_COLHEAD, 5);
put '[font.properties("Arial","Bold Italic",9,,true,,,,,,4,1)]';
%sel_cells(&TITLE_PLUS_COLHEAD, 6, &TITLE_PLUS_COLHEAD,);
put '[font.properties("Arial","Bold Italic",9,,true,,,,,,5,1)]';
%sel_cells(&TITLE_PLUS_COLHEAD, 7, &TITLE_PLUS_COLHEAD, 8);
put '[font.properties("Arial","Bold Italic",9,,true,,,,,,4,1)]';
%sel_cells(&TITLE_PLUS_COLHEAD, 9, &TITLE_PLUS_COLHEAD, 10);
put '[font.properties("Arial","Bold Italic",9,,true,,,,,,5,1)]';
%sel_cells(&TITLE_PLUS_COLHEAD, 11, &TITLE_PLUS_COLHEAD, 11);
put '[font.properties("Arial","Bold Italic",9,,true,,,,,,6,1)]';
```

The following SAS code instructs Excel to use subscripts in the column headings for the unit average remediation levels.

```
put '[select("r4c9:r4c10")]';
put '[font.properties("Arial","Bold",9,,,true,,,,,3,2)]';
** subscript 2 characters starting in column 3;
```

The border function puts a single thin line (5th parameter = '1') under columns 4 and 5 in row 3 under "Non-detects" and under columns 7 and 8 in row 3 under "Detected." The border function puts a double line (5th parameter = '6') under all columns on row 4. The function puts a medium thick line (first parameter = '2') as an outline for the entire table.

```
** put borders around columns and rows;
%sel_cells(&FIRST_ROW_COL_HEAD, 4, &FIRST_ROW_COL_HEAD, 5);
put '[border(,,,1)]'; ** 1=single thin line under each row ;
%sel_cells(&FIRST_ROW_COL_HEAD, 7, &FIRST_ROW_COL_HEAD, 8);
put '[border(,,,1)]';
%sel_cells(&TITLE_PLUS_COLHEAD, 1, &TITLE_PLUS_COLHEAD, &NUM_COLUMNS);
put '[border(,,,6)]'; ** 6=double line under row ;
%sel_cells(&FIRST_ROW_COL_HEAD, 1, &LAST_ROW_DATA, &NUM_COLUMNS);
put '[border(2)]'; ** puts medium thick line as outline for entire table;
```

4.11 SET MARGINS, HEADERS AND FOOTERS

The `page.setup` function specifies the left, right, top and bottom margins, portrait or landscape orientation, footers and headers. The font size, left or right justification, font type and text "DRAFT" for the header are specified in the first parameter, while the second parameter controls these same specifications for the footer. The eleventh parameter is a '2' for landscape orientation or '1' for portrait. The thirteenth parameter is the scale number representing the percentage to increase or decrease the size of the sheet so all text will fit onto a page.

```
put %unquote(%str('%[page.setup(%nrquote("&9&R&"Arial""DRAFT"),%nrquote("&9&C&"
"Arial""Page &P of &N&9&R&"Arial"),1,1,1,1,,false,true,false,2,,100)]%'));
```

4.12 FREEZE FRAMES

If the table is longer than one page, the column headers can be displayed in Excel on the second and subsequent pages using the `freeze.pages` function.

```
%sel_cells(&FIRST_ROW_DATA, 1, &LAST_ROW_DATA, &NUM_COLUMNS);
put %unquote(%str('%[freeze.panes(true,0,&TITLE_PLUS_COLHEAD)]%'));
```

4.13 USING MATHEMATICAL SYMBOLS WITH TEXT

Different fonts can be used for specific characters within the same cell. The last row of each station begins with "SOR" where the sums of ratios are calculated. The text in the SAS data set reads, "SOR = S Mean/RLUA =". The following SAS code tells Excel to use the symbol font for the 7th character in the string ("S") that will display the mathematical summation symbol capital sigma (Σ). In addition, the 16th and 17th characters in the string ("UA") are subscripted. The cell above the SOR in the last column is underlined to denote a partial sum of that column.

```
%macro SOR;
%do k = 1 %to &N_SOR;
  data _null_;
    file sas2xl;
    %sel_cells(&SOR_ROW&k., &NUM_COLUMNS, &SOR_ROW&k., &NUM_COLUMNS);
  put '[border(,,,1,1)]'; ** 1=single thin line over and under each row, col 11 ;
  %sel_cells(&SOR_ROW&k., 10, &SOR_ROW&k., 10);
  put '[format.font("Arial",9,false,false,false,1,false,false)]';
  put '[font.properties("Symbol",,9,,,,,,,,,7,1)]'; ** changes "S" to sigma;
  put '[font.properties("Arial",,9,,,true,,,,,16,2)]'; ** subscript;
  put '[alignment(4)]'; ** 4=right justify;
  run;
%end;
%mend SOR;
%SOR
```

4.14 LAST FORMATS

Several macros defined in previous sections are now invoked. It is important to autofit the first column before inserting the analysis type group headings since they may be longer than the chemical names. Then the `autofit` macro can be invoked for the remaining columns after the numeric cells in these columns have been formatted.

```
%autofit(1, 1);
%headings;
%bold_all(bolds, 9);
%sci_all(scientific);
%superscript_all(superscripts);
%dec3_all(dec3);
%comma_all(commas);
%autofit(2, &NUM_COLUMNS);
```

4.15 FOOTNOTES

The `put_footnotes` macro outputs footnotes at the end of the table. The footnotes can be formatted with the footnotes in superscript italic font while specific words can be italicized, bolded or underlined within the same cell. Some footnotes can be output only if a criterion is met. For example, the footnote stating that results exceeding

criteria are shown in bold is output only if there are cells bolded in that table by testing the N_BOLDS macro variable. The first _null_ data step outputs the text of the footnotes while the second _null_ data step formats specific words or letters of the footnotes.

```
%macro put_footnotes;
%sel_rows(&FIRST_ROW_FOOTNOTES, 1, &LAST_ROW_FOOTNOTES, &NUM_COLUMNS);
data _null_;
  file recrange;
  put;
  put 'a Fictional data are shown here for the sole purpose of this paper.';
  put 'b Mean is shown to 3 significant digits or the nearest integer. Nondetected
results were used as reported.';
  put 'c Unit average remediation level taken from County Clerk Records by Howard
Sprague (http://www.countycriteria.gov).';
  put 'd The ratio of the mean concentration to the RLUA is not calculated for
pH.';
  put %unquote(%str(%-- Unit average remediation level not available, so
frequencies exceeding level and ratios cannot be calculated.%));
  %if &N_BOLDS > 0 %then %do;
    put 'Results that exceed criteria are shown in bold.';
  %end;
  put 'PCB = polychlorinated biphenyls.';
  put 'SOR = Sum of ratios of each chemical mean to its unit average remediation
level for chemicals with at least one detected result.'; run;

data _null_;      ** set fonts for footnotes;
  file sas2xl;
%sel_cells(&FIRST_ROW_FOOTNOTES, 1, &LAST_ROW_FOOTNOTES, &NUM_COLUMNS);
  put '[format.font("Arial",8,false,false,false,false,1,false,false)]';
  put '[alignment(2)]'; ** 2=left justified;
%sel_cells(&ROW_2_FOOTNOTES, 1, &ROW_5_FOOTNOTES, 1);
  ** superscript footnote in italic Arial font at character position 1;
  put '[font.properties("Arial","Italic",8,,true,,,,,,1,1)]';
%sel_cells(&ROW_4_FOOTNOTES, 1, &ROW_4_FOOTNOTES, 1);
  ** italicize name of document beginning in position 45 for 20 chars long;
  put '[font.properties("Arial","Italic",8,,,,,,45,20)]';
  ** underline URL beginning in position 85 for 29 characters long;
  put '[font.properties("Arial",,8,,,,,true,,,85,29)]';
  %if &N_BOLDS > 0 %then %do;
    %sel_cells(&ROW_7_FOOTNOTES, 1, &ROW_7_FOOTNOTES, 1);
    put '[font.properties("Arial","Bold",8,,,,,,43,4)]'; ** bold the word BOLD;
  %end; run;
%mend put_footnotes;
%put_footnotes
```

5. TABLE FORMATTED

Table 1 is the formatted table in Excel created directly from within SAS.

Table 1. Summary of Quarterly Soil Data^a

Chemical	Units	Freq. of Detection	Non-detects		Mean ^b	Detected		RL _{UA} ^c	Frequency > RL _{UA} ^c		Ratio
			Min	Max		Min	Max				
Old Man Kelsey's Woods											
Field Measurements											
pH ^d	Std Unit	4 / 4			6.3	5.7	6.9	6.5 - 9	1 / 4	--	
Inorganics											
Aluminum	mg/kg	0 / 4	2010	2908	2447			50,000	0 / 0	0	
Lead	mg/kg	4 / 4			411	205	749	500	1 / 4	0.82	
Mercury	mg/kg	2 / 4	10	17.5	158	175	430	325	1 / 2	0.49	
Organic PCBs											
PCB-1254	mg/kg	1 / 4	6	10	12.1	25.6	25.6	210	0 / 1	0.06	
PCB-1260	mg/kg	3 / 4	10	10	56.2	45.0	121	220	0 / 3	0.26	
Organic Volatiles											
Benzene	mg/kg	1 / 4	19	25	42.9	107	107	255	0 / 1	0.17	
Xylenes	mg/kg	0 / 4	1.0E-06	5.0E-06	3.3E-06			1.0E-05	0 / 0	0	
Radionuclides											
Gross Alpha	pCi/g	4 / 4			394	373	475	--	--	--	
Gross Beta	pCi/g	2 / 4	0.5	0.9	68.4	118	154	50	2 / 2	1.37	
								SOR = Σ Mean/RL _{UA} = 3.17			
R.E. Lee Natural Bridge											
Field Measurements											
pH ^d	Std Unit	3 / 3			7.4	6.6	8.9	6.5 - 9	0 / 4	--	
Inorganics											
Aluminum	mg/kg	1 / 4	200	257	1127	4000	4000	50,000	0 / 1	0.02	
Lead	mg/kg	2 / 4	20	21	157	41	547	500	1 / 2	0.31	
Mercury	mg/kg	1 / 4	9.3	17.5	29.3	75	75	325	0 / 1	0.09	
Organic PCBs											
PCB-1254	mg/kg	1 / 4	4	10	10.8	19	19	210	0 / 1	0.05	
PCB-1260	mg/kg	2 / 4	10	10	42	29	119	210	0 / 2	0.20	
Organic Volatiles											
Benzene	mg/kg	2 / 4	19	20	83	23	270	255	1 / 2	0.33	
Xylenes	mg/kg	0 / 3	4.0E-06	9.0E-06	6.0E-6			1.0E-05	0 / 0	0	
Radionuclides											
Gross Alpha	pCi/g	3 / 4	10	10	265	15	423	--	--	--	
Gross Beta	pCi/g	3 / 4	0.5	0.5	26.6	16	45	50	0 / 3	0.53	
								SOR = Σ Mean/RL _{UA} = 1.53			

^a Fictional data shown here for the sole purpose of this paper.

^b Mean is shown to 3 significant digits or the nearest integer. Non-detected results were used as reported.

^c RL_{UA} = Unit average remediation level taken from *County Clerk Records* by Howard Sprague (<http://www.countycriteria.gov>).

^d The ratio of the mean concentration to the RL_{UA} is not calculated for pH.

-- Unit average remediation level not available, so frequencies exceeding level and ratios cannot be calculated.

Results that exceed unit average remediation levels are shown in **bold**.

PCB = polychlorinated biphenyls.

SOR = Sum of ratios of each chemical mean to its unit average remediation level for chemicals with at least one detected result.

6. CONCLUSION

This paper has shown an example SAS program that uses Dynamic Data Exchange (DDE) to create highly customized, formatted tables in Excel. Although more SAS code is usually needed to produce a customized table compared to ODS, the advantages over ODS are many. First, the potential for human error while formatting an Excel table manually is virtually eliminated since data are output directly from SAS into a high quality, report-ready table. Second, the time efficiency gained by having SAS produce tables directly without formatting manually is significant. Third, DDE gives the SAS user complete control over how the table will be formatted down to the minute details. Finally, with the use of SAS macro variables there is practically no limit to the types of formatted tables that can be created in Excel from SAS. Once the SAS program is operational it can easily be adapted to produce similar tables for other projects and applications to gain even more efficiency.

REFERENCES

- Beal, Dennis (2004), "Using Dynamic Data Exchange to Customize Formatted Reports in Microsoft® Excel," *Proceedings of the 12th Annual Southeast SAS Users Group Conference*, Paper DP03.
- Bodt, M. (1996), "Talking to PC Applications Using Dynamic Data Exchange," *Observations*, 5(3), 18-27.
- Microsoft Corporation (1992), *Microsoft Excel Function Reference*, Document Number AB26298-0592.
- SAS Institute (1999), "Technical Support Document #325 – The SAS System and DDE" <<http://ftp.sas.com/techsup/download/technote/ts325.pdf>> (July 19, 2004).
- Schreier, H. (1998), "Getting Started with Dynamic Data Exchange," *Proceedings of the Sixth Annual Southeastern SAS Users Group Conference*, 207-215.
- Vyverman, Koen (2000), "Using Dynamic Data Exchange to Pour SAS Data into Microsoft Excel," *Proceedings of the 18th SAS European Users Group International Conference*.
- Vyverman, Koen (2002), "Using Dynamic Data Exchange to Export Your SAS Data to MS Excel – Against All ODS, Part I," *Proceedings of the 27th Annual SAS Users Group Conference*.

CONTACT INFORMATION

The author welcomes and encourages any questions, corrections, feedback and comments. Contact the author at:

Dennis J. Beal
Statistician / Risk Scientist
Science Applications International Corporation
P.O. Box 2501
151 Lafayette Drive
Oak Ridge, Tennessee 37831
phone: 865-481-8736
fax: 865-481-8714
e-mail: dennis.j.beal@saic.com

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

Other brand and product names are registered trademarks or trademarks of their respective companies.