Macro to Read Mock Shells in Excel and Create a Copy in Rich Text Format

Rakesh Mucha, Sarah Cannon Research Institute, Nashville, TN
Jeffrey Johnson, Sarah Cannon Research Institute, Nashville, TN

ABSTRACT

It is a common practice for statisticians in pharmaceutical industry to create mock shells as part of the statistical analysis plan. It not only helps reviewers to know what tables, listings, and figures (TFLs) they can expect as part of the analysis, but also gives them a chance to recommend necessary changes based on protocol and/or their clinical expertise. It also serves as a reference for SAS programmers to generate actual tables, listings, and figures based on the mock shells. Even though actual TFLs are typically presented in Rich Text Format (RTF), some statisticians prefer producing mock shells in Excel for various reasons like traditionally creating mock shells in Excel, and creation of mock shells in Excel is relatively easy to RTF. But, it is preferable to create mock shells in Rich Text Format for various reasons like sponsor/programmers prefer shells in RTF and actual TFLs will be presented in RTF. In this paper we introduce a macro, xl2rtf, which was developed in SAS® 9.3 and can be used to read mock shells in Excel and creates a copy of shells in RTF. This gives Statisticians the flexibility of presenting mock shells in RTF without having to actually create mock shells in RTF. This paper expects basic SAS programming knowledge for the users so they can make the macro work for their study by updating the in path and out path of the shells, and also make minor updates according to their environment or shells.

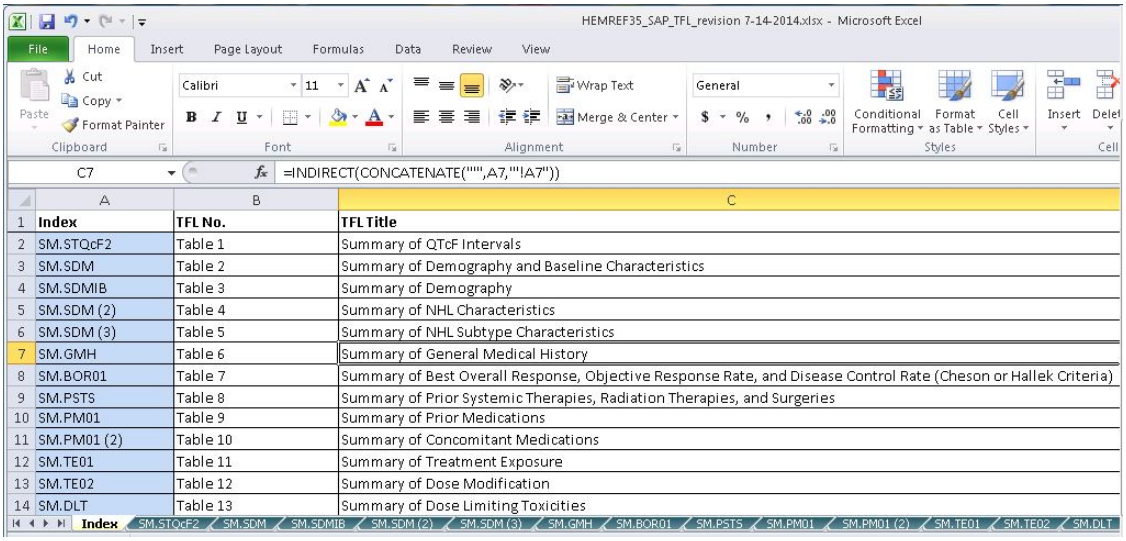
INTRODUCTION

Creating a set of table, and listing mock shells in RTF when you have the same shells in Excel is similar to re-inventing the wheel. Also, creating or updating shells in RTF is relatively time consuming to Excel. The purpose of the macro is to help statisticians and/ or programmers who already have mock table, listing shells in Excel, but wants to present them Rich Text Format. Primary audience of interest are statisticians and/or programmers of pharmaceutical industry who create mock shells in Excel and want to have a copy of them in RTF.

This paper will give a walkthrough for the users including:

1. How can this macro help users
2. Overview of the macro
3. What macro expects from the shell

HOW CAN THIS MACRO HELP USERS



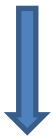
	A	B	C
1	**Index**	**TFL No.**	**TFL Title**
2	SM.STQcF2	Table 1	Summary of QTcF Intervals
3	SM.SDM	Table 2	Summary of Demography and Baseline Characteristics
4	SM.SDMIB	Table 3	Summary of Demography
5	SM.SDM (2)	Table 4	Summary of NHL Characteristics
6	SM.SDM (3)	Table 5	Summary of NHL Subtype Characteristics
7	SM.GMH	Table 6	Summary of General Medical History
8	SM.BOR01	Table 7	Summary of Best Overall Response, Objective Response Rate, and Disease Control Rate (Cheson or Hallek Criteria)
9	SM.PSTS	Table 8	Summary of Prior Systemic Therapies, Radiation Therapies, and Surgeries
10	SM.PM01	Table 9	Summary of Prior Medications
11	SM.PM01 (2)	Table 10	Summary of Concomitant Medications
12	SM.TE01	Table 11	Summary of Treatment Exposure
13	SM.TE02	Table 12	Summary of Dose Modification
14	SM.DLT	Table 13	Summary of Dose Limiting Toxicities

Figure 1. Index tab showing a set of mock shells in Excel

If a set of mock shells and an index tab summarizing all those shells (as shown in Figure 1) is provided in Excel, xl2rtf macro can create a copy in RTF individually for all of those shells starting from SM.STQcF2 through SM.DLT (as shown in Figure 1) or specific shell(s) based on the input given to the macro.

	A	B	C	D	E	F	G	H	I	J
1	Index									
2										
3	XXXXXXXXX, Inc.									Page x of y
4	Protocol:XXXXXXXXXX									
5										
6				Table 41						
7				Summary of QTcF Intervals						
8				Safety Analysis Set						
9										
10										
11				Maximun QTcF (msec)				Change from Baseline QTcF (msec)		
12	Cohort/ Assigned Dose (mg)	Protocol Timepoint	Assessment Timepoint	<480 n(%)	>=480 - <500 n(%)	>=500 n(%)		<30 n(%)	>=30 - <60 n(%)	>=60 n(%)
13										
14	Cohort 1/5 (mg)	Cycle 1 Day 1	Pre-dose	x(xx.x)	x(xx.x)	x(xx.x)				
15		Cycle 1 Day 1	30 min. post-dose	x(xx.x)	x(xx.x)	x(xx.x)		x(xx.x)	x(xx.x)	x(xx.x)
16		Cycle 1 Day 1	1 hr post-dose	x(xx.x)	x(xx.x)	x(xx.x)		x(xx.x)	x(xx.x)	x(xx.x)
17		XXXXXXXXXXXXX	XXXXXXXXXXXXX	x(xx.x)	x(xx.x)	x(xx.x)		x(xx.x)	x(xx.x)	x(xx.x)
18		XXXXXXXXXXXXX	XXXXXXXXXXXXX	x(xx.x)	x(xx.x)	x(xx.x)		x(xx.x)	x(xx.x)	x(xx.x)
19		XXXXXXXXXXXXX	XXXXXXXXXXXXX	x(xx.x)	x(xx.x)	x(xx.x)		x(xx.x)	x(xx.x)	x(xx.x)
20										
21	<insert Cohort>									
22										
23	Note: Baseline value is the mean of the triplicate pre-dose reads in Cycle 1 Day 1.									
24										
25	Program Name: XXXX									
26	Datasets(date): X_XXX.XX, X_XXX.XXX (DDMMMYYYY)									
27	Generation Date/Time : YYYY-MM-DD/HH:MM									
28										
29										

Figure 2. Mock Shell of a Summary Table SM.STQcF2 in Excel



XL2RTF MACRO

XXXXXXXXX, Inc.
Protocol:XXXXXXXXXX **Page x of y**

Table 41
Summary of QTcF Intervals
Safety Analysis Set

			Maximun QTcF (msec)			Change from Baseline QTcF (msec)		
Cohort/ Assigned Dose (mg)	Protocol Timepoint	Assessment Timepoint	<480 n(%)	>=480 - <500 n(%)	>=500 n(%)	<30 n(%)	>=30 - <60 n(%)	>=60 n(%)
Cohort 1/5 (mg)	Cycle 1 Day 1	Pre-dose	x(xx.x)	x(xx.x)	x(xx.x)			
	Cycle 1 Day 1	30 min. post-dose	x(xx.x)	x(xx.x)	x(xx.x)	x(xx.x)	x(xx.x)	x(xx.x)
	Cycle 1 Day 1	1 hr post-dose	x(xx.x)	x(xx.x)	x(xx.x)	x(xx.x)	x(xx.x)	x(xx.x)
	XXXXXXXXXXXXX	XXXXXXXXXXXXX	x(xx.x)	x(xx.x)	x(xx.x)	x(xx.x)	x(xx.x)	x(xx.x)
	XXXXXXXXXXXXX	XXXXXXXXXXXXX	x(xx.x)	x(xx.x)	x(xx.x)	x(xx.x)	x(xx.x)	x(xx.x)
	XXXXXXXXXXXXX	XXXXXXXXXXXXX	x(xx.x)	x(xx.x)	x(xx.x)	x(xx.x)	x(xx.x)	x(xx.x)

<insert Cohort>

Note: Baseline value is the mean of the triplicate pre-dose reads in Cycle 1 Day 1.

Program Name: XXXX
Datasets(date): X_XXX.XX, X_XXX.XXX (DDMMMYYYY)
Generation Date/Time : YYYY-MM-DD/HH:MM

Figure 3. Mock Shell of a Summary Table SM.STQcF2 in Rich Text Format produced by the macro xl2rtf

If the mock shell of Table SM.STQcF2 in Excel looks as shown in Figure 2, the macro will create a copy of the shell in Rich Text Format as shown in Figure 3, however you need to put specifications in excel. The macro can create a copy in RTF for an individual Table, Listing Shell or a set of Tables, and /or Listings shells.

OVERVIEW OF THE MACRO

The basic philosophy behind the macro can be divided into five parts:

1. Assigning a unique sequential number in increments of one to each shell based on the index, and storing them as macro variables.
2. Dividing the shell into six different sections like Header, width/justification, column label, spanning information, body, footnote; and creating a data set for each section.
3. Storing each variable of every section into a macro variable.
4. Calling macro variables while writing PROC REPORT depending on the nature of those macro variables. Macro variables storing information relating to header will be called in TITLE statement; spanning information will be called in COLUMN statement of PROC REPORT; width/justification, column label, body will be called in DEFINE statement of PROC REPORT; and footnote will be called in FOOTNOTE statement.
5. Repeating above steps 2 -4 individually for any number of shells provided to the macro i.e. to all the shells referred by the macro variables created in step 1, one at a time.

```
* Creating a macro variable 'file' which has information about the path and filename
of the excel mock shell;
* Creating a macro variable 'outpath' which has information about the path for the
output RTF file;

%let file= User input 1;
%let outpath= User input 2;
%let titlesta= User input 3;
%let titleend= User input 4;

* Importing the 'index' tab of excel mock shells and creaing a sas data set named
'index';

proc import datafile="&file." dbms=xlsx out=index replace;
getnames=no;
sheet="index";
run;

* creating a macro variable 'numouts' which stores the number of shells in the index
sheet for which the first two letters are either SM or LS;

proc sql noprint;
select count(A) into: numouts from index where substr(A,1,2) in ("SM", "LS");
quit;

%let numouts=&numouts;

%macro out;
* You can subset the shells you need by giving a condition;
* If you prefer a single shell or couple shells or all the shells you can do so with a
condition on what to consider;

data index;
      set index;

      if substr(A,1,2) in ("SM", "LS");
      keep A;
run;

* creating macro variables for each of the output name in the index sheet;
* ex: SM.STQcF2, SM.SDM....SM.DLT will be assigned to macro variable out1, out2
......out13 respectively;
```

```
data index2;
       set index;

       call symput ("out"||strip(put(_n_,best.)),strip(A));
run;

* 'j' to start from 1 and loops through the maximum number of outputs stored in macro
variable 'numouts';
* lets assume this as loop 1 (j=1, so 'out1' will be 'SM.STQcF2') to make things
simple and traceable;

%do j=1 %to &numouts.;

* Creating a sas data set named 'mds1' by importing 'SM.STQcF2' tab of the excel mock
shell file, since j=1 (out1 resolves to SM.STQcF2);

proc import datafile="&file." dbms=xlsx out=mds1 replace;
getnames=no; sheet="&&out&j.";
run;

* creating a macro variable 'nvar' which stores the number of columns in 'MDS1';

proc sql noprint;
select nvar into: nvar from dictionary.tables where libname="WORK" and memname="MDS1";
quit;

%let nvar=&nvar.;

* Renaming all the variables;
* ex: A to _1, B to _2, c to _3 ... i to _9 (since number of variables in SM.STQcF2
are 9 or 'nvar' for j=1 is 9);

proc contents data=mds1 out=cont1(keep=memname name) noprint;
run;

data cont2;
       set cont1;
       name2=strip(name)||"=_"||strip(put(_n_,best.));
run;

proc transpose data=cont2 out=cont3;
       by memname;
       var name2;
run;

data cont4;
       set cont3;
       call symput ("renm", catx(" ", of col1-col&nvar.));
run;

data mds2;
       set mds1;
       rename &renm.;
run;

data mds2;
       set mds2;

       do aa=1 TO 29, 31, 127, 129, 141 to 144, 157, 158;
       %do  i=1 %to &nvar.;
       _&i.=tranwrd(_&i.,byte(aa),"~");
       %end;
       %do  i=1 %to &nvar.;
```

```
      _&i.=tranwrd(_&i.,"~~","~");
      %end;
      end;
run;

data mds2;
      set mds2;
      seq=_n_;
run;

* creating a macro variable 'fn' which has information on the start of the footnote
(which line does footnote start from);
* ex: In Figure1, Footnote starts from line 23 so macro variable 'fn' will be 23;

      proc sql noprint;
      select seq into: fn from mds2 where substr(_1,1,2)="$F"; quit;

* Creating data set:
      ttl to store title information
      jwid to store width, justification of the column header
      hdrc to store spanning or grouping of header information
      hdrd to store column headers
      body to store body of the shell
      fotnot to store footnote information;

data ttl jwid hdrc hdrd body fotnot;
      set mds2;
            if &titlesta.<=seq<=&titleend. then output ttl;
            if seq= %eval(&titleend.+1) then output jwid;
            if seq= %eval(&titleend.+2) then output hdrc;
            if seq= %eval(&titleend.+3) then output hdrd;
            if %eval(&titleend.+4) <=seq<&fn. then output body;
            if &fn.<=seq then output fotnot;
run;

* Repeating '_' 133 times and making it the first footnote;
data ftn1;
     _1 = repeat('_',133);
run;

* To divide any footnote greater than 133 characters so that
      first 133 characters go to the first line of the footnote,
      and next 133 characters go to next line (not next footnote) without stopping
displaying at character 133;

data fotnot;
      set ftn1 fotnot ;
      if _n_=11 and _1=" " then delete;
      if length (_1) >133 then _1p=substr(_1,1,133)||"~"||substr(_1,134);
      else _1p=_1;
      drop _1;
      rename _1p=_1;
run;

* Assigning titles to macro variables;
* ex: XXXXXXXXX, inc. will be assigned to titl1, Protocol: XXXXXXXXXX will be assigned
to titl2 and so on;

data ttl2;
      length tit $256;
      set ttl;
      if seq >7 and _1=" " then delete;
```

```
      if seq=3 then tit="bold j=left "||'"'||strip(_1)||'"'||" "||'j=right bold "Page
x of y"';
      if seq=4 then tit="bold j=left "||'"'||strip(_1)||'"';
      if seq>4 then tit="bold j=center "||'"'||strip(_1)||'"';
      tit=tranwrd(tit,'"',' "');
      call symput ("titl"||strip(put(_n_,best.)),tit);
run;

* Assigning footnotes to macro variables;
* ex: "Note: Baseline value is the mean of the triplicate pre-dose reads in Cycle 1
Day 1." will be assigned to ftnt1 and so on;

data fotnot2;
      length fot $256;
      set fotnot;
      _1=strip(tranwrd(_1,"$F"," "));
      fot="j=left "||'"'||strip(_1)||'"';
      fot=tranwrd(fot,'"',' "');
      call symput ("ftnt"||strip(put(_n_,best.)), fot);
run;

data ttl2_;
      set ttl2;
      _2="ttl";
run;

data fotnot2_;
      set fotnot2;
      _2="ttl";
run;

*  Creating macro variable 'maxtitl' that has information on the maximum number of
titles in the shell;
*  Creating macro variable 'maxftnt' that has information on the maximum number of
footnotes in the shell;

proc sql noprint;
      select count( _2) into:maxtitl from ttl2_;
      select count( _2) into:maxftnt from fotnot2_;
quit;

* Creating variables that has information on when and how to span row headers;

data hdrc2;
      set hdrc;
      %if &nvar <7 %then
      %do;

      %do  i=1 %to &nvar.;
      if scan(_&i.,2,"$")="A1" then
      __&i.='("'||tranwrd(_&i.,"$A1", '" "_____________"')||"_&i.)";

      else if scan(_&i.,2,"$")="A2" then
      __&i.='("'||tranwrd(_&i.,"$A2", '" "_____________________"')||"_&i.
_%eval(&i.+1))";

      else if scan(_&i.,2,"$")="A3" then
      __&i.='("'||tranwrd(_&i.,"$A3", '" "_____________________________"')||"_&i.
_%eval(&i.+1) _%eval(&i.+2))";

      else if scan(_&i.,2,"$")="A4" then
```

```
      __&i.='("'||tranwrd(_&i.,"$A4", '"
"____________________________________________"')||"_&i. _%eval(&i.+1) _%eval(&i.+2)
_%eval(&i.+3))";

      else if scan(_&i.,2,"$")="A5" then
      __&i.='("'||tranwrd(_&i.,"$A5", '"
"_____________________________________________" ')||"_&i. _%eval(&i.+1)
_%eval(&i.+2) _%eval(&i.+3)  _%eval(&i.+4))";

      else if scan(_&i.,2,"$")="A6" then
      __&i.='("'||tranwrd(_&i.,"$A6", '"
"______________________________________________________"')||"_&i.
_%eval(&i.+1) _%eval(&i.+2) _%eval(&i.+3)  _%eval(&i.+4) _%eval(&i.+5))";

      else if scan(_&i.,2,"$")="A7" then
      __&i.='("'||tranwrd(_&i.,"$A7", '"
"_______________________________________________________"')||"_&i.
_%eval(&i.+1) _%eval(&i.+2) _%eval(&i.+3)  _%eval(&i.+4) _%eval(&i.+5))";

      else if scan(_&i.,2,"$")="A8" then
      __&i.='("'||tranwrd(_&i.,"$A8", '"
"_____________________________________________"')
||"_&i. _%eval(&i.+1) _%eval(&i.+2) _%eval(&i.+3)  _%eval(&i.+4) _%eval(&i.+5))";

      else __&i.="_&i.";

      drop _&i.;
      rename __&i.=_&i.;
      %end;
      %end;

      %else
      %do;
      %do  i=1 %to &nvar.;

      if scan(_&i.,2,"$")="A1" then
      __&i.='("'||tranwrd(_&i.,"$A1", '" "__________"')||"_&i.)";

      else if scan(_&i.,2,"$")="A2" then
      __&i.='("'||tranwrd(_&i.,"$A2", '" "_____________"')||"_&i. _%eval(&i.+1))";

      else if scan(_&i.,2,"$")="A3" then
      __&i.='("'||tranwrd(_&i.,"$A3", '" "__________________"')||"_&i. _%eval(&i.+1)
_%eval(&i.+2))";

      else if scan(_&i.,2,"$")="A4" then
      __&i.='("'||tranwrd(_&i.,"$A4", '" "_____________________"')||"_&i.
_%eval(&i.+1) _%eval(&i.+2) _%eval(&i.+3))";

      else if scan(_&i.,2,"$")="A5" then
      __&i.='("'||tranwrd(_&i.,"$A5", '" "_____________________________" ')||"_&i.
_%eval(&i.+1) _%eval(&i.+2) _%eval(&i.+3)  _%eval(&i.+4))";

      else if scan(_&i.,2,"$")="A6" then
      __&i.='("'||tranwrd(_&i.,"$A6", '"
"_____________________________"')||"_&i. _%eval(&i.+1) _%eval(&i.+2)
_%eval(&i.+3)  _%eval(&i.+4) _%eval(&i.+5))";

      else if scan(_&i.,2,"$")="A7" then
      __&i.='("'||tranwrd(_&i.,"$A7", '"
"______________________________"')||"_&i. _%eval(&i.+1) _%eval(&i.+2)
_%eval(&i.+3)  _%eval(&i.+4) _%eval(&i.+5))";
```

```
       else if scan(_&i.,2,"$")="A8" then
       __&i.='("'||tranwrd(_&i.,"$A8", '"
"_____"')||"_&i. _%eval(&i.+1)
_%eval(&i.+2) _%eval(&i.+3)  _%eval(&i.+4) _%eval(&i.+5))";

       else __&i.="_&i.";

       drop _&i.;
       rename __&i.=_&i.;
       %end;
       %end;
run;

data hdrc3;
       set hdrc2;
       mdupvar=" ";

       %do  i=1 %to &nvar.;
       if length(_&i.)>3 then
       do;
       _&i.mkbk="mdupvar;"||strip(substr(scan(compbl(_&i.),5,'"'),4));
       _&i.mkbk2=tranwrd(strip(_&i.mkbk)," ","=' '; ");
       _&i.mkbk3=tranwrd(strip(_&i.mkbk2),")","=' '; ");
       call symput("_&i.mvn",_&i.mkbk3);
       end;
       else
       do;
       call symput("_&i.mvn","mdupvar");
       end;
       %end;
run;

data hdrc3;
       set hdrc3;
       %do  i=1 %to &nvar.;
              drop &&_&i.mvn;
              call symput ("colmn", catx(" ", of _1-_&nvar.));
       %end;
run;

* Assigning labels of column headers to macro variables;
* ex: Cohort/Assigned Dose (mg) will be assigned to labl1, ....>=60 (n%) will be
assigned to labl9;
proc transpose data=HDRd out=HDRd2 ;
       by seq;
       %do  i=1 %to &nvar.;var  _&i.;%end;;
run;

data hdrd3;
       set hdrd2;
       collbl='"'||strip(col1)||'"';
       call symput ("labl"||strip(put(_n_,best.)), collbl);
run;

* Assigning width and justification information of column headers to macro variables;
* ex: LL1.3 will be assigned to jwid1, ....CC.8 will be assigned to jwid9;

data jwid2;
       set jwid;
       %do  i=1 %to &nvar.;
       if upcase(substr(_&i.,1,2))="LL" then _jw&i.="left style =
{cellwidth="||strip(substr(_&i.,3))||"in asis=on just=L}";
```

```
      if upcase(substr(_&i.,1,2))="LC" then _jw&i.="left style =
{cellwidth="||strip(substr(_&i.,3))||"in asis=on just=C}";
      if upcase(substr(_&i.,1,2))="CL" then _jw&i.="center style =
{cellwidth="||strip(substr(_&i.,3))||"in asis=on just=L}";
      if upcase(substr(_&i.,1,2))="CC" then _jw&i.="center style =
{cellwidth="||strip(substr(_&i.,3))||"in asis=on just=C}";
      drop _&i.;
      rename _jw&i.=_&i.;
      %end;
run;

proc transpose data=jwid2 out=jwid3 ;
      by seq;
      %do  i=1 %to &nvar.;var  _&i.;%end;;
run;

data jwid4;
      set jwid3;
      call symput ("jwid"||strip(put(_n_,best.)), col1);
run;

options topmargin=1.0in  bottommargin=1.0in  leftmargin=1.0in  rightmargin=1.0in;
options nodate nonumber papersize=letter orientation=landscape;
ods escapechar="~";

*Calling titles stored in macro variables titl1-titl&maxtitl;

%do i=1 %to &maxtitl.;
title&i. &&titl&i.;
%end;

*Calling titles stored in macro variables ftnt1-titl&maxftnt;

%do i=1 %to &maxftnt.;
footnote&i. &&ftnt&i.;
%end;

* Output path for the RTF file;

ODS rtf FILE="&outpath.\&&out&j...rtf" STYLE=g_styles  ;
ods listing close;

proc report data=body center missing headline headskip nowd split="~" ls = 250 ps =
47;

*Calling macro variable 'colmn' which has information of all the column headers;
      columns &colmn.;

* Calling macro variables jwid, labl which defines label, width and justification of
the column headers;

      %do i=1 %to &nvar.;
      define _&i.   /  &&labl&i. order=data &&jwid&i.;
      %end;

run;
quit;

ods rtf close;
%end;
%mend;

%out;
```

WHAT MACRO EXPECTS FROM THE SHELL

Everybody has their own style of formatting the mock shells. Some leave a line between the header information and body, and others leave couple lines. Also, each shell is different in its own way. Some shells needs to have more information in the body in order to communicate clearly on how the final output with the data would/should look like, where as other need not. Since formatting of shells could vary from study to study or even within the study there should be some indication on identifying different sections (as explained above in Overview of the macro) of the shell like header, column labels, body etc. so macro can work properly. This macro can work for any study that has shell(s) that follows the specifications detailed below.

There are six important specifications/requirements shell(s) need to have/follow, for the macro to work properly and yield expected results:

1. Header information could spread over any number of lines, but the start and end line must be provided as input to the macro variables titlesta and titleend.
2. Frist line after end of title/header (titleend) should have width and justification information for each column.

   Ex: First 'L' of 'LL1.3' in cell A10 indicates column label will be left justified, second 'L' indicates actual content of the column will be left justified and the number '1.3' indicates width of the column in inches.
3. Second line after end of title/header may have any category headers that might be necessary to categorize columns. There must be '$An' sign at end of each category header where 'n' would be the number of columns this category header refers to.
4. Third line after end of title/header must have column labels
5. Body of the shell should start anywhere from, fourth line after end of title/header.
6. Footnote must always start with '$F'.

Figure 4 below is a visual illustration on how and where to have the specifications/requirements mentioned above.

	A	B	C	D	E	F	G	H	I	J
1	Index									
2										
3	XXXXXXXXXX, Inc.									Page x of y
4	Protocol: XXXXXXXXXX									
5										
6	Table 41									
7	Summary of QTcF Intervals									
8	Safety Analysis Set									
9										
10	LL1.3	LL1.3	LL1.4	CC.8	CC.8	CC.8	CC0.1	CC.8	CC.8	CC.8
11				Maximun QTcF (msec)$A3				Change from Baseline QTcF (msec)$A3		
12	Cohort/ Assigned Dose (mg)	Protocol Timepoint	Assessment Timepoint	<480 n(%)	>=480 - <500 n(%)	>=500 n(%)		<30 n(%)	>=30 - <60 n(%)	>=60 n(%)
13										
14	Cohort 1/5 (mg)	Cycle 1 Day 1	Pre-dose	x(xx.x)	x(xx.x)	x(xx.x)				
15		Cycle 1 Day 1	30 min. post-dose	x(xx.x)	x(xx.x)	x(xx.x)		x(xx.x)	x(xx.x)	x(xx.x)
16		Cycle 1 Day 1	1 hr post-dose	x(xx.x)	x(xx.x)	x(xx.x)		x(xx.x)	x(xx.x)	x(xx.x)
17		XXXXXXXXXXXXX	XXXXXXXXXXXXX	x(xx.x)	x(xx.x)	x(xx.x)		x(xx.x)	x(xx.x)	x(xx.x)
18		XXXXXXXXXXXXX	XXXXXXXXXXXXX	x(xx.x)	x(xx.x)	x(xx.x)		x(xx.x)	x(xx.x)	x(xx.x)
19		XXXXXXXXXXXXX	XXXXXXXXXXXXX	x(xx.x)	x(xx.x)	x(xx.x)		x(xx.x)	x(xx.x)	x(xx.x)
20										
21	<insert Cohort>									
22										
23	$FNote: Baseline value is the mean of the triplicate pre-dose reads in Cycle 1 Day 1.									
24										

Figure 4. Mock Shell of a Summary Table SM.STQcF2 in Excel with specifications

In Figure 4. Header started on line 3, ended on line 9, width/justification information is on line 10, category headers are on line 11, column labels are on line 12, and body of the shell started on line 13 and ended before line 23 where footnote started.

ADAPTING THE MACRO

Replace “User input 1” with path and file name of the excel mock shells.

```
Ex: %let file= C:\Users\rne8826\Desktop\HEMREF35\HR35_SAP_TFL_revision 7-14-2014.xlsx;
```

Replace “User input 2” with path where RTF file needs to be saved.

```
Ex: %let outpath= C:\Users\rne8826\Desktop\HR35;
```

Replace “User input 3” with the starting line of header.

```
Ex: %let titlesta=3;
```

Replace “User input 4” with ending line of header.

```
Ex: %let titleend=9;
```

CONCLUSION

This macro will primarily be helpful for the statisticians/programmers who have been designing or would like to design shells in Excel, but want to present them in Rich Text Format. To convert shells from Excel to RTF, all you do is have shells in Excel with specifications, customize the macro according to your environment (updating the paths, input macro variables etc.), and run the macro. While adding specifications to shells, time consuming process is adding column width/justifications (Line 10 of Figure 4) manually. To avoid this tedious work, there are some automated ways to write that information in Excel shells. One way is to use visual basic code, which is out of scope for this paper. Some secondary advantages would be: since information of the shells like header, footnote etc. is stored as macro variables; programmers can make use of those macro variables, rather than actually typing titles, footnotes, column headers etc. which is out of scope and not explained in this paper. There are some limitations to the use of the macro including but not limited to: It doesn’t work for shells that has more than one category headers; it doesn’t accept symbols “≤”/”≥” but will accept “<=”/”>=” etc. Maintenance of the macro is not so difficult for users with intermediate or advanced knowledge of SAS®, and maintenance gets easy with repeated use of the macro.

ACKNOWLEDGEMENTS

We thank Kajdasz Daniel, Shuangli Guo for their valuable suggestions in developing and improvising the macro.

CONTACT INFORMATION

Your comments and questions are valued and encouraged. Please feel free to contact the author at:

Name: Rakesh Mucha
Enterprise: SCRI, Nashville, TN
Address: 3322 West End Avenue, Suite 900
City, State ZIP: Nashville, TN - 37221
Work Phone: 615-329-7626
E-mail: rakesh.mucha@scri-innovations.com,

Name: Jeffrey Johnson
Enterprise: SCRI, Nashville, TN
Address: 3322 West End Avenue, Suite 900
City, State ZIP: Nashville, TN - 37221
Work Phone: 615-329-7288
E-mail: Jeffrey.johnson@scri-innovations.com