ABSTRACT

Biostatisticians devote numerous hours of research time to the development of tables, figures, and listings (TFL) mock shells. The information provided by the mock shells include titles, footnotes, and formatting details, such as column labels, alignments, and widths. Statistical SAS programmers are tasked with transferring this information word by word into the final TFLs. Manual transfer of this metadata can be challenging, labor-intensive, and prone to errors. Several SAS users have written and shared their thoughts on how titles and footnotes can be adopted for TFL production with minimal manual involvement. This paper will discuss and provide a SAS 9.4 macro program to generate ready to use metadata of titles, footnotes, and TFL formatting details from a standard set of TFL mock shells. Also, TFL reporting details such as output labels and program names will be discussed. Automatic transfer of this metadata will eliminate major quality issues and formatting challenges, which in turn will free up more time for statistical programmers and biostatisticians to focus on data analysis.

INTRODUCTION

Often statisticians spend numerous hours developing and reviewing tables, figures, and listings (TFL) mock shells. The information provided by the mock shells include titles, footnotes, and formatting details, such as column labels, widths, and alignments. Statistical programmers are tasked with transferring this information to the final TFLs. Manual transfer of this metadata can be time consuming, tedious, and error prone. Major or minor changes to the mock shells require a programming team member to manually edit effected TFL-generating SAS programs; these edits may result in significant quality issues, such as typos, text omissions, erroneous deletions, changes not saved, changes applied to the wrong program or text, etc. Most TFL errors or reviewer comments are attributed to formatting or erroneous transfer of TFL mock shell metadata. This paper will discuss and provide a SAS 9.4 macro to generate ready to use metadata of titles, footnotes, and formatting details, including column labels, widths, and alignments, from a standard set of TFL mock shells. Also, TFL reporting details such as output labels and program names based on the parent mock shell will be discussed.

MACRO CALL

The initial set up of this macro requires very minimal SAS experience.

```
%tfdh(subset =, span =, type =, tflno =, filename =, path =);
```

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>subset</td>
<td>Used for extracting required columns from the mock shell e.g subset in('F1','F2','F3','F4','F7'), skip 'F5' and 'F6' if not needed for a particular deliverable.</td>
</tr>
<tr>
<td>span</td>
<td>Number of columns to span over. This parameter should not be blank if subset macro parameter above is not blank.</td>
</tr>
<tr>
<td>type</td>
<td>Used for extracting metadata for desired deliverable e.g CSR, ASCO, Cohort review, etc</td>
</tr>
<tr>
<td>tflno</td>
<td>Used for altering TFL numbering criteria. Set the parameter to NEW if new TFL numbers are needed, else set to blank.</td>
</tr>
<tr>
<td>filename</td>
<td>Name of TFL mock shell file name e.g TFL_Shells.xlsx</td>
</tr>
<tr>
<td>path</td>
<td>The path to import the TFL mock shells and to store resulting metadata.</td>
</tr>
</tbody>
</table>

Macro parameters `subset`, `span`, `type` and `tflno` may require frequent updates depending on the desired deliverables and changes in the TFL numbering criteria.
REQUIREMENTS

TFL mock shells are created in Excel and stored as an Excel workbook (.xlsx) file. The first Excel sheet (labeled as the index page) is reserved for the table of contents (see below). Each of the Excel sheets following the index page will be populated with TFL mock shells (one mock shell per sheet).

For the purposes of this program, column A through D should be labeled as shown above. Columns following column D can be labelled as needed by the user. These additional columns may be used to store further TFL identifiers for unique study deliverables. The contents of these identifiers can be used to alter the numbering criteria, by concatenating the text part of column B and any preferred column after column D (see macro call parameters type and tflno).

The program extract above illustrates how an additional column after D can be used to subset and change the numbering criteria of a desired TFL deliverable.

Below is an example of a standard TFL mock shell in Excel.
MACRO EXECUTION AND RESULTS

TITLES

Rows 3 to 9 of the standard mock shell are reserved for titles. This program assumes that the titles in rows 3 and 4 are primarily left aligned with some information possibly aligned to the right. The titles in rows 5 to 9 are aligned to the center. The program extract below can be edited if alternate title alignments are desired.

if _n_ le 8 and _n_ gt 1 then do;
  if _n_ in(2,3) and index ^='' then title='title'||strip(put((_n_ -1),2.))||
    ' j=l '''|strip(index)||
    ' j=r '''|catx(' ',of F2-F%trim(&f))||
    ''; else if _n_ ^=in(2,3) and index ^='' then title =
    'title'||strip(put((_n_ -1),2.))||
    ' j=c '''|strip(index)||
    '; else if _n_ ^=in(2,3) and index ='' then title =
    'title'||strip(put((_n_ -1),2.))||
    '; end;

Note: Row 1 of the mock shell is deleted as soon as the shell is imported to SAS.

FOOTNOTES

The primary row associated with the first footnote depends on the number of rows occupied by the body section of the mock shell. Due to this inconsistency the first footnote will be preceded by $F$. For the purpose of this macro, the last three footnotes will remain as illustrated by the mock shell extract shown previously. The program extract below can be updated as desired.

if substr(F1,1,2)='$F' then do;
  call symput("ffot",put(_n_,3.));/*_n_row associated with the 1st footnote*/
  call symput("flen",input(length(F1),best.));end;
if scan(upcase(F1),1,'/')='GENERATION DATE' then do;
  call symput("efot",put(_n_,3.));/*_n_row associated to last footnote*/
  F1=strip(scan(F1,1,':')||
    ":%sysfunc(date(),ymmddd10.)/&systime'');end;
if scan(F1,1,':')='Datasets(date)' then F1=
  strip(scan(F1,1,':')||
    ":&dataset (&cutoffdt)' ;/*to be assigned later*/
if scan(F1,1,':')='Program Name' then F1=strip(scan(F1,1,':')||
  ": &sid"||
  ": &tfl&i' ;/*&sid=Table No. and _&&tfl&i =excel sheet name*/

Note: This macro assumes that all footnotes are aligned to the left (left justified).
FORMATTING DETAILS: COLUMN LABELS, ALIGNMENTS, AND WIDTHS

Column labels and spanning headers are populated on rows 10 and 11 of the standard mock shell. Column labels for some mock shells may end with (N=xx) to indicate subject population, i.e., big N. Row 10 is reserved for spanning headers ending with $A#$, where # represents the number of columns to span over. Special character tilde (~) specifies the point at which the column label can be split.

Column alignment and widths are populated on row 9 of the standard mock shell. Alignments are specified by the first two letters, for example: C = center and L = left. The first letter aligns the column label and the second letter aligns the final data columns, and the digits following these letters represents the column widths.

Statisticians assign random names to Excel sheets populated with the TFL mock shells. For the purposes of this program, these names will be preceded by SM., SSM., FG., and LS. for summary tables, shift summary table, figures, and listings respectively. SM. commands this program to skip two columns, typically F2 and F3, before the first column with N=xx. The two skipped columns will act as security for future programming and changes to TFL formats.

Below are extracts of column and define statements respectively.

```
<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
</tr>
</thead>
<tbody>
<tr>
<td>TABLE3_SM.DM1</td>
<td>(&quot;S={borderbottomcolor=white borderbottomwidth=.5&quot; F1)</td>
<td>1</td>
<td>3</td>
</tr>
<tr>
<td>TABLE3_SM.DM1</td>
<td>(&quot;Part A 'S=[] 'S={borderbottomcolor=black borderbottomwidth=.5&quot; F4-F5)</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>TABLE3_SM.DM1</td>
<td>(&quot; 'S=[] 'S={borderbottomcolor=white borderbottomwidth=.5&quot; F6-F8)</td>
<td>4</td>
<td>3</td>
</tr>
<tr>
<td>TABLE3_SM.DM1</td>
<td>(&quot;Part B 'S=[] 'S={borderbottomcolor=black borderbottomwidth=.5&quot; F7-F9)</td>
<td>5</td>
<td>3</td>
</tr>
<tr>
<td>TABLE3_SM.DM1</td>
<td>(&quot; 'S=[] 'S={borderbottomcolor=white borderbottomwidth=.5&quot; F9-F9)</td>
<td>7</td>
<td>3</td>
</tr>
</tbody>
</table>
```

Distinct column statement (column B) will be extracted by order (column D) and sorted by id (column C) before being assigned to an independent macro variable.

```
<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>C</th>
</tr>
</thead>
<tbody>
<tr>
<td>TABLE3_SM.DM1</td>
<td>define F1/display &quot;S=[just=left]Characteristic&quot; style(column)=[just=left cellwidth=4.5 in];</td>
<td>3</td>
</tr>
<tr>
<td>TABLE3_SM.DM1</td>
<td>define F4/display &quot;S=[just= center]Cohort 1 ~(N-&amp;tarm4)&quot; style(column)=[just= center cellwidth=0.7 in];</td>
<td>3</td>
</tr>
<tr>
<td>TABLE3_SM.DM1</td>
<td>define F5/display &quot;S=[just= center]Cohort 2 ~(N-&amp;tarm5)&quot; style(column)=[just= center cellwidth=0.7 in];</td>
<td>3</td>
</tr>
<tr>
<td>TABLE3_SM.DM1</td>
<td>define F6/display &quot;S=[just= center]Cohort 3 ~(N-&amp;tarm6)&quot; style(column)=[just= center cellwidth=0.7 in];</td>
<td>3</td>
</tr>
<tr>
<td>TABLE3_SM.DM1</td>
<td>define F7/display &quot;S=[just= center]Cohort 4 ~(N-&amp;tarm7)&quot; style(column)=[just= center cellwidth=0.7 in];</td>
<td>3</td>
</tr>
<tr>
<td>TABLE3_SM.DM1</td>
<td>define F8/display &quot;S=[just= center]Cohort 5 ~(N-&amp;tarm8)&quot; style(column)=[just= center cellwidth=0.7 in];</td>
<td>3</td>
</tr>
<tr>
<td>TABLE3_SM.DM1</td>
<td>define F9/display &quot;S=[just= center]Total ~(N-&amp;tarm9)&quot; style(column)=[just= center cellwidth=0.7 in];</td>
<td>3</td>
</tr>
<tr>
<td>TABLE3_SM.DM1</td>
<td>define F10/display &quot;S=[just= center]Total ~(N-&amp;tarm10)&quot; style(column)=[just= center cellwidth=0.7 in];</td>
<td>3</td>
</tr>
</tbody>
</table>
```

Subject population counts will be assigned to &tarm macro variables at the TFL programming level. Summary variable names to be reported will be preceded by F and variables for empty columns (the white space after the spanning header) will be preceded by b.

REPORTING DETAILS: OUTPUT LABELS AND PROGRAM NAMES

Output labels and program names are created from the table of contents shown previously, by concatenating columns B & C and columns C & A, respectively.

Note: Special characters such as (<, >, ?, etc.) should be avoided.
Maximum Use of TFL Mock Shells, continued

<table>
<thead>
<tr>
<th>ProgName</th>
<th>Label</th>
<th>Order</th>
</tr>
</thead>
<tbody>
<tr>
<td>TABLE1_SM.DS</td>
<td>Table 1 Summary of Patient Disposition</td>
<td>1</td>
</tr>
<tr>
<td>TABLE2_SM.DM</td>
<td>Table 2 Summary of Demographic Characteristics</td>
<td>2</td>
</tr>
<tr>
<td>TABLE3_SM.DM1</td>
<td>Table 3 Summary of Demographic Characteristics</td>
<td>3</td>
</tr>
<tr>
<td>TABLE4_SM.RAE</td>
<td>Table 4 Summary of Treatment Related Adverse Events by Preferred Term</td>
<td>4</td>
</tr>
<tr>
<td>TABLE5_SM.RAE</td>
<td>Table 5 Summary of Treatment Related Adverse Events by Preferred Term</td>
<td>5</td>
</tr>
</tbody>
</table>

TFL output labels (column B) will be extracted by distinct order (column C) and assigned to an independent macro variable (see below).

```
select label into :outlabel separated by ' ' from out.tfdh where order='3'; /*program name or text after _ can be used*/
```

Note: Above sql program snapshot can be repeated to extract titles, footnotes, define, and column statements.

USING/ VALIDATING THE METADATA

The resulting metadata can be easily validated with a dummy dataset whose column names are preceded by F and b to match the column and define statements above. The program below can be used to generate a validation dummy dataset.

```
array av(*) $ 2 F1-F20; /*20 char. variables (F1,...,F20), length = 2*/
array av2(*) $ 2 b1-b20; /*20 char. variables (b1,...,b20), length = 2*/
do a= 1 to dim(av);
   av(a)='' ; av2(a)='' ;
end;
```

FINAL TFL OUTPUT

The formatted TFL shown below is based on a dummy dataset and the metadata discussed above.

```
%Development Innovations
Protocol No.: xxx

Drug: xxx

1 of 1

Table 3
Summary of Demographic Characteristics
Safety Analysis set

<table>
<thead>
<tr>
<th>Characteristic</th>
<th>Part A</th>
<th>Part B</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Cohort 1</td>
<td>Cohort 2</td>
</tr>
<tr>
<td></td>
<td>[n=items]</td>
<td>[n=items]</td>
</tr>
</tbody>
</table>

No available data to support this summary

[1] Patients with multiple categories will be categorized as multiple.
Note: Age is an integer difference in years between date of informed consent and date of birth.
Note: Baseline value is defined as the latest non-missing value on or prior to first study drug administration (CID1).

Program Name: Table3_SM.DM1
Dataset (date): ndataset (stoday)
Generation Date/Time: 2017-02-25 10:10
```

Pre-formatted TFLs give the programming team plenty of time to review and provide comments on the mock shells, which in turn will give the statisticians ample time to make the necessary changes.

OUTPUT LABELS

TFLs labeled with distinct numeric and descriptive text are easy to identify. Reviewing TFLs with meaningless labels can be frustrating and time consuming to the statistician, medical writers, and other Data Science study team members.
Maximum Use of TFL Mock Shells, continued

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Size</th>
<th>Date modified</th>
</tr>
</thead>
<tbody>
<tr>
<td>Table 1 Summary of Patient Disposition.rtf</td>
<td>SAS® Program</td>
<td>7 KB</td>
<td>2/23/2017 7:41 AM</td>
</tr>
<tr>
<td>Table 2 Summary of Demographic Characteristics.rtf</td>
<td>SAS® Program</td>
<td>8 KB</td>
<td>2/23/2017 2:23 PM</td>
</tr>
<tr>
<td>Table 3 Summary of Demographic Characteristics.rtf</td>
<td>SAS® Program</td>
<td>64 KB</td>
<td>2/23/2017 7:37 AM</td>
</tr>
<tr>
<td>Table 4 Summary of Treatment Related Adverse Events by Preferred Term.rtf</td>
<td>SAS® Program</td>
<td>8 KB</td>
<td>2/23/2017 11:58 AM</td>
</tr>
<tr>
<td>Table 5 Summary of Treatment Related Adverse Events by Preferred term.rtf</td>
<td>SAS® Program</td>
<td>8 KB</td>
<td>2/23/2017 11:58 AM</td>
</tr>
</tbody>
</table>

PROGRAM NAMES
Well labeled programs are easy to identify and maintain.

CONCLUSION
Once the TFL mock shells have been approved, a study programming team member can easily execute this macro to generate a metadata of titles, footnotes, formatting and reporting details. Implementing a standard transfer of TFL mock shell’s metadata to the final research summaries will eliminate major quality issues, which in turn will free up more time for statistical programmers and biostatisticians to focus on data analysis.

ACKNOWLEDGMENTS
Thanks to the following colleagues for their contribution to the success of this paper.
David Moorman, Principal Biostatistician/ IIT Manager
Julia Shi, Biostatistician Fellow
Tamara Tester, Biostatistician Fellow

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

Julius Kirui, MS  
Biostatistician / SAS Programmer
Sarah Cannon Development Innovations (SCDI)  
1100 Charlotte Avenue, Suite 800  
Nashville, TN 37203
615.329.7274 (Phone), 615.524.4453 (Direct), 615.524.4453 (Fax)  
Julius.kirui@scresearch.net, Juliuskr3@gmail.com, Sarahcannon.com
APPENDIX 1

Titles, footnotes, define and headers (TFDH) macro program

%macro tfdh(type=, subset=, outlabel=, span=, path=, filename=, toc=, tflno=, nshell=, out=);
LIBNAME XLSLIB "&path\&filename"
data toc;
set XLSLIB."&toc"n(where=(&type ^= ''));
%if &tflno=NEW %then %do;
    yy=compress(TFL_No_,'ka')||''||compress(&type);%end; /*new numbering*/
%else %if &tflno= %then %do;
yy=TFL_No_;%end;
s=length(TFL_No_)+5;
run;
proc sql noprint;
/*tot=total No. of mock shells; yy=Table numbers*/
select count(index),'$'||strip(put(max(s),3.))||'.'
into :tot, :tl
from toc;
select index,put(yy,&tl.)||strip(TFL_Title),put(yy,&tl.)
into :tfl1-%tfl%ntrim(&tot),:title1-%title%ntrim(&tot),:tabno1-%tabno%ntrim(&tot)
from toc;
quit;
%if &nshell eq 0 %then %let i=1;
%else %if &nshell gt 0 %then %do;
    %let i=&nshell;
    %let tot=&nshell;%end;
%do %while (&i<=&tot);
%if &i gt 0 %then %do;
    proc contents data=XLSLIB."&&tfl&i"n out=list(keep=name length) noprint; run;
    proc sql noprint;
    /*grap the last useful column and assign it to macro var f*/
    select max(input(compress(name,'F'), best.)) into :f
    from list where upcase(name) like '%F%' and length gt 1;
    quit;
    data _st(rename=(index=F1)) ;
    length index title $260;
    set XLSLIB."&&tfl&i"n;
    index=strip(tranwrd(index,"'"')) ; /*replace " quotation with ' quotation marks*/
    if upcase(F%trim(&f))='PAGE X OF Y' then F%trim(&f)='`{pageof}';
    if _n_=5 then do;if _n_=5 then index="&tabno11";
        call symput('tflid2',strip(index));
        call symput('sid',strip(compress(index)));%end;
        if _n_ =6 then call symput('tfllabel',strip(index));
    <<Titles>>
        if _n_ in(9,10,11) then id=_n_;
    run;
    %let efot=0;
    data _st;
    set _st;
    <<assigning footnotes>>
    run;
    length label $260 title $260;
    set _st;
    label="&&title&i" ;
    if title ^='';
    run;
    data _st1;
    informat F1-F%trim(&f) $260.; format F1-F%trim(&f) $260.;length F1-F%trim(&f) $260;
    set _st(where=(id ^= .));
    array av(*) F1-F%trim(&f);
    do a=1 to dim(av);
    if id=9 then do; av(a)=tranwrd(av(a),'C','center / '));
        av(a)=tranwrd(av(a),'L','left / '));end;
    if id=10 then do; if av(a) ^='' then av(a)=tranwrd(av(a),'$A',' $$ '));end;
    if id=11 then do; av(a)=tranwrd(av(a),'(N=','~(N= '));
    run;
%end; %end; %end; %end; %end;
%let efot=0;
%end; %end; %end; %end; %end;

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 trademarks of their respective companies.
Maximum Use of TFL Mock Shells, continued

```
av(a)=tranwrk(av(a),'(n=','~(n='));
if scan(av(a),2,~)=''' and length(compress(scan(av(a),1,=''))) ge 2
then av(a)=strip(scan(av(a),1,=''))||=' XX';end;end;
run;
proc transpose data=_st1 prefix=col out=_st1;
id id;
var Fl-F%trim(&f);
run;
%let spin=0;%let lastspin=0;%let lastshift=0;
data _st2;
set _st2 %if %length(&subset) gt 0 %then %do;
  %if %scan(&TFL&i,1,.'')=SM %then %do;(where=(_name_ &subset)); %end;
  %else %if %scan(&TFL&i,1,.'') ^=SM %then %do; %end;%end;
  %else %if %length(&subset) le 0 %then %do; %end;;
retain _ ;
id=_n_;
if coll1 ='' then _=0; _+1;
if coll '~' then _=.;
if _ ^=. then blank+=1;
if coll1 ='' then do coll1='b'||strip(put(blank,3.)); coll0='b';end;
if scan(upcase(coll0),1,~)='(N= XX)' or (scan(upcase(coll0),1,~)='GRADE' and scan(coll0,2,~) in ('0','1','2','3','4','5')) then id=_n_;
call symput('nspin', blank);
if coll0 ~in('','b') then call symput('lastspin', _n_);
run;
proc sql noprint;
select min(id) into :firstarm from _st2 where id ^=.;
select id,compress(scan(col10,2,'$$')),'kd' into :spin1:-spin10, :shift1:-shift10
from _st2 where col10 ^='';
select sum(input(scan(col9,3, '/' ),best..)) into df1 from _st2;
quit;
data _st3;
set _st2;
length define $ 500;
%if %scan(&&TFL&i,1,.'')=SM %then %do;if id ge &firstarm then ids=id+2;
  else ids=id;
  order=strip(put((ids=blank),3.));
if _name_='Fl' then do;
  col9=substr(col9,1,length(col9)-3)||
  strip(put($sysevalf(8.7-&df1)+scan(col9,3,'/' ),best..));
  if coll1 ='' then coll1=''';end;
  if scan(upcase(coll1),2,~)=''(N= XX)' then coll1=scan(coll1,1,='''||
    strip(put("&tarm",6..))||strip(put(order,3.)));||');
  %if %length(&span) gt 0 %then %do;
    if _n_ = &lastspin then coll0=scan(col10,1,'$$')||'$$'||
      strip(put("&span",3.)); %end;
  %else %if %length(&span) le 0 %then %do; %end;%end;
  %else %if %scan(&&TFL&i,1,.'') ^=SM %then %do;
    if _ ^=. then define='define F'||strip(order)||'/display "S={just= ''||
      strip(scan(col9,1,''''))||'"|strip(order)||"cellwidth='||strip(scan(col9,3,''''))||' in'];
  else define='define F'||strip(order)||'/display"style(column)=just=''||
      strip(scan(col9,2,''''))||'cellwidth='||strip(scan(col9,3,''''))||' in ];
  if _ ^=. then col=coll1;
else coll=Fl||strip(order);
call symput('order',compress(put(order,3.)));
Maximum Use of TFL Mock Shells, continued

data _st4;
  length label $ 260 column $ 500;
  set _st3;
  by blank;
  if col10 ^= '' then do;
    sp=input(scan(col10, 2, '$$'), best.);
    sptext=strip(scan(col10, 1, '$$'));
    %eval(&ffot+10); run;
  end;
  label=strip(put(_n_, 2));
  if &ffot ge 0 then do;
    %let dmy=1;
    set _st3;
    %if &nshell ^= 0 then do;
      if &nshell ge 1 then do;
        if _n_ lt &ffot and length(F1) gt 0 then column=strip(put((order, 3.))||''');
        else if _n_ ge &ffot and length(F1) le 0 then column=''' S='| strip(put((order, 3.))||''');
      end;
    end;
  end;
run;

proc sort data=_st4 out=column(keep=label id column)nodupkey;
  by column;
run;
proc sort data=_st4 out=define(keep=label define) by label id;
%if &efot=0 then %let efot=%eval(&ffot+10);

data ft;
  set _st3;
  if _n_=&ffot and length(F1) gt 2 then do;
    if F1 ^= '' then do;
      F1=tranwrd(F1, '''', '''');
      F1=substr(F1, 3, %eval(&flen-2));
    end;
  end;
run;

length footnotes $ 500 label $ 260;
set ft;
  symput("nf",(_n_));
  label="&title&i";
  call symput("nf",(_n_));
  F1=tranwrd(F1, 'YYYY-MM-DD', '&CUTOFDPD');
  if F1 ^= '' then do;
    footnotes='footnote'||strip(put(_n_, 2.))||'' j=1'' ||strip(F1)||'';';
  end;
run;

data xx;
  length ProgName $50 label $260;
  merge title ft define column;
  by label;
  order=&i;
  ProgName=upcase("&sid"||" &tfl&i");
run;
%if &nshell eq 0 then %let dmy=1;
%else if &nshell gt 0 then %let dmy=&nshell;
  if &i le &dmy then do;
    data cdft;set _NULL_; run;
  end;
9

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 trademarks of their respective companies.
Maximum Use of TFL Mock Shells, continued

data cdft; set xx cdft; run;%end;
%let i=%eval(&i+1); %end;
proc sort data=cdft; by order;run;
ods listing close;
ods TAGSETS.EXCELXP file = "&path\TFDH.xls" style=htmlblue options(autofit_height = 'yes');<Ods to Multiple Excel spreadsheets>>
%mend tfdh;

APPENDIX 2

Resulting Excel output after macro execution

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
<th>C</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>progrname</td>
<td>label</td>
<td>order</td>
</tr>
<tr>
<td>2</td>
<td>table1_sm.ds</td>
<td>Summary of Patient Disposition</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>table2_sm.dm</td>
<td>Summary of Demographic Characteristics</td>
<td>2</td>
</tr>
<tr>
<td>4</td>
<td>table3_sm.dm1</td>
<td>Summary of Demographic Characteristics</td>
<td>3</td>
</tr>
<tr>
<td>5</td>
<td>table4_sm.rae</td>
<td>Summary of Treatment Related Adverse Events by Preferred Term</td>
<td>4</td>
</tr>
<tr>
<td>6</td>
<td>table5_sm.rae2</td>
<td>Summary of Treatment Related Adverse Events by Preferred term</td>
<td>5</td>
</tr>
<tr>
<td>7</td>
<td>table6_sm.orr</td>
<td>Summary of Best Overall Response -RECIST</td>
<td>6</td>
</tr>
<tr>
<td>8</td>
<td>figure1_fg.pfs1</td>
<td>Summary of Progression-free Survival (TIFF file)</td>
<td>7</td>
</tr>
<tr>
<td>9</td>
<td>figure2_fg.pfs2</td>
<td>Summary of Progression-free Survival (TIFF file)</td>
<td>8</td>
</tr>
<tr>
<td>10</td>
<td>figure3_fg.wfp</td>
<td>Waterfall Plot of Best Percentage Change from Baseline in Sum of Target Lesions</td>
<td>9</td>
</tr>
<tr>
<td>11</td>
<td>table7_ssm.fqaeh</td>
<td>Summary of Shifts from Baseline to Maximum Post Baseline CTC AE Grade - Hematology</td>
<td>10</td>
</tr>
<tr>
<td>12</td>
<td>table8_ssm.staeh</td>
<td>Summary of Change from Baseline in Laboratory Results - Hematology</td>
<td>11</td>
</tr>
<tr>
<td>13</td>
<td>listing1_ls.dm</td>
<td>Listing 1 - Listing of Patient Demographics and Baseline Characteristics</td>
<td>12</td>
</tr>
<tr>
<td>14</td>
<td>listing2_ls.dh</td>
<td>Listing 2 - Listing of Disease History</td>
<td>13</td>
</tr>
</tbody>
</table>

APPENDIX 3

Using/validating the Metadata

Table 1

<table>
<thead>
<tr>
<th>Disposition</th>
<th>Cohort 1 (n=starm)</th>
<th>Cohort 2 (n=starm)</th>
<th>Cohort 3 (n=starm)</th>
<th>Cohort 4 (n=starm)</th>
<th>Total (n=starm)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Note: Safety Analysis set is defined as the number of patients who received at least 1 dose of protocol treatment.
Note: Efficiency Analysis set is defined as the number of patients who received at least one dose of protocol treatment at the MTD.

Program Name: Table1_SM_DS
Dataset Name: dataset (today)
Generation Date/Time: 2017-02-23/11:46

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 trademarks of their respective companies.
Table 5

<table>
<thead>
<tr>
<th>MedDRA Preferred term</th>
<th>Maximum CTCAE Grade</th>
<th>Total (E=events)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Grade 1</td>
<td>Grade 2</td>
</tr>
</tbody>
</table>

No available data to support this summary

[a] Number (%) of patients with related AEs.
Note: Patients with multiple unique events are counted once per maximum reported grade and sorted by decreasing overall total events by event.

Program Name: Table5 ООL.RAS2
Database(s) (date(s)): edbabac (today)
Generation Date/Time: 2017-02-23/12:08