Creating Customized Patient Profiles using SAS ODS RTF and PROC TEMPLATE

Andrea Ritter, Biostatistics, Quintiles Inc., Morrisville, NC

ABSTRACT

Patient profiles (a.k.a. case report form tabulations) involve many disparate pieces of information about a single subject, displayed in various formats and all contained in a single word document. This article presents an example for producing patient profiles using SAS® PROC TEMPLATE table definitions and the Output Delivery System (ODS) with the DATA step.

INTRODUCTION

Often times, programmers are asked to create a patient profile listing (previously called case report form tabulations) for an FDA submission. These profiles provide regulatory reviewers the ability to view all reported data available for each patient in a study. This information is usually stored in multiple datasets or formats. Some datasets may contain only one record per patient while others may contain multiple records per patient, with some patients having more records than others. To further complicate matters, standards for the appearance of tables and listings are set increasingly higher. Such data lend themselves to a multi-table presentation that can be easily achieved using ODS and PROC TEMPLATE in SAS. While ODS and PROC TEMPLATE are still relative newcomers, their power lies in their flexibility and versatility.

DESCRIPTION

In this example, the data are provided in a format similar to that of the Clinical Data Interchange Standards Consortium (CDISC) Study Data Tabulation Model (SDTM). For purposes of brevity, the profile template includes only four relatively simple tables. The first table includes patient-level information, including Study Number, Site Number, and Patient Number. The next table presents patient demographic information. The final two tables present markedly abnormal findings and are virtually identical in format, apart from displaying different types of findings (laboratory and vital signs).

Notice that the first two tables in the profile use a row label (the label for the data is displayed to the left of the data) as opposed to a column label (such as is used in the two findings tables). To accomplish this, we'll create separate columns for each label. Thus, each data point presented will be comprised of two columns. For example, "Study Number" will encompass a column for the label "Study Number" and a column for the actual study number as defined in the data.

FIGURE 1: DESIRED PATIENT PROFILE

```
Study Number: XXXXXXX Site Number: XXX   Patient Number: XXXXXXXXX

Age (yrs): XX      Gender: XXXXXX       Race/Ethnicity:XXXXXXX
Weight (lb): XXX.X   Height (in): XX.X

Markedly Abnormal Laboratory Values:
Date:        Parameter (Unit):                      Result:   Markedly Abnormal Criteria:
DDMMMYYYY    Standardized Test (Unit)               XX.XX

Markedly Abnormal Vital Signs:
Date:        Parameter (Unit):                      Result:   Markedly Abnormal Criteria:
DDMMMYYYY    Standardized Test (Unit)               XX.XX
```

METHOD

The output will be produced using SAS ODS RTF coding and table templates created in PROC TEMPLATE to simulate the provided example.

STEP 1: DEFINE THE STYLE TEMPLATE

The first step in this process is to define the style template. PROC TEMPLATE enables one to customize the appearance or style of the output, including font and margins.

The below code defines

- where the template will be stored (work.templat);
- what standard SAS style is used as the starting point, or parent (styles.rtf);
- the page margins (1 inch all around);
- the layout of tables (no frame around the outside, no grid rules inside, no spacing between cells, add a dollop of padding around the text within each cell);
- font styles (all based on an 8pt Courier New font;
- Titles and footers are defined as vertically top-justified, horizontally left-justified, and will use the font defined as "TitleFont2" (Courier New, 8pt) by default. Headers and Footers on the other hand, such as will be used for row and column headings in tables, will also be **bold**, as defined in the "HeadingFont".

```
ods path  work.templat(write) sasuser.templat(update) sashelp.tmplmst(read);
proc template;
define style Styles.Profile / store=work.templat(write);
   parent = styles.rtf;
   style Body from Body /
      leftmargin = 1.0in  rightmargin = 1.0in  topmargin = 1.0in
      bottommargin = 1.0in  rules=none  frame=void;
   style Table from Output /
      frame = void  rules = none  cellspacing = 0  cellpadding = 3pt;
   replace fonts from Fonts /
      'TitleFont2'          = ("Courier New",8pt)
      'TitleFont'           = ("Courier New",8pt)
      'StrongFont'          = ("Courier New",8pt,bold)
      'EmphasisFont'        = ("Courier New",8pt,italic)
      'FixedEmphasisFont'   = ("Courier New",8pt,italic)
      'FixedStrongFont'     = ("Courier New",8pt,bold)
      'FixedHeadingFont'    = ("Courier New",8pt,bold)
      'BatchFixedFont'      = ("Courier New",8pt)
      'FixedFont'           = ("Courier New",8pt)
      'HeadingEmphasisFont' = ("Courier New",8pt,italic)
      'HeadingFont'         = ("Courier New",8pt,bold)
      'DocFont'             = ("Courier New",8pt);
   replace TitlesAndFooters from TitlesAndFooters/
      font=Fonts('TitleFont2')
      vjust=T just=L  cellspacing=0  cellpadding=3pt;
   style SystemTitle from TitlesAndFooters / protectspecialchars=off;
   style SystemFooter from TitlesAndFooters / protectspecialchars=on;
   replace Cell from Cell / font=Fonts('DocFont')  protectspecialchars=off;
   replace HeadersAndFooters from HeadersAndFooters /
      font=Fonts('HeadingFont')  protectspecialchars=off  background=white;
   style RowHeader from HeadersAndFooters;
   style ColumnHeader from HeadersAndFooters;
   style Data from Cell;
end;
run;
```

STEP 2: DEFINE THE TABLE TEMPLATE

The next step defines three table templates. The first template (Reports.PATIENT) defines the patient-level table (first row in FIGURE 1). The second template (Reports.DEMO) defines the demographics table with 2 rows and 10 columns (5 labels and 5 variables). The third template may be used for any findings data. We will use it for both laboratory results and vital signs. Using proc template, this generic table (Reports.FIND) is defined to have four columns with an optional, dynamically-defined header. Each column relates to a specific variable in the data and will have a column heading predefined in the template, unlike the demographics table, which presents a separate column for the label to the left of each variable. The code for these three templates is shown below.

```
** Define Patient-Level Table;
** This table has 6 columns, 3 columns of labels (mixed case variables) and
** 3 columns of text (upper case variables);
define table Reports.PATIENT / store=Profile.templat;
  column StudyNumber STUDYID SiteNumber SITEID PatientNumber SUBJID;

  define column StudyNumber;
    ** Column will always take the value defined in the "compute as" statement;
    compute as 'Study Number: ';
    print_headers=OFF; **do not print a column header;
    just=left;
    ** use the RowHeader style defined above in the style template;
    style=RowHeader{CellWidth=1in};
  end;
  define column STUDYID;
    ** Column is not computed here and is not "generic", so when called, the
    ** template will expect a variable named STUDYID;
    print_headers=OFF;
    just=center;
    style=Cell {CellWidth=0.75in}; ** use the Cell style defined above.
  end;
  define column SiteNumber;
    compute as 'Site Number: ';
    print_headers=OFF;
    just=right;
    style=RowHeader{CellWidth=1.25in};
  end;
  define column SITEID;
    print_headers=OFF;
    just=center;
    style=Cell{CellWidth=0.75in};
  end;
  define column PatientNumber;
    compute as 'Patient Number: ';
    print_headers=OFF;
    just=right;
    style=RowHeader{CellWidth=1.25in};
  end;
  define column SUBJID;
    print_headers=OFF;
    just=center;
    style=Cell{CellWidth=1in};
  end;

  newpage=off;
  center=off;
  balance=off;
end;

** Define Demographics;
** This table is similar to the patient-level table - labels are to the left of
** the data, so labels will be defined in their own columns;
** Table has 10 columns, 5 columns of labels (mixed case variables) and
** 5 columns of text (upper case variables);
define table Reports.DEMO / store=Profile.templat;
```

```
   column AgeLab AGEC Gender SEXC RaceLab RACEETHN WgtLab WEIGHT HgtLab HEIGHT;

   define column AgeLab;
      compute as 'Age (yrs): ';
      print_headers=OFF;
      just=left;
      style=RowHeader{CellWidth=0.75in};
   end;
   define column AGEC;
      print_headers=OFF;
      just=left;
      style=Cell{CellWidth=0.5in};
   end;
   define column Gender;
      compute as 'Gender: ';
      print_headers=OFF;
      just=right;
      style=RowHeader{CellWidth=0.75in};
   end;
   define column SEXC;
      print_headers=OFF;
      just=left;
      style=Cell{CellWidth=0.75in};
   end;
   define column RaceLab;
      compute as 'Race/Ethnicity: ';
      print_headers=OFF;
      just=right;
      style=RowHeader{CellWidth=1.25in};
   end;
   define column RACEETHN;
      print_headers=OFF;
      just=left;
      style=Cell{CellWidth=2in};
      glue=-1; **forces the table to wrap to a new row with next column;
   end;
   define column WgtLab;
      compute as 'Weight (kg): ';
      print_headers=OFF;
      just=left;
      style=RowHeader{CellWidth=0.95in};
   end;
   define column WEIGHT;
      print_headers=OFF;
      just=left;
      style=Cell{CellWidth=0.75in};
   end;
   define column HgtLab;
      compute as 'Height (cm): ';
      print_headers=OFF;
      just=right;
      style=RowHeader{CellWidth=1in};
   end;
   define column HEIGHT;
      print_headers=OFF;
      just=left;
      style=Cell{CellWidth=0.75in};
   end;

   newpage=off;
   center=off;
   balance=off;
end;

** Define Findings Table - Markedly Abnormal Values (used for Labs and
** Vitals). This table has 4 columns and an optional header row. Text for
** header is defined dynamically through the variable _HEAD;
```

```
define table Reports.FIND / store=work.templat(write);
   dynamic _HEAD;
   column DATE PARAM RESULT MACRIT;

   ** optional header - define value of dynamic variable _HEAD in data step,
   ** when calling this report template;
   define header hdr1;
      text _HEAD;
      space=1;
      spill_margin;
      just=left;
      style=ColumnHeader;
   end;
   define column DATE;
      ** Column will have a label, or header, in the first row.
      ** Header is defined here and "print_headers" is turned on;
      header="Date:";
      print_headers=ON;
      just=left;
      style=Cell{CellWidth=1.25in};
      blank_dups=on;
      ** Column is defined as "generic" so any variable can be used to fill
      ** this column in the template. The variable will be named when the
      ** template is called in
```

```
step 3 below;
      generic;
   end;
   define column PARAM;
      header="Parameter (Unit):";
      print_headers=ON;
      just=left;
      style=Cell{CellWidth=2.25in};
      blank_dups=on;
      generic;
   end;
   define column RESULT;
      header="Result:";
      print_headers=ON;
      just=left;
      style=Cell{CellWidth=1in};
      generic;
   end;
   define column MACRIT;
      header="Markedly Abnormal Criteria:";
      print_headers=ON;
      just=left;
      style=Cell{CellWidth=2in};
      generic;
   end;

   newpage=off;
   center=off;
   balance=off;
end;
run;
```

STEP 3: PRODUCE THE PROFILE

The third step in this process is to produce the patient profile for a single patient. Once it is determined that the table templates are acceptable, the profile can be reproduced for multiple patients. For this purpose, a single profile is produced using a macro. The macro code is shown below. For ease of reference, code relating to the templates defined above is highlighted in **red**.

```
options orientation=portrait nodate nonumber missing=' ' nocenter
   papersize=letter leftmargin=1in rightmargin=1in topmargin=1in
   bottommargin=1in;

ods escapechar="~";

*** MACRO RUNTAB(x) will run a single patient profile for USUBJID=x ***;
%macro runtab(x);

   title;
   footnote;
   ods listing close;

   ** Output Unique Subject Identifiers into the macro variable [USUBJID];
   **(Used in defining patient-specific Filename below);
   proc sql noprint;
   select compress(USUBJID) into :USUBJID from sdtm.DM where USUBJID="&x";
   quit;

   filename _rtf_
      "C:\Sponsor\Protocol\Profiles\Output\%sysfunc(compress(&USUBJID)).doc";

   ods rtf file=_rtf_ newfile=none nokeepn startpage=no record_separator=none
     headery=770 footery=770 style=styles.Profile /*Defined in Step 1 above*/;

   ** J= means justify, L for left, C for center, R for right;
   ** ~ is the ODS escape character defined above;
   ** {THISPAGE} and {LASTPAGE} are RTF-specific codes for producing pagination
   ** fields in Word;
   title1    j=L "Patient Profile &USUBJID";
   footnote1 j=L "Source: [C:\Sponsor\Protocol\Profiles\Programs] Profiles.sas"
   footnote2 j=R "Page ~{thispage} of ~{lastpage}";

   ** Patient-Level Table **;
   ods rtf anchor='PatientInfo';
   data _NULL_;
      set sdtm.DM;
      where USUBJID="&x";

      ** These columns are not generic, so they are expecting variable names
      ** matching those defined in the template above;
      file print ods=(template='Reports.PATIENT'
           columns=( STUDYID SITEID SUBJID ));
      put _ods_;
   run;


   ** Demographics **;

   ** Grab Baseline Height and Baseline Weight from the VS SDTM **;
   proc transpose data=sdtm.VS out=HTWT;
      where VSTEST in ("HEIGHT","WEIGHT") and VSBLFL="Y";
      by STUDYID USUBJID;
      id VSTEST;
      var VSSTRESC;
   run;

   ods rtf anchor='Demographics';
```

```
data _NULL_;
   merge sdtm.DM HTWT;
      by STUDYID USUBJID;
   where USUBJID="&x";

   length AGEC SEXC RACEETHN $200;
   if not missing(AGE) then AGEC=put(AGE,2.);
   else AGEC='--';

   if SEX='F' then SEXC='Female';
   else if SEX='M' then SEXC='Male';
   else SEXC='Unknown';

   if RACE="OTHER" then RACEETHN=cats(RACEOTH,'/',ETHNIC);
   else if not missing(RACE) then RACEETHN=cats(RACE,'/',ETHNIC);
   else if missing(RACE) then RACEETHN="Race Unknown/"||ETHNIC;

   ** These columns are not generic, so they are expecting variable names
   ** matching those defined in the template above;
   file print ods=(template='Reports.DEMO'
        columns=( AGEC SEXC RACEETHN WEIGHT HEIGHT ));
   put _ods_;
run;

** Markedly Abnormal Laboratory Values **;
ods rtf anchor='Labs';
data _NULL_;
   set sdtm.LB;
   where USUBJID="&x" and LBNRIND in ('ABNORMAL' 'HIGH' 'LOW');

   ** Combine LBTEST and LBSTRESU as follows: Test Name (Units);
   length LBTESTU $200;
   if not missing(LBSTRESU)
      then LBTESTU=strip(LBTEST) || ' (' || strip(LBSTRESU) || ')';
      else LBTESTU=strip(LBTEST);

   ** These columns are defined as generic in the template and so can
   ** be set to take whatever variables are required. Notice how this
   ** template call differs from the Patient-Level and Demographics template
   ** calls;
   file print ods=(template='Reports.FIND'
        dynamic=( _HEAD="Markedly Abnormal Laboratory Values:" )
        columns=( DATE = LBDTC     (generic=on)
                 PARAM = LBTESTU   (generic=on)
                RESULT = LBSTRESC  (generic=on)
                MACRIT = LBNRIND   (generic=on) ));
   put _ods_;
run;

** Markedly Abnormal Vital Signs **;
** (Similar to Labs above, but using data from sdtm.VS);
ods rtf anchor='Vitals';
data _NULL_;
   set sdtm.VS;
   where USUBJID="&x" and VSNRIND in ('HIGH' 'LOW');

   length VSTESTU $200;
   if not missing(VSSTRESU)
      then VSTESTU=strip(VSTEST) || ' (' || strip(VSSTRESU) || ')';
      else VSTESTU=strip(VSTEST);

   file print ods=(template='Reports.FIND'
        dynamic=( _HEAD="Markedly Abnormal Vital Signs:" )
        columns=( DATE = VSDTC     (generic=on)
                 PARAM = VSTESTU   (generic=on)
                RESULT = VSSTRESC  (generic=on)
                MACRIT = VSNRIND   (generic=on) ));
      put _ods_;
   run;


   ods rtf close;
   ods listing;
%mend runtab;
%runtab(XYZ-101-1001);
```

RESULTS

The resulting profile document is shown below in **FIGURE 2**. Notice how, in the “Markedly Abnormal Laboratory Values” table, the parameter value of “GGT (U/L)” is presented on the first row, but not on the following two rows. This is because the report template for that column has **`blank_dups=on`**. A similar result occurred in the date column of the “Markedly Abnormal Vital Signs” table. In order to display these duplicated values, simply update the template code to set **`blank_dups=off`**.

FIGURE 2: PATIENT PROFILE

```
Study Number:      XYZ          Site Number:     101        Patient Number:      1001

Age (yrs):  52        Gender: Female        Race/Ethnicity: WHITE/NON HISPANIC

Weight (kg):  52.2         Height (cm): 1.65


Markedly Abnormal Laboratory Values:
Date:              Parameter (Unit):                 Result:        Markedly Abnormal Criteria:
27DEC2009          GGT (U/L)                         321            >= 90 U/L
04JAN2010                                            275            >= 90 U/L
17JAN2010                                            240            >= 90 U/L
18FEB2010          HAEMATOCRIT (L/L)                 0.29           <= 0.32 L/L


Markedly Abnormal Vital Signs:
Date:              Parameter (Unit):                 Result:        Markedly Abnormal Criteria:
04JAN2010          SYSTOLIC BLOOD PRESSURE (mmHg)    190            >=180 AND Increase from
                                                                    Baseline >=20
                   DIASTOLIC BLOOD PRESSURE (mmHg)   110            >=105 AND Increase from
                                                                    Baseline >=15
```

CONCLUSIONS

The above method is provided as a simplistic example of how SAS ODS and PROC TEMPLATE may be used to produce one of the more complicated reports requested in clinical trials. There are benefits to using this approach compared to the traditional PROC REPORT; for instance, creating a table with row labels instead of column labels, such as the demographics table in the example. SAS continues to improve and enhance the features available in PROC TEMPLATE all the time and RTF output may be further enhanced through the use of features available in Microsoft® Office and the use of the RTF language.

REFERENCES

Barbara B. Okerson. 2009. Pleasing the Client: Creating Custom Reports with SAS® ODS LAYOUT and Proc REPORT. SESUG Proceedings. Burmingham, Alabama.

Clinical Data Interchange Standards Consortium. 2010. Study Data Tabulation Model, version 3.1.2 http://www.cdisc.org/

Cynthia L. Zender. 2010. SAS® Style Templates: Always in Fashion. SAS Global Forum Proceedings. Seattle, Washington.

David Shannon. 2002. SUGI 27: To ODS RTF and Beyond. SUGI 27 Proceedings Orlando, Florida.

Michele Ensor, Cynthia L. Zender. 2006. Advanced Output Delivery System (ODS) Topics, Course Notes for Live Web training on 5/22/08 – 5/25/08. Cary, NC: SAS® Institute Inc.

Microsoft Corporation. 1999. Rich Text Format (RTF) Specification, version 1.6. http://msdn.microsoft.com/en-us/library/aa140277.aspx

SAS® Institute Inc. 2010. SAS® Online Product Documentation, version 9.2. http://support.sas.com/documentation/onlinedoc/base/index.html#base92

ACKNOWLEDGEMENTS

I would like to thank my manager, Kim Sandle, for her constant encouragement and support; Sonia Davis and Steve Wright for their help and advice on the paper submission process; Tommy Retzlaff, Jan Pribac and the members of Quintiles Statistical Programming/Biostatistics team for their suggestions and assistance in the development of this paper; the professional support and advice of the SAS support team.

CONTACT INFORMATION

Your comments and questions are valued and encouraged. For more information contact:

Andrea Ritter, M.S.
Senior Statistical Programmer
Statistical Programming, Biostatistics
Quintiles, Inc.
5927 South Miami Boulevard
Morrisville, NC 27560
Office: 919-998-2263
Fax: 919-998-2382
Email: andrea.ritter@quintiles.com