Paper DV05-2013

Using ODS PDF, Style Templates, Inline Styles, and PROC REPORT with SAS® Macro Programs

Patrick Thornton, SRI International, Menlo Park, CA

ABSTRACT

A production system of SAS macro programs is described that modularize the generation of syntax to produce client-quality reports of descriptive and inferential results in a PDF document. The reusable system of macros include programs that save all current titles, footnotes, and option settings, establish standard titles, footnotes and option settings, and initially create the PDF document. Macro programs may be called to generate PROC REPORT syntax to produce various tables of descriptive and inferential results with table numbers, footnotes and customized bookmarks. A modified style template is use to determine the look of the whole document, and a macro program of inline style definitions is used to define the defaults for each type of table. A version of the inline style macro program may be customized to accommodate the needs of each project, and inline style parameters may also be modified for any given macro call. A macro program to end the PDF creates a standard data documentation page, and restores all original titles, footnotes and option settings. This paper is designed for the intermediate to advanced SAS programmer using Foundation SAS Software on a Windows operating system.

INTRODUCTION

This paper documents some of the techniques and concepts I have found important in creating reusable SAS macro programs used for the output of results. Over the course of several SAS projects, I have produce tens of thousands of tabular outputs having a very similar set of common results (Table 1). These results may be generated through a set of core procedures (column 2). Rather than manually creating the syntax for the procedures over and over again, I began to create macro programs that allowed me to regenerate the syntax for different data sets and variable names. Developing knowledge of the tools listed in column 3 allowed me to increase the viability of the macro programs for use across programmers and projects.

Table 1 Results, Procedures and Presentation Tools

Common Results	Core Procedures	Primary Presentation Tools
N and percent	PROC REPORT	Macro Programs
Mean, minimum, maximum, and standard deviation	PROC MEANS	PROC TEMPLATE, DEFINE STYLE
Independent sample t-test	PROC FREQ	System Options
Cross tabulation Chi Square test	PROC TTEST	ODS PDF
Paired sample t-test		ODS OUTPUT
2x2 cross tabulation McNemars test		PROC REPORT

Appendix 1 and 3 show examples of output generated through the system of macro programs using the tools in Table 1. This paper will not present all the syntax necessary to produce these outputs. The system of macro programs is much too long and complicated to cover in 20 pages. As an example, Appendix 2 does show just the arguments and local variables that are defined in the macro program used to produce all the tables in Appendix 1. There are many other supporting macro programs. While the paper cannot present the whole system, it does describe fundamental techniques used with all the primary tools listed in Table 1, lists all the tasks that the application accomplishes, and shows and describes some of the key syntax and macro programs that are used to accomplish the key tasks.

FUNDAMENTALS

This section presents some necessary fundamentals of creating and sharing resources, such as user-defined formats, macro programs and style templates. The section also shows examples of creating and configuring ODS PDF destinations, producing and showing results with PROC REPORT and defining and using inline styles. I will present only what is needed to understand the syntax that is generated by the macro programs. I recommend various resources for learning more about these topics.

CREATING AND SHARING USER-DEFINED FORMATS

PROC REPORT is the primary tool I use for the presentation of results. The following shows a basic syntax for generating the n and percent of observations using PROC REPORT. It also creates a permanent format called TEEN that is used by the second PROC REPORT.

```
❶%let shareit = \\128.......\sharedserverlocation\;
libname shareit "&shareit";
proc format lib= shareit.formats;
   value teen 11-12 = 'Preteen' 13-16 = 'Teen';
run;
❷options fmtsearch=(shareit.formats work library);
proc report data=sashelp.class nowd;
  col sex n pctn;
  define sex/group;
  define pctn/format=percent7.;
  rbreak after/summarize;
run;
proc report data=sashelp.class nowd;
  col age n pctn;
  define age/group format=teen.;
  define pctn/format=percent7.;
  rbreak after/summarize;
run;
```

❶ LIBREF SHAREIT references a folder on our server that all programmers may access. The format TEEN is saved to the catalog SHAREIT.FORMATS.

❷The FMTSEARCH= option directs SAS to first look in the SHAREIT.FORMATS catalog where it will find the format TEEN.

The format is shared by saving it permanently to a shared server location where it can be references with FMSTSEARCH= by programs and programmers running other SAS Sessions. Macro programs and style templates may also both be shared in a similar way.

CREATING AND SHARING MACRO PROGRAMS

The REPORT procedures calculate the n and percent of observations in SASHELP.CLASS within the classifications the variables SEX and AGE (formatted). There are about 132 characters, counting spaces, in each REPORT procedure. The only difference between the two procedures are the names of the variables, SEX and AGE, as well as the FORMAT=TEEN option found in the second procedure. The vast majority of syntax is repeated. If you needed to use the syntax with just 100 different variables, you would have over 13,000 characters and 600 lines of syntax to maintain and update. A much more efficient alternative is to create a macro program that contains the syntax while allowing you to change the data set, variable name, and format option. The following macro program MYPROCREPORT is a simple example.

```
%macro myprocreport(❶lib, ❷data, ❸rowvariable, ❹rowvariablef=);
proc report data=&lib..&data nowd;
  col &rowvariable n pctn;
  define &rowvariable/group &rowvariablef;
  define pctn/format=percent7.;
  rbreak after/summarize;
run;
%mend myprocreport;
%myprocreport(sashelp,class,sex,rowvariablef=)
%myprocreport(sashelp,class,age,rowvariablef=%str(format=teen.));
```

The parameters to the macro program include:

❶ LIB which needs to be the name of a LIBREF

❷ DATA which should be the name of a data set found in the LIBREF specified by LIB

❸ ROWVARIABLE the name of a variable in the data set specified by DATA

❹ The optional parameter ROWVARIABLEF needs to be the format specification for variable specified by ROWVARIABLE

By default the MYPROREPORT macro program was saved in a catalog called WORK.SASMACR and it was automatically available to my session. The following adds the necessary syntax to permanently save the macro

program in the catalog called SHAREIT.SASMACR.

```
%let shareit = \\128.......\sharedserverlocation\;
libname shareit "&shareit";
options ❶ mstored ❷ sasmstore=shareit mcompilenote=all;
%macro myprocreport(lib, data, rowvariable,rowvariablef=) ❸/store;
proc report data=&lib..&data nowd;
  col &rowvariable n pctn;
  define &rowvariable/group &rowvariablef;
  define pctn/format=percent7.;
  rbreak after/summarize;
run;
%mend myprocreport;
```

❶ MSTORED tells SAS to use the LIBREF and catalog specified by SASMSTORE to save a compiled macro program

❷ SASMSTORE= specifies SHAREIT.SASMACR as the catalog to store compiled macro programs. SASMACR is the default catalog if none is specified

❸ /STORE is added to the macro program to tell SAS to compile it. This will cause the macro program to be saved to SHAREIT.SASMACR.

Once the macro program is saved, then the following syntax may be used to make the macro program available to a new SAS session.

```
%let shareit = \\128.......\sharedserverlocation\;
libname shareit "&shareit" ❶access=readonly;
options mstored sasmstore=shareit;
```

The LIBREF SHAREIT is created with a path to the shared folder. It is defined as a ❶ READONLY LIBREF which allows multiple sessions and users to access the compiled macro programs found in the catalog SHAREIT.SASMACR.

DEFINING A PDF DESTINATION

This presentation uses the ODS PDF destination, and the following shows a basic example of sending the output generated from PROC REPORT to a PDF document.

```
❶ods pdf ❷file="&path_export.example1.pdf" ❸columns=2 ❹style=printer;
%myprocreport(sashelp,class,sex,format=)
%myprocreport(sashelp,class,age,format=%str(format=teen.));
❺ods pdf close;
```

❶ The ODS PDF statement opens the destination for output

❷ FILE= dictates the folder and name of the PDF. PATH_EXPORT is a macro variable resolving to a folder on my operating system.

❸ COLUMNS= determines the number of columns in which to display output

❹ STYLE= determines the style template to use in creating the PDF

❺ ODS PDF statement with the CLOSE option stops additional output from going to the PDF, and effectively creates the PDF file

STYLE= is an incredibly power option of the ODS PDF statement because the style may change the overall look of the PDF including many of the feature of all the output generated by many different procedures.

CREATING AND SHARING STYLE TEMPLATES

Style templates play an important role in the presentation of results. They allow you to change the style for the entire destination, for example, setting the lines and line thickness for all tables created in your output, or shading a column in all the tables. SAS provides many style templates. The following syntax will list the styles in the output window that are available from the item store SASUSER.TEMPLAT and SASHELP.TEMPLMST.

```
proc template;
   list styles;
run;
```

PROC TEMPLATE also provides a way to create your own custom style templates. The following example makes a small change to the PRINTER style template and saves it to SHARIT.TEMPLAT as the style MYDEFAULT.

```
%let shareit = \\128.......\sharedserverlocation\;
Libname shareit "&shareit";
ods path ❶shareit.templat(update) sashelp.tmplmst(read);
proc template;
define style mydefault; parent= styles.printer;
❷replace Table from Output /
frame = void
rules = rows
;
end;
run;
ods path ❸sasuser.templat(update) sashelp.tmplmst(read);
```

❶ The ODS PATH option SHAREIT.TEMPLAT(UPDATE) allows the new style template MYDEFAULT to be save to SHAREIT

❷ REPLACE TABLE FROM OUTPUT/ allows syntax occurring after to override specific style attributes of the TABLE element that are found in the PRINTER style, and specifically I chose to change FRAME and RULES

❸ODS PATH SASUSER.TEMPLAT(update) sets the path for updating back to the default location (you may also use ODS PATH RESET to accomplish the same thing)

USING SHARED USER DEFINED FORMAT, MACRO PROGRAMS, AND STYLE TEMPLES

Once user-defined formats, macro programs, and style templates are saved to a shared location they may all easily be referenced and used together.

```
libname shareit "&path_sharedfolder"  access=readonly;
options mstored sasmstore=shareit fmtsearch=(shareit work library) mprint;
ods path shareit.templat(read) sashelp.tmplmst(read);
ods pdf file="&path_export.example1.pdf" columns=2 style=mydefault;
%myprocreport(sashelp,class,sex,format=)
%myprocreport(sashelp,class,age,format=%str(format=teen.));
ods pdf close;;
ods path reset;
```

USING INLINE STYLES

Inline styles may be used to make a great number of changes to PROC REPORT output generated to ODS PDF, and various other destinations. The following is a simple and very valuable inline style that is used to make constant the widths and justify the content of the columns in PROC REPORT.

```
%macro myprocreport2(lib, data, rowvariable,rowvariablef=);
proc report data=&lib..&data nowd;
  col &rowvariable n pctn;
  define &rowvariable/group &format ❶style(column)=[cellwidth=100 just=left];
  define n /style(column)=[cellwidth=100 just=center];
  define pctn/format=shux. style(column)=[cellwidth=100 just=right];
  rbreak after/summarize;
run;
%mend myprocreport2;
```

❶ The keyword STYLE is used with the modifier (COLUMN) followed by square brackets that enclose the various attributes. The attribute CELLWIDTH is used to determine the width of the column in points and JUST is used to set the justification of the values in the column.

If we follow the strategy used to make the macro flexible then then macro program could include parameters for each style:

```
%macro myprocreport3(lib, data, rowvariable,rowvariablef=,
❶rowvarcolumnstyle=%str(style(column)=[cellwidth=100 just=left]),
ncolumnstyle=%str(style(column)=[cellwidth=100 just=center]),
percentcolumnstyle=%str(style(column)=[cellwidth=100 just=right])
);
proc report data=&lib..&data nowd;
  col &rowvariable n pctn;
  define &rowvariable/group &rowvariablef &rowvarcolumnstyle;
  define n /&ncolumnstyle;
  define pctn/format=shux. &percentcolumnstyle;
  rbreak after/summarize;
run;
%mend myprocreport3;
```

❶ Optional parameters were added to the macro program for each inline style, and default values were set for each which allows the call to the macro program to be short if the default styles are used, but requires that the call is much longer if different inline styles are needed.

This section introduced fundamental techniques. The rest of the paper describes key syntax and macro programs that enable the system of the macro programs I created to generate PDF output. There are 4 capabilities that I developed much further to enable the system of macro programs and sections that follow provide more detail:

- Control the overall look of the output with Style Template(s)
- Make standard the page set up and document using System OPTIONS and ODS PDF
- Refine the content and layout of the PROC REPORT table(s)
- Refine the format of the PROC REPORT table(s) with Inline Styles

STYLE TEMPLATE

I used the following style template for the examples shown in the Appendix.

```
ods path shareit.templat(update) sashelp.tmplmst(read);
proc template;
define style example1; parent= styles.printer;
replace fonts /
'TitleFont' = ("Helvetica",8pt,Bold)
'headingFont' = ("Helvetica",7pt,Bold)
'docFont' = ("Helvetica",7pt)
'footFont' = ("Times Roman",7pt)
'FixedStrongFont' = ("Courier",8pt,Bold)
'FixedHeadingFont' = ("Courier",8pt,Bold)
'BatchFixedFont' = ("Courier",5.7pt)
'FixedFont' = ("Courier",8pt);
replace color_list /
'bgH' = white /* row and column header background */
'bgT' = white /* table background */
'bgD' = white /* data cell background */
'fg' = black /* text color */
'bg' = white; /* page background color */
replace Table from Output /
frame = void
rules = rows
cellpadding = 2pt
cellspacing = 0.0pt
borderwidth = 0.2pt
background = color_list('bgT');
end;
run;
ods path reset;
```

MACRO PROGRAMS FOR OPENING ODS PDF DESTINATION

There are many possible ways to name, title, footnote, and specify a PDF file, so I tried to create a standard for myself as well as allowing the macro programs to be flexible. I created a macro program called STARTPDF that is responsible for orchestrating the following tasks:

- Create a LIBREF _DELTA_, if it does not exist, to store temporary data sets
- Save all original titles, footnotes, and system options settings to data sets in _DELTA_
- Set system options applicable for the PDF
- Open PDF destination and set initial ODS PDF options, including file name and document style
- Create standard footnotes

The STARTPDF macro program is shown on the next page and Figure 1 shows a partial log of the program.

Figure 1 Partial Log Creating _DELTA_ and Saving Titles, Footnotes and Option Settings

```
292  ods path shareit.templat(read) sashelp.tmplmst(read);
293  %let mytitle =Using STARTPDF and ENDPDF Other Options;
294  %startpdf(path=&path_export,name=&mytitle,odspdfoptions=%str(style=mydefault columns=1));
MPRINT(TEMPLIBRARY):   options noxwait;
MPRINT(TEMPLIBRARY):   x "md ""C:\DOCUME~1\PTHORN~1\LOCALS~1\Temp\SAS Temporary
Files\_TD4520\_4_""";
MPRINT(TEMPLIBRARY):   libname _delta_ "C:\DOCUME~1\PTHORN~1\LOCALS~1\Temp\SAS Temporary
Files\_TD4520\_4_";
NOTE: Libref _DELTA_ was successfully assigned as follows:
      Engine:        V9
      Physical Name: C:\DOCUME~1\PTHORN~1\LOCALS~1\Temp\SAS Temporary Files\_TD4520\_4_
MPRINT(SAVETITLES2):  ;
MPRINT(SAVETITLES2):   proc sql;
MPRINT(SAVETITLES2):   create table _delta_.m_titles as select * from dictionary.titles;
NOTE: Table _DELTA_.M_TITLES created, with 1 rows and 3 columns.

MPRINT(SAVETITLES2):   quit;
NOTE: PROCEDURE SQL used (Total process time):
      real time           0.01 seconds
      cpu time            0.01 seconds


MPRINT(SAVETITLES2):   title;
MPRINT(SAVETITLES2):   footnote;
MPRINT(STARTPDF):  ;
MPRINT(SAVEOPTIONS):  ;
MPRINT(SAVEOPTIONS):   proc sort data=sashelp.voption out=_delta_.m_ooption;
MPRINT(SAVEOPTIONS):   by optname;
MPRINT(SAVEOPTIONS):   run;

NOTE: There were 394 observations read from the data set SASHELP.VOPTION.
NOTE: The data set _DELTA_.M_OOPTION has 394 observations and 6 variables.
NOTE: PROCEDURE SORT used (Total process time):
      real time           0.01 seconds
      cpu time            0.01 seconds


MPRINT(SAVEOPTIONS):   data _delta_.m_ooption;
MPRINT(SAVEOPTIONS):   set _delta_.m_ooption;
MPRINT(SAVEOPTIONS):   length keyword $1060.;
MPRINT(SAVEOPTIONS):   keyword = getoption(optname,'keyword');
MPRINT(SAVEOPTIONS):   run;

NOTE: There were 394 observations read from the data set _DELTA_.M_OOPTION.
NOTE: The data set _DELTA_.M_OOPTION has 394 observations and 7 variables.
NOTE: DATA statement used (Total process time):
```

The complete STARTPDF macro program is shown here.

```
%macro startpdf (
path=,
name=,
cleartitles=y,
clearfootnotes=y,
systemoptions= %str(options orientation=portrait nocenter),
odspdfoptions=
);
    %local startpdf_programmer startpdf_programname;
    %let startpdf_programmer =;
    %let startpdf_programname =;

    /*saves the current titles */
    ❶ %savetitles2(cleartitles=&cleartitles,clearfootnotes=&clearfootnotes);

    /*saves the current options */
    ❷ %saveoptions;

    /*sets default options */
    &systemoptions;

    %createfootnote(clearfootnotes=&clearfootnotes,timedate=%nrquote(%currenttimedat
e));

    /*Escape character*/
    ODS escapechar='^';

    /*if global macro variable programmer does not exist then creat my own*/
    %if %symexist(programmer) = 0 %then %let startpdf_programmer =&sysuserid;
    %else %let startpdf_programmer =&programmer;

    /*if global macro variable programmer does not exist then create my own*/
    %if %symexist(programname) = 0 %then %let startpdf_programname =unknown;
    %else %let startpdf_programname =&programname;

    %if "&name" = "" %then %let name =mypdf.pdf;
    %else %let name =&name..pdf;

    filename mypdf "%pathvalidation(&path)&name";
    ods pdf file=mypdf %createodspdfstatement(odspdfoptions=&odspdfoptions);
    data _delta_.log;
        set sashelp.VEXTFL (where=(scan(xpath,-1,"\") ="&name"));
           length programmer programname file path $200.;
        programmer = "&startpdf_programmer";
           programname = "&startpdf_programname";
           file = "&name";
           path = substr(xpath,1,index(xpath,"&name")-1);
           bogus =1;
    run;
%mend;
```

Figure 1 shows that STARTPDF calls macro programs ❶ SAVETITLES2, and ❷ SAVEOPTIONS. The macro program TEMPLIBRARY is called by both SAVETITLES2 and SAVEOPTIONS.

The first task of creating a temporary LIBREF is possibly one you did not expect; however, STARTPDF needs to coordinate the creation of data sets to store the original titles, footnotes and option settings without risk of writing over an existing data set. Dilorio (2010) created and presented the following macro program as a robust solution to this problem. If the LIBREF _DELTA_ does not exist, then a folder is created in the WORK folder, and _DELTA_ is created to reference that path. The following shows how I saved a version of the Dilorio macro program to my shared server location.

```
%let shareit = \\128.......\sharedserverlocation\;
Libname shareit "&shareit";
❶options mstored ❷sasmstore=shareit;
%macro templibrary❸/store;
  %local path;
  %if ❹%sysfunc(libref(_delta_)) ne 0 %then %do;
     ❺ %let path = %sysfunc(pathname(work))\_&sysindex._;
     ❻ options noxwait;
     ❼ x "md ""&path.""";
     ❽ libname _delta_ "&path";
  %end;
%mend;
```

❶ MSTORED tells SAS to use the library and catalog specified by SASMSTORE to save a compiled macro program

❷ SASSTORE= specifies SHAREIT.SASMACR as the catalog to store compiled macro programs, SASMACR is the default catalog if none is specified

❸ /STORE is added to the macro program to tell SAS to compile it. This will cause the macro program to be saved to SHAREIT.SASMACR

❹ If a LIBREF called _DELTA_ does not exist, then go on to create it

❺ Assign to PATH the full path to the WORK library plus a child folder with a unique index e.g. "_12_"

❻ Option NOXWAIT is set to allow SAS to create DOS processing without pausing for a user response

❼ Submit a DOS command to make the new directory resolved from PATH

❽ Creates a LIBREF called _DELTA_ that references the new folder resolved from PATH

The macro TEMPLIBRARY is used by any macro program that needs to create a temporary data set, thus it is called by SAVETITLES2.

```
%macro savetitles2(cleartitles=y,clearfootnotes=y) /store;
❶%templibrary;
❷proc sql;
    create table _delta_.m_titles as select * from dictionary.titles;
quit;
❸%if &cleartitles = y %then %do;
    title;
%end;
❹%if &clearfootnotes = y %then %do;
    footnote;
%end;
%mend savetitles2;
```

❶ _DELTA_ LIBREF is created if it does not exist

❷ Current titles and footnotes are saved to _DELTA_.M_TITLES from DICTIONARY.TITLES

❸ Current titles are removed because the default value of CLEARTITLES=y

❹ Current footnotes are removed because the default value of CLEARFOOTNOTES = y

Similar to the SAVETITLES2, the SAVEOPTIONS macro program saves the option settings to a data set in _DELTA_.

```
%macro saveoptions /store;
    ❶ %templibrary;
    ❷ proc sort data=sashelp.voption out=_delta_.m_ooption; by optname; run;
    ❸ data _delta_.m_ooption;
          set _delta_.m_ooption;
          length keyword $1060.;
          keyword = getoption(optname,'keyword');
    run;
%mend saveoptions;
```

❶ _DELTA_ LIBREF is created if it does not exist

❷ Current options are saved to _DELTA_.M_OPTION from DICTIONARY.OPTIONS via SASHELP.VOPTION view

❸ Using KEYWORD, the options are saved in a format that is needed for resetting the options

Figure 2 is the continuation of the log generated by STARTPDF.

Figure 2 Partial Log Opening ODS PDF and Saving File Information

```
MPRINT(STARTPDF):   options orientation=portrait nocenter;
MPRINT(STARTPDF):   ODS escapechar='^';
MPRINT(CREATEFOOTNOTE):   options nodate nonumber;
MPRINT(CREATEFOOTNOTE):   footnote1 height=8pt j=right f=arial "Page ^{thispage} of ^{lastpage}";
MPRINT(CREATEFOOTNOTE):   footnote2 height=8pt j=right f=arial "9:08 AM  Thursday, July 26, 2012";
MPRINT(STARTPDF):  ;
MPRINT(STARTPDF):   filename
MPRINT(STARTPDF):   mypdf "C:\PROJECTS\SASRESOURCES\MYSASPAPERS\WUSS\WUSS2012\SAS MACRO PROGRAMS PROC
REPORT\export\Using STARTPDF and ENDPDF Other Options.pdf";
MPRINT(STARTPDF):   ods pdf file=
MPRINT(CREATEODSPDFSTATEMENT):  mypdf style=mydefault columns=
MPRINT(STARTPDF):  1;
NOTE: Writing ODS PDF output to DISK destination "MYPDF", printer "PDF".
MPRINT(STARTPDF):   data _delta_.log;
MPRINT(STARTPDF):   set sashelp.VEXTFL (where=(scan(xpath,-1,"\") ="Using STARTPDF and ENDPDF Other
Options.pdf"));
MPRINT(STARTPDF):   length programmer programname file path $200.;
MPRINT(STARTPDF):   programmer = "pthornton";
MPRINT(STARTPDF):   programname = "ODSPDFSTART.SAS";
MPRINT(STARTPDF):   file = "Using STARTPDF and ENDPDF Other Options.pdf";
MPRINT(STARTPDF):   path = substr(xpath,1,index(xpath,"Using STARTPDF and ENDPDF Other Options.pdf")-1);
MPRINT(STARTPDF):   bogus =1;
MPRINT(STARTPDF):   run;

NOTE: There were 1 observations read from the data set SASHELP.VEXTFL.
      WHERE SCAN(xpath, -1, '\')='Using STARTPDF and ENDPDF Other Options.pdf';
NOTE: The data set _DELTA_.LOG has 1 observations and 14 variables.
NOTE: DATA statement used (Total process time):
      real time           0.12 seconds
      cpu time            0.00 seconds
```

System Options & Default Footnote

FILENAME and ODS PDF

Saving file metadata for later use

System options are set both through a parameter to the macro program STARTPDF and also in the CREATEFOOTNOTE macro program.

```
%macro createfootnote(clearfootnotes=,timedate=)/store;
    %if &clearfootnotes = y %then %do;
    ❶ options nodate nonumber;
    footnote1 height=8pt j=right f=arial "Page ❷ ^{thispage} of ^{lastpage}";
    footnote2 height=8pt j=right f=arial ❸ "&timedate";
    %end;
%mend createfootnote;
```

❶ OPTIONS NODATE NONUMBER removes the default date and page number from the upper right corner of the output.

❷ Escape character defined previously as '^' is used to get SAS to pass the special codes THISPAGE and LASTPAGE to PDF in order to obtain page numbering in the footnote (e.g. "Page 1 of 2")

❸ TIMEDATE resolves to a time and date as passed to the macro program

The call to CREATEFOOTNOTE in STARTPDF is the following:

```
%createfootnote(clearfootnotes=&clearfootnotes,timedate=%nrquote(%currenttimedate));
```

CLEARFOOTNOTES is a parameter in STARTPDF, so a default value of y causes the CREATFOOTNOTE macro to create the standard footnote. The value of the TIMEDATE parameter results from a call to a macro program called CURRENTTIMEDATE.

```
%macro currenttimedate(fmt=weekdate29,tm=timeampm8) /store;
  %local d t ;
  %let d = %sysfunc( date( ), &fmt. );
  %let t = %sysfunc( time( ), &tm. );
  &t  &d
%mend currenttimedate;
```

The log in Figure 2 also shows that a FILENAME MYPDF is created and is used in the FILE= option of the ODF PDF statement. The use of FILENAME allows me to obtain the path to my PDF file from the metadata even when the path is not supplied directly by the user. The metadata is also important because I chose to create a documentation page at the end of my PDF rather than having extended documentation in the footnote of every page. As a result, I need access the documentation in a macro program that creates the ODS PDF CLOSE statement. I am saving the metadata for the MYPDF FILENAME found in DICTONARY.EXTFILES via the view SASHELP.VEXTFL to the data set _DELTA_ .LOG.

To create FILENAME MYPDF in STARTPDF, a macro program is called in the statement:

```
filename mypdf "%pathvalidation(&path)&name";
```

If PATH resolves to a valid path then the PATHVALIDATION macro program returns the path, otherwise it returns no path. In Figure 2, the path provided by the user was valid, if it was not, or if PATH was left blank, then SAS will save the PDF file to a default folder, and that folder path will be obtained from metadata the same way it was obtained when specified by a user. The macro program PATHVALIDATION is shown here.

```
%macro pathvalidation(path) /store;
/*if path supplied*/
❶ %if &path ne %then %do;
     /*if path exists*/
    ❷%if %sysfunc(fileexist(&path)) = 1 %then %do;
           /*remove final slash if it exists*/
           ❸%if %substr(&path,%length(&path),1) ne \ %then %do;
                 %let path = &path.\;
           %end;
           ❹&path
    %end;
%end;
%mend pathvalidation;
```

❶ If PATH does not resolve to blank then proceed with validating the path

❷ If PATH resolves to a valid path then continue to prepare the path

❸ If resolved value of PATH does not end with a "\" then add it

❹ Return the resolved value of PATH

The ODS PDF statement is built through a call to the macro program CREATEODSPDFSTATEMENT.

```
ods pdf file=mypdf %createodspdfstatement(odspdfoptions=&odspdfoptions);
```

The macro program CREATEODSPDFSTATEMENT is shown here:

```
%macro createodspdfstatement(odspdfoptions=) /store;
❶%if "&odspdfoptions" = "" %then %do;
  style=minimal startpage=never columns=2 author="&sysuserid"
%end;
%else %do;
❷   &odspdfoptions
%end;
%mend createodspdfstatement;
```

❶ If ODS PDF options are not supplied by the user then set default options

❷ Set the user supplied options

MACRO PROGRAMS FOR CLOSING THE ODS PDF DESTINATION

ENDPDF is the macro program responsible for coordinating the following tasks that occur after all output has been written to the ODS PDF destination. The tasks are:

- Create a final page for the document with extensive footnotes for documentation
- Close ODS PDF destination
- Reset original titles, footnotes and system options

Figure 3 shows a partial log output from the macro program ENDPDF.

Figure 3 Partial Log of ENDPDF

```
1314  %endpdf;
MPRINT(ENDPDF):   data _null_;
MPRINT(ENDPDF):   set _delta_.log;
MPRINT(ENDPDF):   call symput('path',strip(path));
MPRINT(ENDPDF):   call symput('file',strip(file));
MPRINT(ENDPDF):   date =datepart(modate);
MPRINT(ENDPDF):   call symput('date',strip(put(date,weekdate29.)));
MPRINT(ENDPDF):   run;

NOTE: There were 1 observations read from the data set _DELTA_.LOG.
NOTE: DATA statement used (Total process time):
      real time           0.00 seconds
      cpu time            0.00 seconds


MPRINT(ENDPDF):   title4 "Documentation";
MPRINT(ENDPDF):   footnote1 "File: Using STARTPDF and ENDPDF Other Options.pdf";
MPRINT(ENDPDF):   footnote2 "Path: C:\PROJECTS\SASRESOURCES\MYSASPAPERS\WUSS\WUSS2012\SAS MACRO PROGRAMS
PROC REPORT\export\";
MPRINT(ENDPDF):   footnote3 "Date: Thursday, July 26, 2012";
MPRINT(ENDPDF):   proc report nowd data=_delta_.log;
MPRINT(ENDPDF):   col xpath;
MPRINT(ENDPDF):   define xpath/noprint;
MPRINT(ENDPDF):   run;

NOTE: There were 1 observations read from the data set _DELTA_.LOG.
NOTE: PROCEDURE REPORT used (Total process time):
      real time           0.00 seconds
      cpu time            0.00 seconds
```

The macro program ENDPDF is shown here.

```
%macro endpdf /store;
❶data _null_;
    set _delta_.log;
    call symput('path',strip(path));
    call symput('file',strip(file));
    date =datepart(modate);
    call symput('date',strip(put(date,weekdate29.)));
run;
title4 "Documentation";
❷footnote1 "File: &file";
footnote2 "Path: &path";
footnote3 "Date: &date";
❸ proc report nowd data=_delta_.log;
  col xpath;
  define xpath/❹noprint;
run;
❺ods pdf close;
❻filename  mypdf clear;
❼%returntitles;
❽%returnoptions;
proc delete data=_delta_.log; run;
%mend;
```

❶ The DATA step uses the data set _DELTA_.LOG created by STARTPDF to save the documentation as macro variables

❷ The macro variables are resolved in three footnotes

❸ PROC REPORT is used to produce the page without generating visible output

❹ NOPRINT as an option on the DEFINE statement of the only column variable, XPATH, prevents visible output

❺ The ODS PDF destination is closed

❻ FILENAME MYPDF is cleared

❼ Macro program RETURNTITLES is called to restore original session titles

❽ Macro program RETURNOPTIONS is called to restore original session option settings

```
%macro returntitles /store;
title; footnote;
data _null_;
      ❶ set _delta_.m_titles;
      ❷if type = "T" then
      call execute ("Title"||trim(left(number))||" '"||strip(text)||"';");
      else if type = "F" then
      call execute("Footnote"||trim(left(number))||" '"||strip(text)||"';");
run;
❸proc delete data=_delta_.m_titles; run;
%mend;
```

❶ M_TITLES, the data set created by the macro program SAVETITLES2, is used to obtain the original titles in the SAS session

❷ If TYPE = “T” or “F” then CALL EXCUTE is used to generate a TITLE or FOOTNOTE statement , respectively, where NUMBER and TEXT are used to produce the statement

❸ M_TITLES is deleted from _DELTA_

```
%macro returnoptions /store;
          /*capture current-new options*/;
          ❶proc sort data=sashelp.voption out=_delta_.m_noption; by optname; run;
          /*compare new options to previous and set back to previous*/
          data _null_;
                ❷merge _delta_.m_ooption _delta_.m_noption (keep=optname setting
                rename=(setting=csetting));
                by optname;
                ❸if setting ne csetting then do;
                call execute("option "||trim(keyword)||";");
                end;
          run;
          proc delete data=_delta_.m_noption _delta_.m_ooption; run;
    %mend;
```

❶ Save current option settings returned by SASHELP.VOPTION to _DELTA_.M_NOPTION

❷ MERGE _DELTA_.M_NOPTION by OPTINAME to the previous options that were stored in _DELTA_.M_OOPTION by the macro program SAVEOPTIONS

❸ If the option settings are different then use CALL EXCEUTE to generate an OPTION statement that restores the original option

USING THE STARTPDF AND ENDPDF MACRO PROGRAMS

Once the macro programs have been saved to the shared folder, then STARTPDF and ENDPDF can be used across projects to quickly create PDF files with standard page setup and documentation. The original titles, footnotes and options in the SAS session are reset so that any output created after the PDF is not changed.

```
libname shareit "&path_sharedfolder"  access=readonly;
options mstored sasmstore=shareit fmtsearch=(shareit work library) mprint;
ods path shareit.templat(read) sashelp.tmplmst(read);
%let mytitle =Using STARTPDF and ENDPDF Other Options;
%startpdf(path=&path_export,name=&mytitle,odspdfoptions=%str(style=mydefault columns=1));
title "&mytitle";
%myprocreport3(sashelp,class,sex,rowvariablef=);
%endpdf;
ods path reset;
```

DEVELOPING INLINE STYLE MACRO PROGRAMS

Inline styles can be included in a number of locations within PROC REPORT syntax to alter the appearance of the PROC REPORT table as shown in the ODS PDF destination. The style could be hard-coded in a macro program; however, that limits the flexibility. The following macro program enables inline styles to be set through parameters to the macro program, while also allowing styles to be set and used across many different macro programs.

```
%macro report_crosstab (lib, data, rowvariable,rowvariablef=,
❶rowvarcolumnstyle=,
ncolumnstyle=,
percentcolumnstyle=,
❷globalstyle=n
);
%let me =;
/*get macro program name*/
❸ %if &globalstyle = n %then %let me = &sysmacroname;
/*set inline styles based on macro program and then on global defaults*/
❹%inlinestyle_procreport(&me);
proc report data=&lib..&data nowd;
  col &rowvariable n pctn;
  define &rowvariable/group &rowvariablef &rowvarcolumnstyle;
  define n /&ncolumnstyle;
  define pctn/format=shux. &percentcolumnstyle;
  rbreak after/summarize;
run;
%mend;
```

❶Parameters to a macro program are used to set the inline styles, but they default to missing in the %MACRO statement

❷ By default GLOBALSTYLE is set to n which macro specific styles to be used

❸ When GLOBALSTYLE is n macro variable ME is set to the name of the macro program (e.g. REPORT_CROSSTAB) using the system macro variable SYSMACRONAME

❹ A macro program called INLINESTYLE_PROCREPORT is called to set the values for any inline style parameter that is missing, thus default inline styles are applied unless the user supplies specific inline styles in the macro program call

The following syntax shows two calls to the macro program REPORT_CROSSTAB. The first call specifically sets the inline style for the percent column, while the second call relies solely on the default inline styles that are set by INLINESTYLE_PROCREPORT.

```
%report_crosstab(SASHELP,CLASS,SEX,
percentcolumnstyle=%str(style(column)=[cellwidth=100 just=right]));
%report_crosstab(SASHELP,CLASS,SEX);
```

In order to only set the macro variables ROWCOLUMNSTYLE, NCOLUMNSTYLE, and PERCENTCOLUMNSTYLE to default inline styles when they are not specified we need to test if they are missing. The following macro program is used by the INLINESTYLE_PROCREPORT macro program to test for missing style specification.

```
%macro setinlinestyle(name,style);
 /*If macro variable exists*/
❶%if %symexist(&name) %then %do;
     /*If macro variable is missing then set style*/
    ❷%if "&&&name"= "" %then %let &name =&style;
%end;
%mend;
```

❶ If the resolved value of name is the name of a macro variable then continue

❷ If the value of macro variable that name resolves to is missing then set it to the resolved value of style

The following is the INLINESTYLE_PROCREPORT macro that is responsible for setting the default inline style if no inline style is specified in the REPORT_CROSSTAB call.

```
%macro inlinestyle_procreport(calling_program);
        /*Set inline style for specific calling program*/
        ❶%if &calling_program = REPORT_CROSSTAB %then %do;
                /*Only set style if sytle is not already set*/
                  ❷%setinlinestyle(rowvarcolumnstyle, %str(style(column)=[cellwidth=100 just=Right] )
                  )
                  /*set n column style for all reports*/
                  ❸ %setinlinestyle(ncolumnstyle,%str('n' style(column)=[ cellwidth=50 just=center])
                  )
                  /*set percent column style for all reports*/
                  ❹ %setinlinestyle(percentcolumnstyle,%str('%' style(column)=[cellwidth=80 just=right])
                  )
        %end;
         /*Set any global inline style defaults that are not already set*/
        ❺ %inlinestyledefaults
%mend inlinestyle_procreport;
```

❶ If the calling program equals REPORT_CROSSTAB then go on to set specific default inline styles

❷ Call to SETINLINESTYLE macro program shown previously, so if the ROWCOLUMNSTYLE macro variable exists and is missing then it is assigned the inline style specified in the second parameter to SETINLINESTYLE

❸ ❹ Call SETINLINESTYLE macro program for NCOLUMNSTYLE and PERCENTCOLUMNSTYLE macro variables, respectively, with the effect of assigning a default inline style to each if they are missing

❺ Call to a macro program called INLINESTYLEDEFAULTS which sets global defaults for any inline style macro variables that are still missing, thus allow macro programs other than REPORT_CROSSTAB to use the same macro variable names for inline styles

The following is the syntax for the INLINESTYLEDEFAULTS macro program.

```
%macro inlinestyledefaults;
    /*Only set style if style is not already set*/
    %setinlinestyle(rowvarcolumnstyle, %str(style(column)=[cellwidth=100 just=left])
    )
    /*set n column style for all reports*/
    %setinlinestyle(ncolumnstyle,%str('n' style(column)=[cellwidth=50 just=center])
    )
    /*set percent column style for all reports*/
    %setinlinestyle(percentcolumnstyle,%str('%' style(column)=[cellwidth=80
just=right])
    )
%mend inlinestyledefaults;
```

This framework of macro programs, INLINESTYLE_PROCREPORT, INLINESTYLEDEFAULTS, and SETINLINESTYLE may be exploited by adding:

- New macro programs, such as REPORT_CROSSTAB, and new appropriate inline styles
- New macro variables to store inline styles
- More sophisticated inline styles

All the default inline styles for the macro program REPORT_CROSSTAB are shown in Appendix 4.

DEVELOPING MACRO PROGRAMS FOR PROC REPORT

There are a number of tasks that REPORT_CROSSTAB needs to orchestrate to better serve its purpose:

1. Create a LIBREF _DELTA_, if it does not exist, to store temporary data sets
2. Check for the existence of user-supplied LIBREF, data set, and variables
3. Get variable(s) type, label and user defined format
4. If no user -supplied inline styles, then set default inline styles
5. Create or increment table number
6. Create an anchor for the PDF bookmark
7. Set the label for PDF bookmark
8. Create a temporary data set from user-supplied data set with a variable having a constant value which is needed to create a single bookmark in the PDF (Lawhorn, 2011)
9. Create PROC REPORT syntax including the resolution of macro variables having inline styles

The complete set of macro programs for this section is too long and involved to present here but an example of the PROC REPORT syntax that is generated by the REPORT_CROSSTAB macro program is shown below, starting with the following DATA step.

```
❶ ods proclabel="Table 1 Formatted Numeric Variable (Child's Age) ";
 data _delta_.class;
 set work.class;
❷ _tableofcontents=1;
 run;
```

❶ ODS PROCLABEL sets the label for the bookmark in the PDF document

❷ The variable _TABLEOFCONTENTS is created and set to 1. This is necessary to for PROC REPORT to use the variable to remove bookmarks

The following syntax was used to generate table 1 shown in Appendix 1, and serves as example of the syntax generated as a result of the system of macro programs. The inline style were all generated through a system of macro program described in the previous section, and the extended styles shown in Appendix 4. The actual PROC REPORT syntax has been descripted in detail by Miller (2011).

```
proc report nowd data=_delta_.class SPLIT='~' ❶contents="" missing
completerows style(lines)=header{background=white asis=on font_size=7pt
font_face="Helvetica"
font_weight=bold just=left} style(header)={just=left} style(column)={just=right};
   col ( "Table 1 Formatted Numeric Variable (Child's Age) "
_tableofcontents age dumcol n pctn age = n2 age = pctn2 );
   define _tableofcontents / group noprint;
   define age/ group order= data preloadfmt noprint id contents=""
format=AGEF. ;
   define dumcol / '' computed flow style(column)=[cellwidth=100 just=left]
width=30 format = $30. ;
   define n / 'n' format=7.0 style(column)=[background=graydd cellwidth=50
just=center] style(header) = {just=center} ;
   define pctn / '%' format=7.0 style(column)=[cellwidth=80 just=right]
style(header) = {just=center} format=shux. ;
   define n2 / analysis n "n" format=7.0 style(column)=[background=graydd
cellwidth=50 just=center] style(header) = {just=center} ;
   define pctn2 / analysis pctn '%' format=7.0 style(column)=[cellwidth=80
just=right] style(header) = {just=center} format=shux. ;
   ❷ break before _tableofcontents/ contents="" page ;
   compute dumcol / char length=50;
   dumcol = put(age,AGEF.);
   if _BREAK_="_RBREAK_" then dumcol = 'Total';
   else if missing(age) then dumcol="Missing";
   endcomp;
   rbreak after / ol skip summarize suppress page contents="" ;
   compute after;
   line @1 "Exclude % Missing Rows";
   ;
   endcomp;
   run;
```

❶ CONTENTS="" removes the second level PDF bookmark that is created by default by PROC REPORT

❷ _TABLEOFCONTENTS is used in a BREAK statement with a CONTENTS="" to remove the third level bookmark

CONCLUSION

The system of macro programs described in part here has been used successfully to generate 1000s consistent PDF documents with100s of tables per document (e.g. Appendix 1 & 3). Sections have described the fundamentals of sharing and reusing FORMATS, macro programs, and style templates. Later section have presented macro programs to standardize the opening, closing, page setup and documentation of a PDF and set global, macro program and call specific inline styles (Appendix 4).

ACKNOWLEDGEMENTS

I would like to thank Ethan Miller for his contribution to the development of the PROC REPORT syntax, and for his continual support and encouragement for the development of this system of macro programs. I would also like to thank Michelle Woodbridge believing in and supporting our work across several projects.

RECOMMENDED READING

Carpenter A. (2007). Carpenter's Complete Guide to the SAS® REPORT Procedure. Cary, NC: SAS Institute, Inc.

Carpenter A. (2012). PROC REPORT Basics: Getting Started with the Primary Statements. Proceedings of the SAS Global Forum 2012, Orlando, FL.

DiIorio, F. (2010). %whatChanged: A Tool for the Well-Behaved Macro. Proceedings of the Southeast SAS Users, Savanna, GA.

Fehd, R. (2005). A SASautos Companion: Reusing Macros. Proceedings of the SAS Users Group International, Philadelphia, PA

Haworth, L. (2005). SAS® with Style: Creating your own ODS Style Template for PDF Output. Proceedings of the Thirtieth Annual SAS User Group International Conference, Philadelphia, PA

Lawhorn, B. (2011). Let's Give 'Em Something to TOC about: Transforming the Table of Contents of Your PDF File. Proceedings of the SAS Global Forum, Las Vegas, NV

Miller, E. (2011). "Repo" Your Missing Data Using PROC REPORT. Proceedings of the SAS Global Forum, Las Vegas, NV

Thornton, S. P. (2009). Using PROC REPORT to Cross-Tabulate Multiple Response Items. Proceedings of the Seventeenth Annual Western Users of the SAS® Software Conference, San Jose, CA.

Thornton, S. P. (2009a). Capturing and Presenting PROC TTEST Results using ODS and PROC REPORT. Proceedings of the Seventeenth Annual Western Users of the SAS® Software Conference, San Jose, CA.

Thornton, S. P. (2010). Essential SAS® ODS PDF. Paper presented at the Eighteenth Annual Western Users of the SAS® Software Conference, San Diego, CA.

CONTACT INFORMATION

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

S. Patrick Thornton Ph.D., Sr. Social Science Programmer
SRI International, Center for Education and Human Services
Phone: 650 859-5583
Fax: 650 859-2861
Email: patrick.thornton@sri.com
Web: www.sri.com

APPENDIX 1: EXAMPLE OUTPUT FROM REPORT_CROSSTAB MACRO

Example Output from Macro Program Application
Frequencies and Cross-Tabulations

Table 1 Example of Unformatted Numeric Variable (Child's Age)

	n	%	n[1]	%[1]
Missing	30	5.3%	0	0.0%
11	60	10.5%	60	11.1%
12	150	26.3%	150	27.8%
13	90	15.8%	90	16.7%
14	90	15.8%	90	16.7%
15	120	21.1%	120	22.2%
16	30	5.3%	30	5.6%
Total	570	100.0%	540	100.0%

[1] Exclude % Missing Rows

Table 2 Example of Unformatted Character Variable (Child's Ethnicity)

	n	%	n[1]	%[1]
Missing	302	53.0%	0	0.0%
Pacific Islander	132	23.2%	132	49.3%
Latino	74	13.0%	74	27.6%
White	26	4.6%	26	9.7%
Native American	9	1.6%	9	3.4%
African American	27	4.7%	27	10.1%
Total	570	100.0%	268	100.0%

[1] Exclude % Missing Rows

Table 3 Example of Formatted Numeric Variable (Child's Age)

	n	%	n[1]	%[1]
Pre Teen	210	36.8%	210	38.9%
Teen	330	57.9%	330	61.1%
Missing	30	5.3%	0	0.0%
Total	570	100.0%	540	100.0%

[1] Exclude % Missing Rows

Table 4 Example of Formatted Character Variable (Child's Ethnicity)

	n	%	n[1]	%[1]
Missing	302	53.0%	0	0.0%
Latino	74	13.0%	74	27.6%
White	26	4.6%	26	9.7%
Other	168	29.5%	168	62.7%
Total	570	100.0%	268	100.0%

[1] Exclude % Missing Rows

Table 5 Example of User Specified Format Numeric Variable (Child's Age)

	n	%	n[1]	%[1]
11-12	210	36.8%	210	38.9%
13-14	180	31.6%	180	33.3%
15-16	150	26.3%	150	27.8%
Missing	30	5.3%	0	0.0%
Total	570	100.0%	540	100.0%

[1] Exclude % Missing Rows

Table 6 Example of User Specified Format Character Variable (Child's Ethnicity)

	n	%	n[1]	%[1]
Missing	302	53.0%	0	0.0%
White	26	4.6%	26	9.7%
Other	242	42.5%	242	90.3%
Total	570	100.0%	268	100.0%

[1] Exclude % Missing Rows

Table 7 Example of Two Formatted Numeric Variables (Child's Age and Service Level)

	Three Levels of Service							
	Low		Medium		High		Total	
	n	%[1]	n	%[1]	n	%[1]	n	%[1]
Pre Teen	59	39.6%	56	31.8%	95	44.2%	210	38.9%
Teen	90	60.4%	120	68.2%	120	55.8%	330	61.1%
Missing	9	0.0%	7	0.0%	14	0.0%	30	0.0%
Total	158	100.0%	183	100.0%	229	100.0%	570	100.0%

[1] Exclude % Missing Rows

Table 8 Example of Two Unformatted Numeric Variables (Child's Age and Service Level)

	Three Levels of Service							
	0		1		2		Total	
	n	%[1]	n	%[1]	n	%[1]	n	%[1]
Missing	9	0.0%	7	0.0%	14	0.0%	30	0.0%
11	18	12.1%	14	8.0%	28	13.0%	60	11.1%
12	41	27.5%	42	23.9%	67	31.2%	150	27.8%
13	16	10.7%	40	22.7%	34	15.8%	90	16.7%
14	24	16.1%	35	19.9%	31	14.4%	90	16.7%
15	38	25.5%	36	20.5%	46	21.4%	120	22.2%
16	12	8.1%	9	5.1%	9	4.2%	30	5.6%
Total	158	100.0%	183	100.0%	229	100.0%	570	100.0%

[1] Exclude % Missing Rows

Table 9 Example of Two Numeric Variables User Specified Formats (Child's Age and Service Level)

	Three Levels of Service					
	Low		Other		Total	
	n	%[1]	n	%[1]	n	%[1]
11-12	59	39.6%	151	38.6%	210	38.9%
13-14	40	26.8%	140	35.8%	180	33.3%
15-16	50	33.6%	100	25.6%	150	27.8%
Missing	9	0.0%	21	0.0%	30	0.0%
Total	158	100.0%	412	100.0%	570	100.0%

[1] Exclude % Missing Rows

Table 10 Example of Character Column Variable User Specified Format (Child's Age and Ethnicity)

	Character Ethnicity							
	Missing		White		Other		Total	
	n	%[1]	n	%[1]	n	%[1]	n	%[1]
Pre Teen	112	39.4%	8	34.8%	90	38.6%	210	38.9%
Teen	172	60.6%	15	65.2%	143	61.4%	330	61.1%
Missing	18	0.0%	3	0.0%	9	0.0%	30	0.0%
Total	302	100.0%	26	100.0%	242	100.0%	570	100.0%

[1] Exclude % Missing Rows

APPENDIX 2: PARAMETERS AND LOCAL VARIABLE FOR MACRO PROGRAM REPORT_CROSSTAB

```
%macro report_crosstab(
lib,                    /*LIBREF*/
data,                   /*data set*/
rowvariable,            /*discrete row variable on which to obtain a frequency*/
colvariable,            /*discrete col variable on which to obtain a frequency*/
tabletitle=,     /*title of the table, if blank will use label of row variable*/
rowvariablef=,   /*format for the variable on which to obtain a frequency*/
rowvariablel=,   /*label of variable to appear in Table title*/
roworderfmt=data preloadfmt,    /*default to sorting and formatting row variable*/
colvariablef=,   /*format of column variable*/
colvariablel=,   /*label of variable of column variable*/
colorderfmt=data preloadfmt,     /*default to sorting and formatting column
                                 variable*/
mean=,                  /*if =y then puts mean etc...of rowvariable in footnote*/
dumrowz=Missing,        /*Value shown in table row when variable is missing*/
procreportoptions=missing completerows, /*default PROC REPORT will show missing and
all ROWS*/

rowdescending=,   /*sort order. SAS default is ascending. Can make this descending*/
foot2=,                 *Depreciated. Used to add a second footnote to the table*/
rowvarcolumnstyle =,  /*set a specific inline style for DEFINE of row variable*/
colvarcolumnstyle=, /*set a specific inline style for DEFINE of col variable*/
ncolumnstyle =,  /*set a specific inline style for DEFINE of N */
percentcolumnstyle=, /*set a specific inline style for DEFINE of PCT */
ncolumnstyle2 =, /*set a specific inline style for DEFINE of alias N2 exclude
missing*/

percentcolumnstyle2=,   /*set a specific inline style for DEFINE of alias PCT2
exclude missing*/

procreport_lines_style=,       /*set PROC REPORT line style*/
procreport_header_style=,      /*set PROC REPORT header style*/
procreport_footline1_content=, /*table footnote 1*/
procreport_footline2_content=, /*table footnote 2*/
about=                         /*show help*/
);

%local
error      /*Set by macro doesdatasetexist. 0 or a problem statement*/
me         /*will be set to name of this macro*/
rowvariablet /*set to N or C for type of row variable by doesdatasetexist*/
rowvariablefa /*user defined format assigned to the variable from doesdatasetexist*/
rowvariablela /*label assigned to the variable retrieve by doesdatasetexist*/
rowvariableforalias /*name of the variable to use as the source of the aliases.
if row variable is numeric then rwo variable is used, else numeric _flg is created
and set to 1 when character row variable is missing*/
colvariablefa /*format assigned to the col variable retrieve by doesdatasetexist*/
colvariablela/*label assigned to the col variable retrieve by doesdatasetexist*/
colvariablet /*set to N or C for type of col variable by doesdatasetexist*/
defaultformat/*a dummy row variable has to be create to add TOTAL label row
variable, so the dummy row format is either 8. for numeric or $25 for character*/

tablename  /*name of table set by %helpfulinfo*/
tablerequirements
abouttext
macrocall
;
```

APPENDIX 3: ADDITIONAL EXAMPLE OUTPUT FROM VARIOUS OTHER MACRO PROGRAMS

Example Output from Macro Program Application
Descriptive Statistics and Various Inferential Tests

Table 11 Average by Classification (Average Child's Age by Service)

	n	n[1]	Mean	Min	Max	STD
Low	158	149	13.396	11	16	1.589
Medium	183	176	13.364	11	16	1.383
High	229	215	13.126	11	16	1.485
Total	570	540	13.278	11	16	1.485

[1] Number reporting

Table 12 Change in Percent (Children with Health Risk)

	Intake		Follow-up	
Values	n	%	n	%
No	281	49.3%	281	49.3%
Yes	289	50.7%	289	50.7%
Total	570	100.0%	570	100.0%

McNemars=0.0000 DF=1 p=1.0000

Table 13 Paired T-Test (Child's Weight)

	MEAN	MIN	MAX	Std Dev	Std Error
Baseline	2.081	0	15	3.160	.
6-Months	6.064	0	18	4.632	.
Difference	3.984	-13	18	5.173	0.217

Paired T-Test n=570 t=18.386 DF=569 p=0.0000 PooledSD=3.9651 D=1.0047

Table 14 Chi Square Test

Three Levels of Service		n	%	% Column 1	% Column 2
High	No	112	19.6%	48.9%	39.9%
	Yes	117	20.5%	51.1%	40.5%
Low	No	76	13.3%	48.1%	27.0%
	Yes	82	14.4%	51.9%	28.4%
Medium	No	93	16.3%	50.8%	33.1%
	Yes	90	15.8%	49.2%	31.1%
		570	100.0%		

df= 2 Chi-Square = 0.274 p= .8720

Table 15 Independent Samples T-Test (Child's Weight by Age)

Condition	n	Mean	Min	Max	STD	STDERR
Pre Teen	210	1.755	0.0	14.8	3.110	0.215
Teen	330	2.242	0.0	14.9	3.128	0.172
Diff (1-2)	_	-0.487	_	_	3.121	0.276
Overall	570	2.081	0.0	14.9	3.160	0.132

t value=-1.768 DF=** p value=0.0776 Cohen's D=0.1525

APPENDIX 4 MACRO PROGRAM SPECIFYING INLINE SYTLES FOR REPORT_CROSSTAB

```
%macro inlinestyle_procreport(calling_program);
%local currentdestination;
/*determine whether PDF is a destination*/
%let currentdestination = ;
    proc sql noprint;
      select destination into: currentdestination  from
    dictionary.destinations where index(destination,"PDF") > 0;
    quit;
/*Set default styles specific to the calling macro*/
%if &calling_program = REPORT_CROSSTAB %then %do;
    %setinlinestyle(rowvarcolumnstyle,
    %str(style(column)=[cellwidth=300 just=left] width=30 format = $30.)
    )
     %if &currentdestination =PDF %then %do;

           %setinlinestyle(ncolumnstyle,
           %str("n" format=7.0 style(column)=[background=graydd cellwidth=50
just=center]
        style(header) = {just=center})
           )

    %setinlinestyle(percentcolumnstyle,
    %str('%' format=7.0 style(column)=[cellwidth=70 just=right] style(header) =
{just=center}
    format=shux.)
    )
           %setinlinestyle(ncolumnstyle2,
           %str("n^{super 1}" format=7.0 style(column)=[background=graydd
cellwidth=50 just=center]
        style(header) = {just=center})
           )

           %setinlinestyle(percentcolumnstyle2,
           %str("%^{super 1}" format=7.0
               style(column)=[cellwidth=80 just=right] style(header) = {just=center}
format=shux.)
           )

           %setinlinestyle(procreport_footline1_content,
           %str(line @1 "^{super 1} Exclude % Missing Rows")
           )

    %end;
    %else %do;

           %setinlinestyle(ncolumnstyle2,
           %str("n" format=7.0 style(column)=[background=graydd cellwidth=50
just=center]
        style(header) = {just=center})
           )

           %setinlinestyle(percentcolumnstyle2,
           %str('%' format=7.0 style(column)=[cellwidth=80 just=right]
           style(header) = {just=center} format=shux.)
           )
    %end;

%end;
%inlinestyledefaults;
%mend inlinestyle_procreport;
```