PharmaSUG China 2015 – 74

Application of Output Delivery System in Creating Customized Targets

Yuan Wang, Fountain Medical Development, Inc., Nanjing, China

Abstract

The default results from SAS STAT PROC contains many useful information. Organizing these output objects in SAS can be confusing, customizing the analysis summary results into required clinical output would be more tedious. In fact SAS Output Delivery System (ODS) is a powerful tool in dealing with all these issues. In this paper, we will present how to use SAS ODS to select a statistic output, convert a processed output into a dataset, organize them into a pretty RTF output and add some special symbols to explain the statistic model.

Introduction

The Output Delivery System (ODS) was created for SAS users to customize the content of output, choose how output is formatted, and customize the appearance. It gives you flexibility to generate, format, and reproduce SAS procedure and DATA step output. Since ODS should be a huge topic and it is impossible to cover them in depth at the same time. This workshop will take a fast-track overview on the application of ODS.

Demonstration shows as below:

1. Brief Introduction of Output Objects and ODS Destinations.
2. Use ODS TRACE to obtain the table names the procedure create and ODS SELECT/EXCLUDE to specify the tables that ODS selects or excludes.
3. Convert a processed output into a dataset by ODS OUTPUT.
4. Create your own style with TEMPLATE procedure.
5. Organize them into a pretty and customized RTF output.

Output Objects and Output Destinations

There is some confusion in the nomenclature of ODS: Many people have no idea of the conceptual distinction between Output Object and Output Destination. Actually SAS procedures produce the Output Object no matter what type of destination you designate while Output Delivery System delivers to various ODS destinations, according to the default specifications for the procedure or to your own ones.

Output objects have associated information and attributes. Each output object consists of two component parts:

I. Data Part: Which consists of the results computed by a SAS procedure (Numeric or Character).

II. Template Part: Which contains rules for formatting and displaying the results (Table Definition).

Each output object has an associated template that defines its presentation format. You can modify the presentation of the output by using the TEMPLATE procedure to alter these templates or to create new templates.

The Output destination is a SAS dataset with the same structure you'd get by running a DATA step.

When you invoke a SAS procedure, the procedure sends all output to the Output Delivery System. ODS then routes the output to all open destinations such as Listing, PDF, HTML, RTF, CSV or Output.

ODS TRACE: Identify the Output Objects

ODS produces an output object by combining data from the data component with a table definition. The trace record provides information about the data component, the table definition, and the output object. For any SAS procedure, you can use the ODS TRACE statement to find the descriptive information about each specified output object, such as its name, label, template and path.

```
/* Ods Trace*/
ods trace on;
proc ttest data=demo;
    class treat;
    var height;
run;
ods trace off;
```

This produces the following in the Log:

```
Output Added:
```

```
-------------
Name:       Statistics
Label:      Statistics
Template:   Stat.TTest.Statistics
Path:       Ttest.HEIGHT.Statistics
-------------

Output Added:
-------------
Name:       ConfLimits
Label:      Confidence Limits
Template:   Stat.TTest.ConfLimits
Path:       Ttest.HEIGHT.ConfLimits
-------------

Output Added:
-------------
Name:       TTests
Label:      T-Tests
Template:   Stat.TTest.TTests
Path:       Ttest.HEIGHT.TTests
-------------

Output Added:
-------------
Name:       Equality
Label:      Equality of Variances
Template:   Stat.TTest.Equality
Path:       Ttest.HEIGHT.Equality
-------------

Output Added:
-------------
Name:       SummaryPanel
Label:      Summary Panel
Template:   Stat.TTest.Graphics.Summary2
Path:       Ttest.HEIGHT.SummaryPanel
-------------

Output Added:
-------------
Name:       QQPlot
Label:      Q-Q Plots
Template:   Stat.TTest.Graphics.QQ2
Path:       Ttest.HEIGHT.QQPlot
```

ODS SELECT/EXCLUDE: Maintain a selection list for one destination and an exclusion list for another

Suppose that you want to display only the tables of ttest estimates from the procedure. You can give any of the following statements (before invoking TTEST preocedure) to display as needed. For this example, these statements are equivalent in order to modify the overall selection lists:

```
/* Ods Select*/
ods select where=(_name_ = “TTests”);
ods select Ttest.HEIGHT.TTests;
ods select HEIGHT.TTests;
ods select TTests (persist);
```

The first ODS SELECT statement specifies by a where expression that only output objects with the word “TTests” in their name. The second statement specifies the full qualified path while the third and fourth statements specify the partial path. In the last statement PERSIST option is used to keep the output-object that precedes PERSIST in the selection list, even if the DATA or PROC step ends, until you explicitly modify the list. Please take note that you must enclose PERSIST in parentheses..

ODS EXCLUDE is also a type of ODS Control Statement just as ODS SELECT which specifies output objects to exclude from ODS destinations. ODS SHOW can be used to write the overall selection list to the SAS log

ODS OUTPUT: Create the Output Dataset

Once finding out the output ODS table name for a PROC, you may need to use ODS OUTPUT statement to create the output dataset. However, if you specify the NOPRINT option, the procedure may not send any output to the Output Delivery System.

```
/* Ods Output*/
ods output ttests=ttests equality=equality;
ods exclude all;
proc ttest data=demo;
    class treat;
    var age height weight;
quit;
```

The sample code above shows how to get PROC TTEST output into an ODS output dataset for processing. The ODS OUTPUT statement writes the ODS table “ttests” and “equality” to SAS datasets called ttests and equality. In order to limit the amount of displayed output, the ODS exclusion list is set to ALL.

PROC TEMPLATE: Create your own ODS Style Template

SAS provides several default styles which can be automatically used. Besides you can use STYLE= option to chose the required one.

Table 1 Demographics

Parameter	Treatment Group			
Statistics	**Active**	**Placebo**	**Total**	**P-Value [a]**
Age (years)				0.xxx
n	xx	xx	xx	
Mean (SD)	xx.xx (xx.xx)	xx.xx (xx.xx)	xx.xx (xx.xx)	
Median	xx.xx	xx.xx	xx.xx	
Q1-Q3	xx.xx – xx.xx	xx.xx – xx.xx	xx.xx – xx.xx	
Min - Max	xx.xx – xx.xx	xx.xx – xx.xx	xx.xx – xx.xx	
Height (in)				0.xxx
n	xx	xx	xx	
Mean (SD)	xx.xx (xx.xx)	xx.xx (xx.xx)	xx.xx (xx.xx)	
Median	xx.xx	xx.xx	xx.xx	
Q1-Q3	xx.xx – xx.xx	xx.xx – xx.xx	xx.xx – xx.xx	
Min - Max	xx.xx – xx.xx	xx.xx – xx.xx	xx.xx – xx.xx	
Weight (lb)				0.xxx
n	xx	xx	xx	
Mean (SD)	xx.xx (xx.xx)	xx.xx (xx.xx)	xx.xx (xx.xx)	
Median	xx.xx	xx.xx	xx.xx	
Q1-Q3	xx.xx – xx.xx	xx.xx – xx.xx	xx.xx – xx.xx	
Min - Max	xx.xx – xx.xx	xx.xx – xx.xx	xx.xx – xx.xx	

[a]: P value is based on T-Test

Table 1. Table Shell

However, the default output style is likely not ideal for requirement from sponsor. Usually we need modify it with PROC TEMPLATE to produce output with different attributes. Let's look at the example program to learn how to create your own style as required in this customized table.

```
1. proc template;
2.      define style styles.myrtf;
        parent=styles.rtf;
3.      style fonts/
            "TitleFont"= ("Arial", 9pt, Bold)
            "TitleFont2"= ("Arial", 9pt, Bold)
            "FixedFont"= ("Arial", 9pt, Bold)
            "StrongFont"= ("Arial", 9pt, Bold)
            ……
         ;
4.       style color_list/
            "link" = blue
            "bgH" = white
            "fg" = black
            "bg" = white
         ;
5.       style body/
             bottommargin = 1in
             topmargin = 1in
             rightmargin = 1in
             leftmargin = 1in
         ;
6.       style table/
              frame = hsides
             rules = group
             cellpadding = 1
             cellspacing = 1
             borderwidth = 1
         ;
     end;
quit;
```

Following parts which have been identified with numbers are basic structure or important items need to be paid attention.

1. Invoke the TEMPLATE procedure.
2. Create a new style named styles.myrtf that inherits from styles.rtf . Aspects of the customized style now can be modified as necessary to meet the requirements. Please take note that there are many aspects and attributes, most of which do not need to be modified and specified in the code.
3. The style statement is used to supply attributes to the style element. Here it defines a style element with name fonts.
4. A list of font colors need be modified. Here the cryptic color names like "fg" and "bg" are used by the style definition to apply these colors to various parts of the output.
5. Body margins setting.
6. Create the style element table. Frame = box | above | below | hsides | vsides | lhs | rhs | void: Specifies which borders should appear around the perimeter of the table. Rules = all | cols | rows | groups | none: Specifies which borders should appear within the table.

ODS RTF and PROC REPORT: Organize the final data into a pretty and customized RTF output.

ODS ESCAPECHAR

Once the final dataset and customized rtf style have been created, we can use PROC REPORT with ODS RTF to organize them into a pretty output as needed. During the process you may come up against some difficulties. Before specify the problem and solution, here I need to mention the ODS ESCAPECHAR statement which can be used to define and use a specific escape character to perform in-line formatting., Anytime ODS encounters the character you specify, it will interpret whatever follows the character to be a sequence or string of control characters that will have an impact on the final output file.

Titles and footnotes

As is known to all, we can add the title and footnote by using title/footnote statements just before REPORT procedure. Besides, ODS RTF TEXT= statements may be another way to add a single text. Before invoking the REPORT procedure, you can add the following statement to add the title just up the table. However, in order to get the titles repeat on each page, the title statement is more practical. Moreover, BODYTITLE option can be used to get the specific titles and footers in the body of the table.

```
title1 j=c "Table 1 Demographics";

ods rtf text="@S={font=('Arial',10pt,bold) outputwidth=100% just=c}Table 1
Demographics";
```

Please take note that "@" here should be an escape character and the function of text in s={} is just to modify the title font and alignment. The use the footnotes can be the same way as above.

Add the page number as a footnote

Generally, we need to specify the current and total page number in the final output. The following code is an example which can meet the requirement.

```
footnote1 j=c "Page @{thispage} of @{lastpage}";
```

Inserting a border underneath "Treatment Group"

For the header "Treatment Group" just above the two treatment columns in table shell, you can add the character string just before the two variables with parentheses in the column statement. Besides, the rtf code with escape character can be used to insert the underline. The sample code showed as below.

```
columns ("Treatment Group @R/RTF'\fs0\brdrb\brdrs\brdrw20" active placebo);
```

Indents in RTF and conditional styling

Another functionality which is often needed, is the use of indents. There is no doubt that, we can create the dataset with leading space for the corresponding columns or rows and "ASIS" option can be used to achieve the indentation. However there is one more simple way. COMPUTE Blocks allow certain data step statements, including IF-THEN. Furthermore, CALL DEFINE sets attributes of report columns. The below code just shows how to use COMPUTE and CALL DEFINE to indent the text as required and conditionally make the text bold. Pay attention that protectspecialchars=off is to keep ODS from protecting text from interpretation by the program.

```
compute name;
    if seq = 0 then call define(_col_,"style","style=[font_weight=bold]");
    else call define
    (_col_,"style","style=[pretext='\ql\li400'protectspecialchars=off]");
endcomp;
```

Sometimes special format such as subscript and superscript is needed. In this paper @{super [a]} is a example to add [a] as an superscript.

Conclusion

SAS Output Delivery System has become one essential tool in creating our clinical SAS productions for years. From this paper you can see the strong and powerful function of ODS. Here we just demonstrate the basic options and functions by examples. You can take advantage of the advanced features of ODS to organize the output and customize the report with styles and templates.

References

[1] Carol Matthews.2013."Pretty Please?! Making RTF Output "Pretty" with SAS". Pennsylvania.

[2] Sonia Extremera.2011." How to Create a Custom Style". Madrid.

[3] Haworth, Lauren. 2004. "SAS® with Style: Creating your own ODS Template for PDF Output". South San Francisco.

[4] Myra A. Oltsik.2008."ODS and Output Data Sets: What You Need to Know". New York.

Contact Information

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

Name : Yuan Wang

Enterprise : Fountain Medical Development, Inc.

Address : Room 403, Building 43, No.70, Headquarter Base, Phoenix Road,
Jiangning District, Nanjing, China

Email : yuan.wang@fountain-med.com