Paper RIV-04

SAS® Proc Report and ODS ExcelXP Tagsets to Produce Customized Excel Output Without DDE

Mira Shapiro, Consultant, Bethesda, MD

Abstract

After completing what appeared to be a routine request for a data-driven report with Proc Tabulate, one of my tools of choice, the additional requirement of an asterisk in selected cells in the requested Excel spreadsheet became an opportunity to explore some other techniques and add some new skills to my SAS repertoire. While it would have been possible to take the Proc Tabulate output and use DDE to create the desired spreadsheet, I decided to use Proc Report and ODS ExcelXP Tagsets. The solution to producing this report required the use of many features of Proc Report including across variables, nested columns, call define, and style. The most interesting challenge was that the requester wanted the results delivered via an Excel spreadsheet with an asterisk in some of the columns based on two variables that were not being displayed in the report. The solution evolved into a short SAS program with a Proc Report step doing all of the data summarization and the majority of the formatting. This paper will briefly outline the DDE approach that might have been taken and will detail the solution that makes use of Proc Report and ODS ExcelXP Tagsets.

This presentation is for SAS Users who want to use some of the advanced features of Proc Report and gain an understanding of how to exploit Proc Report's underlying column structure. In addition, in a client server SAS implementation where DDE is not feasible, this approach gives an alternative for producing customized Excel output.

Introduction

As experienced SAS users know, there is always more than one way to approach a particular programming task. When I was asked to produce a weekly report that summarized events that fell into multiple nested categories, for a variety of locations, my initial solution was Proc Tabulate. The data-driven report would have a different number of rows and or columns each week based on what was reported in the proceeding week or month. After completing what I thought was a nearly finished prototype, I was asked to compare two other variables not displayed in the report, and based on the result, put an asterisk (*) next to the counts in selected cells in the resulting Excel spreadsheet. While using DDE was suggested, I had just participated in a SAS webinar on ExcelXP Tagsets and wanted to put my new found knowledge to work. After exploring all my options and, some long email exchanges with SAS gurus, I decided that an approach using Proc Report, Picture Formats and ODS ExcelXP Tagsets would be my solution of choice.

In this paper I will first describe the desired output, the structure of the data used and the challenges I faced. The original report used confidential medical data. So, for demonstrating these techniques, I have constructed a data set as described in the next section and displayed in Figure 1.

I will briefly describe the Proc Tabulate / DDE approach that I might have taken, and then the rest of this paper will focus on the details of the Proc Report / ODS ExcelXP tagsets solution. The goal of this discussion is to have the reader gain an understanding of Proc Report's underlying column structure and call define usage, thereby enabling them to exploit Proc Report's flexibility for their complex reporting needs. In addition, the reader will gain an understanding of how to create customized SAS output and direct it to an Excel spreadsheet using ExcelXP tagsets.

The Scenario and Dataset

For demonstrating these techniques, a dataset was created for this paper by extracting the variables make and model from the sashelp dataset cars supplied with SAS 9.2. Additional variables: state_registered, state_purchased, and number_purchased, were added to the data set and multiple records were created. In addition, a variable footnote_code was created and was coded as 1 if state_registered and state_purchased were different and 0 if they were equal. The value of footnote_code will be used to determine which cells in the resulting Excel spreadsheet will require an asterisk.

Figure 1: Sample Data

Make	Model	state_registered	state_purchased	footnote_code
Acura	MDX	VT	ME	1
Acura	RSX Type S 2dr	NY	NY	0
Acura	TL 4dr	VA	MA	1
Acura	3.5 RL 4dr	NY	NY	0
Acura	RSX Type S 2dr	ME	ME	0
Acura	TSX 4dr	VT	NY	1
Acura	3.5 RL 4dr	MA	MA	0
Acura	MDX	NY	NY	
Acura	TSX 4dr	VT	ME	
Acura	TL 4dr	NY	NY	
Acura	MDX	ME		
Acura	RSX Type S 2dr			
Acura	TL 4dr			
Acura				

Proc Tabulate Worked, But What About the Asterisk?

Initially the desired report was created very quickly with the Proc Tabulate code shown below but without the reference to the footnote code. The variables were properly nested by using the Proc Tabulate Table statement as shown in Figure 2 and exported to an Excel spreadsheet (Figure 3) through use of ExcelXP tagsets which will be described in a later section of this paper. When the report recipient requested that an asterisk be placed in the cell in the Excel spreadsheet, the footnote code variable was created and added to the Proc Tabulate code with the expectation that there would be a way to incorporate it into the cell. When all efforts to do that failed, the idea of using Proc Report was suggested by members of the SAS Technical Support Staff.

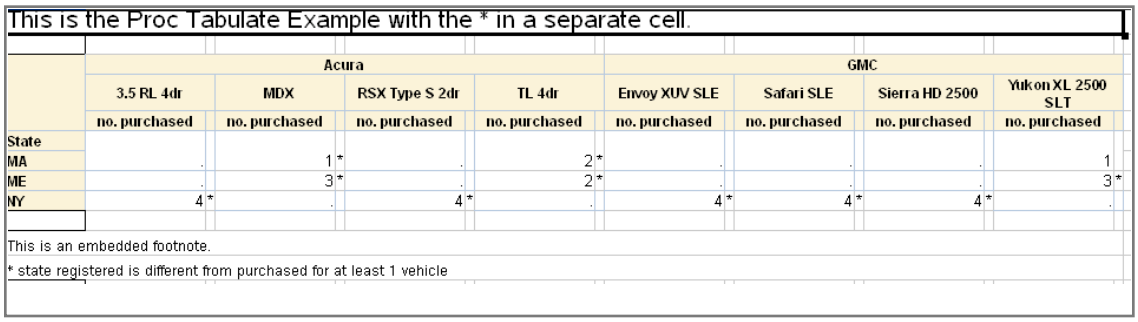
Figure 2: Initial reporting attempts using Proc Tabulate

```
Ods tagsets.ExcelXP
path='c:\Documents and settings\mira\desktop'
file='tabulate_example_two.xml'
style=sesug;
ods tagsets.ExcelXP options(embedded_titles='yes' embedded_footnotes='yes');
run;

Ods tagsets.ExcelXP options(sheet_name='Car_test');
title;
title1 font=calibri h=5 "This is the Proc Tabulate Example with the * in a
separate cell.";
footnote1 font=calibri h=2 j=l "This is an embedded footnote.";
footnote2 font=calibri h=2 j=l "* state registered is different from
purchased for at least 1 vehicle";
proc tabulate data=test4;
class state_purchased make model;
classlev state_purchased/s=[cellwidth=75];
var footnote_code number_purchased;
table state_purchased='State',make=' '*(model=' '*(number_purchased='no.
purchased'*sum=' ' footnote_code=' '*sum=' '*f=footcode.));

run;
ods tagsets.ExcelXP close; run;
```

Figure 3: The Excel Spreadsheet created with the Proc Tabulate code above.



This is the Proc Tabulate Example with the * in a separate cell.

	Acura				GMC			
	3.5 RL 4dr	MDX	RSX Type S 2dr	TL 4dr	Envoy XUV SLE	Safari SLE	Sierra HD 2500	Yukon XL 2500 SLT
	no. purchased	no. purchased	no. purchased	no. purchased	no. purchased	no. purchased	no. purchased	no. purchased
State								
MA	.	1 *	.	2 *	.	.	.	1
ME	.	3 *	.	2 *	.	.	.	3 *
NY	4 *	.	4 *	.	4 *	4 *	4 *	.

This is an embedded footnote.

* state registered is different from purchased for at least 1 vehicle

The Dynamic Data Exchange (DDE) Approach

DDE, which has been around since the late 1980's, originally was designed so that Microsoft Windows applications would be able to share data. Given that there are many newer, easier-to-use and more elegant approaches to producing SAS output in an Excel spreadsheet format, why does anyone still use DDE? The simple answer is that DDE allows for taking a pre-defined Excel template and directing output to specific cells in the spreadsheet using PUT statements.

The DDE approach involves getting the data in the "shape" that is required, writing a filename statement, and then using a Data _Null_ datastep with a file statement to direct the output. The Put statements to direct the results into the spreadsheet cells are coded within the Data _Null_ step. There are many papers, two of which are listed in the reference section of this paper, that describe the DDE approach, so it will not be repeated here. In addition, given the data-driven nature of this report and the high probability that the customer will be moving to a server-based SAS implementation, it made more sense to use the Proc Report approach, which will be detailed in the next sections. One of the disadvantages of the DDE approach is that both Excel and SAS need to be open on the same system for it to be utilized. Given that many companies use server-based rather than PC-based SAS implementations, the DDE approach is not feasible in those environments.

The Proc Report Solution

The desired output (figure 11) seems simple, so why all the fuss? Notice that the asterisks are properly placed in the Excel spreadsheet cells and the numbers that are without asterisks are properly aligned so that when cells with the asterisk and those without are in the same column, the numbers will line up. In addition, since the report is data-driven, it will contain a different number of rows and columns based on the data that is available at the time of execution, thereby allowing for the report to be run in production with minimal intervention.The SAS code for the report is shown in Figure 4.

Figure 4: SAS Code to create the desiired Excel spreadsheet using ExcelXP Tagsets, Proc Report and Macro

```
Ods tagsets.ExcelXP
path='c:\Documents and settings\mira\desktop'
file='sesug_new3.xml'
style=sesug;
ods tagsets.ExcelXP options(embedded_titles='yes'
embedded_footnotes='yes');
run;

Ods tagsets.ExcelXP options(sheet_name='Car_test');
title;
title1 font=calibri h=5 "This is the Proc Report Example Using Across
Variables and Embedded Titles";
footnote1 font=calibri h=2 j=l "This is an embedded footnote.";
footnote2 font=calibri h=2 j=l "* state registered is different from
purchased for at least 1 vehicle";
%macro create;
%let startcol=2;
%let varsleft=1;
%let varsunder=3;
proc report data=test4 nowd out=testweek ls=256 nocompletecols;
column state_purchased make, model, (number_purchased footnote_code
star);
define state_purchased/group "State" order=data
style(column)=[protectspecialchars=off cellwidth=.75IN];
define make/across " " ;
define model/group across " ";
define number_purchased/analysis "No. Purchased" sum
format=no_star. style=[protectspecialchars=off tagattr="format:@"
cellwidth=1IN];
define star/noprint;
define footnote_code/ noprint analysis " " sum;
compute star;
%do i=&startcol %to %eval(&make_across_ct*&model_ct*&varsunder+&varsleft)
  %by &varsunder;

  if _c%eval(&i+1)_=>1
       then
        Call Define("_c%eval(&i.)_",'format','with_star.');
            else if _c%eval(&i)_>=1 then
                  Call Define("_c%eval(&i.)_",'format','no_star.');
%end;
endcomp;
quit;
%mend create;
%create;
run;
ods tagsets.ExcelXP close; run;
ods pdf file="c:\Documents and settings\mira\desktop\sampledata.pdf";
```

Proc SQL to Count the number of Rows and Columns for the report using the current dataset

In order to make the report truly data-driven, it is necessary to ascertain how many rows and columns will be required for the represent the current dataset. Here we use Proc Sql / Count Distinct to count the number of models and makes of cars that exist in the current dataset and create macro variables. These values are used in the Proc Report step to loop through the underlying column structure to place the asterisks in the appropriate cells.

Figure 5: Proc SQL code used to determine the of rows and columns in this week's report

```
* count the number of rows and columns to be used in proc
report;
Proc SQL noprint;
      Select count(distinct model)
      into :model_ct
      from test4;
      Quit;
* create the macro variable for proc report do loop;
      %put &model_ct;
      Proc SQL noprint;
      Select count(distinct make)
      into :make_across_ct
      from test4;
      Quit;
* create the macro variable for proc report do loop;
      %put &make_across_ct;
```

Proc Report Options

Like most SAS Procedures, Proc Report has a multitude of options, which are fully explained in the SAS documentation and further explained in detail in many of the excellent SAS papers found online. For this report three Proc Report options (Figure 6) were specified: NOWD, NOCOMPLETECOLS and LS. Proc Report can be executed in an interactive windowing environment, which is the default. It you choose, as was done in this case, to turn off the interactive window, this is done by specifying the option NOWINDOWS, which can be abbreviated as NOWD. The default in Proc Report for Across variables, those that span several nested columns, is COMPLETECOLS. In this case Proc Report will display all possible combinations of the across variables, resulting in an unwieldy report with many empty cells. To turn off this feature, as was desired for this report, the option NOCOMPLETECOLS is specified. The other option specified for this report is linesize, which is abbreviated as LS. This option is used to specity how long a line in the report may be.

Figure 6: Proc Report options specified in this report

```
proc report data=test4 nowd out=testweek ls=256 nocompletecols;
```

Column statement and nesting

The Proc Report column statement (Figure 7) is used to layout the report by specifying the variables to use for the report and how they are nested. The variable usage and whether or not they are displayed in the report is further refined through the define statements and compute statements explained in the next two sections.

In this report, the nesting is evident in the column statement through the use of parentheses, but the full structure of the report is not fully defined until examining the define and compute options.

Figure 7: The Proc Report column statement indicates column order and nesting.

```
column state_purchased make, model, (number_purchased footnote_code star);
```

Define statement specification

In this report the Across, Group, Analysis, Noprint, format,style, and style(column) define statement options are used (Figure 8).Specifying a variable as across indicates that it spans nested columns; in the case of this report, Make spans the model columns. The specification of group indicates to Proc Report that the value of this variable is to be summarized. The Analysis attribute is used to indicate that some arithmetic is to be performed for the specified variable, and noprint indicates that the variable is used for definition or computation only and not to be displayed in the report.

Since the final output is going to reside in an Excel spreadsheet the style options specified are Excel-specific commands. Column width specifies the exact column size, so that the report recipient does not have to readjust the Excel columns upon opening the report. “Proctectionspecialcharacters=off”, “tagattr=”format:@”]” combined with the picture formats described in the next section allow for proper alignment of the asterisk and numbers in the Excel cells.

Figure 8: The Proc Report define statements to refine report variable definitions.

```
define state_purchased/group "State" order=data
style(column)=[protectspecialchars=off cellwidth=.75IN];
define make/across " " ;
define model/group across " ";
define number_purchased/analysis "No. Purchased" sum
format=no_star. style=[protectspecialchars=off tagattr="format:@"
cellwidth=1IN];
define star/noprint;
define footnote_code/ noprint analysis " " sum;
```

Proc Format, Picture format to incorporate asterisk into the report.

The picture formats (Figure 9) “no-star” and “star” specify how to display the numbers and asterisks in the Excel cells. “9 ” tells Excel to display the number, formatted without an asterisk, with a space following. “9*” tells Excel to display the numbers requiring an asterisk, with the asterisk following the digit. Specifying the space in the cells where the asterisk is not required,] allows all of the digits in the spreadsheet columns to be lined up.

Figure 9: Proc Format, picture formats to specify asterisk/no asterisk in an excel cell.

```
*formats for star and no star;
proc format;
picture no_star(round)
low-high='9&amp;nbsp;';
picture with_star(round)
low-high='9*';
```

Compute Block to Traverse the Underlying Columns and Determine Asterisk Placement and Call Define to use the Picture Format

The heart of this Proc Solution lies in the compute block code (Figure 10). As discussed earlier, Proc Report’s default is the complete column option which creates columns for each across variable combination. When the nocompletecols option is specified, although the empty columns are not displayed, they still exist in Proc Report’s

underlying column structure. In order to access the appropriate column to include the asterisk for this report, it is necessary to understand this column structure and traverse the structure with some sort of looping mechanism.

For this report, the Proc Report code was contained within a macro, so that macro variables could be exploited for this process. First of all, earlier in the program we counted the number of makes and models that will appear in this week's report and created the macro variables make_across_ct and model_ct. In addition, %let statements were used to specify how many columns are to be skipped, varsleft, and how many columns are nested underneath, varsunder. The %do and %by statements make use of these variables to control how to traverse Proc Report's underlying column structure.

The Call Define statements combined with the %if syntax, provide the mechanism to check the value of the footnote_code variable and determine whether or not an asterisk should be added to the output.

Figure 10: Proc Report Compute Block Syntax

```
compute star;
%do i=&startcol %to %eval(&make_across_ct*&model_ct*&varsunder+&varsleft)
  %by &varsunder;

  if _c%eval(&i+1)_=>1
       then
        Call Define("_c%eval(&i.)_",'format','with_star.');
            else if _c%eval(&i)_>=1 then
                  Call Define("_c%eval(&i.)_",'format','no_star.');
%end;
endcomp;
```

Using ODS ExcelXP Tagsets

Now that Proc Report has created the output that we envisioned, we need to place the output in a file that Excel can open and read. It turns out that an efficient way to do that is to use ExcelXP Tagsets to create an XML file that then can be opened in Excel. As with all ODS output in SAS, the statements require the destination, options, and ODS close statement surrounding the report code. Figure 11 contains the code that was used to create the XML file from the Proc Report-created output.

In this case we specified embedded titles and footnotes so that we can include the titles and footnotes in the Excel spreadsheet. In addition the path, filename and sheet name are specified. Among other options, the user has the option to specify style. In this case, Proc Template was used to create style SESUG incorporating a specific color scheme.

Figure 11: ExcelXP Tagsets

```
Ods tagsets.ExcelXP
path='c:\Documents and settings\mira\desktop'
file='sesug_new3.xml'
style=sesug;
ods tagsets.ExcelXP options(embedded_titles='yes'
embedded_footnotes='yes');

Ods tagsets.ExcelXP options(sheet_name='Car_test');


......Program statements


ods tagsets.ExcelXP close; run;
```

Finally, the Report!

The final report is shown in Figure 10. The asterisks and numbers line up, titles and footnotes are embedded, and only columns that have at least one non-empty cell are included.

Figure 10: The Resulting Excel Output

This is the Proc Report Example using Across variables and Embedded Titles

	Acura				GMC			
	3.5 RL 4dr	MDX	RSX Type S 2dr	TL 4dr	Envoy XUV SLE	Safari SLE	Sierra HD 2500	Yukon XL 2500 SLT
State	No. Purchased	No. Purchased	No. Purchased	No. Purchased	No. Purchased	No. Purchased	No. Purchased	No. Purchased
ME	.	3*	.	2*	.	.	.	3*
NY	4*	.	4*	.	4*	4*	4*	.
MA	.	1*	.	2*	.	.	.	1

This is an embedded footnote.

* state registered is different from purchased for at least 1 vehicle

Conclusion

Given that there are often numerous ways to create the same results in SAS, it is sometimes useful to re-examine your approach and apply a different technique that enhances your SAS knowledge and provides a more elegant solution to the problem at hand. In this example, the added requirement of adding an asterisk in selecting cells in the Excel spreadsheet provided an opportunity for me to enhance my knowledge of Proc Report and to introduce Proc Report and ExcelXP tagsets to a group of SAS users that were primarily using DDE for their Excel output.

Isn't this a lot of trouble simply to give the report recipient an asterisk in a few Excel spreadsheet locations? Not really. This technique can be used for traffic lighting reports and a host of other data-driven reporting needs. Once understood, this approach is an easy-to-use approach that can be achieved with minimal coding and provides a flexible approach to customizing your Excel or other ODS destination files.

References

Paper 276-2007
Beyond the Basics: Advanced PROC REPORT Tips and Tricks
Allison McMahill, , SAS Institute, Inc., Cary, NC
http://support.sas.com/rnd/papers/sgf07/sgf2007-report.pdf

Paper 141-2010
So You're Still Not Using PROC REPORT. Why Not?
Ray Pass, BJC HealthCare
Daphne Ewing, Auxilium Pharmaceuticals, Inc.
http://support.sas.com/resources/papers/proceedings10/141-2010.pdf

Paper 173-2008
Creating Complex Reports
Cynthia L. Zender, SAS Institute, Inc., Cary, NC
http://www2.sas.com/proceedings/forum2008/173-2008.pdf

Paper 188-2008
PROC REPORT:
Compute Block Basics – Part II Practicum
Arthur L. Carpenter
California Occidental Consultants
http://www2.sas.com/proceedings/forum2008/188-2008.pdf

Paper 153-2010
Traffic Lighting Your Multi-Sheet Microsoft Excel Workbooks
the Easy Way with SAS®
Vincent DelGobbo, SAS Institute Inc., Cary, NC
http://support.sas.com/resources/papers/proceedings10/153-2010.pdf

Paper 12-28
Generating Custom Excel Spreadsheets using ODS
Chevell Parker, SAS Institute, Cary, NC
http://www2.sas.com/proceedings/sugi28/012-28.pdf

Paper 099-2007
Funny ^Stuff~ in My Code: Using ODS ESCAPECHAR
Cynthia L. Zender, SAS Institute Inc., Cary, NC
http://www2.sas.com/proceedings/forum2007/099-2007.pdf

Paper 154-31
Step-by-Step in Using SAS® DDE to Create an Excel Graph Based on N
Observations from a SAS Data Set
Choon-Chern Lim, Mayo Clinic, Rochester, MN
http://www2.sas.com/proceedings/sugi31/154-31.pdf

Paper 022-31
SAS®-with-Excel Application Development: Tools and Techniques
LeRoy Bessler, Assurant Health, Milwaukee, Wisconsin, USA
http://www2.sas.com/proceedings/sugi31/022-31.pdf

Acknowledgements

I would like to give special thanks to Barbara Okerson & Andrea Wainwright-Zimmerman, the section co-chairs, for accepting my abstract and paper and to Cynthia Zender (SAS), Jerry Leonard (SAS), Kirk Paul Lafler (Software Intelligence Corporation) and Vicki Jeffries (GDIT) for their suggestions and patience in answering my many questions. I would also like to thank my husband, David Rosen and my son, Marc Rosen for their eleventh hour proofreading efforts.

Contact Information

Mira Shapiro mira.shapiro@gmail.com