Paper AA07-2014

Creating an Easy to Use, Dynamic, Flexible Summary Table Macro with P-Values in SAS® for Research Studies

Amy Gravely, Center for Chronic Disease Outcomes Research, A VA HSR&D Center of Innovation, Minneapolis, MN
Barbara Clothier, Center for Chronic Disease Outcomes Research, A VA HSR&D Center of Innovation, Minneapolis, MN
Sean Nugent, Center for Chronic Disease Outcomes Research, A VA HSR&D Center of Innovation, Minneapolis, MN

ABSTRACT

SAS Macros are useful and can save time with routine, repetitive tasks such as creating customized tables in situations when the exact same table is needed repeatedly, but the data changes. However, often flexibility is desired so that the same macro can be used to create similar but not identical tables for different research studies, within the same study or to allow one to quickly accommodate requested changes. This paper presents an easy to use, flexible, dynamic summary table macro that includes p-values. This macro incorporates flexible features that the user specifies through macro parameters: the number and type of row variables, the number of columns, the type of tests one would like p-values from, and the spacing of the table. Currently chi-squared tests, Fisher's exact tests, ANOVA F-tests and t-tests are available and any combination of these can be used. The macro also has dynamic features. The macro automatically provides the column totals, the number of missing values, and assigns SAS formats to the column variable based on the dataset. The macro will also automatically pull in the correct p-value for t-tests depending on whether or not the variances are equal. It is also extremely easy to use. The user simply enters 18 parameter values and a nice looking rtf summary table is produced. Not only can the macro be used immediately for many different types of situations and saves time, but it is also written in a modular format so that one can easily add to it. Users must know the general idea of a SAS macro and how to run one.

INTRODUCTION

MOTIVATION FOR THE MACRO

The impetus for creating this macro came from working on many word tables for journal manuscripts or for data summarization and exploration purposes and having to cut and paste, hand enter or do a lot of coding using ODS statements to get the desired output from raw SAS output. When changes to the table are requested such as such as when the data is modified or when variables are recoded, the raw SAS output has to be recreated and then the word table also has to be re-hand entered. This is not time efficient. A smaller but not inconsequential motivation was that often when giving non-statisticians raw SAS output it is not clear what they should be looking at and often this output is pages and pages long. Even for situations where the summary information is not intended for a manuscript table 1, it is more effective to give someone 1-2 pages of clear output rather than pages and pages of output where they might not know which p-value to look at or which percent they should be focusing on, etc. Additionally, whether multiple changes are requested or not, the use of macros allows for efficient, clean SAS code.

I checked the literature to see if such a macro already exists. First, the macro needed to be easy to use and transparent in that a lot of time could not be spent figuring out what the macro does or how to use it. Second, the macro needed to be downloadable for immediate use. Third, the macro needed to incorporate a lot of flexibility so that it could be applied to a wide variety of summary table situations with the general goal of creating a typical table 1. For example, one study might have a treatment variable with 5 levels whereas another study might have a treatment variable with 3 levels. Additionally, one study might want to see the missing data column and another might not. Fourth, the macro needed to include dynamic features in that anything that could be automated was. For example, if the treatment variable level names are different between studies, it was desired that the macro automatically update the table accordingly without explicit user inputs. One study might have treatment groups that are called Treatment A, Treatment B and Treatment C whereas another study might have treatment groups that are called placebo and active. Although there are many table macros out there, most of the examples I found were either dated, did not provide the flexibility desired, were not easy to use, or could not be downloaded for immediate use. The closest thing that I could find to what I wanted was this article: %SummaryTable: A SAS Macro to Produce a Summary Table in Clinical Trial, by Yinmei Zhou, but the macro could not be located for download and was not exactly what I was looking for. Additionally, this article was written in 2006 which inspired me to investigate newer coding techniques.

THE CURRENT MACRO

This paper illustrates the creation of a flexible, dynamic, easy to use macro that that can create summary tables with p-values. This macro produces a typical "table 1" as found in many journal manuscripts. One example of this is a comparison of demographic variables (dependent variables) across treatment groups (one independent variable) to see if treatment groups are balanced. Another example is a summary table showing how demographic variables vary by your independent variable of interest such as race. Yet another example is comparing responders to non-responders to see if they differ.

The current macro adds to what is out there in that it provides ease of use, flexibility, transparency and dynamic automation all within one macro. Most of the literature on this topic is for producing the exact same table over and over and less literature exists about creating flexible, dynamic macros that can be applied to different situations. While ideas and pieces came from the existing literature, this macro is original with an original approach to the macro organization and includes all of the following combined into one macro that cannot be found in another published macro. First the macro is extremely 1) **easy to use.** The user inputs merely 18 macro parameters that are intuitive and a nice looking rtf table is produced. 2) The macro is **transparent**. **Flexible features specified through macro parameters** include**:** 3) the number and type (categorical or continuous) of row variables (your dependent variables), 4) the order in which to display the row variables, 5) the type of tests that you would like p-values from for each row variable (currently chi-squared tests, Fisher's exact tests, ANOVA F-tests and t-tests are available and any combination can be used), 6) the spacing of the table (specifying how many inches wide you want each column to be which is very intuitive), 7) which columns in the final table you would like to see (current choices are: all, nopvalue, nomiss, justtotal, justcolvargroups, nomtot), 8) if you want to see the variable name, the variable label or both in the first column, 9) if you want to see row percentages or column percentages for your categorical row variables, 10) if you want landscape or portrait orientation, 11) the title of the table and 12) where you would like to store the final table that is produced. 13) Additionally you can create any style through PROC TEMPLATE and enter it in the style parameter, use a pre-existing SAS style, or use the default which is RTF "style". Even if you choose a different style, your final table will still be an rtf "document". **Dynamic Features that are automated based on the data** include: 14) column totals are automatically produced, 15) the number of independent variable levels or treatment groups is automatically calculated and there is no limit, 16) the number of missing values for each variable are automatically produced (listed in the #M column), 17) and the macro will automatically list any categorical format that is needed for the table including categorical row variables and column variable levels. 18) The order in which you enter the variables will automatically be retained so that the variable order in the final table is exactly the order in which you enter the variable names into the varnames, type and tests parameter lists. 19) Additionally, this macro will automatically pull in the correct p-value for t-tests depending on whether or not the variances are equal.

FULL EXPLANATION OF EACH PARAMETER IN THE MACRO

%macro table 1;	
dataset,	The dataset that you want to run the table on, for example: temp.anlaysis.
col,	Your column variable, example: treatmentgroup.
numvar,	The total number of row variables, example: 5 (no limit).
varnames,	**LIST PARAMETER.** The row variable names separated by a space, if you entered 5 in the numvar parameter, than you must list exactly 5 variables here. List these in the order that you would like them to appear in the table.
type,	**LIST PARAMETER.** The type of each variable in varnames separated by a space. MUST BE IN THE SAME ORDER AS VARNAMES. Choices: CATE or CONT.
test,	**LIST PARAMETER.** The test you want for each variable in varnames separated by a space. MUST BE IN THE SAME ORDER AS VARNAMES. Choices: CHISQ, FISHER, ANOVA AND TTEST.
rowcol,	For categorical row variables do you want to see row percents or column percents? Choices: colpercent rowpercent.
orientation,	Do you want landscape or portrait orientation? Choices: portrait or landscape.
filepathname,	Folder where you want your final RTF table to be stored, the author has not experimented with other options other than .rtf, example: Z:\FirstTry\firsttry34.rtf.
title,	Title of your table.
reportcol,	Which columns do you want to see? Choices: all, nopvalue, nomiss, justtotal, justcolvargroups or nomtot.
labelvarname,	In the first column labeled "Variable" do you want to see the label, the variable name or both? Choices: label, varname or both.
style,	Must enter rtf as the default, but you can also enter any style created through PROC TEMPLATE or any already existing SAS style.
These are for indicating how wide in INCHES you want each column in the final outputted RTF table to be so it looks nice on the page.	
variablelength,	How many inches do you want the variable column to be? Start with 1.5.
missinglength,	How many inches do you want the missing column to be? Start with .40.
formatlength,	How many inches do you want your levels column to be (this column lists out the categorical row variable formatted values)?
totallength,	How many inches do you want your total column to be?
collength);	How many inches do you want your treatment levels or grouping variable levels to be?

THE OVERALL APPROACH TO WRITING THE MACRO WITH CODE EXCERPTS

STEP 1: Enter the 18 parameters into the macro.

Note that three of the macro parameters are list parameters: varnames, type and test. The rest of the parameters are single parameters and are processed in the usual way. The list parameters where appropriate make the macro easier to use. Here is the SAS code that reads in the parameter lists:

```
%macro full(variable , type1 ,  test1);
      %local count word cntr cut;
      %let count=1;
      %let word=%qscan(&variable,&count,%str( ));
      %do %while(&word ne);
      %let count=%eval(&count+1);
      %let word=%qscan(&variable,&count,%str( ));
        %end;
      %let count=%eval(&count-1);
        %do cntr = 1 %to &count;
        %let cut = %scan(&variable , &cntr, ' ');

      data new&cntr; length variables $50;  variables="&cut";  run;
      %end;

      data all; set new1 - new&numvar; run;
      %local count word cntr cut;
      %let count=1;
      %let word=%qscan(&type1,&count,%str( ));
      %do %while(&word ne);
      %let count=%eval(&count+1);
      %let word=%qscan(&type1,&count,%str( ));
      %end;
      %let count=%eval(&count-1);
      %do cntr = 1 %to &count;
      %let cut = %scan(&type1 , &cntr, ' ');

      data two&cntr;      length type $50;  type="&cut";  run;
      %end;

      data all2; set two1-two&numvar; run;

      …repeat for test, not shown

      data all5; merge all all2 all3; run;
%mend full;

%full (&varnames,&type, &test); run;
```

STEP 2: Create a dataset containing the list parameters, the order of the list parameters and the formats for each categorical row variable.

number	variable	test	type	catformats
1	BL_DEMO_MARITALSTATUS	FISHER	CATE	MARITAL.
2	SWLS_TOTAL2	ANOVA	CONT	
3	BL_DEMO_INCOME	CHISQ	CATE	INCOME.
4	BL_DEMO_DEGREE	CHISQ	CATE	DEGREE.
5	BL_DEMO_WORKING	CHISQ	CATE	YESNO.

The goal of step 2 is to produce the above table which is named “all9” in the macro if you want to look at it to be sure that you have specified your macro parameters correctly. Putting this type of information into a dataset can make subsequent steps easier since working with a dataset is familiar territory for most statisticians and programmers. Columns 1-4 are created from step1. In order to add the format names for any categorical row variable you can utilize SAS I/O functions. The code below will produce a table full of metadata such as formats, labels, etc. which can then be merged into the dataset above. The formats can then be made into macro variables which allow the formats to be displayed in the final table and it also allows for stacking of variables to produce the final table.

Here is some example I/O code:

```
ods output variables=varinfo;
proc contents data=&dataset;
run;
```

The flexibility and dynamic nature of the macro is enhanced by the fact that the macro automatically keeps track of the order in which you entered the categorical variable names into the varnames parameter. One does not have to keep all categorical variables together or all ANOVA test variables together, etc.

STEP 3: Process (loop) through the table created in step 2 one row at a time, sending each variable and format to the appropriate statistical macro (chisq, fisher, anova or ttest).

The macro will step through the table one row at a time. For each row, if it finds a categorical variable with a chi-square test requested it will send the variable name (as a macro variable) and format (as a macro variable) to the chi-square macro. If it finds a categorical variable with a fisher’s exact test requested it will send the variable name (as a macro variable) and format (as a macro variable) to the fisher macro, etc. and then move to the next row and do the same.

Making the variable names and formats into macro variables looks something like this:

```
data _null_;
set all9 end=last;
call symputx('V'||left(_n_), variable);
call symputx('Test'||left(_n_), test);
call symputx('Type'||left(_n_), type);
call symputx('Format'||left(_n_), catformats);
if last then call symputx('Num', _n_);
%put &Num;
run;
```

After each row is processed, a table like the following will be produced for each variable:

variable	numiss	levels	freqperc	Expressive_ writing	Control_ writing	Non_ writing	prob2
BL_DEMO_MARITALSTATUS	0	Never married/single	10 (20)	6 (30)	0 (0)	4 (30.77)	0.0926
		Married/partnered	32 (64)	11 (55)	14 (82.35)	7 (53.85)	
		Divorced/separated	8 (16)	3 (15)	3 (17.65)	2 (15.38)	

Note: Inside the separate macros for CHISQ, FISHER, ANOVA and TTEST you will find the obvious PROCs and ODS OUTPUT statements. Each of these macros is completely self-sufficient (I describe this as being written in modular format) in the event that someone only wants t-tests or only wants chi-squared tests. This makes it easy if you want to add a test of your own as well. A bi-product of making each of these sub-macros within the larger table macro self-sufficient is that one has to include the same 4 global macro variables within each so that they can be used later in the PROC REPORT code. These 4 variables are: gtotaln, gvarlist10, gcoltot and gtotal which store the total N for the sample, the column level format names (which is technically stored as a macro list), the column level totals and the number of column levels respectively. These are dynamic features of the macro. The macro can adjust to any inputs and update the table accordingly.

One technique used to create these global macro variables is to utilize SAS dictionary tables and views which allow you to make macro variable lists out of column names (gvarlist10) and count the number of column levels or treatment groups (gtotal).

```
data vars;
set sashelp.vcolumn;
where memname = 'WIDECONT1T';
keep varnum name; if varnum>2;
run;

data vars2; set vars;
varnum2=varnum-2; drop varnum;
run;

data _null_;
set vars2 end=last;
call symput ('gcolumn'||left(put(_n_, best.)), name);
if last then do;
call symput('total', left(put(_n_, best.)));

%do j = 1 %to &total;
%put &&gcolumn&j;
%global gtotal;
%let gtotal=&total;
%put &gtotal;
%end;

data _null_;
length allvars $1000;
retain allvars ' ';
do i = 1 to &gtotal;
set vars2 ;
if name ^= ' ' then do;
allvars = trim(left(allvars))||' '||left(name);
call symput('varlist10', allvars);
end;

end;
run;
end;
run;

%global gvarlist10;
%let gvarlist10=&varlist10;
%put &gvarlist10;
```

STEP 4: Stack the resulting datasets created for each variable and run PROC REPORT to create the final RTF table.

After the macro loops through each row in the table listed in step 2 and sends each variable and format to the appropriate macro to get it's p-value (chisq, fisher, anova or ttest) and outputs a table for each variable as shown above in step 4, all of these are stacked in order so you have a dataset that looks exactly how you want the final table to look. Finally, PROC REPORT is used to make an RTF file from this final stacked table which allows one to add a title, headers, and more with a nice format. This step tends to be different than most other table 1 macros in that most of them use PROC REPORT at an earlier step.

RULES TO USING THIS MACRO

In order for the macro to work correctly, the following rules must be adhered to:

- Every variable (categorical or continuous) must be numeric. This macro does not work with character variables.
- Every categorical row variable must have a SAS format attached.
- The column variable levels must be coded as 1, 2, 3, and so on and it is best if this variable has a SAS format attached. Therefore even though your column variable may initially be something like intervention vs.

usual care, coded as 1 and 0, as is often the case with two arm RCTs, you will need to recode this to 1 and 2 in order to use this macro. There is not an explicit limit to the number of columns you can have (or the number of levels to your treatment variable).

- The formatted names that go with the levels of your column variable will be displayed at the top of the final table and cannot exceed 32 characters.
- You can run any combination of chi-squared tests, Fisher's exact tests, ANOVAS and t-tests including all of just one of them, but keep in mind that you would never want to run both an ANOVA and a t-test within the same table – that wouldn't make sense since if your column variable has 3 or more levels you would run an ANOVA and if your column variable has two levels you would run a t-test.
- So as to avoid picking a dataset name that is also inside the macro just create a temp dataset called analysis and run the macro on that dataset.
- Your column variable (whose levels appear at the top of the table) must be categorical with at least 2 categories.
- The total number of row variables parameter (numvar) must match the number of variables you enter into varnames.
- The order that you enter variables into the varnames parameter corresponds to the order you fill out type and test. Putting these in the same order is imperative.
- For the style parameter you must enter rtf (which I am considering to be the default). If you would like to use a different template style either provided by SAS or created by you, simply change this from rtf to whatever your style is named. Note that no matter which style you choose, this macro produces an rtf document, not to be confused with rtf style.

Notes:

- You probably don't want to run a Fisher's exact test on a large dataset.
- Only varnames, type and test are list parameters, the rest are single parameter inputs.
- For the t-test only, the *p*-value will automatically be pooled if the variances are equal and satterthwaite if the variances are not equal.
- Reportcol has 6 choices: all, nopvalue, nomiss, justtotal, justcolvargroups, nomtot
- Labelvarname has 3 choices: label, varname or both
- The orientation, rowcol, test and type parameters also have choices as listed on page 3.
- Variablelength, missinglength, formatlength, totallength and collength are for the report itself. How wide do you want each of these columns to be in INCHES? This can used to make sure that your whole variable name stays on one line or to space the table out to look nicer on a page as each table will be different. The goal of this macro is to not have to do anything else to the table after you run it.

EXAMPLES OF INPUTS AND OUTPUTTED TABLE

EXAMPLE CODE:

```
%table1 (see, treatmentgroupn, 5,
bl_demo_maritalstatus swls_total2 bl_demo_income bl_demo_degree bl_demo_working,
cate cont cate cate cate,
fisher anova chisq chisq chisq,
colpercent,
landscape,
Z:\FirstTry\firsttry34.rtf,
TABLE 1:  Demographics by Treatment (count, column %) Unless Otherwise Noted,
all,
varname,
rtf,
2.1,
.40,
2.0,
1.0,
1.0)
```

PRODUCES THIS TABLE:

TABLE 1: Demographics by Treatment (count, column %) Unless Otherwise Noted

Variable	#M		Total N=50	Expressive_ writing N=20	Control_ writing N=17	Non_ writing N=13	P-Value
BL_DEMO_MARITALSTATUS	0	Never married/single	10 (20)	6 (30)	0 (0)	4 (30.77)	0.0926
	.	Married/partnered	32 (64)	11 (55)	14 (82.35)	7 (53.85)	.
	.	Divorced/separated	8 (16)	3 (15)	3 (17.65)	2 (15.38)	.
SWLS_TOTAL2	0	Mean	22.02	23.05	21.82	20.69	0.6878
	.	Std Dev	7.61	8.21	7.69	6.85	.
	.	Median	22.50	26.00	23.00	21.00	.
	.	Minimum	8.00	8.00	10.00	8.00	.
	.	Maximum	35.00	35.00	31.00	31.00	.
BL_DEMO_INCOME	0	$10,000 to $20,000	7 (14)	3 (15)	0 (0)	4 (30.77)	0.0909
	.	$20,000 to $40,000	12 (24)	6 (30)	5 (29.41)	1 (7.69)	.
	.	$40,000 to $60,000	9 (18)	1 (5)	6 (35.29)	2 (15.38)	.
	.	More than $60,000	18 (36)	9 (45)	5 (29.41)	4 (30.77)	.
	.	I prefer not to answer	4 (8)	1 (5)	1 (5.88)	2 (15.38)	.
BL_DEMO_DEGREE	0	GED or High School diploma	4 (8)	2 (10)	2 (11.76)	0 (0)	0.3538
	.	Some college or post high school training	20 (40)	10 (50)	4 (23.53)	6 (46.15)	.
	.	College diploma	13 (26)	4 (20)	4 (23.53)	5 (38.46)	.
	.	Advanced degree	13 (26)	4 (20)	7 (41.18)	2 (15.38)	.
BL_DEMO_WORKING	0	No	9 (18)	4 (20)	1 (5.88)	4 (30.77)	0.2038
	.	Yes	41 (82)	16 (80)	16 (94.12)	9 (69.23)	.

Data shown from a VA study used with permission, a subset of data is shown.

CONCLUSION

In conclusion, this macro is available for immediate use allowing users from many domains to increase efficiency. It saves a huge amount of time when producing summary tables while ensuring accuracy, code efficiency and creating user friendly output. A few applications that come to mind in addition to the research statistician are the statistical consultant charging by the hour or the business analyst who needs to produce summary tables for decisions in real-time. The user will not need to spend a lot of time figuring out how to input the macro parameters or how the macro works as the macro is easy to use and the approach to the coding of the macro is transparent. One could add to the macro, replicate this approach for another macro, utilize some of the techniques described, or simply use the macro. Flexibility is highlighted by its many choices that one can decide upon when entering the parameter values. Dynamic features include automation based on the data without any inputs from the user. Some techniques used to achieve these desirable characteristics were highlighted such as using list parameters, utilizing SAS I/O functions and SAS dictionary tables and views. This macro is a work in progress. Next steps might entail: adding a nonparametric comparison test for non-normally distributed continuous variables or allowing choices for the display of ANOVA and TTESTs where one viewing option is mean (sd) and the other option is the way it is what is currently displayed above with Mean and Std Dev listed on separate rows. One could create several styles through PROC TEMPATE, perhaps including highlighting and then easily and quickly plug them into the style macro parameter. Additionally, one could expand this macro to accommodate domain analysis situations when appropriate.

REFERENCES:

Carpenter A (2005), Storing and Using a List of Values in a Macro Variable, *SUGI 30,* Paper 028-30.

Carpenter A (1997), Resolving and Using &&var&i Macro Variables, Proceedings of the Twenty-Two Annual SAS Users Group International Conference.

Lafler K (2005), Exploring DICTIONARY Tables and Views, *SUGI 30,* Paper 070-30.

Lewandowski D (2008), A Step-by –Step Introduction to PROC REPORT, *SAS Global Forum,* Paper 079-2008.

Sevick C (2006), "Table 1" in Scientific Manuscripts; Using PROC REPORT and the ODS System. *WUSS.*

Suwen L, Ma S, Li R, Lan B (2013), Using CALL SYMPUT to Generate Dynamic Columns in Reports, *SAS Global Forum,* Paper 198-2013.

Zhou Y, Zhang L, Hancock M (2006), %Summary Table: A SAS Macro to Produce a Summary Table in Clinical Trial., *Pharmaceutical Industry SAS Users Group,* Paper AD13.

ACKNOWLEDGEMENTS:

This material is the result of work supported with resources and the use of facilities at the Minneapolis VA Health Care System.

RECOMMENDED READING:

SAS® Macro Language 1: Essentials,
SAS® Macro Language 2: Advanced Techniques

CONTACT INFORMATION:

Comments, questions and requests for the macro are welcomed and greatly appreciated:

Amy Gravely
Center for Chronic Disease Outcomes Research, A VA HSR&D Center of Innovation
Minneapolis VA Health Care System
Building 9, Mail code 152
5445 Minnehaha Ave. South
Minneapolis, MN 55417
amy.gravely@va.gov