PharmaSUG2012 – Paper CC18

One Macro Call to a Table with both Frequency and Summary Elements from a Subject-Level Dataset

Wayne Zhong, Octagon Research Solutions Inc.

ABSTRACT

A Subject-Level dataset such as ADSL has one observation per subject with data appropriate to the format, such as demographics, baseline characteristics, certain disposition events, and other one record per subject numeric or character data. Tables summarizing this data corresponding have both frequency and summary statistics sections, and normally the tables are produced in individual sections and assembled together. The code to do each section is not complex, which is why this table is a good candidate for a macro to produce.

This paper presents the full code of the macro, built with simplicity of the macro call in mind. The call requires only 3 parameters: the input dataset, a variable used to define table columns, and a list of mixed numeric/character variables, one for each section of the table. Adding optional features allow full control over the table presentation, with the end goal of making the program header the lengthiest section of the program.

INTRODUCTION

Suppose a SAS® program is written to summarize the information in the dataset in Display 1. The variable STAGE will determine column placement, and the remaining variables will be summarized with summary stats or frequencies as appropriate. Although not displayed, every record represents a unique person so the number of records represent an accurate population count for the purpose of percent calculation.

adsl.sas7bdat

	Stage at Diagnosis (STAGE)	Sex (SEX)	Age (AGE)	Age Category (AGECAT)	Discontinuation Reason (DSREASC)
1	IV	M	55	<65 Years	DISEASE PROGRESSION
2	III	F	63	<65 Years	DISEASE PROGRESSION
3	III	F	71	>=65 Years	DISEASE PROGRESSION
4	III	F	68	>=65 Years	INTERCURRENT ILLNESS
5	III	F	66	>=65 Years	DISEASE PROGRESSION
6	II	F	59	<65 Years	DISEASE PROGRESSION
7	IV	M	56	<65 Years	DISEASE PROGRESSION
8	III	M	53	<65 Years	DISEASE PROGRESSION
9	III	F	65	>=65 Years	DISEASE PROGRESSION
10	IV	F	71	>=65 Years	INTERCURRENT ILLNESS

Display 1

Even without using any macros, this program would be simple if a bit long and repetitive. If single function macros exist to produce summary and frequency sections, then the program reduces to macro calls for each section and some code to combine them. The natural next step is to condense this process into one macro and one call.

The following presents one approach that emphasizes ease of use. The code has been simplified for presentation purposes, some features and all error checking code were removed to make the code more transparent. This means the code is ready for use, but will crash if incorrect parameters are given and may overwrite existing global macro variables.

DESIGN INPUTS

Designing a macro for ease of use starts with the input parameters. There is no one way that is universally best, and words like intuitive and functional are completely subjective. The best approach is to create a draft design and prepare lots of pennies for people's thoughts. The following macro call generates the dataset shown in Display 2. The author welcomes comments and suggestions.

```
%wide(indsn=adsl,
      cols= stage="II":"III":"IV":"II" "III" "IV",
      rows= age#"Subject Age, years"/
            sex="M":"F":"U"#"Subject Sex"/
            agecat/
            dsreasc#"Symbol test (fs)(pd)(eq)");
```

VIEWTABLE: Work.Wide

	desc	_1	_2	_3	_4	order
1	Subject Age, years					1
2	n	1	6	3	10	1
3	Mean (SD)	59.0 (NA)	64.3 (6.19)	60.7 (8.96)	62.7 (6.62)	1
4	Median	59.0	65.5	56.0	64.0	1
5	Min, Max	59.0, 59.0	53.0, 71.0	55.0, 71.0	53.0, 71.0	1
6	Subject Sex					2
7	M	0	1 (16.7%)	2 (66.7%)	3 (30.0%)	2
8	F	1 (100.0%)	5 (83.3%)	1 (33.3%)	7 (70.0%)	2
9	U	0	0	0	0	2
10	Age Category					3
11	<65 Years	1 (100.0%)	2 (33.3%)	2 (66.7%)	5 (50.0%)	3
12	>=65 Years	0	4 (66.7%)	1 (33.3%)	5 (50.0%)	3
13	Symbol test /#=					4
14	DISEASE PROGRESSION	1 (100.0%)	5 (83.3%)	2 (66.7%)	8 (80.0%)	4
15	INTERCURRENT ILLNESS	0	1 (16.7%)	1 (33.3%)	2 (20.0%)	4

Display 2

The arrangement of columns is specified with the **cols=** parameter using the variable name, an **=** sign, and the variable values to take for each column delimited by **:**. More than one value may be specified for each column, for example the fourth column combines the first three.

The **rows=** parameter is used to specify each section in the table, delimited by ***/***. For each section, the first word is the variable name to use, this may be followed by an **=** or **#** sign. Whether a summary stat or frequency section is produced is automatically determined using the variable type. An **=** sign may be used to list the order of values in a frequency section, otherwise alphabetical order is used. A **#** sign may be used to specify the section label, otherwise the variable label is used.

In case a delimiter is needed as text, a set of replacement values, for example (fs) (pd) (eq), is recognized and processed appropriately by the macro.

MACRO – METADATA & PRE-PROCESSING

The following sections cover the entire macro code with detailed comments.

```
%macro wide(indsn=, outdsn=wide,
            cols=,
            rows=);

%local colvar collist rowvar rowlist i ncol label2 loop;
```

The variable type (N or C) and label is needed in many parts of code so embedding a helper macro allows this macro to be self contained. In some environments global macros may already exist with the same functions. This macro places the type and label of the variable specified in global macro variables.

```
%macro meta(var);
%global type label;
%local num dsid rc;
%let dsid=%sysfunc(open(&indsn,i));
%let num=%sysfunc(varnum(&dsid,&var));
%let type=%sysfunc(vartype(&dsid,&num));
%let label=%sysfunc(varlabel(&dsid,&num));
%let rc=%sysfunc(close(&dsid));
%mend meta;
```

The first step is to process the columns specified into an easier format. The approach used here is to create a temporary dataset TEMP1 with new variable _COL, containing the numbers 1 through 4 for each column. Note that both character and numeric column variables can be used by providing quoted or numeric values. A loop through the list provided and the OUTPUT statement creates _COL values, at the same time the total number of columns is stored in &ncol.

```
%let colvar=%scan(&cols,1,=);
%let collist=%scan(&cols,2,=);
%let i=1;
data temp1;
    set &indsn;
    %do %while (%length(%scan(&collist,&i,:))>0);
        if &colvar in (%scan(&collist,&i,:)) then
            do;
                _col=&i;
                output;
            end;
        %let ncol=&i;
        %let i=%eval(&i+1);
    %end;
run;
```

Another step that can be done now is create population counts for each column using _COL and &ncol.

```
proc sql noprint;
    %do i=1 %to &ncol;
        %global n&i;
        select count(_col) into :n&i from temp1(keep=_col where=(_col=&i));
    %end;
quit;
```

MACRO – SUMMARY STATS HELPER MACRO

Taking advantage of the column variable _COL in temporary dataset TEMP1, it takes a MEANS procedure to generate the statistics, a DATA step to process the numbers into the desired format, and a TRANSPOSE procedure to get columns in order. The labels added during the DATA step become values of the _LABEL_ variable in the transposed dataset, and a rename to DESC carries them to the output dataset.

```
%macro sum(var);
proc means data=temp1(keep=_col &var) noprint nway;
    class _col;
    var &var;
    output out=temps1 n=n median=median mean=mean std=sd min=min max=max;
run;

data temps2;
    length n1 msd m1 mm $100 ;
    set temps1;
    if mean ne . and sd ne . then msd=strip(put(mean, 12.1))||
                                  " ("||strip(put(sd,12.2))||")";
    else if mean ne . and sd=. then msd=strip(put(mean, 12.1))||" (NA)";
    if min ne . then mm=strip(put(min,12.1))||", "||strip(put(max,12.1));
    if n ne . then n1=strip(put(n,best12.));
    if median ne . then m1=strip(put(median, 12.1));
    keep _col n1 msd m1 mm;
    label N1='n' MSD='Mean (SD)' M1='Median' MM='Min, Max';
run;

proc transpose data=temps2 out=temp2(drop=_name_ rename=(_label_=desc));
    id  col;
    var n1 msd m1 mm;
run;
%mend sum;
```

MACRO – FREQUENCY HELPER MACRO

The first portion of the frequency macro is simple, a FREQ procedure captures the counts, a SQL procedure creates character variables and sorts by DESC, subsequently a TRANPOSE procedure creates columns using _COL by DESC order.

```
%macro freq(var, list);
%local i;
proc freq data=temp1(keep=_col &var where=(&var ne '')) noprint;
    table _col*&var/out=tempf1;
run;

proc sql noprint;
    create table tempf2 as
        select _col, &var length=100 as desc,
            strip(put(count,best12.)) length=100 as freq
            from tempf1 order by desc;
quit;

proc transpose data=tempf2 out=tempf3(drop=_name_);
    var freq;
    by desc;
    id _col;
run;
```

Processing a list of values supplied with the character variable adds complexity to the macro. A shell dataset with values provided in the list is first generated with a loop, taking care to process any replacement strings used for delimiters. A SQL LEFT JOIN merge keeps both the order and values specified in the shell. Some variable juggling allows the usage of **b.*** since only values from the shell dataset is wanted in variable DESC.

```
%if %length(&list)>0 %then %do;
    data tempshell;
        length desc $100;
        %let i=1;
        %do %while (%length(%scan(&list,&i,:))>0);
            desc=%scan(&list,&i,:);
            desc=tranwrd(tranwrd(tranwrd(desc,'(fs)','/'),'(pd)','#'),'(eq)','=');
            c=&i;
            output;
            %let i=%eval(&i+1);
        %end;
    run;

    proc sql noprint;
        create table tempf4(drop=desc1 c) as
            select a.desc, b.*, a.c from
            tempshell a left join tempf3(rename=(desc=desc1)) b on a.desc=b.desc1
            order by c;
    quit;
%end;
%else %do;
    data tempf4;
        set tempf3;
    run;
%end;
```

Finally, a simple %DO loop uses the population counts generated earlier to calculate percents for each column or in case of missings fill in 0s.

```
data temp2;
    set tempf4;
    %do i=1 %to &ncol;
        if _&i='' then _&i='0';
```

```
        else  &i=strip( &i)||' ('||
                 strip(put(input(_&i,best12.)/&&n&i*100,8.1))||'%)';
    %end;
run;
%mend freq;
```

MACRO – PUT TOGETHER FREQ & SUM SECTIONS

Finally, the **rows=** parameter is ready to be processed. As might be expected by now, the delimiters used demand a loop to generate each section but first an empty dataset is prepared to catch each section generated.

```
data &outdsn;
    stop;
run;

%let i=1;
%do %while (%length(%scan(&rows,&i,/))>0);
    %let loop=%scan(&rows,&i,/);
```

Breaking down each section further is necessary to call the helper macros. The %meta call gives the variable type and label. The label is used if none is specified for the section.

```
    %let rowvar=%scan(&loop,1,=#);
    %let rowlist=%scan(%scan(&loop,2,=),1,#);
    %meta(&rowvar);

    %let label2=%scan(&loop,2,#);
    %if %length(&label2)=0 %then %let label2="&label";
```

Depending on the variable type, the appropriate helper macro is called. The label for the section is placed in one observation dataset TEMPL.

```
    %if &type=N %then %sum(&rowvar);
    %else %freq(&rowvar,&rowlist);

    data templ;
        desc=&label2;
    run;
```

Both TEMPL and the dataset produced by the helper macro call (always called TEMP2) are added to the final output dataset. The index for the loop is used for the order, indention of the section content is added, and text replacement for delimiter symbols is processed.

```
    data &outdsn;
        length desc _1-_&ncol $100;
        set &outdsn templ(in=a) temp2(in=b);
        if a or b then order=&i;
        if b then desc='     '||desc;
        desc=tranwrd(tranwrd(tranwrd(desc,'(fs)','/'),'(pd)','#'),'(eq)','=');
    run;

    %let i=%eval(&i+1);
%end;
```

Deleting temporary datasets is made easy by specifying the TEMP prefix.

```
proc datasets library=work;
    delete temp:;
quit;

%mend wide;
```

ONE MORE THING

While the macro provided will generate an output dataset, it feels like it is missing one more step, printing an output file. This feature is one of the removed because output processing differ greatly by environment. While the code is can't be provided, the idea can be. It will mean adding a parameter and – since we are all experts at loops by now – a delimited list, for example:

```
print= %nrstr(" ": "Type II|(N=&n1)":
                   "Type III|(N=&n2)":
                   "Type IV|(N=&n3)":
                   "Total|(N=&n4)")
```

Macro quoting with %nrstr is necessary because the macro variables' values are generated by the macro. Now suppose some columns need to be grouped together, another parameter could be added:

```
advprint= desc ("Group Header" _1 _2) _3 _4
```

What about setting decimal places, setting number of lines per page, not calculating frequency percents, changing the indent amount, or any number of other options that might be needed? Adding features is easy, knowing when to stop isn't.

CONTACT INFORMATION:

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

Author Name:	Wayne Zhong
Company:	Octagon Research Solutions Inc.
Address:	585 East Swedesford Road, Wayne, PA
Work Phone:	(610) 535-6500 x5535
Email:	wzhong@octagonresearch.com
Web:	www.octagonresearch.com

APPENDIX

```
%macro wide(indsn=, outdsn=wide,
            cols=,
            rows=);

%local colvar collist rowvar rowlist i ncol label2 loop;

/* MACRO: give variable metadata of &var */
%macro meta(var);
%global type label;
%local num dsid rc;
%let dsid=%sysfunc(open(&indsn,i));
%let num=%sysfunc(varnum(&dsid,&var));
%let type=%sysfunc(vartype(&dsid,&num));
%let label=%sysfunc(varlabel(&dsid,&num));
%let rc=%sysfunc(close(&dsid));
%mend meta;

%let colvar=%scan(&cols,1,=);
%let collist=%scan(&cols,2,=);

/* create new column variable _COL */
%let i=1;
data temp1;
    set &indsn;
    %do %while (%length(%scan(&collist,&i,:))>0);
        if &colvar in (%scan(&collist,&i,:)) then
            do;
                _col=&i;
                output;
            end;
        %let ncol=&i;
        %let i=%eval(&i+1);
    %end;
run;

/* population counts */
proc sql noprint;
    %do i=1 %to &ncol;
        %global n&i;
        select count(_col) into :n&i from temp1(keep=_col where=(_col=&i));
    %end;
quit;

/* MACRO: create summary stats from &var by _COL */
%macro sum(var);
proc means data=temp1(keep=_col &var) noprint nway;
    class _col;
    var &var;
    output out=temps1 n=n median=median mean=mean std=sd min=min max=max;
run;

data temps2;
    length n1 msd m1 mm $100 ;
    set temps1;
    if mean ne . and sd ne . then msd=strip(put(mean, 12.1))||"
("||strip(put(sd,12.2))||")";
    else if mean ne . and sd=. then msd=strip(put(mean, 12.1))||" (NA)";
    if min ne . then mm=strip(put(min,12.1))||", "||strip(put(max,12.1));
    if n ne . then n1=strip(put(n,best12.));
    if median ne . then m1=strip(put(median, 12.1));
    keep _col n1 msd m1 mm;
```

```
    label N1='n' MSD='Mean (SD)' M1='Median' MM='Min, Max';
run;

proc transpose data=temps2 out=temp2(drop=_name_ rename=(_label_=desc));
    id _col;
    var n1 msd m1 mm;
run;
%mend sum;

/* MACRO: create freq stats from &var by &list and _COL */
%macro freq(var, list);
%local i;
proc freq data=temp1(keep=_col &var where=(&var ne '')) noprint;
    table _col*&var/out=tempf1;
run;

proc sql noprint;
    create table tempf2 as
        select _col, &var length=100 as desc,
            strip(put(count,best12.)) length=100 as freq
            from tempf1 order by desc;
quit;

proc transpose data=tempf2 out=tempf3(drop=_name_);
    var freq;
    by desc;
    id _col;
run;

%if %length(&list)>0 %then %do;
    data tempshell;
        length desc $100;
        %let i=1;
        %do %while (%length(%scan(&list,&i,:))>0);
            desc=%scan(&list,&i,:);
            desc=tranwrd(tranwrd(tranwrd(desc,'(fs)','/'),'(pd)','#'),'(eq)','=');
            c=&i;
            output;
            %let i=%eval(&i+1);
        %end;
    run;

    proc sql noprint;
        create table tempf4(drop=desc1 c) as
            select a.desc, b.*, a.c from
            tempshell a left join tempf3(rename=(desc=desc1)) b on a.desc=b.desc1
            order by c;
    quit;
%end;
%else %do;
    data tempf4;
        set tempf3;
    run;
%end;

data temp2;
    set tempf4;
    %do i=1 %to &ncol;
        if  &i='' then  &i='0';
        else _&i=strip(_&i)||' ('||strip(put(input(_&i,best12.)/&&n&i*100,8.1))||'%)';
    %end;
run;
%mend freq;
```

```
/* empty dataset to build on */
data &outdsn;
    stop;
run;

/* loop through variables in &rows */

%let i=1;
%do %while (%length(%scan(&rows,&i,/))>0);
    /* &loop holds one variable for one section */
    %let loop=%scan(&rows,&i,/);

    /* break &loop down into variable and list */
    %let rowvar=%scan(&loop,1,=#);
    %let rowlist=%scan(%scan(&loop,2,=),1,#);
    %meta(&rowvar);

    /* see if label for section is specified, if not use variable label */
    %let label2=%scan(&loop,2,#);
    %if %length(&label2)=0 %then %let label2="&label";

    /* call summary stats of freq stats depending on vartype */
    %if &type=N %then %sum(&rowvar);
    %else %freq(&rowvar,&rowlist);

    /* temp section label dataset */
    data templ;
        desc=&label2;
    run;

    /* append sections one by one, indent section and convert delimiter replacements
*/
    data &outdsn;
        length desc _1-_&ncol $100;
        set &outdsn templ(in=a) temp2(in=b);
        if a or b then order=&i;
        if b then desc='     '||desc;
        desc=tranwrd(tranwrd(tranwrd(desc,'(fs)','/'),'(pd)','#'),'(eq)','=');
    run;

    %let i=%eval(&i+1);
%end;

/* delete temp datasets */
proc datasets library=work;
    delete temp:;
quit;


%mend wide;
```