Paper PO10

Chic FREQ and UNIVARIATE

Jasmin Fredette, ICON Clinical Research

ABSTRACT

There are many standard and often-used SAS® procedures for reporting results from clinical trials. The aim of the efficient SAS programmer is to be able to macroize the programs that generate regular tables, thus reducing rework and repetitive programming. This paper presents two flexible and dynamic macros utilizing the PROC FREQ and PROC UNIVARIATE procedures. These macros allow the user to produce commonly used statistical output, including flexible control over formatting and output options. Intended audience – Beginner to Intermediate.

INTRODUCTION

The content of most tables generated for clinical trials largely results from two basic SAS procedures: PROC FREQ and PROC UNIVARIATE. However, due to the nature of the data and because of preferences in terms of reporting styles and formatting, these two procedures need to be called differently using the various options available, and the output often has to be handled differently based on the specifications of each project.

The intent of this paper is to introduce two flexible macro procedures that allow you to specify the required details in terms of reporting and formatting while also getting comprehensive and easily readable output in an efficient way. The user can either dynamically specify commands or simply leave the most common default macro variable values to produce a table satisfying usual reporting conventions. The procedures we present can also be used with alignment macro procedures so that the numbers are consistently and legibly reported throughout the tables. Since statistical tests generated by these two procedures require more programming steps and are project specific, they can be inserted separately between the macro calls should the specifications request to present specific p-values.

The first macro procedure we present focuses on PROC FREQ and allows you to present in different parts of the same output the percentages calculated using a denominator either from the overall population or from a subpopulation. It also allows the user to produce a total line, adjust the labels, display the percentages or not, customize the formatting, etc.

The second macro procedure facilitates the production of output containing formatted results from PROC UNIVARIATE. Several reporting patterns are suggested and fit almost all possible presentation styles and allow you to specify appropriate formats imposed by the data to be reported. The labels can obviously be customized, and indentation can be controlled as well.

COMMON BASIS

First of all, both the *_FREQ_* and *_UNIV_* macros need to be fed with the parameters '*var*', the variable of interest on which the UNIVARIATE or the FREQ procedure is to be performed, and '*inds*', the input dataset to be used. The '*var*' parameter can be either numeric or character for the *_FREQ_* macro, but it obviously has to be numeric for the *_UNIV_* macro. The only constraint to the entry dataset is to have a variable called '*col*' indicating the column to which each record belongs. Values for this variable must be positive integers. The first information the macros look for is the number of columns to be reported. If the parameter '*colnb*' is not specified, the macro automatically finds the number of distinct values within the '*col*' variable from the '*inds*' input dataset. If the entry dataset does not include records for at least all columns, '*colnb*' must be specified so all the columns show up in the output dataset. We define the macro variable '*NBCOL*' with the following code:

```
%global NBCOL;
 PROC SQL noprint;
    select compress(put(count(distinct col),best.)) into: NBCOL
    from &inds
    where col ge 1;
 quit;
 %if %UPCASE("&colnb") ne "NONE" %then %do; %let NBCOL=&colnb; %end;
```

Another feature from both macros is the '*block*' parameter that can be used to tag the output dataset. The value assigned to this parameter becomes a variable in the output dataset and can be used for sorting purposes when gathering separate parts of the final output.

THE *_FREQ_* MACRO

Since we often need to calculate frequencies and percentages on subsets of data with flexibility, two types of where statements are provided with the *_FREQ_* macro. The '*wh_bef*' allows users to compute the percentages before subsetting, and the '*wh_aft*' indicates percentage calculations performed after subsetting. Both can be used within the same macro call.

In *Table 1.1* below, the first macro call includes gender as the variable of interest and outputs the first two lines. No where statement is used, since the denominators for the percentage calculations are the total numbers for the entire population included in the entry dataset. For the subcategory presented below the frequencies per gender, we are interested in the percentages of females with Menstrual Irregularities. This requires the usage of the '*wh_aft*' parameter, which first subsets on female subjects and then calculates the percentages, with the total number of female subjects being the denominator. The third information we're interested in is the percentages of females experiencing irregularities by age category. We use '*wh_aft*' to subset on women and '*wh_bef*' to subset on menstrual irregularities. This excludes women without irregularities from the count, but they are still being part of the denominator while performing the percentage calculations.

Table 1.1

	(N=10)	(N=100)
Male	1 (10.0%)	90 (90.0%)
Female	9 (90.0%)	10 (10.0%)
Female	9	10
Menstrual Irregularities	6 (66.7%)	8 (80.0%)
Less than 40 years old	3 (33.3%)	8 (80.0%)
More than 40 years old	3 (33.3%)	0 (0.0%)

As we see on the line just above Menstrual Irregularities in *Table 1.1*, it is common to present a total row for subcategories in order to display the denominators. The '*totline*' parameter from the *_FREQ_* macro allows for a total line when changing the parameter from its default value. When '*totline*' is not specified no totals are displayed.

Most of the time, when the variable of interest is numeric, a specific format needs to be applied so that the rows are properly identified. This can be handled by specifying the format using the '*fmt*' parameter. A label variable is then created to reflect the desired layout.

From time to time percentages aren't needed, so we added a specific option not to present percentages. Changing the default value of parameter '*pct*' will remove percentages from the output dataset.

THE *_UNIV_* MACRO

The presentation of UNIVARIATE results is usually quite straight–forward, although the structure and the statistic displayed can differ from one project to the other. With the parameter '*pres_typ*', you can chose amongst two of the most common presentation styles used in the pharmaceutical industry. The two different layouts that can be assigned to '*pres_typ*' is provided in Appendix I. Although the two styles we propose may not cover entirely the specific presentation requirement, it is very easy to modify with only a basic knowledge of macro programming.

To present the results from PROC UNIVARIATE, we adhere to the following conventions: It is well accepted to report statistics such as minimum, maximum and quartiles with the same number of decimals seen in the raw data. Since the mean and the median are calculated, they are more often presented with one more decimal. The standard deviation and the standard error are more sensitive, and are commonly displayed with two more decimals than the values from the raw data.

Since the level of precision is not always the same for all records of the variables susceptible to be passed to *_UNIV_* , the macro expects the user to provide a value for '*fmt'*, the format which reflects the level of precision usually displayed for the variable of

interest. The macro will then assign the appropriate format to the various statistics as described in the paragraph above. It performs this task and creates macro variables *fmt1* and *fmt2* as follows:

```
DATA _null;
 fmta=scan("&fmt", 1,'.'); fmtb=scan("&fmt", 2,'.');
 if fmtb=0 then fmta=fmta+1;
 fmt1=put(fmta+1, best.) || "." || put(fmtb+1, best.);
 fmt2=put(fmta+2, best.) || "." || put(fmtb+2, best.);
 call symput('fmt1', compress(fmt1));
 call symput('fmt2', compress(fmt2));
run;
```

ALIGNMENT

Alignment is probably the most tedious task when reporting statistics from the UNIVARIATE procedure. To get a better presentation, the alignment based on the decimal point gives a neat layout. To achieve such results, alignment macros designed for the particular type of output are required. The alignment macros we suggest here all start by defining variables $maxus_i$ for column i ($i=1, 2, \ldots, nbcol$), which stores the maximum length before the decimal encountered in the entire column. Depending on the '*pres_typ*' chosen, the corresponding SAS code creating the $maxus_i$ variables could vary slightly,, but follows the pattern below, where *&var* is the dataset name obtained from the PROC TRANSPOSE performed after PROC UNIVARIATE.

```
%do k=1 %to &nbcol;
 PROC SQL noprint;
            select max(length(compress(scan(col&k,1,'.')))) into:  maxus&k
            from &var
            where indexc(col&k,',') = 0;
 quit;
%end;
```

The variable *align* finds how many spaces are needed for every number to be aligned.

```
do i=1 to dim(col);
   align=maxus[i] - length(compress(scan(col[i],1,.')));
end;
```

The results are then patched (padded?) with the appropriate number of blank spaces according to the format of the output file, usually .doc or .rtf.

CONCLUSION

The FREQ and UNIVARIATE procedures from SAS are widely used to produce statistical summaries from clinical trials. However, to follow specific presentation guidelines that can change from one sponsor to another, adjustments have to be made so that the final results are reported as required. We presented here two flexible macros allowing SAS programmers to easily produce reports corresponding to various layout requirements. We tried to cover most of the patterns and requirements we have seen over the past few years, tested different versions of our macros, and noticed great improvement in efficiency. As everybody knows, there is always room for improvement. We may eventually see a reporting pattern using PROC FREQ or PROC UNIVARIATE we did not consider, but we're confident we've covered most of the presentation requirements commonly seen in the pharmaceutical industry.

REFERENCE

SAS Institute Inc. 2007. *SAS Global Forum Paper Presentation Guidelines*, Cary, NC: SAS Institute Inc., 2007.

ACKNOWLEDGMENTS

The author would like to take the opportunity to thank his colleagues Stephen Hunt and David Carr for their encouragement, their review and their advice during the redaction process.

CONTACT INFORMATION

Your comments and questions are value and encouraged. You can contact the author at

Fredettej@iconus.com

Appendix I

Values for the *pres_typ* parameter

pres_typ=1	*pres_typ*=2
N	N
Mean	Mean (SE)
SD	Median
Median	SD
Min, Max	Q1, Q3
	Range

Appendix II

```
%macro _freq_(inds=, var=, block=, fmt=NONE, pct=YES, totline=NO,
                    wh_aft=NONE, wh_bef=NONE, outds=NONE, colnb=NONE, align=YES);
***GET THE TOTAL NUMBER OF COLUMNS ***;
%global NBCOL;
 proc sql noprint;
     select compress(put(count(distinct col),best.)) into: NBCOL
     from &inds
     where col ge 1;
 quit;
 %if %UPCASE("&colnb") ne "NONE" %then %do; %let NBCOL=&colnb; %end;
 ***FREQUENCIES***;
%if %UPCASE("&wh_aft") ne "NONE" %then %do;
  proc freq data=&inds (where=(&wh_aft)) noprint;
            tables col /out=n&var;
  run;
  %if %UPCASE("&wh_bef") ne "NONE" %then %do;
   proc freq data=&inds (where=(&wh_aft. and &wh_bef)) noprint;
            tables col*&var /out=c&var (keep=col &var count rename=(count=val));
   run;
  %end;
  %else %do;
   proc freq data=&inds (where=(&wh_aft)) noprint;
            tables col*&var /out=c&var (keep=col &var count rename=(count=val));
   run;
  %end;
%end;
%else %do;
 proc freq data=&inds noprint;
            tables col /out=n&var;
 run;
 %if %UPCASE("&wh_bef") ne "NONE" %then %do;
  proc freq data=&inds (where=(&wh_bef)) noprint;
            tables col*&var /list missing out=c&var (keep=col &var count
                                                     rename=(count=val));
  run;
 %end;
 %else %do;
  proc freq data=&inds noprint;
            tables col*&var /out=c&var (keep=col &var count rename=(count=val));
  run;
 %end;
%end;
```

```
data &var.frq;
 length cntprc $200;
 merge n&var c&var;
  by col;
 %if %UPCASE("&pct") ne "NO" %then %do;
            prc=(val/count)*100;
            cntprc=put(val, 4.0) || " (" || put(prc, 5.1) || "%)";
 %end;
 %else %do;
            cntprc=put(val, 4.0);
 %end;
run;
proc sort data=&var.frq;
 by &var;
run;
proc transpose data =&var.frq out=&var prefix=col;
 var cntprc;
 id col;
 by &var;
run;
***Check if &var.frq is empty***;
%local NOBSERV;
proc sql;
         select nobs into:NOBSERV
            from dictionary.tables
            where libname='WORK' and (lowcase(memname)="&var.frq" or
                     UPCASE(memname)="&var.FRQ");
quit;
```

```
%if &NOBSERV >= 1 %then %do;
***FORMATING***;
data &var;
 array col{*} $40 col1-col&nbcol;
 length lbl $200;
 set &var;
 block=&block;
 %if %UPCASE("&fmt") ne "NONE" %then %do;
            lbl=" " || put(&var, &fmt..);
            ord=&var;
 %end;
 %else %do;
            lbl=" " || &var;
            ord=_N_;
 %end;
 if compress(_NAME_) in("." "" " ") or compress(lbl) in(" ." " " "" ".")  then do;
   lbl=" Missing"; ord=.;
 end;
 do i=1 to dim(col);
  %if %UPCASE("&pct") ne "NO" %then %do;
            if trim(left(col[i])) in ("" " ") then col[i]="  0 (  0.0%)";
  %end;
  %else %do;
            if trim(left(col[i])) in ("" " ") then col[i]="  0";
  %end;
 end;
run;
*** TOTAL LINE ***;
data n&var;
 set n&var;
 countc=put(count, best.);
run;
proc transpose data =n&var out=tot&var prefix=col;
 var countc;
 id col;
run;
data tot&var;
 array col{*} $40 col1-col&nbcol;
 length lbl $200;
 set tot&var;
 block=&block;
 lbl=" N" ;
 ord=0;
 do i=1 to dim(col);
  %if %UPCASE("&pct") ne "NO" %then %do;
   if trim(left(col[i])) in ("" " ") then col[i]="  0 (  0.0%)";
  %end;
  %else %do;
   if trim(left(col[i])) in ("" " ") then col[i]="  0";
  %end;
 end;
run;
```

```
***RTF ALIGMENT ***;
 %if %UPCASE("&align")="YES" %then %do;
 %do k=1 %to &nbcol;
  proc sql noprint;
                select length(compress(col&k)) into:  maxuse&k
                from tot&var;
  quit;
 %end;
 data &var(drop=i);
  length align $40.;
  set &var;
  array col{*} $40 col1-col&nbcol;
  array maxuse{*} maxuse1-maxuse&nbcol;
  %do j=1 %to &nbcol;
   maxuse[&j]=&&&maxuse&j;
  %end;
  do i=1 to dim(col);
   algn=maxuse[i] - length(compress(scan(col[i],1,')));
   if algn=1 then align="\~"; else if algn=2 then align="\~\~  "; else
if algn=3 then align="\~ \~";
   if algn > 0 then col[i]=trim(left(align))|| trim(left(col[i]));
  end;
 run;
 %end;
%end;
%if %UPCASE("&outds") ne "NONE" and &NOBSERV >= 1 %then %do;
  data &outds;
   set &var;
  run;
%end;
%mend _freq_;
```

Appendix III

```
%macro _univ_(inds=,var=, pres_typ =1, block=, fmt=, colnb=NONE);
 ***SET FORMATS***;
 data _null;
  fmta=scan("&fmt", 1,'.'); fmtb=scan("&fmt", 2,'.');
  if fmtb=0 then fmta=fmta+1;
  fmt1=put(fmta+1, best.) || "." || put(fmtb+1, best.);
  fmt2=put(fmta+2, best.) || "." || put(fmtb+2, best.);
  call symput('fmt1', compress(fmt1));
  call symput('fmt2', compress(fmt2));
 run;
 ***GET THE TOTAL NUMBER OF COLUMNS ***;
 %global NBCOL;
 proc sql noprint;
      select compress(put(count(distinct col),3.0)) into: NBCOL
      from &inds
      where col ge 1;
 quit;
 %if %UPCASE("&colnb") ne "NONE" %then %do; %let NBCOL=&colnb; %end;
 ***DESCRIPTIVE STATS***;
 proc sort data=&inds out=inds;
  by col;
 run;
 proc univariate data=inds noprint;
  var &var;
  by col;
  output out=&var.0 n=N mean=Mean stdmean=SE std=SD  median=Median
                          Q1=Q1 Q3=Q3 min=Min max=Max;
 run;
 proc transpose data=&var.0 out=&var.0(DROP=_LABEL_) prefix=val;
  var N Mean SE SD Median Q1 Q3 Min Max;
  id col;
 run;
```

```
 ***FORMATTING***;
 data &var(keep=block lbl col: ord);
  length lbl $200;
  array val{*} val1-val&nbcol;
  array rval{*} rval1-rval&nbcol;
  array col{*} $40 col1-col&nbcol;
  set &var.0;
  retain rval1-rval&nbcol;
  block=&block;
  if UPCASE(_NAME_)="N" then do;
   ord=1;lbl=" N"; do i=1 to dim(val); col[i]=put(val[i], 4.0); end; output;
  end;

%if &pres_typ =1 %then %do;
  else if UPCASE(_NAME_)="MEAN" then do;
            ord=2; lbl=" Mean";
                      do i=1 to dim(val); col[i]=put(val[i],&fmt1); end; output;
  end;
  else if UPCASE(_NAME_)="SD" then do;
            ord=3;lbl=" SD";
            do i=1 to dim(val); col[i]=put(val[i],&fmt2); end;output;
  end;
  else if UPCASE(_NAME_)="MEDIAN" then do;
            ord=4;lbl=" Median";
            do i=1 to dim(val); col[i]=put(val[i],&fmt1); end;
            output;
  end;
  else if UPCASE(_NAME_)="MIN" then do;
            do i=1 to dim(val); rval[i]=val[i]; end;
  end;
  else if UPCASE(_NAME_)="MAX" then do;
            ord=4;lbl=" Min, Max";
do i=1 to dim(val); col[i]=put(rval[i],&fmt) || ", " || put(val[i],&fmt); end;
output;
  end;
%end;
```

```
%else %if &pres_typ =2 %then %do;
  else if UPCASE(_NAME_)="MEAN" then do;
            do i=1 to dim(val); rval[i]=val[i]; end;
  end;
  else if UPCASE(_NAME_)="SE" then do;
            ord=2; lbl=" Mean (SE)";
            do i=1 to dim(val);
                      col[i]=put(rval[i],&fmt1) || "(" || compress(put(val[i], &fmt2)) || ")"; end;         output;
  end;
  else if UPCASE(_NAME_)="MEDIAN" then do;
            ord=3;lbl=" Median";
do i=1 to dim(val); col[i]=put(val[i],&fmt1); end;
output;
  end;
  else if UPCASE(_NAME_)="SD" then do;
            ord=4;lbl=" Std";
do i=1 to dim(val); col[i]=put(val[i],&fmt2); end;
output;
  end;
 else if UPCASE(_NAME_)="Q1" then do;
            do i=1 to dim(val); rval[i]=val[i]; end;
  end;
  else if UPCASE(_NAME_)="Q3" then do;
            ord=5;lbl=" Q1, Q3";
do i=1 to dim(val); col[i]=put(rval[i],&fmt) || ", " || put(val[i],&fmt); end;
output;
  end;
  else if UPCASE(_NAME_)="MIN" then do;
            do i=1 to dim(val); rval[i]=val[i]; end;
  end;
  else if UPCASE(_NAME_)="MAX" then do;
            ord=6;lbl=" Range";
do i=1 to dim(val); col[i]=put(rval[i],&fmt) || ", " || put(val[i],&fmt); end;
output;
  end;
%end;
 run;
```

```
***RTF ALIGMENT***;
%do k=1 %to &nbcol;
 proc sql noprint;
                select max(length(compress(scan(col&k,1,'.')))) into: maxus&k
                from &var
                where indexc(col&k,',') = 0;
                select max(length(compress(scan(col&k,1,',')))) into: maxuse&k
                from &var
                where indexc(col&k,',') ne 0;
 quit;
%end;
data &var(drop=i);
 length align $40.;
 set &var;
 array col{*} $40 col1-col&nbcol;
 array maxuse{*} maxuse1-maxuse&nbcol;
 %do j=1 %to &nbcol;
  maxuse[&j]=max(&&&maxus&j,&&&maxuse&j);
 %end;
 do i=1 to dim(col);
  if (length(compress(scan(col[i],1,'.'))) < maxuse[i] and indexc(col[i],',') = 0)  then do;
   algn=maxuse[i] - length(compress(scan(col[i],1,.')));
   if algn=1 then align='\~';
   else if algn=2 then align='\~\~  ';
   else if algn=3 then align='\~ \~';
   col[i]=trim(left(align))|| trim(left(col[i]));
  end;
 end;
run;
%mend _univ_;
```