ABSTRACT
Embedding code within SAS® macro definitions is frequently used to generate output datasets for analysis of multiple variables in clinical studies involving treatment groups. Typically, SAS Procs such as Means and Freq are applied to individual variables within a macro, which itself might be nested within another macro to sequentially analyze a series of variables. Although such code is extremely useful and flexible, such programs can suffer from macro complexity or macromania. When inherited by subsequent programmers, such programs can prove timely to modify when table specifications change.

In this paper, I exhibit how statistics calculated upon multiple variables can be summarized from features of ODS output datasets. Easy to read, standard SAS code is used to create a dataset for output to a desired file type. In one example, subject counts and within treatment group percentages are obtained from Proc Freq to create an output dataset for multiple categorical variables. Another example demonstrates the combining of ODS outputs from Proc Univariate to create a summary dataset of descriptive statistics for variables with continuous values.

INTRODUCTION
A common task of clinical data programming is the production of tables containing statistical summaries on variables of interest. Limitations within the outputs generate by SAS Procs are typically ameliorated by clever use of embedding such Procs within macros. In this approach, a macro parameter is used to code the variable on which the Procs operate. Additional macro scripting might control the observations on which the Procs operate. A surrounding or wrapper macro is sometimes used to sequentially execute the Procs for a series of user specified variables. The resulting datasets are then concatenated or merged in order to create a dataset with the desired statistics for all specified variables. A typical partial example follows:

```sas
%macro combinedata;
    proc sql noprint;
        select trim(left(put(max(trtgroup), best.))) into :maxtrtgrp from ref.indata
            where trtgroup<99;
    data alldata (keep=subject treatcode itptx r1-r&novar);
        set ref.indata;
        where valueset = 'y' and indicator = 'y';
        %do i=1 %to &maxtrtgrp; r&i=(trtgroup = &i); %end;
        r&novar=(trtgroup in (%do i = 1 %to &maxtrtgrp; &i  %end; ));
    proc univariate data=alldata; by treatcode;
        var r1 - r&novar;
    output out = stats sum = sum1 - sum&novar; run;
    data final (keep=treatcode fgrp1-fgrp%eval(&maxtrtgrp+1));
        set stats; by treatcode;
        %do ix = 1 %to &tottrt;,
            if &&n&ix=0 or sum&ix = 0 then p&ix = 0;
            else p&ix=sum&ix / &&n&ix * 100;
        if p&ix=0 then fgrp&ix = trim(left(put(sum&ix, 4.))) || ' (0)';
        else fgrp&ix=trim(left(put(sum&ix,4.)))||'(\mid'||trim(left(put(p&ix,4.1)))||')';
        %end; run;
%mend combinedata;
%combinedata;
```

This programming technique has proven usefulness in generating a wide variety of outputs.

Frequently, new projects inherit these programs. A programmer beginning work on such a project may face a rather daunting task in adapting macro code to a modified table specification and design, especially when program annotation is weak due to tight timelines. Much of the macro approach was initiated before the advent of ODS outputs but continues as an accepted convention. In this paper, I provide examples of how such ODS output
datasets can be used to create a summary dataset prepared for output to a table. The coding removes the need for virtual all of the macro parameters and structure and is thus more transparent to troubleshooting and modification.

USING ODS OUTPUT DATASETS WITH PROC FREQ TO CREATE TABLES WITH COUNTS AND PERCENTS

EXAMPLE DATA AND REQUESTED TABLE

My first example is the creation of summary tables for counts and percentages frequently requested when subjects are classified by categorical variables.

Consider an example dataset where subjects are classified as either receiving a ‘Placebo’ or ‘Treatment’ and are classified for 3 coded medical history variables (prpcd, pircd and bldcd). The data of interest are in numeric codes, which are decoded by numeric formats. Note in this example that all 3 medical history variables have the same possible values and are decoded by the same format.

```
proc format;
  value trtfmt 0='Placebo' 1='Treat' 2='Total';
  value medhisf 0='Never' 1='Rare' 2='Common' 3='Usual' 66='Often' 99='Absent';
run;

data test;
  format trtcd trtfmt. prpcd pircd bldcd medhisf. ;
  input subjid $ sexcd $ trtcd   prpcd  pircd   bldcd ;
datalines;
  0130 M 1 3 1 3
  0132 F 1 3 0 3
  0330 M 0 3 1 0
  0331 F 0 2 3 3
  1230 F 1 0 . 0
  1231 M 1 3 2 1
  1232 F 1 3 2 2
  1633 F 0 2 2 1
  1634 F 1 0 0 0
  1637 F 1 0 1 3
  1830 F 1 2 3 2
  1831 F 1 3 0 3
  1832 M 1 3 3 2
  1930 F 0 2 . 0
  230 F 1 3 0 0
  2730 M 1 2 1 0
  3630 M 1 3 2 0
  3632 M 0 1 2 3
  3730 M 1 66 3 2
  3830 F 1 3 0 0
  3832 F 1 0 . 0
  3931 F 1 3 3 3
  3932 M 1 2 2 3
  3933 M 1 66 . .
  4330 M 1 66 1 2
run;
```

The requested table is a summary of counts and percentages for the three medical histories. Structure of this table is shown below in part.

<table>
<thead>
<tr>
<th>Medical History = prpcd</th>
<th>Placebo</th>
<th>Treatment</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>n</td>
<td>%</td>
<td>n</td>
</tr>
<tr>
<td>Never</td>
<td>3</td>
<td>(13.0)</td>
<td>6</td>
</tr>
<tr>
<td>Rare</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Common</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Usual</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Missing</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>
**PROC FREQ OUTPUT USING OUT=**

For the data above counts of subject classification could be performed as follows using Proc Freq. Because each out= statement is restricted to one table request, three tables statements are required to produce all the output datasets.

```r
ods listing;
proc freq data=test;
tables trtcd*prpcd / out=prpout outpct sparse missing;
tables trtcd*pircd / out=pirout outpct sparse missing;
tables trtcd*bldcd / out=bldout outpct sparse missing;
run;
```

Each output dataset has the following structure (eg. bldout)

<table>
<thead>
<tr>
<th>trtcd</th>
<th>bldcd</th>
<th>COUNT</th>
<th>PERCENT</th>
<th>PCT_ROW</th>
<th>PCT_COL</th>
</tr>
</thead>
<tbody>
<tr>
<td>Placebo</td>
<td>.</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0.000</td>
</tr>
<tr>
<td>Placebo</td>
<td>Never</td>
<td>2</td>
<td>8</td>
<td>40</td>
<td>22.222</td>
</tr>
<tr>
<td>Placebo</td>
<td>Rare</td>
<td>1</td>
<td>4</td>
<td>20</td>
<td>50.000</td>
</tr>
<tr>
<td>Placebo</td>
<td>Common</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0.000</td>
</tr>
<tr>
<td>Placebo</td>
<td>Usual</td>
<td>2</td>
<td>8</td>
<td>40</td>
<td>25.000</td>
</tr>
<tr>
<td>Treat</td>
<td>.</td>
<td>1</td>
<td>4</td>
<td>5</td>
<td>100.000</td>
</tr>
<tr>
<td>Treat</td>
<td>Never</td>
<td>7</td>
<td>28</td>
<td>35</td>
<td>77.778</td>
</tr>
<tr>
<td>Treat</td>
<td>Rare</td>
<td>1</td>
<td>4</td>
<td>5</td>
<td>50.000</td>
</tr>
<tr>
<td>Treat</td>
<td>Common</td>
<td>5</td>
<td>20</td>
<td>25</td>
<td>100.000</td>
</tr>
<tr>
<td>Treat</td>
<td>Usual</td>
<td>6</td>
<td>24</td>
<td>30</td>
<td>75.000</td>
</tr>
</tbody>
</table>

The keywords listed as options in the Tables statement gives certain desirable qualities to the output datasets. ‘Missing’ includes subjects with missing medical history codes in the calculation of percents. If values are present in at least one of the classification values of variables in the tables statement, then ‘sparse’ outputs 0 counts for values that are not present in the input data (e.g. the ‘Common’ row for the placebo group in the above example). Finally, ‘outpct’ produces the PCT_ROW variable. This variable is useful because the table specification calls for counts and percentages, where the percentage is defined as the within trtgroup value (e.g. above, the medical code ‘Never’ within the Placebo group has a count of 2 which is 40% of the total placebo subjects (N=5)).

**PROC FREQ OUTPUTS USING ODS**

Instead of producing outputs through out=, datasets can be produced using ODS output objects. One such object CrossTabFreqs is created when a tables statement is requested for combinations of two or more variables. This output produces a similar dataset to out=, but with some additional information. The code for this follows:

```r
ods listing close;
ods trace on;
ods OUTPUT CrossTabFreqs =Frqdat1 (rename=(prpcd=MedHist))
     CrossTabFreqs2=Frqdat2 (rename=(pircd=MedHist))
     CrossTabFreqs3=Frqdat3 (rename=(bldcd=MedHist));
proc freq data=TEST;
tables trtcd*(prpcd pircd bldcd) / missing;
run;
ods output close; ---;
```

Again, three tables are requested but only one tables statement is needed. ODS creates three output objects and these are given the dataset names Frqdat1-3. Because the table specification calls for the creation of one summary table, these datasets are to be concatenated into one dataset. For this reason, in each output the medical history variables are renamed to the common name MedHist. In addition, the function of keywords ‘sparse’ and ‘outpct’ is automatic with CrossTabFreqs so only the keyword ‘missing’ in included, again to use missing values in the calculation of percents. The following is a sample of the ODS dataset for the medical history variable bldcd.
### POST PROCESSING OF ODS OUTPUT DATASETS

To create the dataset required for output of the requested table, some post ODS processing is necessary. The following code concatenates the three output datasets and restructures the data for ease of output.

```latex
\begin{verbatim}
data all;
set frqdat1 frqdat2 frqdat3;
if trtcd=. then trtcd=2;    *** A;
if _type_='11' then Percent=RowPercent; *** B;
if (_type_='11' or _type_='01') and MedHist=. 
    then MedHist=99;    *** C;
if _type_ in ('10','00') then delete;  *** D;
display=trim(left(put(frequency,3.)))||'  ('||trim(left(put(Percent,4.1)))||')';
keep table trtcd MedHist display;
run;
\end{verbatim}
```

Some simple 'if' statements aid the post-processing. A) since treatment code (trtcd) is never missing, values of '.' in the output datasets indicate an observation for the Total column. The _TYPE_ variable could also be used. B) The percent value requested in the table is the within column percentage. This value is provided by the RowPercent variable except when the observation is for a 'Totals' value where the value desired is Percent. Thus, for Placebo and Treatment observations (_TYPE_=11), the RowPercent values are mapped to the Percent variable, which now contains the output percent for all observations. C) The table requests that missing medical history values be counted and labeled as 'Absent'. Therefore, this statement maps these rows to the format for 99. D) Finally, column totals are not requested so, for data simplicity, these values are eliminated using the _TYPE_ variable.

The dataset created from the above code can then be sorted and transposed (Proc Transpose by MedHist with id trtcd) on the variable ‘display’, which has been created by combining the counts (frequency) and percents into a character variable. (Code not shown.) The resulting dataset has the structure and variables for output to the tables file in rtf format.
CAVEAT FOR VALUES NOT REPRESENTED IN DATA
Although Proc Freq produces 0 containing counts for table requests with no observations, it does not return observations when data is not present for a table variable (see Phillips 2005). For instance, imagine an early-in-study scenario where Placebo data is not yet available so the input data is composed only of the treatment group observations. A Placebo column, however, is desired for the draft table. Either one of the following actions could be coded. 1) Proc Freq does not have the preloadfmt keyword option to create an output dataset with all combinations of tables statement variables. To handle this situation, the user could use a data step to output a simple scaffold structured dataset with 0 counts and percents for all combinations of the tables variables. This dataset can then be updated or merged with the actual output dataset from Proc Freq so that the 0 containing values in the scaffold dataset would not be overwritten unless data are present. 2) Use of Proc Means without a var statement will produce counts of class variables and the preloadfmt option in the class statement with the completetypes option in the Proc Means statement would produce 0 containing counts. Percents, however, are not available in Proc Means so they need to be derived in post Proc processing. In addition, the column specific percents in this example would require careful capture of the totals needed to calculated these type of values.

USING ODS OUTPUT DATASETS WITH PROC UNIVARIATE TO PRODUCE TABLES OF DESCRIPTIVE STATISTICS
Now consider a request for a table containing typical statistics on quantitative variables. The table specifications might be as follows:

<table>
<thead>
<tr>
<th>bleeding times</th>
<th>Placebo</th>
<th>Treatment</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Female</td>
<td>Male</td>
</tr>
<tr>
<td>Subjects</td>
<td>3</td>
<td>1</td>
</tr>
<tr>
<td>Mean</td>
<td>4.1</td>
<td>0.0</td>
</tr>
<tr>
<td>Min Max</td>
<td>[0.0 9.3]</td>
<td>[0.0 0.0]</td>
</tr>
</tbody>
</table>

Below are part of the input data. Read as in the Proc Freq example above.

0130 M 1 3.4 8.1 3.1
0132 F 1 3.5 0 3.5
0330 M 0 3.5 8.7 0
0331 F 0 2.8 3.5 3.0
1230 F 1 0 . 0
1231 M 1 1.6 2.8 6.1
1232 F 1 1.6 2.8 2.2

The following code saves several ODS outputs as datasets that, altogether, contain the statistics requested.

ods listing close;
ods output Moments=samp
   basicmeasures=basics (where=(locmeasure in ('Median','Mode')))
   BasicIntervals=Intervals
   Quantiles=Bounds;
proc univariate data=test cibasic;
class trtcd sex;
var prp pir bld;
run;

Once these datasets are examined, a general strategy can be developed and coded to extract the required variables and data. An example of this code with annotation follows:

******************************************************************************
get sample size, Mean, Std Dev - put Std Err in separate ds to permit concatenation
******************************************************************************;
data samp2 (keep=trtcd sex varName Label1 LocValue rename=(Label1=LocMeasure))
   semean (keep=trtcd sex varName Label2 Locvalue2 rename=(Label2=LocMeasure Locvalue2=Locvalue));
set samp;
Locvalue=input(left(cValue1),10.);
Locvalue2=input(left(cValue2),10.);
output samp2;
if Label2='Std Error Mean' then output seemean; run;
*********************************************************************************
extract [upper tertile, max] & [lower tertile, min] into separate datasets - create
var setn for proper combining vars in merge
*********************************************************************************;
data Lower (drop= Estimate LocValue2 q3 Quantile)
Upper (drop= Estimate LocValue q1 Quantile);
set Bounds;
LocValue=Estimate;
LocValue2=Estimate;
q1=Quantile; q3=Quantile;
if Quantile in ('0% Min','100% Max') then setn=2;
if Quantile in ('25% Q1','75% Q3') then setn=1;
if Quantile in ('75% Q3','0% Min') then output Lower;
if Quantile in ('25% Q1','100% Max') then output Upper; run;
proc sort data=lower; by trtcd sex varname setn;
proc sort data=upper; by trtcd sex varname setn; run;
data limits;
merge Lower Upper;
by trtcd sex varname set;
Locmeasure=compress(q1||'-'||q3);
drop setn; run;
**********************************************************************************
95% conf intervals of the mean
*********************************************************************************;
data confint;
set intervals;
rename lowerCL=LocValue upperCL=LocValue2;
LocMeasure='CI_95';
where parameter='Mean';
drop parameter estimate; run;

Once parameters are extracted and rearranged, the datasets are merged.

data all;
length LocMeasure $16;
set samp2 (in=S) basics (in=B drop=VarMeasure VarValue) seemean (in=SE) confint
Limits (in=L);
length display $12;
if S or B or SE then display=put(Locvalue,6.1);
if S and Locmeasure='N' then display=put(Locvalue,4.);
if L then display=['trim(left(put(Locvalue,6.1)))||''||trim(left(put(Locvalue2,6.1)))];
*** order output as in shell, only display these stats *****;
select (Locmeasure);
when ('N') ordvar=1;
when ('Mean') ordvar=2;
when ('Std Deviation') ordvar=3;
when ('Std Error Mean') ordvar=4;
when ('Median') ordvar=5;
when ('25%Q1-75%Q3') ordvar=6;
when ('0%Min-100%Max') ordvar=7;
otherwise; end;
if ordvar ne .;
TrtGender=trim(left(put(trtcd,trtfmt.)))||'_'||sex;
drop trtcd sex; run;
**CONCLUSION**

This paper demonstrates some instances where ODS output objects can be used to create SAS datasets. These datasets are typically more useful than those available using the `out`= option, and can be used on multiple variables in one Proc step. The code can substitute for commonly employed macro-enclosed Procs that "loop through" the Proc code for each variable in sequential macro calls. Unlike complexly scripted macros, this code has a familiar presentation. It can be more easily understood, modified and adapted to other scenarios.

Purposely, the examples provided have no macro variables. The astute reader, however, will see opportunity to make the code more dynamic through macro coding. For instance, the list of variables to be analyze may be varying and coding of the tables statement in Proc Freq, or the class and var statement of Proc Univariate, with macro variables and statements could improve versatility.

Finally, further customization of output could be achieved through Proc Template to create customized output objects. In the pharmacology environment, this extension is typically more a utility and would required project group agreement and maintenance.

**REFERENCES**


**ACKNOWLEDGMENTS**

The author would like to thank Kari Kroeger for her assistance. The author also thanks Paul LaBrec for his help and encouragement.

**CONTACT INFORMATION**

Rod Norman  
i3 StatProbe  
10052 Mesa Ridge Court  
San Diego, CA 92121  
858 431 3017  
E-mail: rodney.norman@i3statprobe.com

SAS and all other SAS Institute Inc. product or service names are registered trademarks or trademarks of SAS Institute Inc. in the USA and other countries. ® indicates USA registration.