Paper 150-28

Efficient Reporting with Large Numbers of Variables: a SAS® Method

Jonathan R. Goddard, Health Care Research Unit, University of Southampton, UK

ABSTRACT

Large surveys or clinical trials will often generate many variables which need to be reported in a similar way. Tabulating a large number of categorical variables by a particular grouping variable is one such task. This paper presents a macro method of performing repetitive reporting tasks without having to explicitly name or count all the variables involved. This method is made more useful by the ability of SAS software (in version 8.2) to write output directly to a rich-text file, via the Output Delivery System command, ODS RTF. The generated reports can be written to a single Microsoft Word file, together with an indexed table of contents. The code behind the idea is explained and illustrated with an example from a data set supplied with SAS. The paper discusses base SAS (including macros) and is primarily aimed at IBM PC and Microsoft Windows users with intermediate SAS programming skills.

INTRODUCTION

Analyzing data from a large survey or clinical trial can be very time-consuming, as there can be hundreds of variables that need reporting. The variables will usually need to be handled in a similar way, e.g. frequencies and percentages, overall and by a grouping variable. To avoid the repetitious coding you can write a single-parameter macro, but the multiple macro calls required can still make things awkward. In addition, the document management involved in collating the output can be considerable. These problems can be largely eliminated through the use of a dynamic macro loop and ODS RTF output.

METHOD

To illustrate the method in a meaningful way, I will use a data set supplied with the SAS System. Consider the data set CARS, stored in the SASUSER library. This is a survey of car owners, and contains categorical data (one row per respondent). Suppose you want to tabulate each characteristic by the marital and family status of the owner. The method requires that all relevant variables be labeled and formatted, so some pre-processing is required:

```
proc format;
value origf 1='American' 2='Japanese'
            3='European';
value sizef 1='Small' 2='Medium' 3='Large';
value typef 1='Family' 2='Sporty' 3='Work';
value homef 1='Own' 2='Rent';
value sexf 1='Male' 2='Female';
value incf 1='1 income' 2='2 incomes';
picture percf (round) 0-100='000%'; run;

data example;
set sasuser.cars (keep = norigin--nincome
                         nsex marital);
label  norigin = 'Origin of car'
       nsize = 'Car size'
       ntype = 'Car type'
       nhome = 'Home owning status'
       nsex = 'Sex'
       nincome = 'Number of incomes'
       marital = 'Marital and family status';
format norigin origf. nsize sizef. ntype
       typef. nhome homef. nsex sexf.
       nincome incf.;
run;
```

The PICTURE format has nothing to do with the CARS data set, but will be the way percentages are displayed in the output (to the nearest whole number, and with the '%' suffix).
The first step is to create an empty data set containing just the variables required for repetitive processing.

```
data varlist;
set example (keep = norigin--nsex);
stop;
run;
```

The STOP command means that no data will be copied to the data set: all you need are the names. The list operator (--) means 'all the variables from NORIGIN to NSEX'. The grouping variable, MARITAL, is not kept, as you do not want it cross-tabulated with itself. The variables required should, ideally, be together in the source data set, and in a desirable order for reporting. If this is not the case, you would need to use more than one variable list in the KEEP option, and perhaps a LENGTH statement to reorder the variables.
The second step is to use the system data set SASHELP.VCOLUMN (actually a SAS view) to read these variable names, labels and formats as records of a data set. This is like an output data set from the CONTENTS procedure.

```
data _null_;
set sashelp.vcolumn
     (where=(libname="WORK" and memname =
     "VARLIST")) end=last;
label = translate(label,"'",'"');
call symput('var' || trim(left(put(_n_,8.))),
             trim(name));
call symput('lab' || trim(left(put(_n_,8.))),
             trim(left(label)));
call symput('fmt' || trim(left(put(_n_,8.))),
             trim(format));
if last then
     call symput('n',trim(left(put(_n_,8.))));
run;
```

The first CALL SYMPUT statement generates one macro variable for each variable to be analyzed. The macro variable name will be different for each row, with a numerical suffix derived from _N_, the observation counter (|| is the concatenation operator). The suffix is important as it gives a unique reference to each variable. This will help us to access it later. The first variable name will be assigned to the macro variable 'var1', the second to 'var2', and so on.
You repeat this process to capture the labels and formats for each variable. Before this happens, the TRANSLATE function replaces any double quotes in the labels with single quotes. This is necessary as the labels will be used in the titles, and the TITLE statements will use double quotes. The LEFT and TRIM functions are for tidiness. You will be using the labels as part of a title, and use of these functions will avoid unsightly gaps.
The END=LAST option on the SET statement is used so that you can assign the total number of variables to be analyzed to a separate macro variable, 'n'.
If all the variables in a data set are to be tabulated, and they are already in a suitable reporting order, then you do not need the first DATA step. In this example, your SET statement in the DATA _NULL_ step could become:

```
set sashelp.vcolumn
   (where = (libname = "WORK" and memname =
   "EXAMPLE" and name ne "marital"))
   end=last;
```

The next step is to set up a reporting process (e.g. the TABULATE procedure) inside a macro loop:

```
%macro repeat1;

%do i = 1 %to &n;
  title "Table &i: &&lab&i, by family status";
  proc tabulate data=example;
  class marital &&var&i;
  tables (&&var&i all), (marital all)*
     (n pctn<&&var&i all>='%'*f=percf.);
  run;
%end;

%mend;
%repeat1;
```

The macro references are resolved twice for each execution of the loop. When the macro variable i equals 1, &&var&i resolves to &var1 on the first pass, and the name of the first analysis variable on the second. The %DO loop runs from 1 to &n. This way, the programmer does not need to specify how many variables there are: SAS has already counted this in the previous step.
Executing this macro will give an exhaustive series of cross tabulations. Whilst this is useful, you can enhance this (in SAS 8.2) by writing the results to a Word file with a hyperlinked table of contents. This will be particularly useful if the number of variables to process is large. You write an enhanced macro to do this:

```
%macro repeat2;

data toc;
length tabledesc $200 pageno $100;
%do i = 1 %to &n;
  tabledesc = "&&lab&i, by family status";
  pageno=compbl('{\field {\*\fldinst PAGEREF '
                || "var&i" || ' \\h}}');
  output;
%end;
run;

options orientation=portrait nonumber nodate;

ods rtf file="Tabulation of several
variables, by family status.rtf"
style=printer;

title "Table of contents (page numbers
hyperlinked)";
footnote j=c "{\field {\*\fldinst PAGE
              \\*MERGEFORMAT}}";

proc report data=toc nowd
      style={protectspecialchars=off
      rules=groups frame=hsides};
column tabledesc pageno;
define tabledesc / display 'Table' left
       style={cellwidth=10cm just=left};
define pageno / display 'Page'
       style={cellwidth=2cm just=right};
run;

%do i = 1 %to &n;

  ods rtf bookmark="var&i";
  title "Table &i: &&lab&i, by family status";

  proc tabulate data=example;
  class marital &&var&i;
  tables (&&var&i all), (marital all)*
         (n pctn<&&var&i all>='%'*f=percf.);
  run;

%end;

ods rtf close;
```

```
%mend;

%repeat2;
```

The first data set created (TOC) becomes the table of contents. The special characters in the variable PAGENO will cause the page number to be hyperlinked to the relevant table in Word. The first ODS RTF statement opens a rich-text file for subsequent output. This file type (.RTF) will be associated with Word. The first output to go to this file is the table of contents (the REPORT procedure step). The subsequent outputs are from the PROC TABULATE step. The title of each table will appear in a Word header, and the page number will appear in the footer. The NONUMBER and NODATE options are activated to prevent the default behavior of system titles appearing in a text box in the top right corner. At the end of the loop, the ODS RTF CLOSE command closes the rich-text file.

FIGURE 1

Table of contents (page numbers hyperlinked)

Table	Page
Table 1: Origin of car, by family status	2
Table 2: Car size, by family status	3
Table 3: Car type, by family status	4
Table 4: Home owning status, by family status	5
Table 5: Number of incomes, by family status	6
Table 6: Sex, by family status	7

The page numbers may not appear in the table of contents when you first open the file in Word. If this is the case, select the column of page numbers in Word (place the cursor anywhere in this column and choose Table | Select | Column) and press F9 to update the fields. Furthermore, if you have the 'Update fields' print option selected (Tools | Options | Print), the fields will update themselves before printing. Moving the mouse over the page number will show it to be a hyperlink, which can take you straight to the associated table.
The bookmark name (in the second ODS RTF statement) is used to link the table to the table of contents entry. It needs to be short (30 characters is the maximum for a Word bookmark), unique, and the same as the name embedded in the PAGENO variable.

FIGURE 2

Table 1: Origin of car, by family status

	Marital and family status									
	Married		Married w Kids		Single		Single w Kids		All	
	N	%	N	%	N	%	N	%	N	%
Origin of car										
American	37	36%	52	47%	33	30%	6	40%	128	38%
Japanese	51	50%	44	40%	63	57%	8	53%	166	49%
European	14	14%	15	14%	15	14%	1	7%	45	13%
All	102	100%	111	100%	111	100%	15	100%	339	100%

The appearance of the table and the positioning of the paper margins can be customized through the use of styles in the TEMPLATE procedure. Displayed above is the SAS-supplied style called PRINTER. You can, for example, change the style to

eliminate shading and the vertical lines. See the reference at the end of this paper (and the PROC TEMPLATE Help) for details of how to customize the style.
Notice that you have made use of the macro variables VAR1 through VARn (variable names), and LAB1 through LABn (labels), but not FMT1 through FMTn (formats). The latter are still worth creating, however, as they can prove useful for trapping exceptions. In many situations, you will have variables coded 1 for 'Yes' and 0 for 'No'. It may be more aesthetically pleasing if the 'Yes' appears first, and so you may want to conditionally reverse the order in which the levels appear, i.e.

```
%if &&fmt&i = NYF. %then %do;
   class &&var&i / descending;
%end;
```

Also, you may want to use a different format from the one originally defined:

```
%if &&fmt&i = SIZEF. %then %do;
   format &&var&i NEWF.;
%end;
```

(Note that SASHELP.VCOLUMN capitalizes the format name, even though it was originally entered in lower case).
You may want to present an abbreviated categorization for some variables. For example, a variable with the responses: Strongly Agree | Agree | Disagree | Strongly Disagree, can be summarized as Agree | Disagree, by replacing the format with a suitable alternative.
Instead of using the macro loop to create several small tables, you can position it inside PROC TABULATE to create one long table. This is worth considering, as Word will, by default, repeat the header rows of long tables which span more than one page. It will also repeat the title for subsequent pages, using a Word header. To do this, the steps in the macro can be modified as follows:

```
data toc;
length tabledesc $200 pageno $100;
tabledesc = "Breakdown of all variables, by
family status";
pageno = compbl('{\field {\*\fldinst PAGEREF
                ' || "BIGTABLE" || ' \\h}}');
output;
run;

ods rtf bookmark="BIGTABLE";
title "Breakdown of all variables, by family
status";

proc tabulate data=example;
class marital
   %do i = 1 %to &n;
    &&var&i
   %end;;
tables (
         %do i = 1 %to &n;
            &&var&i all
         %end;), (marital all)*
         (n pctn<%do i = 1 %to &n;
                   &&var&i
                 %end; all>='%'*f=percf.);
run;
```

Three loops in all are used inside the TABLES statement. An extract from the table produced is shown below:

FIGURE 3

Breakdown of all variables, by family status
(abridged version)

	Marital and family status									
	Married		Married w Kids		Single		Single w Kids		All	
	N	%	N	%	N	%	N	%	N	%
Origin of car										
American	37	37%	50	46%	32	29%	6	40%	125	37%
Japanese	51	50%	44	40%	62	57%	8	53%	165	49%
European	13	13%	15	14%	15	14%	1	7%	44	13%
All	101	100%	109	100%	109	100%	15	100%	334	100%
Car size										
Small	50	50%	37	34%	58	53%	6	40%	151	45%
Medium	42	42%	51	47%	40	37%	8	53%	141	42%
Large	9	9%	21	19%	11	10%	1	7%	42	13%
All	101	100%	109	100%	109	100%	15	100%	334	100%

Another variation on this technique uses nested loops to cross tabulate a series of row and column variables. To do this, you need to set up two data sets, one for row variables and one for column variables, and create two sets of macro variables. The code to define the macro variables therefore becomes:

```
data rows (keep=nhome--nsex)
     cols (keep=norigin--ntype);
set example;
stop;
run;

data _null_;
set sashelp.vcolumn (where = (libname =
     "WORK" and memname = "ROWS")) end=last;
call symput('varr' ||
  trim(left(put(_n_,8.))),trim(name));
call symput('labr' ||
  trim(left(put(_n_,8.))),trim(left(label)));
call symput('fmtr' ||
  trim(left(put(_n_,8.))),trim(format));
if last then
  call symput('nr',trim(left(put(_n_,8.))));
run;

data _null_;
set sashelp.vcolumn (where = (libname =
     "WORK" and memname = "COLS")) end=last;
call symput('varc' ||
  trim(left(put(_n_,8.))),trim(name));
call symput('labc' ||
  trim(left(put(_n_,8.))),trim(left(label)));
call symput('fmtc' ||
  trim(left(put(_n_,8.))),trim(format));
if last then
  call symput('nc',trim(left(put(_n_,8.))));
run;
```

Inside the macro definition, the table of contents data set definition will now be:

```
data toc;
length tabledesc $200 pageno $100;
%do r = 1 %to &nr;
  %do c = 1 %to &nc;
    tabledesc = "&&labr&r, by &&labc&c";
    pageno = compbl('{\field {\*\fldinst
      PAGEREF ' || "Row&r._by_Col&c" || '
      \\h}}');
    output;
  %end;
%end;
run;
```

The output itself will be generated by the following code:

```
ods rtf file="Crosstab matrix.rtf"
        style=printer;

%do r = 1 %to &nr;
  %do c = 1 %to &nc;
    ods rtf bookmark="Row&r._by_Col&c" ;
    ods noproctitle;
    title "Frequencies of &&labr&r, by
     &&labc&c";
    proc freq data=example;
    tables &&varr&r * &&varc&c / nopct;
    run;
  %end;
%end;

ods rtf close;
```

The first table in the series is reproduced below:

FIGURE 4

Home owning status, by Origin of car

Table of nhome by norigin				
nhome(Home owning status)	norigin(Origin of car)			
Frequency Row Pct Col Pct	American	Japanese	European	Total
Own	93 38.43 74.40	111 45.87 66.87	38 15.70 86.36	242
Rent	32 34.41 25.60	55 59.14 33.13	6 6.45 13.64	93
Total	125	166	44	335
Frequency Missing = 4				

CONCLUSION

Although the examples presented here have used mainly PROC TABULATE, they are just an illustration of the technique. If a greater degree of customization is required, reports can be made by forming a data set of summary statistics from the relevant procedures and displaying this using PROC REPORT (with ODS RTF). An example would be a survey containing a lot of Yes/No variables. It will be wasteful of space to report the 'No' frequencies and percentages, so an alternative to PROC TABULATE could be the FREQ procedure with output data set options or ODS OUTPUT statements. When customizing a report in this way, an additional DATA step may be required to concatenate the summary data sets together prior to reporting.

The technique of the dynamic macro loop to count and process the variables can still be applied, however.
With a fairly small amount of code, a large amount of reporting has been done. In fact, the amount of coding stays the same, regardless of the number of variables. The code is generic and re-usable, and furthermore, the reports it produces are collated and indexed into a single Word file. This is a substantial bonus, as most sponsors will happily accept a report in this format.

REFERENCES

Shannon, David. "To ODS RTF and Beyond", *Proceedings of the Twenty-Seventh Annual SAS® Users Group International Conference*. April 2002. <http://www2.sas.com/proceedings/sugi27/p001-27.pdf> (November 15, 2002).

CONTACT INFORMATION

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

Jonathan R. Goddard
Health Care Research Unit
University of Southampton
Southampton General Hospital
Southampton
SO16 6YD
United Kingdom
Email: J.R.Goddard@soton.ac.uk
Web: http://www.som.soton.ac.uk/staff/jrg/