A Few Ways to Use Macro Variables, Comparison Operators, and “other=” Format Statements to Shorten Your Programs
Scott Burroughs

A macro will be used which takes all of the variable names in a data set and makes them into one string macro variable separated by spaces. Applications for this will be shown such as KEEP statements, macro calls, and formats.

Programmers from the “old” days called them assignment statements. SAS calls them comparison operators. Usually used in the IF-THEN-ELSE type of logical operations or just plain assignment statements, comparison operators can be used in other ways to shorten your code otherwise done in a more tedious manner.

Fun(?) With Macro Variables

Let’s say you want to list out or put into a macro a list of variables that’s at least reasonably long. But throughout the program you don’t want to type them all in several times for different analyses. Well, you could type them once into a macro string and do it that way, but what if you have 75 variables? And what if you need to reference individual variables for some reason or the names of the variables change from program to program? The only limitation in this exercise is that all of the variables need to be of the same type, character or numeric. Partly for this reason, your data set needs to have only the character or numeric variables that you will be using for the string and they need to be in the order you want them for use. In the example, I will be using a data set with various symptoms of an ailment under test for a pharmaceutical drug. The data set is named SYMP.

```sas
%MACRO mac_var(ds=);
data _null_; set &ds; if _n_ = 1 then do;
array char{*} _character_; length name $ 8 ; * could be longer for Version 8 *;
do i=1 to dim(char);
call vname(char{i},name);
call symput('v'||compress(put(i,3.)),name );
output;
end;
call symput('dimvar',i-1);
end;
%MEND;
/* this macro appends all of the individual */
/* &v1-&&v&dimvar variables to one string */
%MACRO listvar;
%local i w1 arg;
%do i=1 %to &dimvar;
%let w1 = &&v&i ;
%let arg = &arg &w1;
%end;
&arg
%MEND;
/* Now &sy will be the macro variable string of variables */
%let sy=%listvar;

Using the ‘other=’ statement in PROC FORMATS

Now let’s use the individual macro variables in a PROC FORMAT situation. Let’s say that you have a macro that needs a different format name for each processed variable and each format has to be the name of the variable, AND it can’t just use your own FORMAT statement. (I know this sounds far-fetched, but it used to be this way in my company). For example, you have 15 symptoms, each with the same severity format and each needs a format attached to it with the name of the variable (e.g. $ache.). Well, I’ve seen programs like this:

```sas
proc format;
value $ache
‘0’ = ‘None’
‘1’ = ‘Mild’
‘2’ = ‘Moderate’
‘3’ = ‘Severe’
.
.
.
value $pain
‘0’ = ‘None’
‘1’ = ‘Mild’
‘2’ = ‘Moderate’
‘3’ = ‘Severe’
run;
```

Just a little bit redundant.

Our huge company-wide format library already has that same format with the name $vsycd., so there’s no need to retype it in. If you want to copy a format with one name to one with a different name, it can be done like this:

```sas
proc format;
value $newfmt
other = [$vsycd.];
run;
```

Now, using this method and the macro variable assignments done previously, we can assign 15 new formats with the $vsycd. format in one short macro:

```sas
%MACRO McRowe;
proc format;
%do i=1 %to &dimvar;
value $&&v&i
other = [$vsycd.];
%end;
run;
%MEND McRowe;
```

Now, I’m at my deadline to submit this and I’m not sure what I was going to do with the KEEP statement mentioned in the abstract other than something like
this: (keep=subject &sy), but I'll make sure to put it in
the presentation.

"Fun" with Comparison Operators

Otherwise known as assignment statements. Let's say
you have a bunch of Yes/No variables from, say, a
questionnaire and you want to know how many Yes
responses each person had. How might you do this?
You can't use an N function or a COUNT function,
because they are used on numeric variables. So you
could convert each Y/N variable to 1/0 to use the
SUM(of ) function, but if we don't have to…. Let's do it
this way:

data x;
  set y;
  numyes=0;
array uhwray {*} $ x1-x100;
  do i=1 to dim{uhwray};
    numyes=numyes+(uhwray{i}='Y');
  end;
run;

Voila!

Here's another example. Let's say you have 3
treatments, and the treatment variable is character.
You want to use a PUT statement in a DATA _NULL_
for your table. Instead of this:

if tmt='A' then put @15 value;
if tmt='B' then put @25 value;
if tmt='C' then put @35 value;

do this:

put @(((tmt='A')+2*(tmt='B')+3*(tmt='C'))*10)+5) value
 @;

Biography:

Scott has been a statistician for GlaxoSmithKline
(formerly Glaxo Wellcome) in Research Triangle Park,
NC since 1994. He has programmed in SAS
extensively since 1992. He has a B.S. and an M.S. in
Statistics from Virginia Tech.

Scott Burroughs
GlaxoSmithKline
5 Moore Drive
17.1463J
Research Triangle Park, NC 27709
smb10565@gsk.com