

Paper 030-30

A Sample Macro for Creating a List on the Fly

Debbie Miller, National Park Service, Denver, CO

ABSTRACT

The SAS® macro facility is a great tool for adding flexibility to your programs. You can create a macro that will generate pieces of code on the fly, and then use that macro to insert the code later in your program. By doing this, you can alter the way a program works without having to hard-code the changes. Even a macro that generates a small section of code can be very helpful if you are using a program repeatedly. This paper offers a simple example of a macro that illustrates this idea. This macro generates a list that is used in a WHERE statement inside a data step to select the records that are read from another data set. This paper is intended for SAS users of all skill levels.

INTRODUCTION

SAS macro programming can be used to generate pieces of code on the fly that are used later in a program. The macro facility has its own language elements that allow you to assign a set of SAS programming statements to a macro name. The macro can be referenced later in a program to insert the desired statements. With the macro facility you can also assign text-strings to macro variables, either inside a macro or outside in open code. These are very useful tools for adding flexibility to a program.

As I data analyst I find that I perform a lot of repetitive tasks, and thus I have several programs that I run over and over again. Each time I run one of these programs, I am interested in producing output for a different subset of some particular data set parameter. For example, I often run a program that produces separate charts displaying air quality data obtained from 100 different data collection sites. I do not, however, need (or want) to produce charts for all 100 sites each time I run the program, and the set of sites I am interested in will vary from run to run. I also do not want to have to change the code each time I run the program to specify a different set of sites. One way to solve this is to use a short macro to generate a list of the desired sites on the fly. The macro is used inside a data step to limit an input data set to just the sites of interest.

OBTAIN THE LIST MEMBERS

In my application, I first have to decide the subset of sites for which I want to produce charts. In order to do this, I have to maintain somewhere a list of these sites, and then bring this information into SAS. I keep this list in a very simple Excel file containing a single column of site names. Below is an example of how this file looks.

sitelist
BOAP1
BRET1
BRIG1
CHAS1
LOST1

Sitelist is just the name of a variable whose values are the names of the sites whose data I want to display. The SAS program begins by importing this Excel file into a small SAS data set called *site_names*.

```

/* import excel file with names of sites */
proc import out= work.site_names
  datafile= "c:\requests\miller\sugi 30\site names.xls"
  dbms=excel2000 replace;
  getnames=yes; /* tells SAS to get the variable names from the
                 first row of the Excel file */
run;

```

The SAS data set *site_names* now contains a list of the names of the sites I am interested in. I want to use these site names later inside of a WHERE statement in a data step. Since the site names are the values of a character variable, **sitelist**, I will need to have single quotes around each site name when the list is used in a WHERE statement. They are added to each site name in a data step using the CATS function. This function concatenates character strings while removing leading and trailing blanks. Here it concatenates the single quote character with the value of the variable **sitelist**. This is similar to using the TRIM and LEFT string functions together, but it runs faster. The result is stored in the **site** variable.

```
data site_names;
  length site $7;
  set site_names;
  site=cats("'",sitelist,"");
  keep sitelist site;
run;
```

Now the *site_names* data set looks like this:

sitelist	site
BOAP1	'BOAP1'
BRET1	'BRET1'
BRIG1	'BRIG1'
CHAS1	'CHAS1'
LOST1	'LOST1'

Next the program determines how many site names are in the *site_names* data set, and stores this number as a text string to a macro variable. The macro variable is referenced later in the program by using the name of the macro variable preceded by the ampersand (&) symbol. When SAS sees the ampersand, it resolves the macro variable and replaces it with its value in the code.

The SET statement option END= creates a temporary data set variable that indicates when the last record of the data set has been reached. When this condition is met, a CALL SYMPUT statement is used to assign the value of the SAS automatic variable **_n_**, which contains the data record number, to the macro variable **total**. In this example, where there are 5 site names in the data set, the macro variable **total** will contain the value '5'.

```
data _null_;
  set site_names end=lastrec;
  if lastrec then do;
    call symput('total',_n_);
  end;
run;
```

CREATE THE MACRO

The macro *assign_names* is then created. Macros begin with a %MACRO statement that names the macro, and end with a %MEND statement. Individual macro programming statements begin with a %. The macro statements are not executed until invoked with a statement of the form %*macro-name*.

The first step in the macro is a do-loop that takes each site name in turn and assigns it to a one of a series of sequentially named macro variables--**site1**, **site2**, **site3**, etc. In this example there are 5 sites of interest, so 5 macro variables are created. The loop variable is a macro variable 'n' that runs in value from 1 to the total number of sites determined above (5). Macro variables assigned inside of a macro are normally only available within that macro, so a GLOBAL statement is required so that all of the **siten** macro variables will be available outside of the macro.

The DATA _NULL_ step is placed inside the do-loop, and thus is executed 5 times, each time reading a different line of the data set *site_name*. Each time it executes, it assigns the current site name from the

site_names data set to the next macro variable **siten** through a CALL SYMPUT statement. The text in quotes in the first part of the CALL SYMPUT statement, “**site&n**”, specifies the name of the macro variable to be created. The text in this particular example is placed in double quotes so that SAS will resolve the macro variable **n** when creating the variable name. SAS thus sees the variable names as **site1**, **site2**, etc. SAS does not resolve macro variables placed in single quotes, and would therefore read the variable reference as ‘**site&n**’ instead of ‘**site1**’. The text appearing in the CALL SYMPUT statement after the comma, **site**, specifies the data set variable whose value becomes the value of the new macro variable. Finally, the macro is invoked with a **%assign_names** statement.

```
%macro assign_names;

%do n=1 %to &total;

%global site&n;
data _null_;
  set site_names;
  if &n=&n_; /* keep only the nth data record */
  call symput("site&n",site);
run;

%end; /* end of the %do-loop */

%mend assign_names;

%assign_names;
```

Once the **siten** macro variables have been created and assigned their values, a second, shorter macro named ‘*create_sitelist*’ produces the list of site names. This macro also uses a do-loop that runs from 1 to the number of sites; this time the macro variable ‘**k**’ is used as the loop’s index variable. In each iteration of the loop, the code “**&&site&k**” is resolved by SAS to generate the name of a site surrounded by single quotes. The double ampersand causes SAS to resolve each macro variable in two steps. In the first step, “**&&site&k**” is resolved to “**&site1**”, assuming that the value of **k** is ‘1’. In the second step, the macro variable “**&site1**” is resolved to the value of the macro variable, which in this case is ‘BOAP1’.

```
%macro create_sitelist;
%do k=1 %to &total;
  &&site&k
%end;
%mend create_sitelist;
```

The ‘*create_sitelist*’ macro does not actually run until invoked with a **%create_sitelist** statement. The macro called ‘*create_sitelist*’ can now be used in a data step to limit the input data set to only the sites of interest. The macro is called inside of a WHERE statement. When the statement **&create_sitelist** is resolved it generates the list of sites, with each name enclosed in single quotes.

```
/* use the macro to limit a data set */

libname lib1 'c:\requests\miller\sugi 30';

data improve;
  set lib1.test_data;
  where site in (%create_sitelist);
run;
quit;
```

Thus, when this code is executed, the WHERE statement in the data step looks like this:

```
where site in ('BOAP1' 'BRET1' 'BRIG1' 'CHAS1' 'LOST1');
```

The result is that only data from the desired sites are brought in from the permanent SAS data set called TEST_DATA.

CONCLUSION

With fairly simple macro coding, it is possible to generate useful pieces of code on the fly that can be used later in a program. This allows your program to automatically modify itself to suit your needs, and can save you from having to hard-code changes to a program each time you run it. This macro is just one very small example of the many ways in which macro coding can add flexibility to your programs.

REFERENCES

SAS Institute Inc. (1999), *SAS Macro Language: Reference, Version 8*, Cary, NC: SAS Institute Inc.

CONTACT INFORMATION

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

Debbie Miller
National Park Service – Air Resources Division
12795 W. Alameda Parkway
Lakewood, CO 80228
Work Phone: (303) 987-6947
Fax: (303) 969-2822
Email: debbie_c_miller@nps.gov

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.

Other brand and product names are trademarks of their respective companies.