Paper IT-71

Store and Recall Macros with SAS ® Macro Libraries

John M. Myers, Virginia Commonwealth University

ABSTRACT

When you store your macros in a SAS macro library, you can recall them with your SAS programs and share them with other SAS programmers. SAS macro libraries help you to reduce the time it takes to develop new programs by using code that has been previously tested and verified. Macro libraries help you to organize your work by saving sections of code that you can reuse in other programs. Macro libraries improve your macro writing skills by focusing on a specific task for each macro. Macro libraries are not complicated – they are just a way to store macros in a central location. This paper will give examples of how you can build macro libraries using %INCLUDE files, AUTOCALL libraries, and STORED COMPILED MACRO libraries.

INTRODUCTION

Building a SAS macro library begins with choosing a location to store it. This could be in your personal directory for a private macro library. If you plan to share your macros with others, the macro library needs to be in a shared location such as a shared disk on a PC network or a shared directory on a UNIX server. For shared macro libraries, procedures need to be developed to coordinate with other users.

In this paper, I will describe three methods to store and recall SAS macros.

1. %INCLUDE files
2. AUTOCALL library
3. Stored compiled macro library

%INCLUDE FILES

The first method to build a SAS macro library is to use %INCLUDE files. A %INCLUDE file is a text file where you can store any SAS code. In this paper, I will be storing SAS macro definitions. Each macro should be stored in a separate file with extension “sas”. Each file should have the same name as the macro. The location for the SAS macro library should be dedicated to store only SAS macros.

In the first example, I build a text file that will be used in the following examples.

Example 1.1 shows how to store a macro (red text) in a file. In this example, I used the same name “calc11” for the file ① and the macro ②. It is not necessary to use SAS for this job. You could use your text editor to do this.

```
*------------------------------------------------*;
* Example 1.1 Store macro in a directory         *;
*------------------------------------------------*;
options dlcreatedir;
libname lib11 "C:\SESUG14\IT\Myers\mymacros";
filename file11 "C:\SESUG14\IT\MYERS\mymacros\calc11.sas"; ①
data _null_;
length sasstmt $80.;
input sasstmt & $;
file file11;
put sasstmt $;
datalines4;
%macro calc11; ②
%put begin macro calc11;
***;
%put end macro calc11;
%mend calc11;
;;;;
run;
```

Once your macro is stored in a text file, it can be recalled by any SAS program using the %INCLUDE statement.

Example 1.2 shows how to call a macro using %INCLUDE. The %INCLUDE statement ② tells SAS to read the file ① and insert the statements into the program. Then, SAS compiles the macro but it is not executed until you call it ③ later in your program. The option SOURCE2 on the %INCLUDE statement tells SAS to print the statements on the log. You can omit this option if you do not want to see the statements.

```
*-------------------------------------------------------*;
* Example 1.2 Call macro using %include                 *;
*-------------------------------------------------------*;
filename file12 "C:\SESUG14\IT\MYERS\mymacros\calc11.sas"; ①
%include file12 / source2; ②
***;
%calc11; ③
```

Example 1.3 shows another way to code the %INCLUDE statement. In this example, the file name is given in the %INCLUDE statement ①.

```
*-------------------------------------------------------*;
* Example 1.3 Call macro using %include                 *;
*-------------------------------------------------------*;
%include "C:\SESUG14\IT\MYERS\mymacros\calc11.sas" / source2; ①
***;
%calc11;
```

In the previous examples, you need to code the file name with the complete path either in a FILENAME statement or in the %INCLUDE statement. Another way to do this would be to use an aggregate location. With this syntax, you can call many different macros with one FILENAME statement.

Example 1.4 shows how to code the %INCLUDE statement using an aggregate location. For this syntax, the directory name is given on the FILENAME statement ①. The file name is given in parentheses after the FILEREF on the %INCLUDE statement ②. Usually if you want a pointer to a directory in SAS you would use a LIBNAME statement, but in this case you need to use a FILENAME statement. Also, this syntax requires that you use “sas” as the extension for the file name.

```
*-------------------------------------------------------*;
* Example 1.4 Call macro using %include                 *;
*-------------------------------------------------------*;
filename file14 "C:\SESUG14\IT\MYERS\mymacros\"; ①
%include file14(calc11) / source2; ②
***;
%calc11;
```

When SAS compiles the macro, it is stored in a temporary catalog named SASMACR on the WORK directory. After doing that once, you can call the macro again in the same program (or session) without having to repeat the %INCLUDE statement. After the program (or session) terminates, all the temporary files are deleted, so you need to use %INCLUDE statement in every program where you want to use the macro.

NOTES FOR %INCLUDE

1. %INCLUDE must start on a statement boundary (after a semicolon).
2. %INCLUDE is a global statement so it cannot be used in conditional logic.
3. Avoid using %INCLUDE in macros (nesting) because the inner macro will be compiled every time the outer macro is called.

AUTOCALL LIBRARY

The second method to build a SAS macro library is an AUTOCALL library. These macros are stored the same way as you would for %INCLUDE.

Example 2.1 shows how to store a macro (red text) in a file that will be used in later examples. Again, it is not necessary to use SAS for this job. You could use your text editor to do this.

```
*------------------------------------------------*;
* Example 2.1 Store macro in directory           *;
*------------------------------------------------*;
options dlcreatedir;
libname lib11 "C:\SESUG14\IT\Myers\mymacros";
filename file21 "C:\SESUG14\IT\MYERS\mymacros\calc21.sas";
data _null_;
length sasstmt $100.;
input sasstmt & $;
file file21;
put sasstmt $;
datalines4;
%macro calc21;
%put begin macro calc21;
***;
%put end macro calc21;
%mend calc21;
;;;;
run;
```

An AUTOCALL library is another way to have an aggregate location for your macros. With an AUTOCALL library, you can call many different macros with one FILENAME statement.

Example 2.2 shows how to call a macro from an AUTOCALL library. The FILENAME statement ② points to the directory where the macros are stored. Each macro is stored in a separate file with the same name as the macro and file extension “sas”. The system options statement ① gives the option “mautosource” which tells SAS that you want to use an AUTOCALL library and the option “sasautos” which tells SAS where to look for your library. SAS does not read the AUTOCALL library until you call the macro ③, then SAS will find, read, compile and execute the macro.

```
*------------------------------------------------------*;
* Example 2.2 Call macro using autocall library       *;
*------------------------------------------------------*;
options mautosource sasautos=file22; ①
filename file22 "C:\SESUG14\IT\MYERS\mymacros\"; ②
***;
%calc21; ③
```

When SAS compiles the macro, it is stored in a temporary catalog named SASMACR on the WORK directory (the same as %INCLUDE files). You can call the macro again in the same program (or session) without having to compile it again. After the program (or session) terminates, all the temporary files are deleted. If you call the macro in another program, SAS will find, read and compile it again.

You can have more than one AUTOCALL library.

Example 2.3 stores a macro (red text) in a separate directory “mymacros2” ① for the next example. Again, it is not necessary to use SAS for this job. You could use your text editor to do this.

```
*------------------------------------------------*;
* Example 2.3 Store macro in directory           *;
*------------------------------------------------*;
options dlcreatedir;
libname lib11 "C:\SESUG14\IT\Myers\mymacros2";
filename file23 "C:\SESUG14\IT\MYERS\mymacros2\calc23.sas"; ①
data _null_;
length sasstmt $100.;
input sasstmt & $;
file file23;
put sasstmt $;
datalines4;
%macro calc23;
%put begin macro calc23;
***;
%put end macro calc23;
%mend calc23;
;;;;
run;
```

Example 2.4 shows how to call macros from more than one AUTOCALL library. The option SASAUTOS ① contains a list of FILEREF’s which point to the AUTOCALL libraries.

```
*-------------------------------------------------------------*;
* Example 2.4 Call macros from more than one autocall library *;
*-------------------------------------------------------------*;
options mautosource sasautos=(file24,file25); ①
filename file24 "C:\SESUG14\IT\MYERS\mymacros\";
filename file25 "C:\SESUG14\IT\MYERS\mymacros2\";
***;
%calc21; *** called from file24 ***;
***;
%calc23; *** called from file25 ***;
```

STORED COMPILED MACRO LIBRARY

The third method to build a SAS macro library is to use a stored compiled macro library. When you tell SAS to use a stored compiled macro library, SAS will build a permanent catalog (or use existing catalog) named SASMACR in the directory that you choose. There can be only one catalog named SASMACR per directory but you can store many macros in that catalog.

Example 3.1 shows how to store macros (red text) in a stored compiled macro library. The LIBNAME statement ② gives the location where the library is stored. The options statement ① gives the MSTORED option which tells SAS to use the stored compiled macro library and the SASMSTORE option which tells SAS where to find the library. On the %MACRO statement ③, the STORE option tells SAS to save the compiled macro, the SOURCE option tells SAS to save the macro definition and the DES option allows you to give a short description for the macro.

```
*------------------------------------------------------------*;
* Example 3.1 Store macro in stored compiled macro library  *;
*------------------------------------------------------------*;
options dlcreatedir mstored sasmstore=lib31; ①
libname lib31 "C:\SESUG14\IT\MYERS\mymacros\"; ②
%macro calc31 / store source des="calculation 31"; ③
%put begin macro calc31;
***;
%put end macro calc31;
%mend calc31;
%macro calc32 / store des="calculation 32";
%put begin macro calc32;
***;
%put end macro calc32;
%mend calc32;
%macro calc33 / store source des="calculation 33";
%put begin macro calc33;
***;
%put end macro calc33;
%mend calc33;
%macro calc34 / store des="calculation 34";
%put begin macro calc34;
***;
%put end macro calc34;
%mend calc34;
```

Example 3.2 shows how to call a SAS macro from a stored compiled macro library. The libname statement ② gives the location of the library. The options statement ① tells SAS that you want to use a stored compiled macro library and where to look for it.

```
*------------------------------------------------------------*;
* Example 3.2 Call macro from stored compiled macro library  *;
*------------------------------------------------------------*;
options mstored sasmstore=lib32; ①
libname lib32 "C:\SESUG14\IT\MYERS\mymacros\"; ②
***;
%calc31;
```

You can have more than one stored compiled macro library.

Example 3.3 stores a macro (red text) in a separate directory “mymacros2” ① for the next example. Again, it is not necessary to use SAS for this job. You could use your text editor to do this.

```
*------------------------------------------------------------*;
* Example 3.3 Store macro in stored compiled macro library  *;
*------------------------------------------------------------*;
options dlcreatedir mstored sasmstore=lib35;
libname lib35 "C:\SESUG14\IT\MYERS\mymacros2\"; ①
%macro calc35 / store source des="calculation 35";
%put begin macro calc35;
***;
%put end macro calc35;
%mend calc35;
```

Example 3.4 shows how to use more than one stored compiled macro library. In the LIBNAME statement ②, you give a list of LIBREF’s. In the SASMSTORE option ①, you give the LIBREF which contains the list of other LIBREF’s.

```
*-------------------------------------------------------------------*;
* Example 3.4 Using more than one stored compiled macro library  *;
*-------------------------------------------------------------------*;
options mstored sasmstore=lib38; ①
libname lib36 "C:\SESUG14\IT\MYERS\mymacros\";
libname lib37 "C:\SESUG14\IT\MYERS\mymacros2\";
libname lib38 (lib36 lib37); ②
***;
%calc34; *** called from lib36 ***;
***;
%calc35; *** called from lib37 ***;
```

CATALOG PROCEDURE

The CATALOG procedure provides functions that you can use to manage SAS catalogs. This procedure allows you to copy, rename or delete entries in a SAS catalog. It also has a CONTENTS statement which allows you to print the names of the entries in a SAS catalog.

Example 4.1 shows how to use the CONTENTS statement to print the names of the macros in a stored compiled macro library.

```
*------------------------------------------------------------*;
* Example 4.1 List entries in SASMACR using proc catalog *;
*------------------------------------------------------------*;
libname lib41 "C:\SESUG14\IT\MYERS\mymacros\";
proc catalog;
contents catalog=lib41.sasmacr;
quit;
```

Output from SAS proc CATALOG is shown in Figure 1. This report gives the names of the macros stored on the catalog but it does not tell you which macros have stored macro definitions (source code).

```
                                The SAS System

                          Contents of Catalog LIB41.SASMACR

#    Name      Type              Create Date           Modified Date     Description
-------------------------------------------------------------------------------------
1    CALC31    MACRO    27Jul14:21:10:59       27Jul14:21:10:59       calculation 31
2    CALC32    MACRO    27Jul14:21:10:59       27Jul14:21:10:59       calculation 32
3    CALC33    MACRO    27Jul14:21:10:59       27Jul14:21:10:59       calculation 33
4    CALC34    MACRO    27Jul14:21:10:59       27Jul14:21:10:59       calculation 34
```

Figure 1. Output from SAS procedure CATALOG

%COPY STATEMENT

You can use the %COPY statement to print a macro definition from a stored compiled macro library to the SAS log or to an external file.

Example 4.2 shows how to print macro definition on SAS log with %COPY.

```
*-------------------------------------------------------*;
* Example 4.2 Print macro definition on SAS log        *;
*-------------------------------------------------------*;
libname lib42 "C:\SESUG14\IT\MYERS\mymacros\";
options mautosource sasmstore=lib42;
%copy calc31 / source;
```

MACRO FINDSOURCE

The macro FINDSOURCE will print the names of macros in a stored compiled macro library similar to proc CATALOG. It also shows which macro definitions were stored in the library. Source code for this macro is given in the Appendix of this paper.

Example 4.3 shows how to use macro FINDSOURCE. The %INCLUDE statement ② gives the location of the FINDSOURCE macro definition. The parameter MACROLIB in the macro FINDSOURCE ③ gives the location of the the stored compiled macro library ①.

```
*-------------------------------------------------------*;
* Example 4.3 List macros in SASMACR using FINDSOURCE *;
*-------------------------------------------------------*;
libname lib43 "C:\SESUG14\IT\MYERS\mymacros\"; ①
%include "C:\SESUG14\IT\MYERS\findsource.sas"; ②
%findsource(macrolib=lib43); ③
```

Output from macro FINDSOURCE is shown in Figure 2. This report gives a list of the macros on the stored compiled macro library and it tells you which macros have stored macro definitions (source code).

```
                                    The SAS System

                              Contents of Catalog lib43.sasmacr

#   Name   Type   Create Date          Modified Date         Description    Source

1  CALC31  MACRO  27Jul14:21:10:59     27Jul14:21:10:59     calculation 31   yes
2  CALC32  MACRO  27Jul14:21:10:59     27Jul14:21:10:59     calculation 32   no
3  CALC33  MACRO  27Jul14:21:10:59     27Jul14:21:10:59     calculation 33   yes
4  CALC34  MACRO  27Jul14:21:10:59     27Jul14:21:10:59     calculation 34   no
```

Figure 2. Output from macro FINDSOURCE

SEARCH SEQUENCE

Different methods of storing SAS macros can be used in the same SAS program. The sequence below gives the order of the search when a macro is called.

Search sequence for macro call:

1. Macros compiled during the current session (work.sasmacr)
2. If options MSTORED and SASMSTORE=mylib are used, then the stored compiled macro library (mylib.sasmacr)
3. If options MAUTOSOURCE and SASAUTOS=(myfile) are used, then the autocall library (myfile)
4. the stored compiled macros in SASHELP library (sashelp.sasmacr)

SUMMARY OF METHODS

The summary SAS options, statements and locations for each method is given in table 1.

#	Method	System Options to store	System Options to recall	Macro Options to store	Statements	Location of macro definition	Location of compiled macro
1	%Include File				Filename myfile "Filename"; or Filename myfile "Directory";	Directory	work.sasmacr
2	Autocall Library		mautosource sasautos= myfile		Filename myfile "Directory";	Directory	work.sasmacr
3	Stored Compiled Macro Library	mstored sasmstore= mylib	mstored sasmstore= mylib	store source	Libname mylib "Directory";	Directory or mylib.sasmacr	mylib.sasmacr

Table 1. Summary SAS options, statements and locations for each method.

CONCLUSION

SAS macro libraries give you methods that you can use to store and reuse SAS macros. These methods allow you to save time by storing sections of code that can be used to develop new programs. They also allow you to share SAS macros with other SAS programmers. These methods can improve the quality of your programs by using code that has been previously tested and verified.

REFERENCES

Carpenter, Arthur L. (2002), “Building and Using Macro Libraries”, *Proceedings of the Twenty-seventh Annual SAS Users Group International Conference*, Paper 17-27, Cary, NC, SAS Institute

Carpenter, Arthur L. (2004), *Carpenter’s Complete Guide to the SAS Macro Language, Second Edition*, Cary, NC, SAS Institute

Heaton, Ed and Sarah Woodruff (2009), “Implementing User-Friendly Macro Systems”, Proceedings of the Southeast SAS Users Group, Atlanta, GA

Huang, Jie and Tracy Lin (2009), “SAS Macro Autocall and %Include”, Proceedings of the Southeast SAS Users Group, Atlanta, GA

Stojanovic, Mirjana, Dorothy Watson and Donna Hollis (2006), “A Practical Approach to the Stored Compiled Macro Facility in a Clinical Trial Environment”, Proceedings of the Southeast SAS Users Group, Atlanta, GA

CONTACT INFORMATION

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

Name: John M. Myers
Enterprise: Virginia Commonwealth University
Address: 800 East Leigh Street
City, State ZIP: Richmond, VA 23219
Work Phone: 804-828-8108
E-mail: jmmyers@vcu.edu

TRADEMARK INFORMATION

APPENDIX: SOURCE CODE FOR MACRO FINDSOURCE

```
%*------------------------------------------------------------*;
%* Macro findsource: List entries in sasmacr with source info  *;
%*------------------------------------------------------------*;

%macro findsource(macrolib=mylib);

%local mname nname ix mexist rc xname work;

%*----------------------------------------------------*;
%* build table of macro names using proc catalog       *;
%*----------------------------------------------------*;

ods output Catalog_Random=Catalog_Random;

ods listing close;

proc catalog ;
contents catalog=&macrolib..sasmacr;
quit;

ods listing;

%*----------------------------------------------------*;
%* create macro variable containing macro names        *;
%*----------------------------------------------------*;
```

```
data _null_;

length mvar nvar nval $8.;
length mval $2000.;

nname = 0;
mval = " ";

do until(eof=1);
set Catalog_Random end=eof;
nname = nname + 1;
mval = trim(left(mval)) || " " || trim(left(objname));
end;

mvar = "mname";
call symput(mvar,mval);

nvar = "nname";
nval = trim(left(put(nname,9.)));
call symput(nvar,nval);

stop;
run;

%put nname= &nname;
%put mname= &mname;

%*----------------------------------------------------*;
%* get work directory                                 *;
%*----------------------------------------------------*;

%let work = %sysfunc(getoption(work));

%put work= &work;

filename tempfile "&work/tempsource.txt";

%*----------------------------------------------------*;
%* find source using copy                             *;
%*----------------------------------------------------*;

%let mexist = ;

%let rc = %sysfunc(fexist(tempfile));
%if &rc = 1 %then %do;
  %let rc = %sysfunc(fdelete(tempfile));
  %end;

%do ix = 1 %to &nname;

%let xname = %scan(&mname,&ix);

%copy &xname / library=&macrolib source out=tempfile;

%let rc = %sysfunc(fexist(tempfile));
%if &rc = 1 %then %do;
  %let mexist = &mexist yes;
  %let rc = %sysfunc(fdelete(tempfile));
  %end;
%else %let mexist = &mexist no;

%end;

%let rc = %sysfunc(fexist(tempfile));
%if &rc = 1 %then %do;
  %let rc = %sysfunc(fdelete(tempfile));
```

```
  %end;

%put nname= &nname;
%put mexist= &mexist;

%*-------------------------------------------------------*;
%* build merge data set                                  *;
%*-------------------------------------------------------*;

data sourceinfo;
length mval $2000.;
length mext $2000.;
length macname $20.;
label macname = "Name";
label source = "Source";
keep macname source;
mval = trim(left("&mname"));
mext = trim(left("&mexist"));
do ix = 1 to &nname;
macname = scan(mval,ix);
source  = scan(mext,ix);
output;
end;
stop;
run;

%*-------------------------------------------------------*;
%* create merge variable macname                         *;
%*-------------------------------------------------------*;

data Catalog_Random;
set Catalog_Random;
length macname $20.;
label macname = "Name";
macname = objname;
run;

%*-------------------------------------------------------*;
%* merge catalog and copy info                           *;
%*-------------------------------------------------------*;

data catalog_report;
merge Catalog_Random (in=in1) sourceinfo (in=in2);
by macname;
run;

%*-------------------------------------------------------*;
%* print catalog report with source info                 *;
%*-------------------------------------------------------*;

title3 "Contents of Catalog &macrolib..sasmacr";

proc print data=catalog_report noobs label;
var num macname type crdate moddate desc source;
run;

title3;

%mend findsource;
```