Paper 143-2012

AUTOCALL MACRO LIBRARIES

Ken Schmidt, Truven Health Analytics, Ann Arbor, MI

ABSTRACT

The power of SAS Macros can speed up programming time, eliminate errors, as well as insure consistency of assumptions. The presentation will include how to set up a macro directory, how to assign macro names, and provide general descriptions of some of the most useful macros used in a library.

INTRODUCTION

Placing your macros in a library will help to organize your programs and provide a consistent location for all macro definitions. For large projects or in organizations with multiple programmers sharing macros, it would be efficient to set up a standard operating procedure for the macro library. The development of a macro library takes planning, but can be a valuable resource for your organization.

UTILIZING MACRO LIBRARIES

If you write more than the occasional macro, or if you share macros with colleagues, or if you ever define the same macro in different programs/places, you should be using macro libraries. Macro libraries provide the ability to remove the macro definition (%MACRO to %MEND statements) from your programs. By placing the macro definition in a library, other programmers in your group can have access to the same macro definitions. Libraries allow you to effectively share your macro definitions without copying and storing them in multiple locations.

There are three basic types of macro libraries:

- %INCLUDE
- Autocall – macro definitions (%MACRO to %MEND) are stored as code.
- Stored Compiled – compiled macros are stored in permanent catalogs.

Each of the above forms has value and each is worth knowing; however, if you only learn one type, learn to use the autocall macro library. This library is most often used, and it has a number of advantages over the other two forms of libraries.

A macro is defined through the use of the %MACRO and %MEND statements. When these statements are executed, the macro facility performs a macro compilation. This is not really a true compilation, and is little more than a check on macro syntax. The compiled macro definition is then written to a SAS catalog, with an entry type of MACRO. The default catalog is WORK.SASMACR and the entry name will be the name of the macro itself.

SEE ALSO Building and Using Macro Libraries, Chapter 12 in Carpenter's Complete Guide to the SAS Macro Language, 2nd Edition

ESTABLISHING AN AUTOCALL LIBRARY

By default an autocall macro library is automatically made available. This library contains a fairly extensive collection of macros that are provided with SAS. These include macros such as %LEFT, %VERIFY, and %QTRIM.

Two system options, MAUTOSOURCE and SASAUTOS=, are used to control the use of the autocall library. The ability to access an autocall library is turned on with the MAUTOSOURCE system option (by default this option is on). The physical location of the autocall library is specified using the SASAUTOS= system option (example below).

```
OPTIONS MAUTOSOURCE SASAUTOS=SASAUTOS;
```

By default the SASAUTOS= option's value is an automatic composite *fileref* also named SASAUTOS. This *fileref* points to various locations (which locations and how many depends to some extent on your release of SAS, and the products that you lease). These locations are used to house the autocall macro definitions that are supplied by SAS.

You may add your own macro definitions to the autocall library by storing them in one or more locations and then by adding those locations to the SASAUTOS= option. Notice the use of the FILENAME statement, *not* a LIBNAME statement. Under directory-based systems the *fileref* will point to the directory level (example below).

FILENAME MY_MACS "<physical path to *my* macro definitions>";

FILENAME PRJ_MACS "<physical path to the *project* macro definitions>";

FILENAME ORG_MACS "<physical path to the *organization* macro definitions>";

OPTIONS MAUTOSOURCE SASAUTOS=(MY_MACS PRJ_MACS ORG_MACS SASAUTOS);

The only further constraint is that the macro name must match the name of the file that contains the definition. If you were to create the definition for the macro %ABC, the %MACRO ABC statements through the %MEND ABC statement, would be stored in a file named ABC.SAS. On the UNIX OS, the name of the file that stores the macro definition must be in all lowercase characters (abc.sas).

When the %ABC macro is called, SAS will search for the program ABC.SAS in the locations (left to right) specified in the SASAUTOS=option. Once the file is found, the macro definition is included, the %MACRO to %MEND macro definitions compiled, and then the %ABC macro is executed.

While it is possible for the file containing the macro definition to contain code other than just the %MACRO through the %MEND statements, it is not a good idea to do so. By segregating the code so that a given file contains only the definition for the macro for which it is named, macro definitions become much easier to find, and control.

There are a couple of caveats to be aware of when using autocall libraries. First, be very careful to include the automatic *fileref* SASAUTOS. Failure to do so results in the loss of the ability to use autocall macros supplied by SAS. Secondly, be sure to specify the library locations using *filerefs* and not *librefs.* Use the FILENAME statement even though you are pointing to a location and not to a specific file. Although no error is issued when the SASAUTOS= option is specified using a *libref*, the use of a *libref* will cause problems when the library is accessed.

TRACING AUTOCALL MACRO LOCATIONS

As discussed above in establishing an autocall library, it is not uncommon to have an autocall library point to several locations. When a macro is called, SAS searches each location in turn and executes the first copy of the macro that is encountered. You may need to know which location contains the code for the called macro. The MAUTOLOCDISPLAY system option, the default is NOMAUTOLOCDISPLAY, will write the physical location of the macro's definition, whenever a macro is retrieved from an autocall library and is subsequently used.

In the example below the definition for the %OBSCNT macro resides in the directory shown for the *fileref* MYMACS. Each time the macro is called the LOG shows the path to the program (OBSCNT.SAS) containing the macro definition.

FILENAME MY_MACS "&path\sascode\sasmacros";

OPTIONS MAUTOSOURCE SASAUTOS=(MY_MACS SASAUTOS) MAUTOLOCDISPLAY ;

LOG Output:

MAUTOLOCDISPLAY(OBSCNT) : This macro was compiled from the autocall file &path\sascode\sasmacros\obscnt.sas

USING STORED COMPILED MACRO LIBRARIES

Stored Compiled Macro Libraries are only available when turned on with the MSTORED system option. The SASMSTORE= option is then used to allocate the stored compiled macro library. Although the SASMSTORE option accepts only one *libref* can be a concatenated or composite library (example below).

```
 LIBNAME COMPLIB “&path\sascode\storedmacros”;

OPTIONS MSTORED SASMSTORE=COMPLIB;
```

If a stored compiled macro library is available, the /STORE option on the %MACRO statement can be used to direct the compiled macro to the permanent COMPLIB.SASMACR catalog (example below).

```
%MACRO DEF / STORE;

  %put Stored compiled Version of DEF;

%MEND DEF;
```

MACRO LIBRARY SEARCH ORDER

Understanding the macro library search order is crucial to understanding which version of a macro will be executed. When a macro, such as the %ABC macro, is called, SAS must search for the macro’s definition. SAS first looks for the ABC.MACRO entry in the WORK.SASMACR catalog. Then, assuming that is not found in the WORK catalog, and if stored compiled macro libraries are turned on, a search is made for the ABC.MACRO entry in each SASMACR catalog in the *libref* designated by the SASMSTORE= system option. Finally, if a compiled entry has not yet been found, SAS starts a search in the autocall library locations for a program with the name of ABC.SAS. In summary the search order is:

1. WORK.SASMACR
2. Stored compiled macro libraries (COMPLIB.SASMACR in the prior section example)
3. Autocall macro libraries

CONCLUSION

Macro libraries allow the macro developer to store, maintain, and manage large numbers of macros. Using macro libraries is not difficult nor is it overly complicated. With a well-designed macro library your organization will see advantages in consistency along with a central knowledge repository.

A macro library can be an effective time-saving resource for the programmer and analysts, who can avoid the inefficiency of recreating existing macros by accessing them from the library. With a tested and validated macro library, you can do your work more efficiently and with a high rate of productivity.

The responsibility to maintain, add, or approve new macros will depend on your operating environment and organizational goals. Some organizations could establish an approval team and macro approval process. This team is responsible for developing a library’s macro directory structure, standard operating procedures and user documentation.

RECOMMENDED READING

Building and Using Macro Libraries, Chapter 12 in Carpenter’s Complete Guide to the SAS Macro Language, 2nd Edition

REFERENCES

SAS 9.2 Macro Language: Reference, http://support.sas.com, SAS Institute Inc.

Art Carpenter, Carpenter's Complete Guide to the SAS Macro Language, 2nd Edition, SAS Publishing

Kirk Lafler, "SAS Macro Programming Tips and Techniques – Hands-on Workship, SAS Global Forum 2009"

Burlew, Michele, SAS Macro Programming Made Easy, 2nd Edition, SAS Publishing

COPYRIGHT AND PERMISSIONS

CONTACT INFORMATION

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

Name:	Ken Schmidt
Company:	Truven Health Analytics
Address:	777 East Eisenhower Parkway
City, State ZIP:	Ann Arbor, MI 48108
Work Phone:	734.913.3924
E-mail:	ken.schmidt@truvenhealth.com
Web:	www.truvenhealth.com

TRADEMARK INFORMATION