Paper 064-2012

Assigning a User-Defined Macro to a Function Key

Mary F. O. Rosenbloom, Edwards Lifesciences, LLC, Irvine, California
Kirk Paul Lafler, Software Intelligence Corporation, Spring Valley, California

ABSTRACT

Are you entering one or more of the same SAS® Display Manager System (DMS) commands repeatedly during a session? The DMS offers a convenient way of capturing and saving frequently entered commands in a user-defined macro, and then saving the macro as a function key of your choosing. Are you typing SAS® code for data exploration during program development or validation, only to delete it soon afterwards? If you are, then this code can be placed in a macro, too, and assigned to a function key. This paper illustrates the purpose and steps that you use to assign a user-defined macro to a function key.

INTRODUCTION

During development and validation, when we are in the "getting to know you" stage with the data, we write a lot of very basic code. While developing a program, we may want to run a frequency on all of the categorical variables to examine their values. During validation, one might open an empty program window and write several PROC PRINT calls or use DATA _NULL_ steps to examine the data. In both cases, the code is usually discarded. Another inefficiency that many of us still subscribe to is to either use the "point and click" method to clear the log and output windows, or to repeatedly issue these Display Manager System (DMS) commands throughout a session.

In fact, all of these processes can be placed into macro programs. We can write the macro programs in such a way that they need no input parameters. Then, we assign these macro calls to function keys and suddenly, efficiency is just a key stroke away!

STREAMLINING COMMAND-LINE DMS COMMANDS WITH A MACRO

The macro language is a powerful tool for streamlining frequently entered DMS commands to reduce the number of keystrokes. By embedding a series of DMS commands inside a simple macro, you'll not only save time by not having to repeatedly enter the commands over and over again, but you'll improve your productivity as well. The DMS commands illustrated in Figure 1 are inserted between the %MACRO and %MEND statements in lieu of entering them individually on a command line. The commands display and clear the output window, display, clear and expand the SAS Log to full size, and then position control in the Program Editor window.

```
%MACRO valpostsubmit;
  output; clear;
  log; clear;
  zoom;
  wpgm;
%MEND valpostsubmit;
```

Figure 1. User-defined VALPOSTSUBMIT Macro

ASSIGNING A USER-DEFINED MACRO TO A FUNCTION KEY

The process of reducing keystrokes and enhancing productivity is an important technique used by application developers, programmers, testers, and others. To illustrate the process of saving a macro call to a function key, the user-defined macro, VALPOSTSUBMIT, will be assigned to function key F12 in the KEYS window. To access the KEYS window, simply issue the KEYS command on the command line, or click on TOOLS → OPTIONS → KEYS in the menu bar. Once the KEYS window is open, the desired key is selected from the list and the macro call is entered next to it. The settings are automatically saved when the KEYS window is closed. A partial KEYS window is displayed in Figure 2 to illustrate the process.

Key	Definition
F1	help
F2	reshow
F3	end;
…	…
F10	keys
F11	command focus
F12	**%VALPOSTSUBMIT**

Figure 2. KEYS Window

A WORD ON NOMENCLATURE

When the suggestions from this paper are put into practice, the first thing that one must do is to compile all of the macros that are assigned to the function keys. This can be done either by submitting a SAS program or through an autoexec.sas file. It is extremely important to ensure that the macros defined here do not have the same name as any macros that will be used in the SAS programs being developed or validated. In our examples, all macros and macro variables (global or local) start with the letters "VAL" (short for validation). This choice is arbitrary, but the idea is that within a working group, programmers will agree to designate "VAL" macros for this temporary use only, and will avoid using this nomenclature in production.

CLEARING THE RESULTS VIEWER IN SAS 9.3

The macro VALPOSTSUBMIT can be used to clear the log and output window, but in version 9.3 of SAS, HTML is the default output destination, and while we are able to view graphs and output in this enhanced way, the method for clearing the results that we create is now different. A SAS note was issued on this topic. The SAS Institute suggests closing the HTML destination, and then reopening it using one's registry settings:

```
ods html close;
ods preferences;
```

Figure 3. SAS-suggested Method for Clearing the Results Viewer.

After this code is issued, the results viewer will appear to be unchanged. However, once new output is created, it will be displayed and the old output will no longer be visible.

Rick Wicklin summarized the results of a SAS-L discussion on his blog, suggesting that this code could be used to clear the results viewer:

```
ods html close;
ods html;
```

Figure 4. Rick Wicklin's Suggested Method for Clearing the Results Viewer.

Then Ken Kleinman took that one step further, using GSUBMIT:

```
gsubmit "ods html close; ods html;"
```

Figure 5. Ken Kleinman's Suggested Method for Clearing the Results Viewer.

In this case, the entire statement, including GSUBMIT, is assigned to one of the function keys.

One might want to place this code into a macro, and then assign a function key to call that macro:

```
%macro valClOpHTML;
       log; clear; wpgm;
       submit '
              ods html close;
              ods preferences;
       ';
%mend valClOpHTML;
```

Figure 6. User-defined VALOPCLHTML Macro

The macro, VALOPCLHTML, clears the log and returns to the enhanced editor, but it earns its name by closing ('Cl') and then re-opening ('Op') then HTML destination. When ODS PREFERNCES is specified, the defaults from the user registry are restored. For most users this will, among other things, turn the HTML destination back on, and the for the HTML destination set NEWFILE to "none" (which stacks the results into one document), and change the ODS path back to the work directory. In a moment we will discuss a macro that sets NEWFILE to "proc". If both macros are used in the same SAS session, re-setting this option in this macro becomes important.

Sometimes we want to see only the most recently created results in the results viewer, rather than the cumulative output. The macro VALOPCLHTML, shown in Figure 7, achieves this goal:

```
%macro valHTML1proc;
       odsresults; clear; log; clear; wpgm;
       submit '
          ods html path = "%trim(%sysfunc(pathname(work)))" newfile=proc;
       ';
%mend valHTML1proc;
```

Figure 7. User-defined VALHTML1PROC Macro

This macro differs from VALOPCLHTML in that the HTML destination is not closed. Instead, we issue a DM command to clear the ODS results, which removes the output from the results window (on the left side of most SAS sessions, not to be confused with the results viewer). We issue a command to ensure that the new results will be sent to the work directory using ODS HTML PATH, and we set NEWFILE to "proc" so that a new HTML file is created every time procedural output is generated. This way the results viewer will always show the most recent output, and will not allow the results to stack together.

Together, the macros VALOPCLHTML and VALHTML1PROC can be used to clear the log and effectively clear the results viewer, and to set user preferences for the results viewer to their desired states.

PRINTING THE FIRST TEN OBSERVATIONS FROM THE MOST RECENT DATASET

Often during development and validation of programs, we would like to view a small number of records in a dataset. The macro VAL10OBS will print the first ten observations of the most recently created dataset, shown in Figure 8.

```
%macro val10obs;
  submit
    "proc print data=&syslast (obs=10);
    run;";
%mend val10obs;
```

Figure 8. User-defined VAL10OBS macro

Here, we take advantage of &SYSLAST, which is an automatic macro variable which stores the name and libref of the most recently created dataset in the form *libref.dataset*. This enables us to write the macro with no input parameters, so that it can be assigned to a function key. The full code for this example is shown in the appendix. The built-in dataset SASHELP.CLASS is used. The dataset is processed with a PROC SORT step, and output at the temporary dataset _CLASS, immediately prior to calling this macro with its assigned function key. This ensures that the value of &SYSLAST corresponds to the dataset that we want to print with this macro.

CHECKING THE MOST RECENT DATASET FOR DUPLICATES IN 'BY' VARIABLES

It is often necessary to check a dataset for repeats of 'BY' variables. To do this, we must first ensure that the dataset of interest is sorted by the desired variables, in the proper sequence, and that this was the last dataset created. Then, VALDUPS can be used to check for and then print to the log the first 10 instances of duplicates, see Figure 9.

```
%macro valdups;
  submit
    "***store the most recently used dataset name as a macro variable;
    %let vallastdsn=&syslast;

    ***obtain a dataset containing variables by which the dataset is sorted;
    proc contents data=&vallastdsn out=_valcontents (keep=name sortedby
                 where=(sortedby ne .)) noprint;
    run;

    ***sort the 'by variable' dataset by the sort order;
    proc sort data=_valcontents;
      by sortedby;
    run;

   (Continued on next page)
```

Figure 9. User-defined VALDUPS Macro

```
    (Continued from previous page)

    proc sql noprint;
      ***create a macro variable containing list of by variables;
      select distinct name into :valby separated by ' ' from _valcontents;

      ***create a list of by variables separated by equal signs;
      select distinct name into :valequal separated by '=/ ' from _valcontents;
    quit;

    ***store the last BY variable in a macro;
    data _null_;
      set _valcontents end=last;
      if last then call symput('vallastby',name);
    run;";

  submit
    "***check original dataset for duplicates and send first 10 dups to log;
    data _null_;
      set &vallastdsn;
        by &valby;
      ***keep duplicates;
      if first.&vallastby ne last.&vallastby;
      ***print first 10 duplicates to log;
      if _n_ le 10 then put 'WARNING: Duplicate records exist'/ &valequal.=/;
    run;

    ***delete temporary dataset;
    proc datasets lib=work memtype=data nolist;
      delete _valcontents;
    quit;";
%mend valdups;
```

Figure 9. User-defined VALDUPS Macro (continued)

So far, when we have included DATA steps and PROC steps in the macros, we have had to use a SUBMIT statement, wrap the code in quotes, and then end with a semicolon. In this example, the code exceeded 960 characters, so two submit statements were needed.

PRINTING A SUBSET OF THE DATA

In their book *Validating Clinical Trial Data Reporting with SAS*®, Matthews and Shilling suggest setting a macro variable to a subset of records that you would like to keep an eye on throughout the program, and then printing datasets throughout the program using a WHERE statement referencing the macro variable (Matthews and Shilling 2008, p.81), see Figure 10.

```
%macro valprint;
  submit
    "proc print data=&syslast;
      where &valwhere;
     run;";
%mend valprint;

***run a PROC SORT, so that SYSLAST will refer to the sashelp.class data
(now called _class, but with the same contents);
proc sort data=sashelp.class out=_class;
  by age sex;
run;

***assign a value to VALWHERE - to be used in the WHERE statement;
%let valwhere=%str(name in ('Jane', 'John', 'Joyce'));
%put &valwhere;

***now press the function key to which %VALPRINT is assigned;
```

Figure 10. User-defined VALPRINT Macro

Our objective is to print the records for Jane, John, and Joyce. This is done after each data step in the program during development or validation, to ensure that these special records are handled correctly. Above, the PROC SORT is again used to create the dataset WORK._CLASS, which will automatically be assigned to &SYSLAST. When VALPRINT is called (using a function key) the results will be displayed in the log, as shown in Figure 11.

```
MPRINT(VALPRINT):   proc print data=WORK._CLASS ;
MPRINT(VALPRINT):   where name in ('Jane', 'John', 'Joyce');
MPRINT(VALPRINT):   run;

NOTE: There were 3 observations read from the data set WORK._CLASS.
      WHERE name in ('Jane', 'John', 'Joyce');
NOTE: PROCEDURE PRINT used (Total process time):
```

Figure 11. SAS Log Results from user-defined VALPRINT Macro

REVIEWING THE VALUES OF CHARACTER VARIABLES

In *Cody's Data Cleaning Techniques Using SAS®, Second Edition,* Ron Cody suggests using the _CHARACTER_ keyword with PROC FREQ to examine the values of the character variables in a dataset (Cody 2008, p.6). One might use this macro multiple times while working with a program that incorporates several datasets. As you would expect, this can be placed in a macro, too, taking advantage once again of &SYSLAST, illustrated in Figure 12.

```
%macro valchar;
  submit
    "proc freq data=&syslast;
       tables _character_/ nocum nopercent;
     run;";
%mend valchar;
```

Figure 12. User-defined VALCHAR Macro

SHOWING EXTREME NUMERIC VARIABLE VALUES WITH PROC UNIVARIATE AND ODS

Cody also suggests using PROC UNIVARIATE with ODS SELECT to review the extreme values for numeric variables (Cody 2008, p.34). In VALNUM, we use PROC CONTENTS to create a dataset of non-date numeric variables. Then PROC SQL is used to store the variable names in a list using a macro variable. The macro is referenced in the VAR statement in PROC UNIVARIATE.

```
%macro valnum;
  submit
    "***store the most recently used dataset name as a macro variable;
     %let vallastdsn=&syslast;

     ***obtain a dataset containing only the numeric non-date variables;
     proc contents data=&vallastdsn out=_valcontents
                 (where=(type=1 and index(format,'Date') eq 0))  noprint;
     run;

     ***create a macro variable containing a list of numeric variables;
     proc sql noprint;
       select distinct name into :valnumlst
       separated by ' ' from _valcontents;
     quit;

     ***use ODS and PROC UNIVARIATE to review extreme observations;
     ods select extremeobs;
     title 'Use PROC UNIVARIATE to Review Extreme Observations';
     proc univariate data=&vallastdsn;
       var &valnumlst;
     run;
     title;

    (Continued on next page)
```

Figure 13. User-defined VALNUM Macro

```
    (Continued from previous page)

     ***delete temporary dataset;
     proc datasets lib=work memtype=data nolist;
       delete _valcontents;
     quit;";
%mend valnum;
```

Figure 13. User-defined VALNUM Macro (continued)

A CHEAT SHEET MIGHT BE HANDY

As the number of user-defined macros and macro function keys increases, users might find it helpful to either create a reference (cheat) sheet for themselves, or to keep the KEYS window open. Assigning descriptive macro names and keeping the list parsimonious will help in maintaining a more manageable system.

CONCLUSION

There are a multitude of function keys ready and waiting for an assignment within the SAS System. The examples shown here are meant to inspire each user to develop a system that is useful and efficient for their situation. Do this right, and in no time you too will be pressing "The Easy Button".

ACKNOWLEDGMENTS

The authors would like to thank Andy Kuligowski, SAS Global Forum 2012 Conference Chair, and Sue Douglass and Erik Tilanus, Coder's Corner Chairs for accepting our abstract and paper, as well as for a great conference!

REFERENCES

Burlew, Michele M. (1998), SAS Macro Programming Made Easy, SAS Institute Inc., Cary, NC, USA.

Carpenter, Art (2004), Carpenter's Complete Guide to the SAS Macro Language, Second Edition. SAS Institute Inc., Cary, NC, USA.

Cody, Ron (2008), Cody's Data Cleaning Techniques Using SAS®, SAS Institute Inc., Cary, NC, USA.

Kleinman, Ken. Clear the Results Viewer in SAS 9.3, available at http://kenkleinman.net/home/index.php/sas-and-r-code/sas-tricks/77-clear-the-results-viewer-in-sas-93.html.

Lafler, Kirk Paul (2009), *"Building Reusable and Highly Effective Tools with the SAS® Macro Language,"* PharmaSUG 2009 Conference, Software Intelligence Corporation, Spring Valley, CA, USA.

Lafler, Kirk Paul (2009), *"SAS Macro Programming Tips and Techniques,"* Proceedings of the NorthEast SAS Users Group (NESUG) 2009 Conference, Software Intelligence Corporation, Spring Valley, CA, USA.

Matthews, Carol I. and Shilling, Brian C. (2008) Validating Clinical Trial Data Reporting with SAS®, SAS Institute Inc., Cary, NC, USA.

Rosenbloom, Mary F. O. and Lafler, Kirk Paul (2011), "*Assigning a User-defined Macro to a Function Key*," Proceedings of the Western Users of SAS Software (WUSS) 2011 Conference and Proceedings of the South Central SAS Users Group, Edwards Lifesciences, LLC, Irvine, CA and Software Intelligence Corporation, Spring Valley, CA, USA.

Wicklin, Rick. How to Clear the Output Window in SAS 9.3, available at http://blogs.sas.com/content/iml/2011/08/29/how-to-clear-the-output-window-in-sas-9-3/.

SAS usage note on writing ODS output to the work directory available at http://support.sas.com/kb/45/625.html.

SAS usage note on clearing the results window in 9.3, available at http://support.sas.com/kb/43/911.html.

SAS documentation on SYSLAST, available at http://support.sas.com/documentation/cdl/en/mcrolref/61885/HTML/default/viewer.htm#a000210371.htm.

CONTACT INFORMATION

Comments and suggestions can be sent to:

Mary F. O. Rosenbloom
Edwards Lifesciences, LLC
E-mail: mary.rosenbloom.sas@gmail.com

Kirk Paul Lafler
Software Intelligence Corporation
E-mail: KirkLafler@cs.com

Validation Macros

```
options mprint;

*******************************************************************************;
***Clear the log, return to the editor window;
%MACRO valpostsubmit;
  output; clear;
  log; clear;
  zoom;
  wpgm;
%MEND valpostsubmit;

*******************************************************************************;
***Clear the log, return to the editor window, close and reopen HTML (will clear
       the history in the results viewer after next results);
%macro valClOpHTML;
       log; clear; wpgm;
       submit '
              ods html close;
              ods preferences;
       ';
%mend valClOpHTML;

*******************************************************************************;
***Clear the log, return to the editor window, clear the results window (not the
       viewer), create one HTML file in work directory for each procedural output;
%macro valHTML1proc;
       odsresults; clear; log; clear; wpgm;
       submit '
          ods html path = "%trim(%sysfunc(pathname(work)))" newfile=proc;
       ';
%mend valHTML1proc;

*******************************************************************************;
***print the first 10 observations from the most recent dataset;
%macro val10obs;
  submit
    "proc print data=&syslast (obs=10);
    run;";
%mend val10obs;

***run a PROC SORT, so that SYSLAST will refer to the sashelp.class data (now
       called _class, but with the same contents);
proc sort data=sashelp.class out=_class;
  by age sex;
run;

***now press the function key to which VAL10OBS is assigned;
```

(Continued on next page)

(Continued from previous page)

```
*******************************************************************************;
***examine the most recent dataset for duplicates in "BY" variables;
%macro valdups;
  submit
    "***store the most recently used dataset name as a macro variable;
    %let vallastdsn=&syslast;

    ***obtain a dataset containing variables by which the dataset is sorted;
    proc contents data=&vallastdsn out=_valcontents (keep=name sortedby
                 where=(sortedby ne .)) noprint;
    run;
     ***sort the 'by variable' dataset by the sort order;
    proc sort data=_valcontents;
      by sortedby;
    run;

    proc sql noprint;
      ***create a macro variable containing list of by variables;
      select distinct name into :valby separated by ' ' from _valcontents;

      ***create a list of by variables separated by equal signs;
      select distinct name into :valequal separated by '=/ ' from _valcontents;
    quit;

    ***store the last BY variable in a macro;
    data _null_;
      set _valcontents end=last;
      if last then call symput('vallastby',name);
    run;";

  submit
    "***check original dataset for duplicates and send first 10 dups to log;
    data _null_;
      set &vallastdsn;
        by &valby;

      ***keep duplicates;
      if first.&vallastby ne last.&vallastby;

      ***print first 10 duplicates to log;
      if _n_ le 10 then put 'WARNING: Duplicate records exist'/ &valequal.=/;
    run;

    ***delete temporary dataset;
    proc datasets lib=work memtype=data nolist;
      delete _valcontents;
    quit;";
%mend valdups;

***run a PROC SORT, so that SYSLAST will refer to the sashelp.class data (now
       called _class, but with the same contents);
proc sort data=sashelp.class out=_class;
  by age sex;
run;

***now press the function key to which VALDUPS is assigned;
```

(Continued on next page)

(Continued from previous page)

```
*******************************************************************************;
***print a subset of the data;
%macro valprint;
  submit
    "proc print data=&syslast;
      where &valwhere;
    run;";
%mend valprint;

***run a PROC SORT, so that SYSLAST will refer to the sashelp.class data (now
       called _class, but with the same contents);
proc sort data=sashelp.class out=_class;
  by age sex;
run;

***assign a value to VALWHERE - to be used in the WHERE statement;
%let valwhere=%str(name in ('Jane', 'John', 'Joyce'));
%put &valwhere;
***now press the function key to which VALPRINT is assigned;

*******************************************************************************;
***review values of character variables;
%macro valchar;
  submit
    "proc freq data=&syslast;
      tables _character_/ nocum nopercent;
     run;";
%mend valchar;

***run a PROC SORT, so that SYSLAST will refer to the sashelp.class data (now
       called _class, but with the same contents);
proc sort data=sashelp.class out=_class;
  by age sex;
run;

***now press the function key to which VALCHAR is assigned;

*******************************************************************************;
***Use PROC UNIVARIATE with ODS to show extreme values;
%macro valnum;
  submit
    "***store the most recently used dataset name as a macro variable;
    %let vallastdsn=&syslast;

    ***obtain a dataset containing only the numeric non-date variables;
    proc contents data=&vallastdsn
                  out=_valcontents
                    (where=(type=1 and index(format,'Date') eq 0))  noprint;
    run;

    ***create a macro variable containing a list of numeric variables;
    proc sql noprint;
      select distinct name into :valnumlst separated by ' ' from _valcontents;
    quit;

    ***use ODS and PROC UNIVARIATE to review extreme observations;
    ods select extremeobs;
    title 'Use PROC UNIVARIATE to Review Extreme Observations';
    proc univariate data=&vallastdsn;
      var &valnumlst;
    run;
    title;
```

(Continued on next page)

```
(Continued from previous page)

    ***delete temporary dataset;
    proc datasets lib=work memtype=data nolist;
      delete _valcontents;
    quit;";

%mend valnum;

***run a PROC SORT, so that SYSLAST will refer to the sashelp.class data (now
       called _class, but with the same contents);
proc sort data=sashelp.class out=_class;
  by age sex;
run;

***now press the function key to which VALNUM is assigned;

***end of program***;
```