Custom useful SAS functions using the PROC FCMP procedure

Qixia Shi, Guanyu,su,Fountain Medical Technology Co., ltd, Nanjing, China

ABSTRACT

As a SAS programmer, we usually do some repetitive programming work. Usually, we use DATA STEP or macros to ease writing repetitive program. However, DATA STEP are difficult to be reused in other programs, it will be useful and time-saving if small program units can be reused. Macros are easier to be reused but they are not independent from the main program, they involve non-DATA step syntax, and reuse macros can result in a large number of intermediary data sets being generated .It would be great if we could circumvent the issue of macros generated, as all we really want is one result. Now, multiple subroutines and functions can be declared by using PROC FCMP procedure in SAS 9.2 or higher version. Thanks to the PROC FCMP procedure and RUN_MACRO function, now we can custom independent and reusable subroutines, this enables programmers to read, write, and maintain complex code more easily.

This paper will present some useful functions for handing calculations, standardizing and simplifying code created by using PROC FCMP procedure, using these functions we can use only one statement to achieve what we want, that can reduce much repetitive work and save time.

INTRODUCTION

PROC FCMP provides the ability to write true functions and CALL routines using DATA step syntax. In SAS 9.2 or higher version, FCMP routines can be called like any other SAS function.

SAS macros are designed to generate output and data sets, not values. Macros could only contain macro statements, as they had to resolve their ‘return’ value. Now macros can wear the clothes of a PROC FCMP function, this function executes the macro, and returns a single value to the DATA step.

The first section of this paper describes the syntax and features of PROC FCMP.
The second section presents several examples of creating a FCMP function and calling it from a DATA step. The third section describes how to create and call RUN_MACRO Functions.

1. PROC FCMP SYNTAX.

PROC FCMP can be invoked to create functions and CALL routines. The syntax for the procedure is:

```
PROC FCMP OUTLIB=libname.dataset.package ;①
    routine-declarations;
```

①The OUTLIB= option is required and specifies the package where routines declared in the routine-declarations section are stored.

DECLARING FUNCTIONS:

```
FUNCTION name (parameter-1, …, parameter-N);②
    program-statements;
    RETURN (expression);③
ENDSUB;④
```

②The FUNCTION statement names the function and identifies its incoming arguments. To specify a string parameter, a dollar sign ($) must be placed after the parameter name.

③The RETURN statement identifies the value that is to be returned by the function.

④The ENDSUB statement closes the function definition.

A given PROC FCMP step can contain multiple function definitions.

DECLARING CALL ROUTINES:

CALL routines are also declared within routine-declarations by using the SUBROUTINE keyword instead of the FUNCTION keyword. Functions and CALL routines have the same form, except CALL routines do not return a value and can modify their parameters. The parameters to be modified are specified in an OUTARGS statement. The syntax of a CALL routine declaration is:

```
SUBROUTINE name (parameter-1, …, parameter-N);①
    OUTARGS out-parameter-1, …, out-parameter-N;②
    program-statements;
ENDSUB;
```

① The SUBROUTINE statement defines both the incoming and outgoing arguments.

② When needed the OUTARGS statement is used to let the function know which values are to be returned by the subroutine.

SAS GLOBAL OPTION CMPLIB：

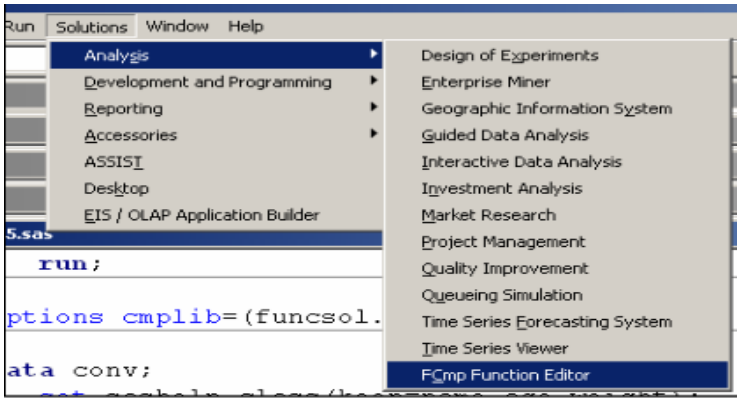
```
options cmplib=libname.dataset;
```

The SAS Global option CMPLIB specifies where to look for previously compiled functions and subroutines. All procedures (including FCMP) that supports the use of FCMP functions and subroutines utilize this global option.

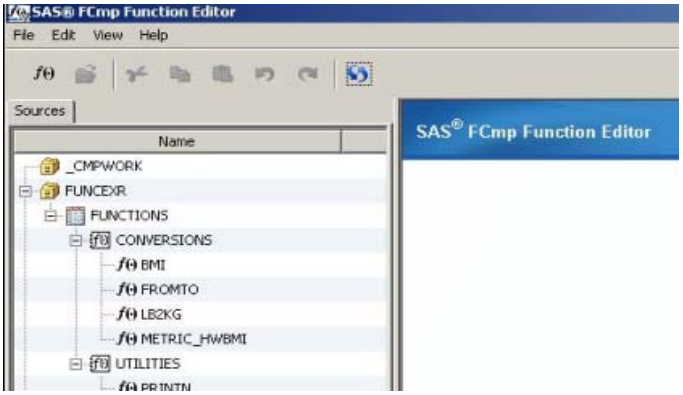
USING THE FCMP FUNCTION EDITOR

The FCMP Function Editor is an interactive tool that is available to you through the display manager.

①Using the pull down menus as shown in Example: Solutions→Analysis→FCMP Function Editor.

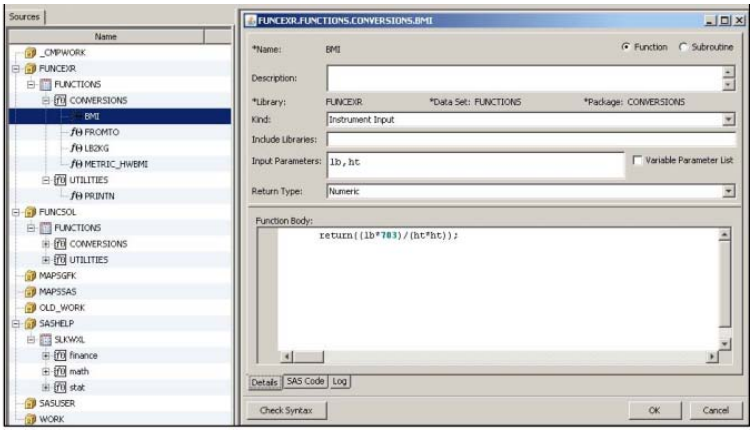


②Once the function editor has Started you will discover that the left panel shows each of the function libraries and the functions that they contain.



③By double clicking on a function in the right panel, you can bring up a dialogue box that contains the function definition.

This definition includes the arguments (parameters) and coding. Tabs at the bottom of this dialogue box allow you to see the full PROC FCMP step that created this function or subroutine.



DELETING FUNCTION DEFINAYIONS

```
proc fcmp outlib=libname.dataset.package;
      deletefunc functionname1;
      deletefunc functionname2;
run;
```

Here the DELETEFUNC statement is used to delete functions from the conversions package which is in the FUNCTIONS data set.

2.EXAMPLES OF CREATING A FCMP FUNCTION

Example 1. Calculation of Study Day

```
/*custom fuctions */
proc fcmp outlib=work.funcs.trial;
      function study_day(start, end);
      day= end - start;
      if day >= 0 then return(day+ 1);
      else return(day);
endsub;
run;
/*call function*/
options cmplib=work.funcs;
data _null_;
      start = '15Feb2006'd;
      today = '27Mar2006'd;
      Day=.;
      ady = study_day(start, today);
      ady2= study_day(start, day);
      put ady=;
      Put ady2=;
run;
```

This example create a function named as study_day which can be used to calculation intervals between two days. The results are shown as follow:

```
268  data _null_;
269      start = '15Feb2006'd;
270      today = '27Mar2006'd;
271      Day=.;
272
273      ady = study_day(start, today);
274      ady2= study_day(start, day);
275      put ady=;
276      Put ady2=;
277  run;

ady=41
ady2=.
NOTE: “DATA 语句” 所用时间（总处理时间）:
      实际时间          0.03 秒
      CPU 时间          0.03 秒
```

From the above results we can see that the log is clean although the variable day is missing. Using function like this we can exclude unexpected warning.

Example 2. Calculation of BMI

```
/*custom fuctions */
proc fcmp outlib= work.functions.conversions;
      Subroutine BMI(height,weight,bmi);
      outargs bmi;
      bmi= weight /(height*height);
      endsub;
run;
/*call function*/
options cmplib=( work.functions);
data test
            input height weight;
            datalines;
            160 42
            180 70
            150 .
            ;
run;
data BMI1;
      set bmi;
      bmi=.;
      call bmi(height/100, weight, BMI);
run;
```

	height	weight	bmi
1	160	42	16.40625
2	180	70	21.604938272
3	150	.	.

```
49  data BMI;
50  set test;
51  bmi=.;
52  call bmi(height/100, weight, BMI);
53  run;

NOTE: There were 3 observations read from the data set WORK.TEST
NOTE: The data set WORK.BMI has 3 observations and 3 variables.
NOTE: DATA statement used (Total process time):
      real time           0.09 seconds
      cpu time            0.17 seconds
```

We can see that above results are correct and the log is clean, we can't see t note like "Missing values were generated" any more. .

Example 3. Check Valid Numeric Function

```
/*custom fuctions */
proc fcmp outlib=work.funcs.myfuncs;
      function numval(var $);
      return (not (notdigit (strip (translate (var,'000','.+-')))
      or count (var,'.')>1 or countc (var,'+-')>1
      or indexc (left (var), '+-')>1 or left (var) in ('+','+.','-','-.')));
      endsub;
run;

/*call function*/
options cmplib=work.funcs;

data lb;
      input lbscresc $;
      if numval(lbscresc) then lbscresn = input (lbscresc, best.);
      else put "USER WAR" "NING: invalid value " lbscresc=;
      datalines;
      1.1
      <5
      -3
      ;
run;
```

	lbscresc	lbscresn
1	1.1	1.1
2	<5	.
3	-3	-3

This simple function can check if a character variable holds a valid numeric value, the results are above. This function will return a Boolean 0 (value is not numeric) or 1 (value is numeric) value, depending on the input data.

Example 4. Get Character Date Function

In this example, the DATE function Return the date in iso8601 format;

```
proc fcmp outlib=work.funcs.myfuncs;
function datevalc(dvar $, dfmt $) $;
      fmtc=compress(dfmt,'0123456789.');
      dvar_=compress (dvar,'ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijkl
      mnopqrstuvwxyz#!$%&()*+-./:;<=>?[\]^_{|}~)');
      yy=strip(put(substr(dvar_,1,4),$4.));
      mm=strip(put(substr(dvar_,5,2),$2.));
      dd=strip(put(substr(dvar_,7,2),$2.));
      length cc$10;
      cc=catx('-',yy,mm,dd);
      return(cc);
   endsub;
run;

options cmplib=work.funcs;
data test;
      LENGTH VISITDTC $20;
      visitdtc='NK';output;
      visitdtc='2016-05-UK';output;
      visitdtc='2013-UNK-UNK';output;
      visitdtc='2014-03-08';output;
      visitdtc='20140523';output;
run;

data date;
      set test;
      LENGTH DTC $20;
      dtc=datevalc(visitdtc,'YYMMDD');
run:
```

The results:

	VISITDTC	DTC
1	NK	
2	2016-05-UK	2016-05
3	2013-UNK-UNK	2013
4	2014-03-08	2014-03-08
5	20140523	2014-05-23

3.THE RUN_MACRO FUNCTION

RUN_MACRO routines give us a way to call the macro inside an FCMP function. The function executes the macro, and returns a single value to the DATA step.

Example 5. Calculation basic statistic.

```
%macro stat;
      %let input_table = %sysfunc(dequote(&in));
      %let var = %sysfunc(dequote(&val));
       %let class = %sysfunc(dequote(&class));
       %let output_table = %sysfunc(dequote(&out));

      proc means data= &input_table nway noprint;
            class &class;
            var &var;
            output out=&out n=n min=min median=median std=std min=min
      max=max;
      run;

%mend;
proc fcmp outlib=sasuser.funcs.sql;
       function get_distinct_values(input_table $, class $,column $, output_table $);
             rc = run_macro('stat',in,class,var,out);
             return (rc);
       endsub;
run;

options cmplib = sasuser.funcs;
data _null_;
       rc = get_distinct_values('sashelp.class',’sex’,'age', 'work.test1');
run;
```

During execution, when the command is reached, the function takes each variable on the RUN_MACRO parameter list and creates a local macro variable with the same name and value. Then the macro is executed, and the values of each macro variable at the end of the macro execution are written back to the corresponding variables in the function. Parameters on the RUN_MACRO command represent two-way communication between the function and the macro. The rc variable returns 0 if the macro was submitted successfully, otherwise it returns an error code.

Example 6. Calculation maximum decimal place.

```
%macro get_decs();
%let dsn = %sysfunc(dequote(&dsn.));
%let var = %sysfunc(dequote(&var.));

data dec;
      set &dsn;
      if find(strip(put(&var,best.)),'.') then
dec=length(scan(strip(put(&var,best.)),2,'.'));
      else dec=0;
run;

proc sql noprint;
      select max(dec) into : dec from dec;
quit;
%let dec=&dec;

proc datasets library = work nolist;
delete dec; quit;
%mend get_decs;

PROC FCMP outlib=work.functions.own;
function decs(dsn $, var $);
rc = run_macro('get_decs', dsn, var, dec);
return(dec);
endsub;
RUN;
options cmplib=work.functions;

data max_dec;
      set sashelp.class;
      decs=decs('sashelp.class','height');
run;
```

REFERENCES:

Dylan Ellis, Mathematica Policy Research, Washington, DC,2013,RUN_MACRO Run! With PROC FCMP and the RUN_MACRO Function from SAS® 9.2, Your SAS® Programs Are All Grown Up

Jason Secosky, SAS Institute Inc., Cary, NC,2007 User-Written DATA Step Functions

Jason A Smith, Argo Analytics Ltd, Cambridge, UK,2014 Define Your Own SAS Functions for Date and Numeric Validation Using PROC FCMP

http://support.sas.com/documentation/cdl/en/proc/61895/HTML/default/viewer.htm#a003181727.htm

http://bbs.pinggu.org/thread-3102239-1-1.html

ACKNOWLEDGEMENTS

I would like to thank Hejing Hong for his timely efforts in reviewing and providing feedback on this paper.

CONTACT INFORMATION:

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

Name: Qixia Shi
Enterprise: Fountain Medical Technology Co., ltd
Address: Room 403, Building 43, No.70, Headquarter Base, Phoenix Road, Jiangning District
City, State ZIP: Nanjing, China 210000
E-mail: qixia.shi@fountain-med.com