PhUSE EU CONNECT 2020

Paper SM08

PROC FCMP or Function you Create, Master and Proceed

Iryna Boiko, Covance, Kyiv, Ukraine

ABSTRACT

SAS offers the wide variety of inbuilt functions for users, but is it a reason to restrict a fantasy of programmers? This paper will prove that the correct answer is 'No' exploring the possibilities of PROC FCMP. It enables you to save the valuable pieces of code and call them further in form of SAS functions. The paper will demonstrate the examples from clinical practice where the user defined functions are reasonable to apply: starting from manipulating with dates to the more complex functions that deals with text transformation. The most part of presented functions could be easily modified depends on project needs. It is a good idea to create so called 'functions library' before project starts to harmonize routine transformations and make results consistent from the beginning.

INTRODUCTION

The SAS Function Compiler (FCMP) procedure enables you to create your own function using DATA step syntax. It can be useful when you have repetitive routines or to program complex code more easily read. In order for SAS to find the custom function, the CMPLIB=option must be specified in the OPTIONS statement. This option names the table where the function is stored within a package. The CMPLIB option is a global option. Once you set this option it remains in effect until you cancel, change, or exit your SAS session.

DATE CONVERSION

To program a custom function then store it and use it as repeatedly as needed we need to specify the table and package that stores the compiled function, which is a required argument in the PROC FCMP statement we specify the function name and the function arguments and the returning value type. In case of nothing mentioned there will be numeric value as a return.

Body of the function contains the main logic of the function that is programmed with statements that SAS language compiler can execute. We are returning the final value which function has produced and ending the PROC FCMP block with ENDSUB. As a result of the execution a dataset is being created and stored in the library.

DATE_TO_NUM FUNCTION

In the first example we will see how with PROC FCMP function character date converts into numeric date using regular expression in the logic. First, we are declaring the library and the table names to store our function with name date_to_num. As an input parameter we are having character type with $ sign and the returning type is numeric so no dollar sign is specified. To use date_to_num function we are declaring cmplib option with required table. And to see function in use we can create a dt_diff variable that contains a quantity of days from dt_start to dt_end.

Also we can create separate function for each date format. This example converts Character date format to Numeric using yymmdd10.

```
proc fcmp outlib= work.funcs.myfuncs;
function date_to_num(dtc $);
       if prxmatch("/\d{4}-\d{2}-\d{2}/o",dtc) = 1 then
       dtn = input(substr(dtc,1,10),yymmdd10.);
       else call missing(dtn);
return(dtn);
endsub;
run;

options cmplib = work.funcs;

data ic;
       set edc.ic_1;
       length dat_ic 8.;
       dat_ic = date_to_num(DSSTDT_RAW);
run;
```

After submitting this in SAS, the log confirms that the function has been saved:

```
NOTE: Function date_to_num saved to work.funcs.myfuncs.
NOTE: PROCEDURE FCMP used (Total process time):
      real time           0.04 seconds
      cpu time            0.02 seconds
```

The output dataset shows that the character values have been converted to numeric as expected:

	Subject	DSSTDT_C	dat_ic
1	1000110006	2019-01-12	12JAN2019
2	1000110011	2019-02-16	16FEB2019
3	1009210012	2019-03-17	17MAR2019
4	1000810013	2019-04-18	18APR2019
5	1007210015	2019-05-19	19MAY2019
6	1000110018	2019-06-22	22JUN2019
7	1001810007	2019-07-15	15JUL2019
8	1001810014	2019-07-18	18JUL2019
9	1000110016	2019-07-19	19JUL2019
10	1000110019	2019-07-22	22JUL2019

DATE_TO_ISO FUNCTION

Bellow example demonstrates similar way to convert date into ISO format. But in this case this is a raw date so value could be with unknown date or month. The returning value for this function is a character type so in the declaration the dollar sign and number of characters are present.

Function date_to_iso(date) can convert Raw Date into ISO format even with missing date digits or values.

```
proc fcmp outlib= work.funcs.myfuncs;

function date_to_iso(dtc $) $20;

       if length(dtc) = 11 and substr(dtc,1,6) = "UN UNK" then
         dtc_f = substr(dtc,8,4)||"-UN-UN";

       else if length(dtc) = 11 and substr(dtc,1,2) = "UN" then
         dtc_f = substr(put(input("01 "||substr(dtc,4),date11.),e8601da.),1,8)||"UN";

       else if length(dtc) >= 10 then dtc_f = strip(put(input(dtc,date11.),e8601da.));

       else if length(dtc) = 6 and substr(dtc,1,2) ne "  " then call missing(dtc_f);

       else if length(dtc) = 6 and substr(dtc,1,2) eq "  " then
         dtc_f = substr(dtc,3,4) || "-UN-UN";

       else if length(dtc) = 8 then dtc_f = substr(dtc,5,4) || "-UN-UN";

       else if length(dtc) = 9 then
         dtc_f = substr((put(input("01"||strip(dtc),date11.),e8601da.)),1,8)||"UN";

       if not missing(dtc) and missing(dtc_f) and ^(length(dtc) = 6
         and substr(dtc,1,2) ne "  ") then
         put "WARNING: Other formats of dtc variable";

return(dtc_f);
endsub;
run;

options cmplib = work.funcs;
```

```
data vendor_func;
       set edc.ic_1(keep=Subject DSSTDT_RAW);
       length ic_dt $20;

       if not missing(DSSTDT_RAW);
       ic_dt = date_to_iso(DSSTDT_RAW);
run;
```

After submitting this in SAS, the log confirms that the function has been saved:

```
NOTE: Function date_to_iso saved to work.funcs.myfuncs.
NOTE: PROCEDURE FCMP used (Total process time):
      real time           0.01 seconds
      cpu time            0.01 seconds
```

The output dataset shows that the character values have been converted to character ISO format as expected:

	Subject	DSSTDT_R..	ic_dt
1	3600111447	UN/MAR/2020	2020-03-UN
2	1002810560	UN UNK 2019	2019-UN-UN
3	1006610313	01/OCT/2019	2019-10-01
4	1000510045	02 AUG 2019	2019-08-02
5	1001210765	02-DEC-2019	2019-12-02
6	1001710919	02/JAN/2020	2020-01-02
7	3600211449	02/MAR/2020	2020-03-02

TEXT MANIPULATION

The third example shows us that function can be used as a mapping tool. When we are having data from different sources, so values could be similar in meaning but different in spelling, stored function can return correct value.

GENDER_FORMAT FUNCTION

Function gender_format(gen_in) can change specified Gender format to requested one.

```
proc fcmp outlib= work.funcs.myfuncs;
function gender_format(gen_in $) $ 10;

       if length(gen_in) = 1 and upcase(gen_in) = 'F' then gen_out = 'Female';
       else if length(gen_in) = 1 and upcase(gen_in) = 'M' then gen_out = 'Male';
       else if length(gen_in) > 1 and upcase(gen_in) = 'FEMALE' then gen_out = 'F';
       else if length(gen_in) > 1 and upcase(gen_in) = 'MALE' then gen_out = 'M';
       else call missing(gen_out);

return(gen_out);
endsub;
run;

options cmplib = work.funcs;

data vendor_func;
       set safload_1(keep=SUBJNO SITEID SEX);
       length Gender $10;
       Gender = gender_format(SEX);
run;

data crf_func;
       set edc.dm_2(keep=Subject SEX);
       length Gender $10;
       Gender = gender_format(SEX);
run;
```

After submitting this in SAS, the log confirms that the function has been saved:

```
NOTE: Function gender_format saved to work.funcs.myfuncs.
NOTE: PROCEDURE FCMP used (Total process time):
      real time           0.00 seconds
      cpu time            0.01 seconds
```

The output dataset shows that the gender values have been mapped into longer or shorter format as expected:

	subject	sex	gender
1	3400111164	M	Male
2	3600110964	F	Female
3	3600211112	U	
4	3600211114	F	Female

PROC FCMP AND MACRO COMPARISON

To see if we would have a major difference in using PROC FCMP with Macro Processing we have changed the number of observations and check whether performance of the same programs logic differ using Proc FCMP and Macros approaches. It was shown that there are no major differences in terms of performance between SAS macro functions and PROC FCMP functions. Basically, presented functions are quiet flexible to be easily modified depends on project needs. We can look at a new angle for choosing a tool that can let you write your code more flexible, readable and useful.

Functions have not been designed to replace macros. All of the things you can do with functions can of course be performed with macros. The aim of using validated functions within clinical programs is to make code easier to read and understand. Code becomes difficult to interpret when datastep syntax and macro syntax are mixed together. This often slows down the error detection and validation process whilst coding. There are performance benefits to functions too, standard Do loops should be used instead of %DO loops for example.

CONCLUSION

At first glance, using PROC FCMP can be replaced by using macro code. But creating a new function with this approach can broaden the understanding of how functions work in SAS. Often happens that we need to deal with a problem which solution can be more universal, convenient or even technically more correct using PROC FCMP. Using validated functions in clinical programming will save time. Repetitive coding can be minimized with the use of functions. The size of programs can be vastly reduced with the use of functions and therefore the validation process can also be reduced. When there is less code to examine in order to locate errors, there will be less rework that needs to happen. All these things added together mean less programming time is required for each study that comes along.

RECOMMENDED READING

1. SAS® Certified Professional Prep Guide: Advanced Programming Using SAS® 9.4
2. Rhoads, Mike. 2012. “Use the Full Power of SAS® in Your Function-Style Macros.” *Proceedings of the SAS Global Forum 2012 Conference.* Cary, NC: SAS Institute Inc.
   Available at http://support.sas.com/resources/papers/proceedings12/004-2012.pdf.
3. Secosky, Jason. 2007. “User-Written DATA Step Functions.” *Proceedings of the SAS Global Forum 2007 Conference.* Cary, NC: SAS Institute Inc.
   Available at http://www2.sas.com/proceedings/forum2007/008-2007.pdf.

CONTACT INFORMATION

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

Iryna Boiko
Covance, Inc.
6 Oleny Telihy str., Building 6
Kyiv, 04112, Ukraine
Work Phone: +38 044 251 3849
Email: iryna.boiko@covance.com
Web: https://www.covance.com