Paper RF13-2014

The Use of PROC FCMP to Manage Your Functions

Jennifer Tsai, MPH, University of Southern California, Los Angeles, CA

ABSTRACT

SAS programming includes many built-in functions that can be programmed within the DATA Step to execute specific tasks. However, to run certain calculations and tasks that are not included in the SAS ® programming procedures, SAS users may choose to write their own user defined functions. PROC FCMP allows programmers to store their written functions in a work library that can then be used in subsequent code similar to how the built-in functions in SAS are used. This procedure provides a streamlined and efficient way of managing complex and tedious codes. The purpose of this paper is to demonstrate how we can utilize Proc FCMP to store and run a user defined function in the DATA Step.

INTRODUCTION

Proc FCMP stores user-defined functions, which can be executed in subsequent DATA Steps. This is similar to how we call up functions that are defined in the SAS user system.

Basic FCMP syntax looks like the following:

```
PROC FCMP OUTLIB = LIBNAME.DATASET.PACKAGE;

FUNCTION FUNCTION-NAME (ARGUMENT... ARGUMENT);
... Program Statements...

RETURN (EXPRESSION);
ENDSUB;
```

APPLICATION OF THE PROC FCMP PROCEDURE

In this example, we look at how Proc FCMP can be used to create a new date variable called “complain_date” to output formatted complain dates a graduate advisor received from her graduate students about their classes in the past year given only the month and day the complaint was received (Table 1).

Table 1. Complain Data

Student Name	Complain_Month	Complain_Day
John	06	27
Mary	07	19
Daniel	07	26
Molly	05	17
Daniel	01	20
Laura	04	11
Joy	08	02
Mark	02	16
Daniel	03	14
Emily	11	24
Owen	09	04
Joy	10	31
Heather	12	10

CREATING THE FUNCTION THROUGH PROC FCMP

Before we can utilize the function, we must first create the specific task we want our function to perform using Proc FCMP. In this example, we want to return a complete complain date with the month, day, and *year* the complain was received given the current date (Program 1).

Program 1. Creating the Create_Date Function Using Proc FCMP

```
LIBNAME mwsug 'T:\MWSUG';

PROC FCMP OUTLIB=mwsug.complain.myfunc;
       FUNCTION create_date (m, d, current_month, current_day);
              if m > current_month then new_date = mdy(m, d, 2013);
              else if m < current_month then new_date = mdy(m, d, 2014);
              else do;
              if d > current_day then new_date = mdy(m, d, 2013);
              else new_date = mdy(M, d, 2014);
       END;
              RETURN(new_date);
       ENDSUB;
run;
```

OUTLIB

The Outlib statement tells SAS where the function is written once the Proc FCMP step ends utilizing a three-level name. In this example, we stored the function in a package called "myfunc". The package is then stored in the data set "mwsug.complain".

PROC FCMP

The Proc FCMP statement informs SAS that we are about to create a user-defined function.

FUNCTION

The Function statement informs SAS to return a value once the data program is run.

FUNCTION NAME & Argument(s)

The Function Name statement specifies the name we created for our task-specific function. Here, we called our function "create_date". Argument(s) included in the program tells SAS the specific argument (i.e., numeric/character constant, variable, or expression) we want. Here, we have four numeric arguments composed of the given complain month and day in the date set, and the current month and current day which will be specified once we run our function. The additional program statements are similar to a regular DATA Step statement that performs the specific tasks we have written. In this case, we have included an "If/Else if" statement that will parse out the data based on the given dates.

RETURN (EXPRESSION)

The Return statement specifies that a value needs to be returned at the end of our function, and the Expression statement is the name we call the value we want returned. Here, we call our return expression "new_date".

ENDSUB

The Endsub statement informs SAS to end the Function.

USING THE PROC FCMP FUNCTION WITH OUR DATASET

Now that our function is written and stored using PROC FCMP, we can now recall the function to be utilized in formatting our data (Program 2).

Program 2. Using Our Created Create_date Function

```
OPTIONS CMPLIB=mwsug.complain;
      Data complaindays2;
            set mwsug.complain;
            complain_date = create_date(cp_month, cp_day, 6, 24);
            FORMAT complain_date mmddyy8.;
      run;
PROC SORT DATA=complaindays2;
            by complain_date;
      run;
      TITLE 'Creating complain_date via user-defined function';
            proc print data=complaindays2;
      run;
```

OPTIONS CMPLIB

The CMPLIB statement informs SAS where to locate the function. Here, our "Create_date" function has been stored in mwsug.complain. If more than one function has been created and stored in different libraries or data sets, multiple libraries or data sets can be specified.

CREATE_DATE

The "Create_date" statement is the name of the function we created and is used like any other previously defined SAS function/statement.

THE RESULTING OUTPUT

Creating complain_date via user-defined function

Obs	Name	cp_month	cp_day	complain_date
1	John	6	27	06/27/13
2	Mary	7	19	07/19/13
3	Daniel	7	26	07/26/13
4	Joy	8	2	08/02/13
5	Owen	9	4	09/04/13
6	Joy	10	31	10/31/13
7	Emily	11	24	11/24/13
8	Heather	12	10	12/10/13
9	Daniel	1	20	01/20/14
10	Mark	2	16	02/16/14
11	Daniel	3	14	03/14/14
12	Laura	4	11	04/11/14
13	Molly	5	17	05/17/14

CONCLUSION

Using Proc FCMP to create a complete date variable is all but one of the many functions a SAS programmer can create. Other common functions using the PROC FCMP procedure include other formatting functions, weight conversions, calculations, macros, etc. The functions can be as simple and as complex as the SAS user desires. Regardless of what the function is utilized for, if the function is not pre-defined in SAS, Proc FCMP allows users to create and manage their own user-defined function(s) that can be used for multiple data sets.

ACKNOLWEDGEMENTS

The author would like to acknowledge the guidance and assistance of Arthur Li.

CONTACT INFORMATION

For any additional comments or questions, please contact the author at:

Jennifer Tsai
University of Southern California
2001 N. Soto Street, 3rd Floor
Los Angeles, CA 90089
tsaijy@usc.edu