PharmaSUG 2024 - Paper PO- 231

BEST FUNCTION EVER: PROC FCMP

Michael Stout, DePuySynthes

ABSTRACT

Base SAS® provides a wealth of character and numeric functions. User defined functions can be created with PROC FCMP (SAS Function Compiler). PROC FCMP is a powerful tool that can invoke SAS procedures and the output delivery system and then return a single value to the calling program. This paper will show how to write and use user defined functions created using the special SAS function RUN_MACRO. Now you can create the best functions ever and deploy them in your SAS environment.

INTRODUCTION

This paper focuses on creating user defined SAS functions. Concepts learned in this paper will give programmers additional tools to simplify complex programming. This paper is for programmers of any skill level that desire to learn more about the power of user defined functions.

Although SAS provides a large library of excellent functions, there have been times when a function I needed did not exist, and I had to write my own code. PROC FCMP is the tool that enables programmers to develop simple and complex custom functions that the programmer can save and share across an organization, thus expanding the number of functions available to the programming community.

The easiest way to understand functions is to think of them as stand-alone programs that run when invoked by the calling program. When a function is called, parameters are passed to the function and program execution is temporarily halted, the function reads the values passed to it, performs some action, returns a single value and control back to the calling program. The calling program then resumes processing.

Functions can do a variety of tasks such as:

- Assign statistics for categorical and continuous variables such as N, Mean, Max, and P-values
- Assign upper and lower confidence levels
- Capture values from ODS output
- Get a list of unique data values
- Score patient outcomes
- Score QOL assessments
- CDISC - remove special characters from data fields
- CDISC - create ISO 8601 formatted dates

Functions are independent stand-alone code. By encapsulating the code used to create the function, programmers do not need to know or understand the underlying code. Functions are easy to use and reduce validation activities.

Creating custom functions is straight forward. This paper will provide syntax for PROC FCMP and two examples. This first example is simple function. The second example is more complex and utilizes the special SAS function named RUN_MACRO.

The programmer can use the same approach to create user defined SAS PROCEDURES. Procedures permit the return of multiple parameters to the calling program. To learn more on user defined procedures, review SAS papers on the subject at www.lexjansen.com.

SYNTAX

The syntax for PROC FCMP is described below. This paper only covers the FCMP statements and parameters discussed in this paper. Read SAS documentation for a complete list of statements and parameters for PROC FCMP.

Following is basic syntax for PROC FCMP:

```
PROC FCMP OUTLIB=;
FUNCTION <name of function>(param_1 <$>, param_2 <$>, param_n <$>) <$> ;
  <SAS Code>
  RETURN(expression);
ENDSUB;
```

The PROC FCMP procedure creates user defined functions. This paper discusses one parameter OUTLIB=. The OUTLIB= parameter points to the library where the compiled function is saved. The FUNCTION statement defines the function name and parameters passed to the function and type of value returned to the calling program. Parameters can be either number or character. By default, parameters are numeric. A dollar sign ($) denotes that the parameter is character. The RETURN statement returns a character or numeric expression to the calling program. Finally, the ENDSUB statement marks the end of the function.

PROC FCMP

PROC FCMP is straight forward and easy to use but underused by the SAS community. Creating user defined functions require minimal coding. The first section of code uses PROC FCMP to define a function to sum two numbers. The second part calls the user defined function from a data step to add two numbers (4 and 6).

The following example sums two numbers and assigns the value to field RESULT:

```
/************************************************************
PART 1 - Define user define function to sum two numbers.
************************************************************/
PROC FCMP OUTLIB=work.funcs.temp;
FUNCTION UDF_Add_Numbers(num1 , num2);
  RETURN(num1 + num2);
ENDSUB;

/************************************************************
PART 2 - Call user defined function UDF_Add_Numbers and assign to Result
************************************************************/
OPTIONS cmplib = work.funcs;   /* location of compiled function */

DATA _NULL_;
  Result   = UDF_Add_Numbers(4,6);
  Put Result=;
RUN;
```

Output 1 shows the output generated by function UDF_Add_Numbers.

```
Result = 10
```

Output 1. Output from PROC FCMP

PROC FCMP USING RUN_MACRO

Calling the special function RUN_MACRO exponentially increases the usefulness of PROC FCMP. RUN_MACRO allows a user to run SAS code such as data step, SAS procedures and Output Delivery System (ODS) and return a value to the controlling program. This affords new programming possibilities.

This example is like the previous one but uses a macro to run SAS code. This program has three parts. The first part defines a macro, the second part defines the FCMP function, and the last part calls the function from a data step.

The following example gets a list of unique values for a given data field and returns the value to calling program:

```
/*************************************************************************
 PART 1 - Define macro called by user defined function
 1. Remove quotes from parameters passed into function
 2. Run PROC SQL to get list of unique values
 3. Check if procedure ran successfully
*************************************************************************/
%MACRO UDF_Get_My_List;
%let dsn = %sysfunc(dequote(&dsn));
%let var = %sysfunc(dequote(&var));
%let dlm = %sysfunc(dequote(&dlm));

%LET List_out  = ;

PROC SQL noprint;
SELECT distinct &var into: List_out separated by "&dlm" from &dsn;
quit;

%LET List_out  = %TRIM(&List_out);

%IF &syserr > 0 %THEN %DO;
%PUT WARNING: Empty list returned for variable &dsn..&var;
%END;
%MEND;


/*************************************************************************
 PART 2 - Run SAS macro UDF_Get_List and return list of values
 1. input character parameters dsn, var and dlm
 2. define character parameter List_out. Set length large enough to
 accommodate list of values
 3. Return List_out to calling program
*************************************************************************/
PROC FCMP OUTLIB=work.funcs.sql;
FUNCTION UDF_Get_List(dsn $, var $, dlm $) $;

/* Define additional parameters used by UF_Get_List_macro */
LENGTH List_out   $ 500;

rc = RUN_MACRO('UDF_Get_My_List', dsn, var, dlm, List_out);
RETURN(List_out);
ENDSUB;
```

```
/*****************************************************************
 PART 3 - Get list of Makes, Types, and Origin of vehicles from
 SASHELP.CARS
*****************************************************************/
OPTIONS cmplib = work.funcs;   /* location of compiled function */

DATA out_file;
  List_Make   = UDF_Get_List('SASHELP.CARS','Make'  ,'|');
  List_Type   = UDF_Get_List('SASHELP.CARS','Type'  ,'+');
  List_Origin = UDF_Get_List('SASHELP.CARS','Origin',':');

  PUT List_Make=;
  PUT List_Type=;
  PUT List_Origin=;
RUN;
```

Output 1 shows the output generated by function UDF_Get_List.

```
List_Make:   Acura|Audi|BMW|Buick|Cadillac|Chevrolet|Chrysler|Dodge|Ford|…

List_Type:   Hybrid+SUV+Sedan+Sports+Truck+Wagon

List_Origin: Asia:Europe:USA
```

Output 2. Output from PROC FCMP Using RUN_MACRO

CONCLUSION

It is easy to create custom functions using PROC FCMP and call them in a DATA step. You have learned how to create user defined functions, call the special RUN_MACRO routine, save functions, and use them in the data step. You now have the skills needed to create your best function ever and share it with the programming community. Get creative and develop functions to simplify code and expand your programming skills.

RECOMMENDED READING

- *SAS® 9.4 and SAS® Viya® 3.5 Programming Documentation*

CONTACT INFORMATION

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

Michael Stout
DePuySynthes
Mstout2@its.jnj.com