Paper CT05

CALL EXECUTE will figure it out: Data-driven Figure Programming

Mitchikou Pearl Tseng, OCS Consulting BV, 's-Hertogenbosch, Netherlands

ABSTRACT

In studies with multiple parameters of interest taken at varying timepoints under different treatment groups, parts, or periods, BY statement and/or employing SAS MACRO in SAS GRAPH® are sometimes not convenient in the creation of repeated figure outputs. But, by incorporating CALL EXECUTE in our figure programming, we can successfully create repeating graphs for which input values are data driven and add specific conditional attributes that are flexible to the user. This paper will discuss a data-driven figure programming approach in the following steps: 1) Creation of a macro for your figure, 2) Addition of new variable(s) to your input dataset for possible specific tweaks to your figure and 3) Making use of DATA STEP + CALL EXECUTE to robustly call the figure macro and apply conditional tweaks to your output.

INTRODUCTION

Graphs are an essential part of clinical trial reporting as it provides better way of understanding the results. It is often that only selected type of graphs are requested for a submission (i.e mean plots, individual concentration plots, etc.) but multiple repeats of it are needed to display results of the different facets of the study. The BY statement of the SAS/GRAPH procedures can be used although the control on the presentation of attributes can be limited.

Creating a macro for the graph of interest is then a solution. However, it can be daunting to have the macro written and called multiple times in the program especially for studies with multiple timing and parameters of interest and it can result in more maintenance to the programmer to ensure that all subsets and tweaks are still correct as the study data changes over time.

CALL EXECUTE routine combined with the created figure macro becomes handy then in this situation as it will allow the macro to be executed and the input macro parameter values to be assigned and stored as new variables in a single DATA step. The general syntax for the routine is:

```
CALL EXECUTE (argument);
```

There are three argument string options and these three are used in this paper:

- a character string, enclosed in quotation marks:
  ```
  CALL EXECUTE ('%macroname;');
  ```
- the name of a DATA step character variable whose value is a text expression or a SAS statement to be generated (enclosure in quotation marks is not required):
  ```
  CALL EXECUTE (varname);
  ```
- a character expression that is resolved by the DATA step to a macro text expression or a SAS statement:
  ```
  CALL EXECUTE ('%macroname ('|| paramvalue ||');');
  ```

FIGURE MACRO CREATION

The first step is the creation of the figure macro. The main objective of the macro creation is to give the programmer freedom on assigning figure attributes such as axis labels, axis values and axis value formats and ease on sub setting the input dataset on varying facets of the study such as study populations, parts, periods, treatment groups or parameter of interest.

As an illustrative example, a macro called ***%plot_series*** which uses the PROC SGPLOT to generate simple line plots is created. The macro parameter descriptions are below while the full program code is in the Appendix I.

PARAMETER	DESCRIPTION
in_data	Name of input dataset
subset	Condition to be applied to the dataset using the WHERE statement
groupvar	Name of the variable to be used to group the data
groupfmt	Name of the format name to be applied to the groupvar variable
xvar	Variable name to be used for the x-axis
xlabel	Text to be used as label for the x-axis
fix_xvalues	List of values to be displayed as ticks on the x-axis
fmt_xvalues	Name of the format to be used for the display of values on the x-axis
yvar	Variable name to be used for the y-axis

ylabel	Text to be used as label for the y-axis
ytype	Type of y-axis to be used, DISCRETE | LINEAR | LOG | TIME
att_m	Name of the dataset containing the discrete attribute map
att_id	Name of the ID variable in the discrete attribute map dataset
legend_title	Text to be used as title for the legend

FIGURE ATTRIBUTES SETUP

Now that we have the graph macro ready and we have identified the desired macro parameters, let's create the driver dataset containing the new variables for which values are the input for the macro parameters.

A one DATA step approach is done in this paper where both the creation of the new variables and the CALL EXECUTE routines are processed. The input dataset for this step is the entire source dataset. It is important to sort the input dataset first and do the next steps only on the unique occurrence of the desired BY variables value. In this case, the last occurrence is selected by utilizing the `LAST.paramn` condition line. The sorting done in here also dictates the sequence of the figure macro calls.

An alternative way of this one DATA step approach is creating a meta dataset which contains the unique BY variables values and using this as the input dataset instead of the entire source dataset. The `LAST.paramn` condition line is not required in this approach.

The new variables that are created in this step are specific on the needs or the conditions that the programmer wants to apply. In the code below, the created `ylabel` variable contains the input values for the **ylabel** macro parameter and its corresponding derived value accounts for the varying parameter unit. The `xvalues` variable is also created to contain the input for the **fix_xvalues** macro parameter. In the example, `xvalues` value depends on the `part` variable value and macro variables xvalues1/xvalues2 are assigned.

```
PROC SORT DATA = in_data;
   BY partn paramn;
RUN;
DATA aa;
   SET in_data END=LAST;
   BY partn paramn;

   LENGTH cmd cmd1 cmd2 $ 2000 ylabel xvalues xformat subset $200;
   IF LAST.paramn THEN DO;

      * Variable(s) for varying y-axis label;
      YLABEL = 'Absolute Value (' || STRIP(paramu) || ')';

      * Variable for varying x-values / x-value format;
      IF partn = 1 THEN do;
            xvalues = "&xvalues1.";
            xformat = 'name_format1';
      END;
        ELSE IF partn = 2 THEN DO;
            xvalues = "&xvalues2.";
            xformat = 'name_format2';
       END;
...
```

CALL EXECUTE ROUTINES

After setting up the new variables that are needed for the macro parameters inputs, the CALL EXECUTE routine(s) can now be performed in the same DATA step. Take note that the processes under this section can be done in a single CALL EXECUTE routine but for the purpose of illustrating the different ways of specifying an argument string, multiple routines were done.

We first need to ensure that we have the correct subsets for each of our macro calls. In the program below, the DATA step process of sub setting the input dataset, `in_data`, based on the BY variables `param` and `partn` is stored in the variable `cmd`. In here, attaching the varying `param` and `partn` values in the variable `cmd` is done using the CATS function. The first CALL EXECUTE routine then uses the variable `cmd` as the argument.

```
* Subsetting the input data to get the unique BY variables occurrence to be used as where
conditions;
cmd = CATS('DATA tmp; SET in_data (WHERE = (paramn =', paramn,' AND partn =', partn, ')); RUN;');
CALL EXECUTE(cmd);
```

The next two CALL EXECUTE routines are two variations of invoking `TITLE` statements. The first one accounts for the `part` and `param` variable values in the title statement by using the concatenation operator "||" while the second one straightforwardly assigns a text for the second title. These two examples cover the first and third options of specifying the argument for the routine as noted in the introduction.

```
* * Title1 setup;
CALL EXECUTE('TITLE1 J=L H=9PT "Study Part: ' || strip(part) || '" J=R "Parameter: ' ||
strip(param) || '";') ;
* * Title2 setup;
CALL EXECUTE('TITLE2 J=R H=9PT "Linear Scale";');
```

The ***%plot_series*** macro call is then stored to `cmd1` variable which is used as the argument to the next CALL EXECUTE routine. In the process of storing the macro call to `cmd1` variable, the CATS SAS function is used and so the previously derived new variables, `xvalues` and `ylabel`, can be assigned as inputs for the macro parameters. If you are still new in using CALL EXECUTE, it is recommended to store the macro call in a variable so it is easy to debug if an issue in the macro is encountered.

It is also recommended to make the CALL EXECUTE 'macro blind' by using the %NRSTR MACRO function to avoid timing issue in resolving macro variables when your figure macro contains some non-macro language constructs, such as CALL SYMPUT or SYMPUTX statements (in a DATA step) or an INTO clause (in PROC SQL). For further explanation of this topic and for the difference of using single or double quotes in string arguments of a CALL EXECUTE routine, please refer to the blog post by Leonid Batkhan (link to the post is in the References).

```
      * * Figure macro call;
      cmd1 = CATS('%nrstr(%plot_series( in_data    = tmp
                          , groupvar    = trtan
                          , xvar        = atptn
                          , xlabel      = Timepoint
                          , yvar        = aval
                          , ytype       = linear
                          , legend_title= Treatment: '
                 , ', fix_xvalues = ', xvalues
                 , ', ylabel      = ', ylabel
                 , '));'
                 );
     CALL EXECUTE (cmd1);
```

If another variety of figure is required, such as semi logarithmic plot or a plot using change from baseline as y-axis variable, the programmer can then create new CALL EXECUTE routine(s) and create associated new variables as the input for the macro parameters within the same DATA step. An additional semi logarithmic plot macro call is demonstrated in the Appendix II.

CONCLUSION

CALL EXECUTE is a useful tool in developing dynamic data-driven graph outputs. As long as there is no change in the dataset structure of your input dataset, your figure program does not need to be modified further even if the data values are updated over time.

REFERENCES

CALL EXECUTE Routine, Retrieved from
https://documentation.sas.com/doc/en/pgmsascdc/9.4_3.5/mcrolref/n1q1527d51eivsn1ob5hnz0yd1hx.htm

Batkhan, L (2017). CALL EXECUTE made easy for SAS data-driven programming. Retrieved from
https://blogs.sas.com/content/sgf/2017/08/02/call-execute-for-sas-data-driven-programming/

Usov, A (2014), Call Execute: Let Your Program Run Your Macro. Retrieved from
https://www.lexjansen.com/phuse/2014/cc/CC06.pdf

Wang, H (2015), Creating Data-Driven SAS® Code with CALL EXECUTE. Retrieved from
https://www.pharmasug.org/proceedings/2015/BB/PharmaSUG-2015-BB15.pdf

ACKNOWLEDGMENTS

The author appreciates Mariska Burger, Jules van der Zalm, John van Bemmelen, Martin Doughty, Luis Soriano, Penelope Mendoza and Thea Valerio for reviewing the manuscript and providing valuable comments for the presentation.

CONTACT INFORMATION

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

Author Name: Mitchikou Pearl Tseng
Company: OCS Life Sciences
Address: Ruwekampweg 2G
City / Postcode: s-Hertogenbosch / 5222 AT
Work Phone: +31 (0)73 523 6000
Email: info.nl@ocs-consulting.com
Web www.ocs-lifesciences.com

APPENDIX I FIGURE MACRO

```
%MACRO plot_series
            ( in_data     =
            , subset      =
            , groupvar    =
            , groupfmt    =
            , xvar        =
            , xlabel      =
            , fix_xvalues =
            , fmt_xvalues =
            , yvar        =
            , ylabel      =
            , ytype       = linear
            , att_m       =
            , att_id      =
            , legend_title=
            );

   PROC SGPLOT DATA = &in_data.
               %IF %LENGTH(&att_m.) > 0 %THEN DATTRMAP = &att_m. ;
               NOAUTOLEGEND
               ;
      %IF %LENGTH(&subset.) > 0 %THEN %DO;
         %STR(WHERE &subset.);
      %END;

      %IF %LENGTH(&groupfmt.) > 0 %THEN  FORMAT &groupvar. &groupfmt;;
      SERIES X = &xvar Y = &yvar /
                                  %IF %LENGTH(&groupvar.) > 0 %THEN  GROUP     = &groupvar.;
                                  MARKERS
                                  %IF %LENGTH(&att_id.) > 0 %THEN  ATTRID    = &att_id.;
                                  LINEATTRS = (PATTERN = 1)
                                  NAME      = "series"
             ;

      XAXIS LABEL       = "&xlabel."
            LABELATTRS  = (COLOR = BLACK FAMILY = ARIAL SIZE = 9 WEIGHT = BOLD)
            VALUEATTRS  = (COLOR = BLACK FAMILY = ARIAL SIZE = 9)
            %IF %LENGTH(&fix_xvalues.) %THEN VALUES      = (&fix_xvalues.);
            %IF %LENGTH(&fmt_xvalues.) %THEN VALUESFORMAT= &fmt_xvalues.;
            FITPOLICY   = STAGGER
            ;

      YAXIS LABEL        = "&ylabel."
            LABELATTRS   = (COLOR = BLACK FAMILY = ARIAL SIZE = 9 WEIGHT = BOLD)
            VALUEATTRS   = (COLOR = BLACK FAMILY = ARIAL SIZE = 9)
            TYPE         = &ytype.
            %IF %upcase(&ytype.) = LOG %THEN LOGSTYLE = LOGEXPAND LOGBASE = 10;
            ;

      KEYLEGEND "series" / NOBORDER
                           LOCATION = INSIDE
                           POSITION = TOPRIGHT
                           TITLE    = "&legend_title."
                           TITLEATTRS = (FAMILY = ARIAL SIZE = 9)
                           VALUEATTRS = (FAMILY = ARIAL SIZE = 9)
            ;

   RUN;
%MEND plot_series;
```

APPENDIX II DATA STEP WITH CALL EXECUTE ROUTINES

```
PROC SORT DATA = in_data;
   BY partn paramn; * sorting dictates the sequence of macro calls;
RUN;
DATA aa;
   * source dataset here is the entire source data ;
   * this source dataset can be changed to a meta dataset which contains the unique desired BY variables;
   SET in_data END=LAST;
   BY partn paramn;

   LENGTH cmd cmd1 cmd2 $ 2000 ylabel xvalues xformat subset $200;

   * the if condition is needed since we used the entire dataset as source ;
   * this step is required so we ensure that we only produce macro calls per unique desired BY variables;
   IF LAST.paramn THEN DO;

      * the following derivation of new variables are specific on the needs or the conditions that the
        programmer wants to apply or control as inputs for the macro parameters;

      * Variable(s) for varying y-axis label;
      ylabel = 'Absolute Value (' || STRIP(paramu) || ')';

      * Variable for varying x-values / x-value format;
      IF partn = 1 THEN do;
            xvalues = "&xvalues1.";
            xformat = 'name_format1';
      END;
        ELSE IF partn = 2 THEN DO;
            xvalues = "&xvalues2.";
            xformat = 'name_format2';
        END;

      * subsetting the input data to get the unique values for the where conditions;
      cmd = CATS('DATA tmp; SET in_data (WHERE = (paramn =', paramn,' AND partn =', partn, ')); RUN;');
      CALL EXECUTE (cmd) ;


      * First sample call - Linear Plot;
      * * Title1 setup - PART and PARAM value are accounted in the title;
      CALL EXECUTE ('TITLE1 J=L H=9PT "Study Part: ' || strip(part) || '" J=R "Parameter: ' || strip(param) ||
'";') ;
      * * Title2 setup - Straightforward assignment of text in the title;
      CALL EXECUTE ('TITLE2 J=R H=9PT "Linear Scale";');
      * * Figure macro call - storing the macro call in cmd1 for easy debugging;
      cmd1 = CATS('%nrstr(%plot_series( in_data    = tmp
                              , groupvar    = trtan
                              , xvar        = atptn
                              , xlabel      = Timepoint
                              , yvar        = aval
                              , ytype       = linear
                              , legend_title= Treatment: '
                  , ', fix_xvalues = ', xvalues
                  , ', ylabel      = ', ylabel
                  , '));'
                  );
      CALL EXECUTE (cmd1);


      * Second sample call - Semi Logarithmic Plot;
      * * Title1 setup - PART and PARAM value are accounted in the title;
      CALL EXECUTE ('TITLE1 J=L H=9PT "Study Part: ' || strip(part) || '" J=R "Parameter: ' || strip(param) ||
'";') ;
      * * Title2 setup - Straightforward assignment of text in the title;
      CALL EXECUTE ('TITLE2 J=R H=9PT "SemiLogarithmic Scale";');
      * * Figure macro call - storing the macro call in cmd2 for easy debugging;
      cmd2 = CATS('%nrstr(%plot_series( in_data    = tmp
                              , groupvar    = trtan
                              , xvar        = atptn
                              , xlabel      = Timepoint
                              , yvar        = aval
                              , ytype       = log
                              , legend_title= Treatment: '
                  , ', fix_xvalues = ', xvalues
                  , ', ylabel      = ', ylabel
                  , '));'
                  );
      CALL EXECUTE (cmd2);

   END;
RUN;
```