Paper 113-2011

%DO Loop – a Simple Dynamic Programming Technique

Yunchao (Susan) Tian, Social & Scientific Systems, Inc., Silver Spring, MD

ABSTRACT

Dynamic programming is an advanced macro topic. This paper presents a simple dynamic programming technique, the %DO loop. Included topics are: 1. Generating a list of variables, such as VAR1A – VAR50A, VAR1B – VAR50B. 2. Labeling a series of variables. 3. Running the same procedure against a series of data sets. 4. Using %IF-%THEN statement within %DO loop to provide logical branches. Each of the topics will be discussed along with examples. This paper is intended for SAS® users with basic knowledge of macro language.

INTRODUCTION

A dynamic program uses the data itself to determine the path and logic of execution and is data independent. Dynamic programming requires the use of certain macro statements, including the macro %DO loop which is used extensively. The iterative %DO is very similar to the DO statement used in the DATA step, but it is not confined to the DATA step. It can be used anywhere inside of a macro. The iterative %DO defines and increments a macro variable. This paper presents a number of examples to demonstrate how to take advantage of the %DO loop to build SAS statements dynamically.

EXAMPLE 1: GENERATING A LIST OF VARIABLES

In some of our projects, we often use SAS data sets with numbered series of variables, such as COL1_LINE1 – COL50_LINE1, COL1_LINE2 – COL50_LINE2, …, and COL1_LINE50 - COL50_LINE50. In a particular task, we may want to calculate the mean values only for the variables COL1_LINE1 – COL50_LINE1. Since these variables do not have a common prefix, we can not list them as COL1_LINE1 – COL50_LINE1 in the VAR statement.

We are facing two options: list these variables one by one in the VAR statement or write a few lines of code to dynamically generate the list of these variables. For a repetition of so many items, the following code is a better solution:

```
%MACRO DO_LIST;
%DO I = 1 %TO 50;
  COL&I._LINE1
%END;
%MEND DO_LIST;
```

Invoking macro DO_LIST in the VAR statement

```
VAR %DO_LIST;
```

produces the following complete VAR statement:

```
VAR COL1_LINE1 COL2_LINE1 ... COL50_LINE1;
```

The advantage of this technique is obvious. It avoids the time-consuming process of typing all these variables and it prevents potential typos. This example has a hard coded number of repetitions. In the next example, I will show you how to generalize this by supplying the number of repetitions as a parameter.

EXAMPLE 2: LABELING A SERIES OF VARIABLES

SAS programmers often need to create labels for a numbered series of variables with a common prefix. Array and DO loops can not be used here since the LABEL statement is a declarative statement. But you can use the %DO loop in a macro like the following to accomplish this:

```
%MACRO DO_LABEL(COUNT);
%DO I = 1 %TO &COUNT;
  PAY&I = "Payment for visit &I"
%END;
%MEND DO_LABEL;
```

Invoking macro DO_LABEL in the LABEL statement

```
LABEL %DO_LABEL(15);
```

produces the following complete LABEL statement:

```
LABEL PAY1 = "Payment for visit 1"
      PAY2 = "Payment for visit 2"
      ...
      PAY15 = "Payment for visit 15";
```

In this example, you supply the stopping value for the index I as the value of parameter COUNT which is the maximum number of doctor visits. This code is more dynamic than the previous example with the hard coded number of repetitions. In a more dynamic solution, the program should include code to determine the number of repetitions.

EXAMPLE 3: RUNNING THE SAME PROCEDURE AGAINST A SERIES OF DATA SETS

In previous examples, the %DO loop is used to generate single SAS statements. It can also be used to dynamically build a series of DATA steps or PROC steps. Let's say you have a series of SAS data sets CLASS1 – CLASS10 that contain students' math scores and you want to calculate the average scores for each of the ten classes. The following code will do the job:

```
%MACRO DO_MEAN;
%DO I = 1 %TO 10;
  PROC MEANS DATA=CLASS&I;
    VAR SCORE;
    TITLE "Average math score of Class &I";
  RUN;
%END;
%MEND DO_MEAN;
```

Invoking macro DO_MEAN generates this code:

```
PROC MEANS DATA=CLASS1;
  VAR SCORE;
  TITLE "Average math score of Class 1";
RUN;

PROC MEANS DATA=CLASS2;
  VAR SCORE;
  TITLE "Average math score of Class 2";
RUN;

...

PROC MEANS DATA=CLASS10;
  VAR SCORE;
  TITLE "Average math score of Class 10";
RUN;
```

The advantage of this technique is that you only need to call the macro once instead of ten times with ten different parameter values.

EXAMPLE 4: USING %IF-%THEN WITHIN %DO LOOP TO PROVIDE LOGICAL BRANCHES

In dynamic coding, you can provide logical branching based on the value of a macro variable. In the previous example, if you do not need the average scores for Class 3 and Class 7, you can use the %IF-%THEN statement with %DO block to cause a logic branch in the processing, as shown below:

```
%MACRO DO_BRANCH;
%DO I = 1 %TO 10;
  %IF &I NE 3 AND &I NE 7 %THEN %DO;

    PROC MEANS DATA=CLASS&I;
      VAR SCORE;
      TITLE "Average math score of Class &I";
    RUN;

  %END;
%END;
%MEND DO_BRANCH;
```

When the macro DO_BRANCH is called, the average scores of all classes except for Class 3 and Class 7 will be calculated.

CONCLUSION

The power of dynamic coding is that it is possible to write programs without hard-coding things like data set and variable names. By writing the programs to be as dynamic as possible, the programs become more reusable. Dynamic programming is an advanced macro topic. It is not always easy to create dynamic programs. While the examples in this paper are fairly straightforward, dynamic applications in the real world can be much more complicated. With practice, the use of these techniques will soon make your programming time much more effective and dynamic programming techniques will eventually become second nature for you.

REFERENCES

Tian, Yunchao. *The Power of CALL SYMPUT - DATA Step Interface by Example*
Proceedings of the 29th Annual SAS Users Group International Conference
Montreal, Canada, May 9-12, 2004

Tian, Yunchao. *A Few Macro Techniques to Shorten Your Programs*
Proceedings of the 16th Annual NorthEast SAS Users Group Conference
Washington, DC, September 7-10, 2003

CONTACT INFORMATION

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

Yunchao (Susan) Tian
Social & Scientific Systems, Inc.
8757 Georgia Avenue, 12th Floor
Silver Spring, MD 20910
Work Phone: (301) 628-3285
Fax: (301) 628-3201
Email: Stian@s-3.com