Paper CC14

%WINDOW: SAS' Diamond In the Rough

Alan Mann, Independent Consultant, Martinsburg, WV

ABSTRACT:
The purpose of this presentation is to show the effectiveness of %WINDOW as an interactive testing interface, and how it has similarities to static web pages with interactive links that are used to pass parameters from the page to the analytic software. The limitations and best applications of %WINDOW are discussed in brief, in order to stimulate discussion in the remainder of the session.

- Basics of %WINDOW in SAS Macro Language;
- Applications of the interface;
- Limitations, such as linkage with compiled macro code as the only way to use % WINDOW;
- Uses as a laboratory interface;
- Uses as a testing interface for future web-enabled applications;
- Comparison with a simple web page;

Getting Started: Basics of %WINDOW in SAS Macro Language

SAS Macros have a structured procedural model that requires any passed parameters to be locally or globally identified and dynamically introduced to the session.

What in the world does that sentence mean? Other than the author might have some professional IT experience, and is fond of 'techspeak'?

In plain English, a macro is a program, or procedural script that is part of a full program, that has a logical process that can be called as a stored routine anywhere in a SAS program or called series of programs. This script can have changing parameters passed to it that alter the output of the program, given the changes in the data steam. Hence, the macro has the ability to be applied as a general script that can be called numerous times to perform the same overall task to different inputs, be they numeric or alpha. Macro variables can also be introduced into memory as constants, able to be called throughout the process and declared but once.

Given: a SAS program to build a multiplication table for elementary school:

```
DATA NULL; file print;
    A    =  2;
    DO I = 1 TO 12;
      x = A * I;
      put A 'X ' I '= ' x;
    END;
RUN;
```

The result is:

```
2 X 1 = 2
2 X 2 = 4
2 X 3 = 6
2 X 4 = 8
2 X 5 = 10
2 X 6 = 12
2 X 7 = 14
```

```
2 X 8 = 16
2 X 9 = 18
2 X 10 = 20
2 X 11 = 22
2 X 12 = 24
```

Build it into a macro:

```
%macro multtbl(t1,rng);

DATA NULL; file print;
     A    = &t1;
     DO I = 1 TO &rng;
       x = A * I;
       put A 'X ' I '= ' x;
     END;
RUN;
%mend multtbl;
%multtbl(2,12);
```

Now, we've got potential, but how many 6 year-olds does it take to bring down a server? Answer: At least one. We need to put a face on this, and in a hurry!

JavaScript and XHTML can give an excellent tool for interactivity, but here, we're looking for economy, and it's ½ hour before class!

```
%window MultTable irow=5 rows=25 icolumn=15 columns=55
          #1 @20 "MULTIPLICATION MAGICIAN" attr = highlight
          #3 @20 "Your wish is my command!"
          #5 @25 "Select Your Times Table Number: " +2  t1 4 attr = underline
          #7 @25 "How Many Times Do You Want To Multiply By?" +2  rng          4 attr = underline
          #9 @30 "Press ENTER 3 Times… "
          #11 @30  @43"To Get Your Times Table! "
           ;

%macro multtbl;
 %display MultTable;
DATA test; file print notitle;

  put "Multiplication Table: the &t1 's" //;
    %DO I = 1 %TO &rng;
              retain x y z;
                    x = %eval(&t1 + 0);
                    y = %eval(&I + 0);
             z = %eval(&t1 * &I);
       put @10 x 'X ' y '= ' z;

    %END;
RUN;
%mend multtbl;
%multtbl;
```

The screen should appear as:

MULTIPLICATION MAGICIAN

Your wish is my command!

Select Your Times Table Number: 12

How Many Times Do You Want To Multiply By? 12

Press ENTER 3 Times…

To Get Your Times Table!

and the result is:

Multiplication Table: the 12 's

12 X 1 = 12
12 X 2 = 24
12 X 3 = 36
12 X 4 = 48
12 X 5 = 60
12 X 6 = 72
12 X 7 = 84
12 X 8 = 96
12 X 9 = 108
12 X 10 = 120
12 X 11 = 132
12 X 12 = 144

Of course, this needs to repeat upon command, but for now, lets take a look at the structure of the %WINDOW interface.

```
%window MultTable irow=5 rows=25 icolumn=15 columns=55
        #1 @20 "MULTIPLICATION MAGICIAN" attr = highlight
        #3 @20 "Your wish is my command!"
        #5 @25 "Select Your Times Table Number: " +2  t1 4 attr = underline
        #7 @25 "How Many Times Do You Want To Multiply By?" +2  rng          4 attr = underline
        #9 @30 "Press ENTER" 3 Times… "
        #11 @30  @43"To Get Your Times Table! "
         ;
```

Basically, you need to invoke the %WINDOW interface, and provide a name that SAS will use to display the interface. We've called ours "MultTable." The term "irow" indicates the Initial Row, and what follows is the maximum number of rows that the window can hold. Following this logic, "icolumn" indicates the Initial Column, and what follows is the maximum number of columns.

The # indicates to the internal pointer that a line will be commencing at the initial row.
The "@" is the familiar pointer control that controls the column placement.

Literal prompts are enclosed within the quotes. The "attr = " adds an effect to the field, such as an underline or a highlight. The "+2" adds spaces from the verbal prompts.

VERY IMPORTANT: The macro variables being passed as parameters are declared in the %WINDOW interface. Here, we are declaring t1 as the multiplier, and rng as the upper limit of iterations.

Is this the limit of %WINDOW's capabilities? Of course not, but this example would make a good cheat sheet for coders to keep in their 'toolkits' .

Applications

If an interactive script is being considered to launch a SAS macro-generated process, %WINDOW is an excellent alternative to those Unix users who have completely forgotten their Shell Script course taken months, or years ago. Manipulating our %WINDOW code:

```
%window MultTable irow=5 rows=25 icolumn=15 columns=55
        #1 @20 "MULTIPLICATION MAGICIAN" attr = highlight
        #3 @20 "Your wish is my command!"
        #5 @25 "Select Your Times Table Number: " +2  t1 4 attr = underline
```

```
          #7 @25 "NOTE: Default is Table Number Times 12"
          #9 @30 "Press ENTER 3 Times… "
          #11 @30  @43"To Get Your Times Table! "
           ;

%macro multtbl;
 %display MultTable;
%include '/multitable/maclib/multimacs.sas'
 %multmac.&&t1;
```

The program will then select a macro from a group of available macros in a SAS code set that will multiply by the self-selected &t1, such as:

```
%multimac12;
DATA test; file print notitle;

   put "Multiplication Table: the &t1 's" //;
     %DO I = 1 %TO 12;
               retain x y z;
                     x = %eval(&t1 + 0);
                     y = %eval(&I + 0);
    z = %eval(&t1 * &I);
       put @10 x 'X ' y '= ' z;

     %END;
RUN;
%mend mutimac12;

%mend multtbl;
%multtbl;
[1]
```

Personally, I prefer the initial calculation-based code, but the illustrates the ability to pass parameters to access programs on call!

Limitations:
The limitations are obvious: in Base SAS, we're limited to SAS code, SAS datasets, or SAS data steps that read external flat-files. However, if we can link to the server that has the SAS data and code processing capabilities, our limitations are only set by the limits of imagination!

Uses as a laboratory interface:
In the experience of this writer, entering statistics from controlled experiments, such as date, time, exposure, dosage, cage number, run number, etc. are parameterized, and excellent applications of %WINDOW! With Versions 8 and above, or via PROC EXPORT, the results of each SAS dataset are easily exported to Excel spreadsheets, or other analytic mediums to allow easy display and inspection/manipulation of data. For example:

```
%window LabEntry irow=5 rows=25 icolumn=15 columns=55
          #1 @20 "Western Sky Labs" attr = highlight
          #3 @20 "Animal Exposure Study"
          #5 @25 "Enter Cage Number: "   +2  c1 4 attr = underline
          #7 @25 "Enter Day :"           +2  d2 4 attr = underline
          #9 @25 "Enter Time: "        +2  t3 4 attr = underline
    #11 @25 "Enter Exposure in ml:" +2  e4 4 attr = underline

           ;
```

Which corresponds to:

```
%macro labent;

 %display LabEntry;
```

```
DATA test; file print notitle;
     retain cagenum daynum timdt expnum;

          cagenum = &c1;
     daynum  = &d2;
     timdt   = &t3;
          expnum  = &e4;

RUN;

PROC REG data=test;
  (SAS statements would follow)
```

As a hyperlink/parameter model:
Uses for %WINDOW would save time and effort in testing hyperlinks, parameter calls, and introduction of external parameters to a back-end SAS operation without the issue of creating a HTML page, which could look like:

```
<html xlmns = http://www.w3.org/1999/xhtml>
          <head>
       <title>Western Sky Labs Entry Page</title>
     </head>
     <form method = “post” action = “cgi-bin/formmail”>
     <body>

       <h1>Western Sky Labs</h1>
            <p> Animal Exposure Study</p>

            <p><label> Enter Cage Number:
               <input name = “c1” type = “text” size = “4” /> </label>

                    <label> Enter Day :
               <input name = “d2” type = “text” size = “4” /> </label>

                    <label> Enter Time:
               <input name = “t3” type = “text” size = “4” /> </label>

                    <label> Enter Exposure in ml:
               <input name = “e4” type = “text” size = “4” /> </label>

            </p>
          </form>
</html> [2]
```

Not as simple as our %WINDOW example, right?

Plus, this just introduces the parameters to a Common Gateway Interface at the “forms method” statement, which directs the parameters to middleware.

Why not just test and script on the server, and at the same time, create a utilitarian script that allows data manipulation and program execution without the hassle of engineering a web page?

You can always write a web page later, after you’ve proofed your code!

CONTACT INFORMATION
Your comments and questions are particularly valued and encouraged. Contact the author at:
Alan Mann
Independent Consultant
725 second Street,
Martinsburg, WV 25401

(304) 671-0475 (voice)
(304) 263-6940 (fax)
sky55@ix.netcom.com

NOTES:

[1] SAS Institute. SAS Macro Language: Reference, Version 8. Cary, NC: Sas Institute, 1999.

[2] Deitel, Harvey M, Deitel, P.J., Nieto, T.R. Internet and World Wide Web: How to Program. . Upper Saddle River, NJ: Prentice-Hall, 2002, pp. 138-140.