Paper 89-25

Hidden Nuggets in Version 8: New Informats, Formats, and Functions

Mike Rhoads, Westat, Rockville, MD

ABSTRACT

Much of the interest in Versions 7 and 8 of the SAS® system has been in major new features such as the Output Delivery System and long variable and table names. Hidden within the "Changes and Enhancements" documentation, however, are many other useful new items. This paper summarizes several enhancements to SAS informats, formats, and functions, and describes how they can make life easier for programmers who know about them.

INTRODUCTION

Informats, formats, and functions -- often referred to affectionately by true SAS aficionados as "IFFs" -- are among the most valuable tools in the SAS system. They enable a skilled SAS programmer to read data in an almost unbelievable variety of representations, and to write data back out in just as many ways, if not a few more. What other piece of software can read in RMF timestamps, octal and hex values, and column-binary files, and write values out as fractions, Julian dates, and Roman numerals? In between using informats to get data into SAS and formats to get data out, you can use over 150 SAS functions to compute accumulated depreciation, convert FIPS codes to state names, generate random variates from a Cauchy distribution, or translate all occurrences of one character to another.

When experienced SAS programmers review programs written by SAS novices, one common theme is the amount of code written to accomplish some task that could be done with a single call to an appropriate function. Such code can take hours to write, and even longer to debug. The solution to this problem is not to memorize the exact syntax of hundreds of IFFs, but rather to maintain a state of awareness regarding the sort of facilities that are available. Whenever you encounter a new task in the areas of data input, output, and manipulation, it never hurts to check whether an informat, format, or function will greatly simplify your program.

An important aspect of maintaining this awareness is to keep up to date with new releases of SAS software. This paper tries to help with this by summarizing some of the new functions, informats, and formats that were added to the SAS system in Version 7. More detailed syntax and explanations can be found in the official documentation.

MORE "FUNCTIONALITY"

A WELCOME IMPROVEMENT TO AN OLD FRIEND

The SCAN function, which enables you to extract a given "word" from a character expression, has always been a useful tool in the SAS software arsenal. While it has been easy to determine the first word in a string or to pull out all words from left to right, it has not been as straightforward to code algorithms where right-to-left processing is required. Version 8 addresses this shortcoming by allowing negative integers to be passed to SCAN. If its 2nd argument is negative, SCAN starts counting from the right rather than from the left, as this example illustrates.

```
DATA _NULL_;
FirstVar   = 'What is the final word';
FinalWord = SCAN(FirstVar,-1);
PUT FinalWord=;
```

A "CONSTANT" SOURCE OF INFORMATION

Have you ever needed to include a standard mathematical value such as pi or Euler's constant in one of your programs? Have you ever wanted to write platform-independent code, but needed to know the smallest possible number that could be stored on your computer, or the largest integer that is safe to store in a particular number of bytes? If so, the new CONSTANT function is for you.

CONSTANT requires one argument, which is a character string keyword indicating the value desired. (With a few of these keywords, you may provide a second optional argument.) The example below shows a few of the possible arguments.

```
DATA _NULL_;
Pi        = CONSTANT('PI');
Euler     = CONSTANT('EULER');
MinVal    = CONSTANT('SMALL');
SafeInt3 = CONSTANT('EXACTINT',3);
PUT _ALL_;
```

ENDLESS POSSIBILITIES (AND PROBABILITIES)

SAS software has always been able to compute factorials, permutations, and combinations, but not in an easy-to-use manner. For instance, most of us who are not mathematicians and have only an occasional need for factorials are unlikely to remember both to use the GAMMA function and to specify as its argument the integer one *greater than* the one whose factorial is desired (or is that *less than*?). There also have not been functions to directly compute the number of permutations or combinations of *n* objects taken *r* at a time. Three new functions in Version 8 make these tasks much more straightforward.

```
DATA _NULL_;
F = FACT(6);
P = PERM(6,2);
C = COMB(6,2);
PUT _ALL_;
```

JUST THE ATTRIBUTES, MA'AM

As SAS software has evolved through the years, it has provided an ever-increasing set of methods for determining and manipulating the attributes of the variables in a SAS data set. When it comes to the variables in the program data vector of a DATA step, however, we have been limited to getting a variable's name and label (by way of the CALL VNAME and CALL LABEL routines). Version 8 provides an extensive set of functions for getting additional information about DATA step variables, such as type, length, informat, and format. (Since their names all start with V, I have taken to calling them the V-functions.) These come in two flavors. The argument to each of the functions in the first set must be a DATA step variable or array reference. Each function in the parallel second set (which can be distinguished because their names all end with X) takes a character string as its argument, which must resolve to the name of a variable. For example, either VFORMATW (Numvar) or VFORMATWX ('Numvar') will return the width of the format for the variable NumVar.

The %MissFill macro below is designed to take a list of variables in a DATA step and to change their values to a string of digits (such as 99997) if they are missing. The length of the fill is to be determined by the length of the variable if the variable is character, or by the width of the variable's format if the variable is numeric. While this is not the most efficient code in the world, it does illustrate how to use several of the V-functions.

```
%MACRO MissFill
  (VarList,FillDigit,FinalDigit);
/* Error handling omitted for brevity */
%LOCAL ThisVar I;
%LET I = 1;
%LET ThisVar = %SCAN(&VarList,&I);
%DO %WHILE (&ThisVar ^=     );
  /* If missing, process according to var
```

```
    type */
      IF MISSING(&ThisVar) THEN DO;
        /* Using VTYPEX instead of VTYPE due to
           V7 bug */
        IF VTYPEX("&ThisVar") = 'N' THEN DO;
          IF VFORMATW(&ThisVar) = 1
            THEN &ThisVar = INPUT("&FinalDigit",1.);
          ELSE &ThisVar =
              INPUT(REPEAT("&FillDigit",
                VFORMATW(&ThisVar)-2)
                ||"&FinalDigit",BEST.);
        END;
        ELSE DO;  /* Character variable */
          IF VLENGTH(&ThisVar) = 1
            THEN &ThisVar = "&FinalDigit";
          ELSE &ThisVar =
              REPEAT("&FillDigit",
                VLENGTH(&ThisVar)-2)||"&FinalDigit";
        END;
      END;
      %LET I = %EVAL(&I + 1);
      %LET ThisVar = %SCAN(&VarList,&I);
    %END;
    %MEND MissFill;

    DATA _NULL_;
    LENGTH  Char1 $ 1  Char2 $ 2  Char6 $ 6;
    FORMAT  Num1 1.    Num2 2.    Num8 8.;
    %MissFill
      (Char1 Char2 Char6 Num1 Num2 Num8, 9, 7)
    PUT _ALL_;
```

"MISSING" IN ACTION

If you were paying close attention to the code in the previous example, you noticed that we slipped in one more new function: MISSING. This function returns a 1 if its argument is missing or a 0 if it's not. MISSING is unique among SAS functions in that its argument may be either a numeric or character variable or expression. It can both add clarity to your code and eliminate the need to remember the collating sequence of special numeric missing values when these are used.

INFORMATS AND FORMATS: DATES FOR A NEW MILLENNIUM

GETTING THEM IN

SAS informats get a modest but welcome addition with the YYMMNw informat. This informat lets you read in year and month values in the form 199910 (or 9910 for you 2-digit-year diehards) and produce a SAS date value, which represents the first day of the specified month.

```
   DATA _NULL_;
   New = INPUT('200001', YYMMN6.);
   PUT 'New millennium starts ' New : WORDDATE.;
```

AND GETTING THEM OUT

The recent focus on Y2K assessment and compliance has revealed some limitations with the traditional SAS date formats. A common need is to get date values onto an external file in the form *yyyymmdd* so that they can be read by other software. Up until Version 7, there was not a date format that would accomplish this directly. Version 7 introduced the YYMMDDxw format. This format is similar to the familiar YYMMDD, except that you can specify the date separator you want by replacing the "x" with a designated letter. The separators available are Blank, Colon, Dash, None, Period, and Slash. If you prefer to order your dates differently, DDMMYYxw and MMDDYYxw formats are also supplied.

```
   DATA _NULL_;
   July4 = '4JUL1999'D;
   PUT 'How to get yyyymmdd output: '
       July4 YYMMDDN8.;
```

HAVE THEM YOUR WAY

But perhaps you want even more flexibility in how you put date values into a report or file. Certainly SAS Institute could never add enough different date formats to satisfy everyone, given the wide variety of ways to present date information. So, they added the concept of "date directives" to PROC FORMAT's PICTURE statement to let us effectively create our own date formats. Each "directive" represents a piece of information about a date -- for instance, %b produces an abbreviated month name, while %B produces the full name of the month. Leading zeroes may be either printed or suppressed. For example, the following code would produce "Monday, September 6":

```
   PROC FORMAT;
     PICTURE MyDateP
       LOW-HIGH='%A, %B %d' (DATATYPE=DATE);

   DATA _NULL_;
     LaborDay = '6SEP1999'D;
     PUT 'Date directives: '
       LaborDay : MyDateP30.;
```

CONCLUSION

Among its many other virtues, Version 8 of SAS software offers a number of new and improved informats, formats, and functions. While they are not likely to end hunger, cure cancer, or make you as rich as Bill Gates (or even Jim Goodnight), using them in appropriate situations can improve your SAS programs, productivity, and general peace of mind. Enjoy!

CONTACT INFORMATION

The author may be reached at:

Mike Rhoads
Westat
1650 Research Blvd.
Rockville, MD 20850
RhoadsM1@Westat.com