

"F-WORDS" for More Productive SAS® DATA Step Programming.

Bill Calvert, Service Research, Inc. Rockville, MD

ABSTRACT:

Do you find your-self using "F-WORDS" when you have to write SAS DATA Step programs? If not, maybe you should expand your vocabulary and/or improve your programming skill. Some common problems you may have encountered in the DATA Step are described with the appropriate "F-WORD" to apply. Specific examples are presented with concomitant discussion of application style and technique. Helpful hints will be provided to facilitate productivity gains. "F-WORDS" such as FUNCTIONS, FORMATS, and FILENAMEs will be considered.

INTRODUCTION:

If you know what a thing is called you can look it up. When you know that something can be done and have mastered some of the vocabulary, the details are not so important until you actually have to produce something. In this tutorial data manipulation problems are presented with suggested solutions using Functions, FORMATS, and FILENAMEs (referred to here as "F-WORDS") in SAS/Base. No effort has been made to present a comprehensive compendium of uses.

Some details and complexities have been glossed over. Each "F-WORD" could easily be the topic for one or more presentations. Instead of being comprehensive, the intent is to expose you to the ideas and reinforce the ideas with simple, easily understood examples. When you have similar data manipulation problems in the future, you will know what to look for in the SAS manuals, or what to ask your local expert. You may even be better equipped to look-up information in the treasure trove of 'Proceedings' such as those produced by SESUG.

FUNCTIONS:

I do not know how I got anything done in the SAS DATA step before I discovered functions. The collection of functions available in the SAS System are often the only reasonable way to accomplish a data manipulation task.

Manipulating Text

Suppose, for example, that you have a data file that includes a list of first names of students in a large school. You need to find out how many students are named 'Sally'. Consider this statement from a DATA step:

```
IF UPCASE(NAME) = 'SALLY'
  THEN COUNT = 1;
```

The UPCASE function assures that both 'Sally' and 'SALLY' are counted, as well as other possibilities such as 'sALLY', etc. that may have occurred as typing mistakes.

Reality is usually more complicated. Assume the variable NAME in a similar data set actually consists of both first name and last name. We could count the number of students with 'Sally' for a first name using the SCAN function:

```
IF SCAN(UPCASE(NAME),1) = 'SALLY'
  THEN COUNT=1;
```

This statement would compare the upper case value of the first word in the variable NAME to 'SALLY' and set the COUNT variable to 1 if a 'Sally' match was found.

The variable COUNT can be added up using your favorite method to find out how often 'Sally' occurs. If further processing is required on the 'Sally' records, COUNT can be used as a Flag to subset the records or trigger special processing.

Beginning Tutorials

Now that you have the idea, consider some of the related functions that may be helpful in parsing and counting text or character strings. You might use LEFT, RIGHT, or REVERSE; they do pretty much what their name suggests. You can also use SUBSTR to select position specific substrings from a larger string. With a little creativity, you can realize the tremendous power of SAS character manipulation functions.

Convert Character to Numeric

The need to convert data of the character type to data of the numeric type, occurs frequently. The reasons for converting types are at least as diverse as the sources of data. Fortunately SAS provides at least two ways to do type conversions, the automatic way, and by the use of functions and formats. (More about formats later.) Each approach has its problems; let the user be wary.

To trigger automatic conversion simply use the character variable as if it were numeric. SAS will do it with varying degrees of complaint registered in the SAS Log. When there are characters other than 0-9 and . in the character type field automatic conversion usually has problems. You need to know what is in the data.

If you have numbers with commas they can be read reliably using the INPUT function and the COMMA format. Consider this statement from a DATA step:

```
BIGNUMBR = INPUT(CHARNUM,COMMA12.);
```

This statement reads the character type field (CHARNUM) containing large numbers with commas, using the COMMA format, and assigns it to the numeric type field BIGNUMBR.

Convert Numeric to Character

To convert numeric type variables to character type variables use the PUT function. There are a large number of standard formats that can be applied. For example:

```
AMOUNT = PUT(CONTRIB,DOLLAR10.2);
```

This statement converts the numeric type variable CONTRIB representing dollars contributed to 'the cause' to a character value complete with dollar sign and decimal point.

Keep in mind that the use of a function to make a conversion between types depends on the data having a consistent format. Inconsistently entered data are difficult regardless of the approach you use. Sometimes you have to make the data more consistent first, to get reliable conversion results.

Take a Quick Sample

Consider one last data manipulation problem. Large complex data sets take lots of processing time. It works best if you use a small part of the data to develop your program. Sometimes taking the first 100 observations is not satisfactory due to the chronological nature of the data, or the sorted order, etc. Solution: take a sample. Use the UNIFORM function to generate a pseudo random number between zero and one with each iteration of the DATA step. Use a specific seed (1117) so you always select the same records. When UNIFORM returns a value within a certain range keep the record. Try this code:

```
IF UNIFORM(1117) GE 0.85 ; ** 15% ;
```

Note that this is a subsetting IF statement. Insert this statement early in your code to select approximately 15% of the records. This is not necessarily a statistically correct sample, but it allows records to be selected from all parts of the data set. Change the seed or change the range to select a different portion of the records. You can use a similar approach to select mutually exclusive samples too. Beyond that, there are other functions, such as RANNOR, which generate random numbers from specific distributions to allow a probability sample to be selected. (For RANNOR the distribution is Normal, with mean equal to zero and standard deviation equal to one).

The Wonderful World of Functions

If you get some time, or need inspiration, browse the function section of your SAS language manual. Functions are available from A to Z. Besides those already mentioned there are functions for arithmetic and statistical operations, probability distributions, date and time issues (how many days between two dates, for example), special purposes (such as PUT and INPUT, discussed here), and even state and ZIP Code functions. Chances are, if you have a problem, there is a function that will help you get it done faster.

FORMATS:

FORMATs are about appearances. FORMATs (and INFORMATs) are a valuable feature of the SAS System. The basic skills required to use FORMATs include defining a simple FORMAT, and using it to influence the way a variable value appears to a SAS procedure (such as PROC FREQ) or how the variable value appears on the output.

Define a FORMAT or two

Suppose you are examining data on drinking behavior in a school aged population. The sponsor of the research wants to know what the drinking behavior patterns are for 3 arbitrary groups, but the data has AGE as a continuous variable. So you define your own format for the age groups:

```
PROC FORMAT ;
** use with age variable
   to set groups ** ;
VALUE F_AGEGRP
    4-12   = 'Child'
    13-18  = 'Teen '
    19-25  = 'Youth'
    OTHER  = 'BAD Data';

VALUE YESNO  0 = 'No '
              1 = 'Yes';

RUN;
```

Notice that a FORMAT (YESNO) was also defined in this step for variables that are coded one for 'yes' and zero for 'No'. In addition to the VALUE statement PROC FORMAT also has a PICTURE statement that can be used to define other kinds of formats. To emphasize the value of the format definition process and to minimize complexity only the VALUE statement is used here.

With the FORMAT F_AGEGRP defined the continuous variable AGE can now appear to PROC FREQ as an indicator of the appropriate age group. The values in AGE are not changed. Also, variables coded with zero/one values can be displayed explicitly as Yes/No using the YESNO FORMAT.

Use the FORMATs

To use the new FORMAT to count binge drinkers (coded in the variable BINGE as zero or one) in each age category try some code like this:

```
PROC FREQ DATA=SKOOL;
TITLE2 'Binge Drinking
        by Age Group';
TABLES AGE * BINGE /CHISQ;
FORMAT AGE F_AGEGRP. BINGE YESNO.;
RUN;
```

Notice that the FORMAT statement assigns the FORMAT defined earlier to the variable AGE. The same statement also includes a format for BINGE to make the zero/one values appear as No/Yes in the output.

When you use FORMATs to label numeric codes (the values of BINGE) so they are understandable on the output, it doesn't matter if the program gets separated from the output. Little things help people work together.

FILENAMES:

The FILENAME statement is used to attach a 'handle' or reference name to an external (non-SAS) file. Once the handle has been defined to SAS you can reference that file by the convenient SAS name. (Eight character names are easier to type, etc.). This feature works well when you need to read data that is captured in an external data file. For example:

```
FILENAME GOODDATA
    'C:\KEYENTRY\PROJECT1\MYDATA.TXT';

DATA NEW ;
  INFILE GOODDATA LRECL=210 PAD ;
  INPUT VAR1 VAR2 VAR3 VAR4 VAR5 ;
  ** DO SOME PROCESSING HERE **;

RUN;
```

This approach is succinct and allows some flexibility in the line lengths for the data and other options that are available using the INFILE statement. This small DATA Step could be done in another way using the CARDS statement and copying the data right into your program. That would make the program much larger (even awkward, if you have much data). Sometimes you need to produce a report or text file in a

Beginning Tutorials

specific format. You might use a FILENAME to identify the external file to your SAS program and write the file using PUT statements.

```
FILENAME REPORT
'C:\FINALTXT\PROJECT1\MYRPT.TXT';

DATA _NULL_ ;
SET SUMDATA;
FILE REPORT LRECL=80 ;
** DEFINE THE PRINT LINE** ;
PUT @10 VAR1 @25 VAR2 1.
    VAR3 @60 VAR4 @70
    VAR5 6.2 ;

RUN;
```

This small program segment takes the results of a data summarization process (SUMDATA) and writes a file (REPORT) with specifically formatted lines containing five variables. This file could then be printed or used in other ways outside the SAS System.

A FILENAME statement can also be used to setup code segments created or maintained by another person or process to be incorporated into your program using %INCLUDE to good effect.

```
FILENAME CRUNCH
'C:\PGMS\CRUNCH.SAS';

** use Tom's number cruncher** ;
%INCLUDE CRUNCH ;
```

This code would insert the code referenced in the FILENAME statement at the position where the %INCLUDE is found. The FILENAME statement must occur in the program stream before the handle is used. Putting them at the beginning of the program is a good idea. Note that there are some restrictions on the use of INCLUDE statements. I try to restrict my INCLUDE files to have only complete statements. I also use a naming convention to differentiate INCLUDEs made of statements from those that consist of complete steps.

These uses of the FILENAME statement are a small beginning. There are other useful ways to apply the FILENAME statement. Try using it in conjunction with PROC PRINTTO to write SAS outputs to a file or with PROC DBF or PROC DIF to convert specific external files. Consider the possibilities.

CONCLUSION:

A select group of SAS features that happen to start with the letter 'F' are presented here with some simple examples of the context of their use. Learning to use these 'F-WORDS' in rudimentary ways can make your DATA step programming more Flexible and efficient. The use of FORMATS and Functions can make your programs easier to understand and better documented (refer to the copious SAS documentation). Some creativity and an inkling to go into new things may be required. For the Forward looking programming analyst, the reward can be in improved productivity and the Fun of conquering new techniques.

REFERENCES:

- SAS Institute Inc., *SAS® Language: Reference, Version 6*, Cary, NC: SAS Institute Inc., 1990.
- Dilorio, Frank C. 1991. *SAS Applications Programming, A Gentle Introduction*, PWS-Kent Publishing Company, Boston, MA.

□SAS is a registered trademark of SAS Institute Inc. Cary, NC, USA