Paper 134-30

A Hands-On Introduction to SAS® DATA Step Programming

Debbie Buck, D. B. & P. Associates, Houston, TX

ABSTRACT

The SAS DATA step is one of the most powerful and versatile software tools available for handling and manipulating data files, especially large data sets. It is also sometimes somewhat confusing to SAS programmers as to what the different statements do, what the correct syntax is for these statements, and how they can help you achieve your goals.

In this workshop we will examine how to get data into a SAS data set, how to create new variables or modify existing variables, how to conditionally handle variables or observations, how to handle a group of variables in the same way, and how to control which observations are written to the SAS data set.

INTRODUCTION

To become familiar with SAS DATA step programming, we will discuss a number of commonly used DATA step statements and then work through exercises utilizing these statements. Some of these will include:

- How to create a SAS data set (DATA statements),
- How to get data into a SAS data set (INFILE/INPUT, SET, MERGE statements),
- How to create or modify variables (Assignment statements),
- How to conditionally execute statements (IF-THEN/ELSE statements),
- How to select specific observations (Subsetting IF statements),
- How to process a group of variables (ARRAY, DO-END statements).

Although there are a number of other statements and options available in the SAS DATA step, the statements above will provide you with a good grasp on working in the DATA step. We will also use some PROCs or other techniques to help you examine your data sets or data values.

Many of these statements could have (and have had) entire papers devoted to them individually, but this presentation is meant as an overview.

In this paper we will look at examples of a sample data set from a blood pressure medication study. This data set includes patient number, gender, race, date of birth, baseline blood pressure, mid-study blood pressure, and final blood pressure. Examples from this presentation were run on the SAS System for Windows, but the SAS statements are applicable to any SAS platform.

CREATING A SAS DATA SET

The first question is “How do I get the data into a SAS data set?”. The data may be in a flat file, existing SAS data set(s), or a file produced by another software program. If the data is in a flat file or existing SAS data sets, this is frequently, but not always, achieved in a DATA step. The DATA step always begins with a DATA statement. The purpose of the DATA statement is to tell SAS that you are starting a DATA step, to name the SAS data set(s) that you are creating, and to indicate whether this is a permanent or temporary SAS data set. The following are examples of DATA statements.

```
DATA NEWPT;              ← temporary SAS data set

LIBNAME DB ‘C:\SAS\MYDATA’;
   DATA DB.NEWPT;     ← permanent SAS data set
```

What is the purpose of the LIBNAME statement shown in this example, and why do we need it? When reading from or writing to a permanent SAS data set, the libref (in this case DB) serves as a nickname or an alias that tells SAS where the SAS data library physically exists. For this example, I am writing a SAS data set named NEWPT to the SAS data library that exists in the subdirectory C:\SAS\MYDATA.

Which statements follow the DATA statement depend upon where and in what form the data exist.

GETTING THE DATA INTO A SAS DATA SET

Is the Data in a Flat File?

What do I mean by a flat or external file? This is generally a text or ASCII file. A simple example of this might be the following layout where each record in the raw data file contains information on patient demographics and blood pressure readings.
Sample record –

```
01M110/10/25100 95 90
```

Variable	Columns	Type
Patient Number	1-2	numeric
Gender	3	M, F
Race	4	1=White 2=Black 3=Oriental 4=Other
Date of Birth	5-12	MM/DD/YY
Baseline BP	13-15	numeric
Mid-Study BP	16-18	numeric
Final BP	19-21	numeric

In order to read this external file into a SAS data set, you need two additional statements. You need to tell SAS where to find the raw data – the INFILE statement, and you need to tell SAS how to read each record – the INPUT statement. An example of the INFILE and INPUT statements is shown below.

```
DATA NEWPT;
 INFILE ‘A:\TEST1.DAT’;            ←  tells SAS where  raw data exists
      INPUT @1 PTNO 2.
             @3 GENDER $1.           ←  Formatted input
             @4 RACE $1.
             @5 BDATE MMDDYY8.
                BP1 13-15             ←  Column input
                BP2 16-18
                BP3 19-21;
RUN;
```

The INFILE statement has a number of options available which are dependent on the file structure, including specifying record length, directions on how to handle differing record lengths, and the ability to specify delimiters between variable values.

The INPUT statement is determined by the layout of the variable values within the records and whether the data are standard character and numeric or nonstandard. What is meant by standard numeric data? Standard numeric data can include integers, decimal points, positive and negative numbers and scientific (E) notation. Nonstandard numeric data include date and time values to be converted to SAS date and time values, hexadecimal and packed decimal data, and data values with embedded commas and dollar signs. The example above uses a combination of column and formatted input. Column input specifies the variable name, a dollar sign to indicate a character variable, and the beginning and ending columns. Formatted input shows the beginning column of the variable using the @ column pointer, the variable name, and the correct informat needed to read the data. Although there are several other types of INPUT statements, column and formatted are two of the most common. The form of the INPUT statement is also determined by whether there is one observation per record line as in the above example, multiple observations per record line, or multiple record lines per observation.

Did the Program Work Correctly?

One habit you should develop is to check the SAS log any time you run a SAS program or section of a program. The SAS log will give you valuable information on the number of observations and variables, as well as any NOTEs, WARNINGs, or ERROR messages. The following is the log from our DATA step above.

Partial SAS Log

```
NOTE: The infile 'A:\TEST1.DAT' is:
         File Name=A:\TEST1.DAT,
         RECFM=V,LRECL=256

   NOTE: 81 records were read from the infile 'A:\TEST1.DAT'.
         The minimum record length was 21.
         The maximum record length was 21.
   NOTE: The data set WORK.NEWPT has 81 observations and 7 variables.
```

For this example, we can see that 81 records were read (which is the number we knew were in the flat file – make sure you know your data) and the NEWPT SAS data set has the expected number of observations and variables. Also, there are no WARNING or ERROR messages. It is tempting to look at any resulting output first if you are working in the SAS Display Manager, since the Output window opens automatically. However, make sure to always examine the log.

Every SAS data set has 2 parts – a descriptor portion and a data value portion. The descriptor portion is like an internal header for the data set and, among other things, contains variable names, variable types (character or numeric), and variable lengths. It also includes any labels or formats associated with the variables. One way you can examine the descriptor portion of a SAS data set is with PROC CONTENTS. Running the following code shows the descriptor portion of the NEWPT temporary SAS data set.

```
PROC CONTENTS DATA=NEWPT;
RUN;
```

Partial Output

```
                                    The CONTENTS Procedure

        Data Set Name: WORK.NEWPT                                 Observations:         81
        Member Type:   DATA                                       Variables:            7
        Engine:        V8                                         Indexes:              0
        Created:       12:07 Saturday, February 5, 2005           Observation Length:   48
        Last Modified: 12:07 Saturday, February 5, 2005           Deleted Observations: 0
        Protection:                                               Compressed:           NO
        Data Set Type:                                            Sorted:               NO
        Label:



                      -----Alphabetic List of Variables and Attributes-----

                               #    Variable    Type    Len    Pos
                               ___________________________________

                               4    BDATE       Num       8      8
                               5    BP1         Num       8     16
                               6    BP2         Num       8     24
                               7    BP3         Num       8     32
                               2    GENDER      Char      1     40
                               1    PTNO        Num       8      0
                               3    RACE        Char      1     41
```

To examine the data value portion of the SAS data set, you can use PROC PRINT to display the actual observations in the data set.

```
PROC PRINT DATA=NEWPT;
RUN;
```

Partial Output

```
Obs    PTNO    GENDER    RACE      BDATE     BP1    BP2    BP3

  1      1       M         1      -12501     100     95     90
  2      2       M         1       -8797      94     90     85
  3      3       F         1        7334      89     78      0
  4      4       F         1        5583      93     93     85
  5      5       F         1       -1768      93      0     90
```

If you are working in the Display Manager, you can also use the Explorer window to examine both the descriptor and data value portions of your data sets. In the Explorer window, if you double-click on Libraries, you will see an icon for each active library. Since we've created a temporary SAS data set named NEWPT, if you double-click on WORK an icon for NEWPT appears. Right-clicking on the icon causes a pull-down menu to appear. Selecting VIEW COLUMNS results in a window that contains variable information from the descriptor portion. Selecting OPEN displays the data value portion.

Is the Data in an Existing SAS data set?

If the data you need already exists in one or more SAS data sets, then the INFILE and INPUT statements are replaced by a SET, MERGE, or UPDATE statement. In this paper we will consider the SET and MERGE statements. Which is the appropriate statement to use?

SET Statements

If you need to bring data in from an existing SAS data set so that you can modify it, then the SET statement will, by default, bring into the new data set all variables and all observations from the existing SAS data set. The form of this DATA step is as follows.

```
DATA new-data-set;
  SET old-data-set;
RUN;
```

Although the examples in this paper use temporary SAS data sets, these could be any combination of permanent and temporary SAS data sets.

In looking at the output in the example above, we see that BDATE (Date of Birth) needs to be displayed differently to be meaningful. We would also like to see the BDATE variable printed with a column header that makes it more clear as to what information this variable contains. Therefore, in our new data set we want to include a FORMAT statement and a LABEL statement.

The form of the FORMAT statement, which assigns a format to be associated with a variable (how to display the variable) is as follows.

FORMAT *variable-name format-name*.;

Note the period in the format-name. This is what tells SAS that it is a format, not a variable name. SAS includes a large number of formats for displaying dates, times, currencies, and other types of data. You can also create your own formats using PROC FORMAT.

The LABEL statement associates a column header with a variable. The form of a LABEL statement is

LABEL *variable-name*='*desired text*';

In SAS Version 8 and above the text portion in a LABEL statement can be up to 256 characters long.

The following code creates a new SAS data set, NEWPT_REV that includes all variables and observations from the existing SAS data set NEWPT. It also associates a format and label with the variable BDATE.

```
DATA NEWPT_REV;
   SET NEWPT;
     FORMAT BDATE DATE9.;
     LABEL BDATE='Date of Birth';
RUN;

PROC PRINT DATA=NEWPT_REV LABEL;
RUN;
```

Partial Output

```
                                          Date of
       Obs     PTNO     GENDER     RACE        Birth     BP1     BP2     BP3

         1        1       M          1     10OCT1925     100      95      90
         2        2       M          1     01DEC1935      94      90      85
         3        3       F          1     30JAN1980      89      78       0
         4        4       F          1     15APR1975      93      93      85
         5        5       F          1     28FEB1955      93       0      90
```

Do you need to concatenate (or stack) the data from two or more SAS data sets? Then the SET statement is also the appropriate statement for this situation. Again, by default, all variables and observations in all of the SAS data sets listed in the SET statement will be included in the new SAS data set. The following is an example of concatenating two existing SAS data sets.

```
DATA NEWPT_REV;
  SET OLDPT1 OLDPT2;
RUN;
```

MERGE Statements

Do you need to combine (or join) two or more data sets by some common variable(s)? Then the MERGE statement is appropriate. The following example joins information from the NEWPT data set with the observations from the data set TREAT, matching information by the variable PTNO. The NEWPT_REV data set will contain all variables and observations from the NEWPT and TREAT data sets.

```
PROC SORT DATA=NEWPT;
  BY PTNO;

PROC SORT DATA=TREAT;
  BY PTNO;

DATA NEWPT_REV;
   MERGE NEWPT TREAT;
    BY PTNO;
RUN;
```

Note that for match-merging, the existing SAS data sets must be sorted by or indexed on the variable(s) in the BY statement.

CREATING OR MODIFYING VARIABLES

Assignment Statements

Now that you can get the data into your new SAS data set, you need to consider whether the data needs to be manipulated in some way, such as creating or modifying variables.

If you need to create new variables, the most common way is with assignment statements. The form of an assignment statement is

variable=expression;

The expression can be a constant, mathematical operation, or a function. SAS has a large number of functions to carry out a variety of operations. The form of a function is as follows.

function-name(*argument1*, *argument2*, etc.)

The arguments for a function are dependent upon the purpose and specific requirements for a given function.

For example, if you need to create a variable named AGE, based on the current date, you can use an assignment statement with a SAS date function in the DATA step.

```
DATA NEWPT_REV;
  SET NEWPT;
    AGE = (TODAY() – BDATE)/365;
RUN;
```

You can also use assignment statements to modify existing variables. The following shows an example of modifying an existing variable.

```
DATA NEWPT_REV;
  SET NEW;
    AGE = (TODAY() – BDATE)/365;
     AGE = ROUND(AGE,1);
RUN;
```

CONDITIONALLY EXECUTING STATEMENTS

IF-THEN/ELSE Statements

IF-THEN statements can be used to conditionally create or modify data. The form of an IF-THEN/ELSE statement is

IF *expression* THEN *statement*;
 ELSE *statement*;

For example, in this case you need to create a new variable to identify at which medical center a patient was enrolled, based on patient number. Patients 1-30 were enrolled at Center 1, patients 31-60 were enrolled at Center 2, and patient numbers 61 and above were enrolled at Center 3. The following IF-THEN/ELSE statements will allow you to conditionally assign values to your new variable, CENTER.

```
DATA NEWPT_REV;
  SET NEWPT;
   AGE=ROUND(((TODAY()-BDATE)/365),1);
    IF PTNO LE 30 THEN CENTER=1;
      ELSE IF PTNO GT 30 AND PTNO LE 60 THEN CENTER=2;
        ELSE IF PTNO GT 60 THEN CENTER=3;
RUN;
```

SELECTING OBSERVATIONS

Subsetting IF Statements

If you need to subset your data based upon the values of one or more variables, you can use a Subsetting IF statement. The form of a subsetting IF is as follows.

IF *expression*;

The expression is evaluated, and if the expression is true, processing continues on that observation. If the expression is false, processing on that observation stops, the observation is not saved to the data set being created, and the DATA step moves on to the next observation.

The following example limits the observations in the data set to those patients from Center 1 who are under 30 years of age.

```
DATA NEWPT_REV;
    SET NEWPT;
     AGE=ROUND(((TODAY()-BDATE)/365),1);
       IF PTNO LE 30 THEN CENTER=1;
         ELSE IF PTNO GT 30 AND PTNO LE 60 THEN CENTER=2;
          ELSE IF PTNO GT 60 THEN CENTER=3;
     IF CENTER=1 AND AGE LT 30;
 RUN;
```

PROCESSING A GROUP OF VARIABLES

ARRAY, DO-END Statements

Sometimes a group of variables needs to be handled in the same way. ARRAY statements can process a group of variables. The DO loop handles the repetitive processing. The form of an ARRAY statement follows.

ARRAY *array-name* {*number of elements*} $ *length list-of-elements*;

The $ is necessary for arrays of character variables. The length is optional.

The form of a DO loop is:

```
DO index-variable = start TO stop;
   SAS statements involving array-name{index-variable}
END;
```

The start and stop values specify which elements in the array should be the starting and stopping points in processing the DO loop.

In this example, blood pressure measurements were recorded in the raw data with a value of zero when the measurement was missing. The ARRAY statement in conjunction with DO-END statements changes the zero values to a missing value for all blood pressure variables.

```
DATA NEWPT_REV (DROP=I);
    SET NEWPT;
     AGE=ROUND(((TODAY()-BDATE)/365),1);
       IF PTNO LE 30 THEN CENTER=1;
        ELSE IF PTNO GT 30 AND PTNO LE 60 THEN CENTER=2;
         ELSE IF PTNO GT 60 THEN CENTER=3;
     IF CENTER=1 AND AGE LT 30;
       ARRAY BP {3} BP1 BP2 BP3;
             DO I=1 TO 3;
              IF BP{I}=0 THEN BP{I}=.;
             END;
RUN;
```

Output

```
Obs    PTNO    GENDER    RACE        Birth    BP1    BP2    BP3    AGE    CENTER
 1       3       F         1     30JAN1980     89     78     .      25       1
```

CONCLUSION

In this workshop we have examined how to get data into a SAS data set, how to create new variables or modify existing variables, how to conditionally handle variables or observations, how to control which observations are written to the SAS data set, and how to handle a group of variables in the same way. Each of the statements we have examined has additional capabilities which were beyond the scope of this presentation. Hopefully, this paper has provided you with guidelines to make SAS DATA step programming user-friendly to you.

REFERENCES

SAS Institute Inc. (1990) *SAS Language and Procedures Guide, Version 6, First Edition*, Cary, NC: SAS Institute Inc.

SAS Institute Inc. (1999) *SAS Online Doc., Version 8*, Cary, NC: SAS Institute Inc.

AUTHOR CONTACT INFORMATION

Debbie Buck
D. B. & P. Associates
Houston, TX 77095
Voice: 281-256-1619
Fax: 281-256-1634
Email: debbiebuck@houston.rr.com