Paper #IN08

Here's the Data, Here's the Report I Want - How Do I Get There?

Deborah Babcock Buck, D. B. & P. Associates, Houston, TX

ABSTRACT

SAS® programmers are often given a set of data and asked to generate a specific report. This can be a daunting task, especially for a relatively new SAS user. Where to start? This process involves two separate, but related phases. Phase one concerns the data itself. What form are the data in – is it raw data, an existing SAS data set or sets, or data from another software program? Also, does the data need to be manipulated, restructured, or summarized to provide the information needed for the SAS procedure? Phase two involves determining which PROC will provide the output in the desired form. Knowing whether the report needs a specific layout, customized information, or statistics is essential in deciding on the most appropriate PROC. This paper will present some guidelines to follow for producing the desired report in a logical and organized manner using Base SAS.

INTRODUCTION

When asked to produce a report using SAS, it is often confusing as to where to start. You may have taken an introductory course in SAS, read some books or manuals about writing SAS programs, or have been lucky enough to have examples of SAS programs on similar projects from co-workers, but you're just not sure where to begin. There are a number of things to consider in outlining the best way to generate the desired report.

- Where is the data, what form is it in, and what needs to be done to get the data into a SAS data set?
- Does the data need to be manipulated or modified in some way to provide the necessary information for the report?
- What procedure or approach in SAS will produce the report you want in the form desired?

The guidelines suggested in this paper were developed using Base SAS. These approaches are all valid with SAS Versions 6-9.

WHAT FORM ARE THE DATA IN?

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 I 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.

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.

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

Sample record –

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

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.

Is the Data in an Existing SAS data set?

If the data you need for your report already exists in one or more SAS data sets, then the INFILE and INPUT statements are replaced by a SET, MERGE, or UPDATE statement. Which is the most appropriate to use?

SET Statement

If you need to bring data in from an existing SAS data set so that you can modify it before generating the report, then the following code with the SET statement will, by default, bring into the new data set NEWPT all variables and all observations in the existing OLDPT1 SAS data set. (Although all of the examples in this section use temporary SAS data sets, these could be any combination of permanent and temporary SAS data sets.)

```
DATA NEWPT;
 SET OLDPT1;
RUN;
```

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;
 SET OLDPT1 OLDPT2;
RUN;
```

MERGE Statement

Do you need to combine (or join) two or more data sets by some common variable(s)? Then the MERGE statement is most appropriate. The following example joins information from the two data sets, matching information by the variable PTNO.

```
DATA NEWPT;
 MERGE OLDPT1 OLDPT2;
  BY PTNO;
RUN;
```

UPDATE Statement

Do you need to update the data in a master file with information from a transaction file? Then the UPDATE statement is the correct approach.

```
DATA NEWPT;
 UPDATE OLDPT TRANPT;
  BY PTNO;
RUN;
```

Why use the UPDATE statement instead of the MERGE statement? The UPDATE statement will allow you to change the values of variables in the master data set, add observations to the master data set, and not replace values in the master data set with missing values unless specifically requested.

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

Is the Data from another Software Program?

If the data exists in another software program, then data can be converted to a SAS data set using the SAS Import Wizard or the SAS/ACCESS product. Because this is platform specific, the topic will not be covered in this presentation.

DOES THE DATA NEED TO BE MANIPULATED TO BE IN THE APPROPRIATE FORM?

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 to make it ready for the SAS procedure that can produce the desired report. Do you need to create new variables, modify existing variable values, summarize data, or subset data?

If you need to create new variables, the most common way is with assignment statements. 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;
 SET OLDPT;
  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;
 SET OLDPT;
  AGE = (TODAY() - BDATE)/365;
   AGE = ROUND(AGE,1);
RUN;
```

IF-THEN statements can be used to conditionally create or modify data. 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 statements will allow you to conditionally assign values to your new variable, CENTER.

```
DATA NEWPT;
 SET OLDPT;
  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;
```

If you need to subset your data based upon the values of one or more variables, you can use a Subsetting IF statement. 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;
 SET OLDPT;
  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;
```

Sometimes a group of variables needs to be handled in the same way. ARRAY statements can process a group of variables. 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;
 SET OLDPT;
  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;
```

If your data needs to be summarized across variables within an observation, this is usually done within the DATA step using an assignment statement. If the data needs to be summarized across observations, this can be accomplished within the DATA step, but is often more easily accomplished within a PROC step. The following example includes the creation of a new variable – MEANBP – which is the mean of BP2 and BP3.

```
DATA NEWPT;
 SET OLDPT;
  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;
         MEANBP=MEAN(BP2,BP3);
RUN;
```

WHAT OUTPUT NEEDS TO BE PRESENTED?

Base SAS has a number of procedures to help you report your data in the form you would like to see it presented. In deciding which procedure is most appropriate for your needs, you must consider a number of factors. These factors include the following.

- Is it a detail or summary report?
- Do you need a specific layout?
- Do you need customized information?
- Do you need cross-tabulations or hierarchical groupings?
- Do you need statistics?

Some of the most frequently used Base SAS procedures include the following.

- SORT – Reorders the data as specified in the BY statement; necessary for some procedures to perform subgroup analyses; also needed when using MERGE or UPDATE statements.
- PRINT – Produces detail listings of data in SAS data set.
- MEANS/SUMMARY - Produces summary statistics, by default - number of observations, mean, standard deviation, minimum, and maximum.
- UNIVARIATE - Produces the above statistics, plus a number of other descriptive statistics including median, mode, percentiles, and the five lowest and highest values.
- FREQ – Generates counts and percentages for one-way and two-way tables; performs statistical tests on frequency data.
- CHART – Produces horizontal and vertical bar charts and pie charts.
- PLOT – Produces two-way plots.

- FORMAT – Creates user-defined formats to associate with data values.
- TABULATE – Produces one, two, and three-way summary tables with selected summary statistics.
- REPORT - Produces detail or summary reports; allows a great deal of flexibility in customizing reports.

When the purpose of the report is primarily for data cleaning, such as checking for outliers or looking for missing or miscoded values, the most commonly used procedures are PROC PRINT for detail reports and PROC MEANS, PROC UNIVARIATE, or PROC FREQ for summary reports. The following code uses PROC FREQ to check the data set for missing values or miscodes in GENDER.

```
PROC FREQ DATA=NEWPT;
 TABLES GENDER;
RUN;
```

OUTPUT FROM PROC FREQ

GENDER	Frequency	Percent
F	51	63.0
M	28	34.6
X	1	1.2
f	1	1.2

After discovering that there are two miscodes for GENDER, those observations can be displaying using PROC PRINT.

```
PROC PRINT DATA=NEWPT;
 WHERE GENDER NOT IN ('M','F');
 VAR PTNO GENDER;
RUN;
```

OUTPUT FROM PROC PRINT

OBS	PTNO	GENDER
8	8	f
28	28	X

For a pictorial look at your data, PROC CHART AND PROC PLOT will produce basic graphics. These procedures can also assist in data checking. The SAS/GRAPH product provides high resolution graphics for presentations.

One procedure that does not produce a report itself, but is often essential to producing customized reports is PROC FORMAT. Although SAS provides a multitude of SAS-defined formats, quite often a format is needed that is not available. PROC FORMAT allows you to create user-defined formats. In this study, we would like to see the GENDER and RACE variables printed with user-defined formats. The following SAS code will produce those formats. Note that it is a two-step process. First the format must be created; then the format must be assigned to the appropriate variable(s).

```
PROC FORMAT;
 VALUE $GENDER 'M'='Male'
               'F'='Female';
 VALUE $RACE '1'='White'
             '2'='Black'
             '3'='Oriental'
             '4'='Other';
RUN;

PROC PRINT DATA=NEWPT (OBS=4);
 VAR PTNO GENDER RACE;
 FORMAT GENDER $GENDER. RACE $RACE.;
RUN;
```

OUTPUT FROM PROC PRINT

OBS	PTNO	GENDER	RACE
1	1	Male	White
2	2	Male	White
3	3	Female	White
4	4	Female	White

PROC TABULATE and PROC REPORT are the real “work-horses” of customized report writing. Both of these procedures produce summary reports, while PROC REPORT can also generate detail reports. PROC REPORT has a number of capabilities that are difficult to find in other procedures. These features include, but are not limited to:

- providing customized information for subgroups or at the beginning or end of a report,
- underlining column headers,
- allowing variable values to extend across more than one line,
- creating new columns of information from existing variables.

The scope of this paper does not allow detailed descriptions of these procedures and how much flexibility each of these permits. However, detailed information on these procedures can be found in SAS documentation. (See references at the end of this paper.) An example of customized reports using PROC TABULATE and PROC REPORT are shown below. The following code which was used to generate Appendix Tables 1 and 2 produce very similar reports. Table 1 (PROC TABULATE) requires less programming code, while Table 2 (PROC REPORT) allows more flexibility in spacing between columns of information, underlining of headers, and customized text (if desired).

```
DATA NEWPT1;
 SET NEWPT;

  CBP3=BP3-BP1;
  IF GENDER='f' THEN GENDER='F';
  IF GENDER='X' THEN GENDER='M';

RUN;

OPTIONS MISSING='0' NONUMBER NODATE LINESIZE=142 PAGESIZE=42;

/**** PROC TABULATE (TABLE 1)****/

PROC TABULATE DATA=NEWPT1 FORMCHAR='           ';
 CLASS GENDER RACE;
 VAR BP1 BP3 CBP3;
 TABLE (RACE ALL),GENDER*
  (BP1*(N*F=3. MEAN*F=6.2 STD*F=6.2)
   BP3*(N*F=3. MEAN*F=6.2 STD*F=6.2)
   CBP3*(N*F=3. MEAN*F=6.2 STD*F=6.2))/RTS=15;
 KEYLABEL ALL='All'
          N='No.'
          MEAN='Mean'
          STD='Std Dev.';
 LABEL BP1='Baseline Diastolic BP'
       BP3='Final Diastolic BP'
       CBP3='Change in Diastolic BP'
       GENDER='Gender';
 FORMAT RACE $RACE. GENDER $GENDER.;
 TITLE1 'APPENDIX TABLE 1';
 TITLE2 'OUTPUT FROM PROC TABULATE';
RUN;
```

```
/**** PROC REPORT (TABLE 2) ****/

PROC REPORT DATA=NEWPT1 NOWD HEADSKIP HEADLINE SPACING=1;
 COLUMN RACE (GENDER,
  (( 'Baseline/Diastolic BP' BP1 BP1=MEANBST BP1=STDBST)
   ( 'Final/Diastolic BP' BP3 BP3=MEANST BP3=STDST)
   ( 'Change in/Diastolic BP' CBP3 CBP3=CMEANST CBP3=CSTDST)));
 DEFINE RACE/GROUP ORDER=INTERNAL WIDTH=6 FORMAT=$RACE. 'Race';
 DEFINE GENDER/ACROSS FORMAT=$GENDER. SPACING=2 'Gender';
 DEFINE BP1/ANALYSIS N WIDTH=3 FORMAT=3. CENTER SPACING=3 'No.';
 DEFINE MEANBST/ANALYSIS MEAN WIDTH=8 FORMAT=8.2 CENTER 'Mean';
 DEFINE STDBST/ANALYSIS STD WIDTH=6 FORMAT=6.2 CENTER 'Std. Dev.';
 DEFINE BP3/ANALYSIS N WIDTH=3 FORMAT=3. CENTER SPACING=3 'No.';
 DEFINE MEANST/ANALYSIS MEAN WIDTH=8 FORMAT=8.2 CENTER 'Mean';
 DEFINE STDST/ANALYSIS STD WIDTH=6 FORMAT=6.2 CENTER 'Std. Dev.';
 DEFINE CBP3/ANALYSIS N WIDTH=3 FORMAT=3. CENTER SPACING=3 'No.';
 DEFINE CMEANST/ANALYSIS MEAN WIDTH=8 FORMAT=8.2 CENTER 'Mean';
 DEFINE CSTDST/ANALYSIS STD WIDTH=6 FORMAT=6.2 CENTER 'Std. Dev.';

 BREAK AFTER RACE/SKIP;

 RBREAK AFTER/ SKIP SUMMARIZE;

 COMPUTE AFTER;
  RACE='ALL';
 ENDCOMP;

 COMPUTE BEFORE;
  LINE @3 ' ';
  LINE @42 57*'=';
  LINE @42 '|The average change in blood' @72 'pressure overall'
       @89 'was -7.6.|';
  LINE @42 57*'=';
  LINE @3 ' ';
 ENDCOMP;

 TITLE1 'APPENDIX TABLE 2';
 TITLE2 ‘OUTPUT FROM PROC REPORT’;
 RUN;
```

An additional procedure in Base SAS that can be used to generate reports is PROC SQL. This procedure is not covered in this paper due to time limitations, but can perform many of the DATA step and PROC step functions and should also be considered when report writing.

CONCLUSION

This paper has attempted to provide some guidelines to follow when developing a SAS program to generate a report, including how to get the data into a SAS data set, how to manipulate the data, and which procedure might be the most appropriate. Some of the approaches in this paper initially utilized SAS programming techniques used under Version 6.12. However, the examples covered in this paper are all valid for SAS Versions 8 and 9, and have not taken advantage of a number of additional capabilities, including long variable names and ODS (Output Delivery System) features. Hopefully, this paper will help guide you in how to generate the report you desire in a logical and organized manner.

REFERENCES

SAS Institute Inc. (1995) *SAS Guide to the REPORT Procedure: Reference, Release 6.11,* Cary, NC: SAS Institute Inc.

SAS Institute Inc. (1990) *SAS Guide to the Report Procedure: Usage and Reference, Version 6*, Cary, NC: SAS Institute Inc.

SAS Institute Inc. (1990) *SAS Guide to TABULATE Processing, Second Edition*, Cary, NC: SAS Institute Inc.

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

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

APPENDIX TABLE 1
OUTPUT FROM PROC TABULATE

	Gender																	
	Female								Male									
	Baseline Diastolic BP			Final Diastolic BP			Change in Diastolic BP			Baseline Diastolic BP			Final Diastolic BP			Change in Diastolic BP		
	No.	Mean	Std Dev.	No.	Mean	Std Dev.	No.	Mean	Std Dev.	No.	Mean	Std Dev.	No.	Mean	Std Dev.	No.	Mean	Std dev.
RACE																		
White	44	94.70	5.52	44	88.23	7.14	44	-6.48	5.59	23	97.17	6.58	24	86.88	9.47	23	-10.57	6.34
Black	7	100.00	1.91	7	93.00	2.94	7	-7.00	4.32	4	98.50	5.45	4	95.75	4.65	4	-2.75	4.57
Other	1	104.00	0	1	100.00	0	1	-4.00	0	1	97.00	0	1	80.00	0	1	-17.00	0
All	52	95.60	5.56	52	89.10	7.00	52	-6.50	5.36	28	97.36	6.23	29	87.86	9.38	28	-9.68	6.69

APPENDIX TABLE 2
OUTPUT FROM PROC REPORT

	Gender																	
	Female									Male								
	Baseline Diastolic BP			Final Diastolic BP			Change in Diastolic BP			Baseline Diastolic BP			Final Diastolic BP			Change in Diastolic BP		
Race	No.	Mean	Std. Dev.	No.	Mean	Std. Dev.	No.	Mean	Std. Dev.	No.	Mean	Std. Dev.	No.	Mean	Std. Dev.	No.	Mean	Std. Dev.
|The average change in blood pressure overall was -7.6.|																		
White	44	94.70	5.52	44	88.23	7.14	44	-6.48	5.59	23	97.17	6.58	24	86.88	9.47	23	-10.57	6.34
Black	7	100.00	1.91	7	93.00	2.94	7	-7.00	4.32	4	98.50	5.45	4	95.75	4.65	4	-2.75	4.57
Other	1	104.00	0	1	100.00	0	1	-4.00	0	1	97.00	0	1	80.00	0	1	-17.00	0
ALL	52	95.60	5.56	52	89.10	7.00	52	-6.50	5.36	28	97.36	6.23	29	87.86	9.38	28	-9.68	6.69