Using PROC REPORT to Summarize Clinical Safety Data

Justina M. Flavin

ABSTRACT

By using the summary and display features available in PROC REPORT, the amount of code needed to generate output can be reduced and often, a more informative table can be produced. This paper will illustrate the use of the REPORT procedure in creating both detail and summary reports of clinical safety data such as adverse events, laboratory results, and concomitant medications.

INTRODUCTION

In preparing Clinical Study Reports of drug development data, several types of data listings are required. Typically, summary tables and short listings of data are needed for inclusion in the text portion of the report, while complete tabulations of all data collected for each patient make up the report appendices. In producing the summary tables, using the statistical options available in PROC REPORT can often reduce or eliminate the need to generate summary statistics from other procedures such as PROC FREQ, PROC MEANS, PROC SUMMARY, or PROC UNIVARIATE that subsequently are printed out using PROC PRINT or DATA _NULL_. For the data tabulations, using PROC REPORT instead of PROC PRINT usually enables more information to be printed in each row by allowing more control over the format of the output such as the ability to specify the number of blank spaces between columns and to wrap long variables. The following are examples of the PROC REPORT code used to accomplish these tasks. Sample output for each example is included at the end of the paper.

EXAMPLE 1: TABULATION OF ADVERSE EVENTS

In this example, a complete listing of adverse event information by patient is desired.

```
PROC REPORT DATA=AE NOWINDOWS SPLIT='$'
            SPACING=1 HEADSKIP HEADLINE MISSING;
  COLUMN PAT AGE TREAT EVENT P_TERM ONSET_DT
         END_DT SEVERE ACTION RELATION;
  DEFINE PAT / ORDER WIDTH=7 'PATIENT' CENTER;
  DEFINE AGE / ORDER WIDTH=3 'AGE' CENTER;
  DEFINE TREAT / ORDER WIDTH=9 'TREATMENT';
  DEFINE EVENT / DISPLAY WIDTH=28 FLOW
                 'INVESTIGATOR TERM';
  DEFINE P_TERM / DISPLAY WIDTH=17 FLOW
                  'PREFERRED TERM';
  DEFINE ONSET_DT / DISPLAY WIDTH=7
                    'ONSET$DATE' CENTER;
  DEFINE END_DT / DISPLAY WIDTH=10
                 'RESOLUTION$DATE' CENTER;
  DEFINE SEVERE / DISPLAY WIDTH=8 'SEVERITY'
                  FORMAT=1. CENTER;
  DEFINE ACTION / DISPLAY WIDTH=7 FLOW
                  'ACTIONS$TAKEN';
  DEFINE RELATION / DISPLAY WIDTH=12 CENTER
            'RELATIONSHIP$TO STUDY$MEDICATION';
  BREAK AFTER PAT / SKIP;
RUN;
```

The PROC REPORT statement invokes the procedure and specifies options to be used in generating the output. The options used are defined as follows:

NOWINDOWS - In an interactive SAS session, this option causes the output to be sent to the OUTPUT window rather than allowing the SAS system to open up the REPORT window.
SPLIT - As with the PRINT procedure, this option defines the character used to determine break points in a long column label.
SPACING - The number of blank spaces between columns.
HEADSKIP - Inserts a blank line between the column labels at the top of each page and the first line of data.
HEADLINE - Adds a dotted line underneath the column labels.
MISSING - Forces the procedure to include records having missing values for group, order, or across variables.

The COLUMN statement lists the variables to be included in the report in the order in which they are to appear.

The DEFINE statement is used to specify the characteristics of the variables included in the report. Each variable is given a DEFINE statement line. After the backslash, options can be specified. A column label may be specified for the variable by enclosing it in double or single quotes and placing it behind the backslash. For a detail report such as this, the variables simply need to be listed; no summary statistics are required, so all variables are defined as DISPLAY or ORDER variables. A DISPLAY variable is printed as it appears in the data set. By default, all character variables are DISPLAY variables. An ORDER variable orders the rows to be printed out according to the ascending formatted values of that variable. The output generated using ORDER variables is similar to that produced when using a BY statement followed by an ID statement in PROC PRINT. The other options used on the DEFINE statement are as follows:
WIDTH - Defines the width of the column.
FLOW - This option allows a character string to wrap to the next line. If the character string contains the split character defined in the PROC REPORT statement, the procedure will break the string at that character as well as at a blank. If a long character string contains neither the split character nor a blank space and its length is longer than the defined width, the procedure will fit as much of the character string on the line as possible and continue it onto the next line.
FORMAT - Defines the format to be used. If none is specified, the format associated with the data in the data set (if any) is used.
CENTER - Centers the column headers and the data within the column.

The BREAK statement creates a summary when the value of the break variable changes. The break location is specified as BEFORE or AFTER the break variable. The SKIP option inserts a blank line between each unique value of the break variable, PAT.

EXAMPLE 2: SUMMARY OF ADVERSE EVENTS

For this example, a summary table of the number of patients reporting adverse events in two treatment groups in three studies is required. The data set consists of one record per patient. The variable EVENT is a dummy variable having a value of '1' if the patient had an adverse event recorded; the value is missing if the patient had no adverse events.

```
PROC REPORT DATA=AE HEADSKIP HEADLINE
            SPLIT='*';
  COLUMN PROTOCOL TREATMNT,(N EVENT PCT) DIFF;
  DEFINE PROTOCOL / GROUP WIDTH=10;
  DEFINE TREATMNT / ACROSS WIDTH=15 '';
```

```
  DEFINE EVENT / ANALYSIS '# PTS*WITH AE';
  DEFINE PCT / COMPUTED FORMAT=8.1
               '% PTS*WITH AE';
  DEFINE DIFF / COMPUTED FORMAT=12.1
                '%*REDUCTION';
  RBREAK AFTER / SUMMARIZE;
  COMPUTE PCT;
    _C4_=(_C3_/_C2_)*100;
    _C7_=(_C6_/_C5_)*100;
  ENDCOMP;
  COMPUTE DIFF;
    DIFF=(1-(_C7_/_C4_))*100;
  ENDCOMP;
  COMPUTE AFTER;
    PROTOCOL='TOTAL';
  ENDCOMP;
RUN;
```

The new variable types used in the DEFINE statements are as follows:
GROUP - Combines all records with the same value of the group variable into one row.
ACROSS - Defines a variable whose values will be used as column headers spanning one or more columns.
ANALYSIS - Denotes a variable for which statistics will be computed. If no statistic is defined, the default statistic is SUM.
COMPUTED - Defines a variable that will be created in a COMPUTE block within the REPORT procedure.

The RBREAK BEFORE/RBREAK AFTER statement is used to create a summary line at the beginning or end of the report. The SUMMARIZE option causes a summary statistic to be computed for all analysis and computed variables.

For this report, the variable TREATMNT is defined as an ACROSS variable that will be printed out with no header. As listed in the COLUMN statement, each value of TREATMNT will span the three columns listed in the parentheses after TREATMNT. When the first and last characters of a header are any of the following: - = _ . * + then the PROC REPORT procedure uses that character to expand the header to fill the space over the columns. Thus to achieve the effect of the dotted lines spanning the width of the three columns, the values of TREATMNT were assigned as '-PLACEBO-' and '-ACTIVE-'. The first column spanned by TREATMNT, N, is simply the number of nonmissing rows having the value of PROTOCOL for that row. EVENT does not have a statistic defined, so by default, it becomes a SUM variable whose value is the sum of the (nonmissing) values for EVENT for each unique value of PROTOCOL. The COMPUTED variables, PCT and DIFF, are calculated by referencing variables by column number. Finally, a value for PROTOCOL in the summary line is assigned by including a COMPUTE AFTER block; if this were not there, the value of PROTOCOL would be blank in the final row.

EXAMPLE 3: TABULATION OF LABORATORY DATA

In this example, the desired output is that of each patient's laboratory results grouped by test name and listed chronologically. The data set consists of one record per patient per laboratory test per time point.

```
PROC REPORT DATA=LAB MISSING NOWINDOWS
                     HEADLINE;
  COLUMN PATNUM TREATMNT TESTTIME TIMEPT
         LABCODE,(DTVIS TMVIS LABVAL);
  DEFINE PATNUM / GROUP 'PATIENT';
  DEFINE TREATMNT / GROUP 'TREATMENT';
  DEFINE TESTTIME / GROUP NOPRINT;
  DEFINE TIMEPT / GROUP 'TIME POINT';
  DEFINE LABCODE / ACROSS '-HEMATOLOGY-';
  DEFINE DTVIS / DISPLAY 'DATE';
  DEFINE TMVIS / DISPLAY 'TIME';
  DEFINE LABVAL / DISPLAY 'RESULT';
  BREAK AFTER PATNUM / SKIP;
RUN;
```

LABCODE is an ACROSS variable with the '-HEMATOLOGY-' header spanning the (multiple) date, time, and result columns. Each unique value of LABCODE spans its own date, time, and result columns. The variable TESTTIME is defined as a nonprinting GROUP variable; it is a numeric variable coded as follows: 1=SCREENING, 2=PRE-INFUSION, 3=END INFUSION, 4=FOLLOW-UP. If TESTTIME were not included in the report definition, the values for TIMEPT would print out in alphabetical order rather than chronological order.

EXAMPLE 4: SUMMARY OF LABORATORY DATA

In the last example, the desired output is a summary table of mean values for each laboratory test at each time point within treatment groups. The data set consists of one record per patient per laboratory test per time point.

```
PROC REPORT DATA=LAB SPLIT='*' MISSING
            NOWINDOWS;
   COLUMN LABCODE LABUNIT TIMEPT
   TREATMNT,('__' LABVAL,N LABVAL,MEAN
                  LABVAL,STD);
  DEFINE LABCODE / GROUP NOPRINT;
  DEFINE LABUNIT / GROUP NOPRINT;
  DEFINE TESTTIME / GROUP NOPRINT;
  DEFINE TIMEPT / GROUP '';
  DEFINE TREATMNT / ACROSS WIDTH=20 '';
  DEFINE LABVAL / ANALYSIS '';
  DEFINE N / FORMAT=3. SPACING=2;
  DEFINE MEAN / FORMAT=8.2 'MEAN';
  DEFINE STD / FORMAT=8.2 'S.D.';
  COMPUTE BEFORE LABUNIT;
    LINE ' ';
    CODEUNIT=COMPBL(PUT(LABCODE,LABCODS.)||
             ' ('||TRIM(LEFT(LABUNIT))||')');
    LINE PUT @15 CODEUNIT $50.;
  ENDCOMP;
RUN;
```

In this example, TREATMNT is an ACROSS variable spanning the columns containing summary statistics for the LABVAL variable. Because '__' is defined as the label within the parentheses, a line will be drawn under each value of TREATMNT; the line will be the length of the three columns of summary statistics. LABVAL is an ANALYSIS variable for which only summary statistics will appear. The LABCODE and LABUNIT variables are defined as nonprinting GROUP variables. The COMPUTE block executes every time a new value of LABUNIT is encountered. The first LINE statement in the COMPUTE block adds a blank line in the output. The second statement creates a new variable, CODEUNIT, which is a concatenation of the LABCODE and LABNUNIT variables. Finally, the last LINE statement in the COMPUTE block is used to write CODEUNIT at column 15 in the output file. The SPACING option on the DEFINE N statement defines the number of blank characters between the column and the column to its left. Use of this option overrides the SPACING option (if any) defined on the PROC REPORT line.

CONCLUSION

As illustrated by these examples, PROC REPORT is a versatile procedure which allows for a concise, compact program. Learning the syntax can at first be frustrating to a novice user, and there is often more than one way of correctly writing the code, especially when creating or referencing summary statistics columns. However, once mastered, PROC REPORT becomes a powerful tool for producing a variety of different types of output. This paper is in no way an attempt to illustrate every feature available in the procedure; rather, it is meant to show several different ways that PROC REPORT has been utilized in clinical data processing.

CONTACT INFORMATION

Justina M. Flavin has over fifteen years experience developing clinical software in the pharmaceutical industry and is an instructor for SAS programming classes at the University of California, San Diego, Extension Division and California State University, San Marcos. She serves on the Executive Committee of SANDS, the San Diego SAS Users Group and was conference chair of Pharma*SUG* '99. Justina has a B.A. in Applied Mathematics from the University of California, San Diego. She may be contacted at justina.flavin@gmail.com.

OUTPUT FOR EXAMPLE 1: TABULATION OF ADVERSE EVENTS

PATIENT	AGE	TREATMENT	INVESTIGATOR TERM	PREFERRED TERM	ONSET DATE	RESOLUTION DATE	SEVERITY	ACTIONS TAKEN	RELATIONSHIP TO STUDY MEDICATION
021-44	58	PLACEBO	ITCHING	PRURITUS	25AUG94	25AUG94	1	1	3
			DEATH	DEATH	24NOV94		3	6	1
024-59	62	ACTIVE	SUBARACHNOID HEMORRHAGE	SUBARACHNOID HEMORRHAGE	19SEP94		3	3 5	1
			BRAINSTEM INFARCT RESULTING IN DEATH	CEREBROVASCULAR ACCIDENT	20SEP94		3	3 5	1
104-52	76	PLACEBO	LEFT LEG AND FOOT TINGLED & FELT NUMB	PARESTHESIA	12JUN93	12JUN93	1	1	4
			FAINTING EPISODE	SYNCOPE	22NOV93	22NOV93	2	5	1
			UPPER RESPIRATORY TRACT INFECTION	PHARYNGITIS	16DEC93	16DEC93	4	4 5 6	1

OUTPUT FOR EXAMPLE 2: SUMMARY OF ADVERSE EVENTS

	PLACEBO			ACTIVE			
PROTOCOL	N	# PTS WITH AE	% PTS WITH AE	N	# PTS WITH AE	% PTS WITH AE	% REDUCTION
110520	1176	1125	95.7	1181	1126	95.3	0.4
182000	1292	1201	93.0	1287	1187	92.2	0.9
275100	1001	957	95.6	1010	938	92.9	2.8
TOTAL	3469	3283	94.6	3478	3251	93.5	1.2

OUTPUT FOR EXAMPLE 3: TABULATION OF LABORATORY DATA

			HEMATOLOGY								
			HEMATOCRIT			HEMOGLOBIN			NEUTROPHILS		
PATIENT	TREATMENT	TIME POINT	DATE	TIME	RESULT	DATE	TIME	RESULT	DATE	TIME	RESULT
J-076	ACTIVE	SCREENING	24NOV93	15:02:00	43.7	24NOV93	15:02:00	15.1	24NOV93	15:02:00	5.62
		PRE-INFUSION	06DEC93	18:11:00	43	06DEC93	18:11:00	15.1	06DEC93	18:11:00	6.8
		END INFUSION	07DEC93	16:00:00	41	07DEC93	16:00:00	13.7	07DEC93	16:00:00	3.43
		FOLLOW-UP	13DEC93	8:20:00	42.2	13DEC93	8:20:00	14.5	13DEC93	8:20:00	3.25
J-078	PLACEBO	SCREENING	24NOV93	16:17:00	37.8	24NOV93	16:17:00	13.4	24NOV93	16:17:00	3.9
		PRE-INFUSION	06DEC93	18:19:00	36.9	06DEC93	18:19:00	12.7	06DEC93	18:19:00	3.36
		END INFUSION	07DEC93	16:10:00	33.9	07DEC93	16:10:00	11.7	07DEC93	16:10:00	2.3
		FOLLOW-UP	13DEC93	8:25:00	38	13DEC93	8:25:00	12.6	13DEC93	8:25:00	3.63

OUTPUT FOR EXAMPLE 4: SUMMARY OF LABORATORY DATA

	ACTIVE			PLACEBO		
	N	MEAN	S.D.	N	MEAN	S.D.
ALK PHOS (U/L)						
SCREENING	625	104.02	77.51	648	98.11	42.79
PRE-INFUSION	566	62.40	41.50	582	58.76	31.08
POST-INFUSION	525	131.77	93.49	550	124.77	81.76
ALK PHOS (UMOL/L)						
SCREENING	63	3.11	1.04	66	2.83	0.99
PRE-INFUSION	56	2.40	0.77	58	2.18	0.55
POST-INFUSION	54	4.75	1.26	55	5.73	2.32
CREATININE (UMOL/L)						
SCREENING	652	96.05	21.57	678	98.25	28.24
PRE-INFUSION	636	96.22	30.98	663	96.35	31.17
POST-INFUSION	592	98.54	28.17	613	102.43	40.14