Counting on Base SAS® Software
Beth Wilson, The United Network for Organ Sharing, Richmond, VA

Abstract

Counting observations can be accomplished in a number of ways, and the method used to create the output is dependent upon the purpose of the report. SAS code is included to demonstrate how the PRINT, FREQ, SUMMARY, MEANS, REPORT and TABULATE procedures can be used in counting. Sample output using each of the procedures is also illustrated.

Some of the reasons for choosing one method versus another are also highlighted. Reasons include visual appearance and uses of the output, flexibility of the procedure and the length of processing time to arrive at the desired report.

Organ donation and transplantation data from the United Network for Organ Sharing is utilized to exemplify the procedures with regard to output. The intended audience is beginning/intermediate SAS users.

Introduction

SAS offers a host of procedures to produce output for counting observations. In many cases, manipulation of the data is necessary to produce the desired report in an understandable format. The report purpose is the driving force to the layout, and the layout determines which SAS procedure(s) should be invoked to produce data summaries that are clear and concise. Additionally, time constraints may play a part in the decision regarding a particular SAS procedure with respect to creating the program, and the processing time to produce the final report.

PROC FREQ

The FREQ procedure is a quick and easy method to produce simple counts and percentages, in addition to other statistics. See Figure 1 for an example of code and output of a single variable frequency. With this procedure you can also create an output data set. This output data set can be used in conjunction with other SAS procedures such as the PRINT procedure (figures 2 and 3) and the REPORT procedure to further manipulate the layout of the report, should the need arise.

In Figure 1, the variable is ORGTYP, meaning Organ Type; the value is PA, meaning Pancreas. The frequency is the number of pancreas transplants performed in the U.S. during an unspecified period of time.

(Figure 1)

```
*OUTPUT---------------------------
Command ===>
O0001 proc freq;
O0002 tables orgtyp/missing input;
O0003 run;

NOTE: The options MISSING, NOFMT were used to minimize the amount of information placed in the output window. (For more information regarding these options, please refer to your SAS manuals.)
```

Utilizing PROC PRINT

The PRINT procedure is another simple process which can be utilized to produce reports of counts in conjunction with the output data set created in the options statement of PROC FREQ. Again, the usefulness of producing counts in this manner is dependent upon the purpose of your output, and may not be appropriate for all applications. For example, you may want to use PROC PRINT when performing multi-variable cross-tabulations. The number of tables produced by PROC FREQ may be inordinately large and thus not useful as a summary report.

Invoking PROC FREQ with the NOPRINT and the OUT= options produces an output data set. Then PROC PRINT can be invoked, using the DATA= option with a SUMBY statement, to summarize the counts created with PROC FREQ.

In Figure 2, ORGTYP is crossed with YEAR. The count represents the number of pancreas transplants performed in the U.S. per year. The OUT= option produced the data set 'D'. (Following the code and output is a partial output from invoking the contents procedure on data set 'D'. Notice the variables COUNT and PERCENT. These are variables created by PROC FREQ, which have now become part of the data set 'D'.)

(Figure 2)

```
*OUTPUT---------------------------
Command ===>
O0001 proc freq data=keeporgtyp year tx_data;  
O0002 tables orgtyp/missing nofmt arrow input;  
O0003 out=d;  
O0004 run;  
O0005 run;
O0006

%%%Alphabetic List of Variables and Attributes%%%  
#    Variable  Type    Len    Pos    Label
---   ---------  ------  ------  -------  
1  COUNT    Num     8     0     Organ Type  
2  ORGTYP   Char     4     4     Frequency Count  
3  PERCENT  Num     8     0     Percent of Total Freq  
4  YEAR     Num     8     4
```

In Figure 3 the PROC PRINT output demonstrates how to produce a report with the COUNT variable obtained from PROC FREQ.
If this were a three variable cross tabulation, PROC FREQ would produce multiple pages of tables, whereas by using PROC PRINT the output would appear very similar to Figure 3, but with an additional column. Many end-users of data prefer to have the information on one page rather than scanning multiple pages.

(Figure 3)

```
<OUTPUT>-------------------------------------
| Command ===
| Number of Pancreas Transplants by Year
| Frequency
| YEAR  Count
| 1992  587
| 1993  774
| 1994  842
| === 2173
```

PROC MEANS

The MEANS procedure can also be used to count observations and produce statistics as well as to produce an output data set. PROC MEANS can be used to count values that are contained within a data set observation. The previous example utilized PROC FREQ to count each observation. In the following example each record/observation in the data set is an organ donor, which may represent multiple organs recovered. When counting the number of kidneys, livers, pancreases, hearts and lungs recovered, one record could represent up to seven observations (2 kidneys, 1 each of livers, hearts, and pancreases, and 2 lungs).

In the code shown in Figure 4, a variable was created for each organ type which can be recovered from one donor. [To help you understand the code in figure 4, read the literal translation of the statements shown in italics above the program editor box.]

By invoking PROC MEANS, the output data set, COUNTS, contains the total number of observations, and the total number of each organ type recovered.

(Figure 4)

Look at line 00004, the literal translation is:

num_ki is equal to 1 if the left kidney recovery disposition (KILDISP) is between the values of 500 and 507 plus 1 if the right kidney recovery disposition (KIRDISP) is between 500 and 507. If the kidney disposition (KILDISP or KIRDISP) value does not fall into those ranges, then the value is 0. Meaning a donor can donate either 0, 1, or 2 kidneys.

(The meaning of these dispositions is pertinent to this discussion, but relates to whether the organs were recovered and transplanted or used for other purposes. If this example seems to be obscure, pretend we are looking at products instead of organs and pretend the dispositions are reasons the merchandise was returned to the store, instead.)

Look at lines 00005 and 00006, translation:

num_li is equal to 1 if the whole liver disposition (LIWDISP) value is between 500 and 507 or either the liver segment disposition (LISDISP) value is between 500 and 507 (meaning a donor can donate a maximum of 1 liver).

Look at lines 00014 and 00015, translation: sumnum_ki num_li num_pa num_hr num_lu is the key line of code which adds the number of organs recovered by each donor and summarizes the data for all observations.

```
<PROGRAM EDITOR>-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-
| Command ===
| 00001 data b;
| 00002 set a;
| 00004 num_ki=('500'<=kildisp<='507')+(500'<=kirdisp<='507') or
| 00005 num_li=('500'<=lirdisp<='507') or
| 00006 ('500'<=lirdisp<='507') or
| 00007 num_pa=('500'<=padisp<='507') or
| 00008 ('500'<=padisp<='507') and (num_hr<='506') or
| 00010 num_lu=('500'<=alldisp<='507')+(500'<=lirdisp<='507');
| 00012 proc means noprint;
| 00013 var num_ki num_li num_pa num_hr num_lu;
| 00014 output data=counts sumnum_ki num_li num_pa
| 00015 num_hr num_lu;--ZOOM---I--
```

Figure 5 demonstrates how output can be displayed by using PROC PRINT for the data set created from PROC MEANS.

(Figure 5)

```
<OUTPUT>-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-
| Command ===
| Number of Organ recovery in 1993
| Number of Number of Number of Number of
| Number Organs Kidneys Livers Pancreases Hearts Lungs
| Recovered Recovered Recovered Recovered
| 9,165 3,765 1,244 2,462 1,462
```

PROC SUMMARY

The SUMMARY procedure is extremely similar to PROC MEANS, however the default is to not produce printed output, in contrast to the default for PROC MEANS.

For these examples, I prefer PROC SUMMARY to PROC MEANS, only because of the names of the procedures, not due to their functionality. In these cases both procedures in code and output are identical, with the exception of the NOPRINT option specified in PROC MEANS. Since these examples are producing sums/summaries, why not just use PROC SUMMARY?

In Figure 6, the same example is used as in Figure 4, to demonstrate the similarities in code between PROC SUMMARY and PROC MEANS. When trying to produce the output, as in Figure 5, the identical code illustrated in Figure 5 would be used in conjunction with PROC SUMMARY, hence the output will also be identical.

This is not to say that you must utilize PROC PRINT in conjunction with PROC MEANS or PROC SUMMARY. The MEANS procedure will produce statistics by default, and you can
specify a PRINT option with the SUMMARY procedure as well. This example demonstrates that you can utilize PROC PRINT with these procedures.

(Figure 6)

```plaintext
01001 proc tabulate data=beth.donr;
01002 run;
```

PROC TABULATE

The TABULATE procedure is another very powerful SAS tool, as it has the capability to create tables with hierarchical variables, making output more concise. This is a major advantage to the TABULATE procedure in comparison to some of the other summary procedures.

PROC TABULATE is particularly beneficial in analyzing multiple variables across time periods. The following example demonstrates counting the number of donors per year, by donor age and gender. This method allows data to be analyzed in one concise table giving the user flexibility of examining trends over time with respect to age and/or gender. It also allows the table and data to be hands-free, rather than having to include many footnotes and other text to incorporate multiple tables, displaying the same information. See figure 7 for sample code and output.

PROC REPORT

The REPORT procedure combines some of the reporting capabilities of PROC PRINT, PROC MEANS, and the TABULATE procedure to generate reports using one procedure. It has a flexible report structure; column widths and column labels can be specified, hierarchical variables, subtotals, and totals can also be created. However, it does not allow for the creation of an output data set. One could argue that since a final report is being created, an output data set is not necessary, but the creation of an output data set of subtotals and totals may also be desirable. Again, the purpose of the report dictates, in many cases, which procedure to use.

Figure 8 demonstrates SAS code and output for the same case as in figure 7 (PROC TABULATE). Notice the appearance of the output in the two figures, as well as the differences in the code.

(Figure 7)

```plaintext
01001 proc tabulate data=beth.donr;
01002 run;
```

(Figure 7, cont’d)

```plaintext
01011 proc summary;
01012 var num_hi num_il num_pa num_hr num_lu;
01014 output out=counts sum(num_hi num_il num_pa num_hr num_lu);
01015 run;
```

(Figure 8)

```plaintext
01011 proc report split="" headskip;
01012 column age;
01013 age=dup('Year '; format=width15 center);@@
01014 age=dup('Gender'; format=width15 center);
01015 age=dup('Gender'; format=width15 center);
01031 run;
```

(Figure 8)
Conclusion

As demonstrated, observations can be counted using a variety of SAS procedures to display the same data in a multitude of ways. The decision to invoke PROC FREQ vs. PROC TABULATE or PROC MEANS vs. PROC SUMMARY, is your choice. The code and the appearance of the output may change, and the interpretations and usefulness of this output can vary widely, depending upon the method you choose. These six procedures can have a powerful impact when appropriately used.

References


Acknowledgements

I would like to acknowledge the following people for their suggestions and support:

Leah Bennett, Ph.D.
Jaime Hampton, B.S.
M.Y. Young Kwon, B.S.
Ray White, M.S.
Jimmy Wilson