

Report Writing, SAS/GRAPH® Creation, and Output Verification using SAS/ASSIST®

Matthew J. Becker, STATPROBE, inc., Ann Arbor, MI

Abstract

Since the release of SAS/ASSIST, SAS® has given users more flexibility in the routine generation of SAS output, including graphs. Novice users can now "point-and-click" their way to table and graph generation. In addition, the code generated by the user's selections is constructed and provided in the SAS Log. If desired, this code can be copied into a program for future submissions. Generating SAS/ASSIST output is also beneficial in verifying table output generated from a SAS program. A user may use a combination of SAS data summarization procedures and a Data _null_ to generate a listing of the summarized data. They can then use SAS/ASSIST to generate a Proc Tabulate of the data and verify that their output is correct.

Introduction

SAS/ASSIST has given SAS users a robust tool to assist them in the daily requirements of their professional positions. Be it an applications programmer for a leading automobile manufacturing company or a single-person data management business, SAS/ASSIST can make the users SAS-life much easier. This paper will touch on a few of the basics of SAS/ASSIST as they pertain to the pharmaceutical industry and STATPROBE, Inc.

STATPROBE, Inc. is a contract research organization whose primary clients are pharmaceutical and medical device companies. SAS is the programming language used to generate the listing and summary tables to support research reports. In addition, SAS is used for graph creation, data management, and statistics. SAS/ASSIST is used to produce output. This output can be used for verification of another program's output or for the report itself. At STATPROBE, Inc. SAS/ASSIST has been used primarily for output verification and validation. Verification and validation are an extremely important subject when dealing with investigational drug reports. These reports are a part of the foundation for a drug to be available to the

general public. Therefore, the data presented must be correct.

Listing Report Writing

STATPROBE, Inc. can provide multiple report formats depending on a client's request. Some clients request Proc Prints of the data in order for them to validate and verify the data entry that may have been done at STATPROBE, Inc. For the moderate to advanced SAS user, this type of report writing is pretty mundane. For a new user, or a novice SAS user, Proc Print and all of its accompanying options may be a source of brain overflow. SAS/ASSIST provides a interactive interface to produce Proc Print output.

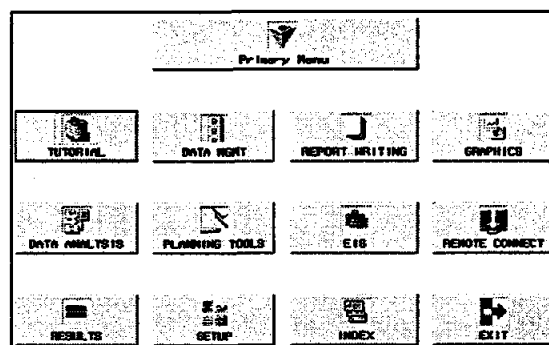


Figure 1. Primary Menu

Figure 1 displays the primary menu of SAS/ASSIST. There are 12 selections a user can make. The selections are the buttons displayed. To produce a Proc Print, the user selects the 'REPORT WRITING' button. Once selected, SAS/ASSIST displays the menu shown in Figure 2.

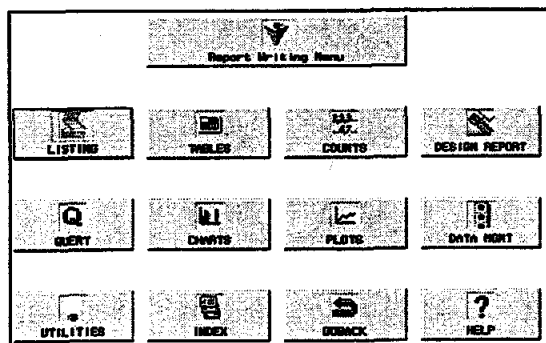


Figure 2. Report Writing Menu

At this point, the user chooses the type of report they would like to produce. For this example, we will concentrate on the 'LISTING' button that produces a Proc Print. Once the user selects the 'LISTING' button, a menu identical to Figure 3 appears.

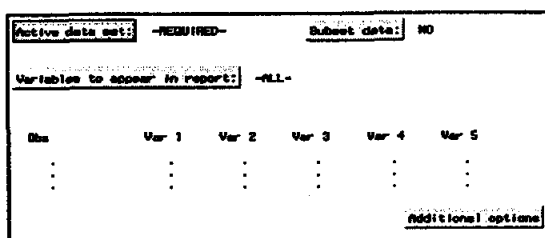


Figure 3. Proc Print Selection Screen

The user interactively selects the data set to print, data subset criteria, variables to appear in the report, and additional options. SAS/ASSIST interactively displays a list box of the data sets available if the user clicks on the 'ACTIVE DATA SET' button. Similarly, list boxes and selection boxes appear if the user selects the other buttons on the selection screen. Figure 4 shows the same screen once selections have been made by the user to define the data set, variables, and data subset conditions for the Proc Print desired.

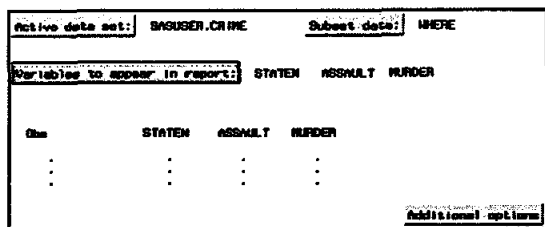


Figure 4. Proc Print Selection Screen with Selections.

The user has selected the data set SASUSER.CRIME, issued a where clause to subset the data, and defined the variables STATEN, ASSAULT, and MURDER to appear in the Proc Print. To generate the report, the

user clicks their right mouse button and selects *Locals->Run* from the pull down menu as seen in Figure 5.

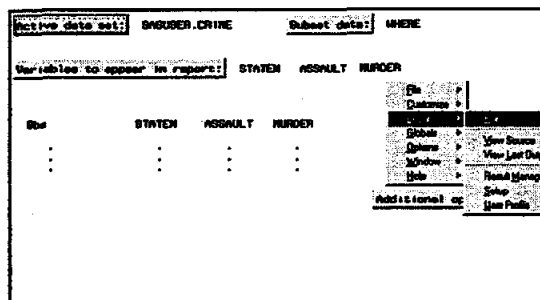


Figure 5. Interactive Submission to Generate Listing.

Once the run has been performed, the output seen in Table 1 is generated.

State name	ASSAULT	MURDER
Alabama	278.3	14.2
Alaska	284.0	10.8
Arizona	312.3	9.5
Arkansas	203.4	8.8
California	358.0	11.5
Colorado	292.9	6.3
Connecticut	131.8	4.2
Delaware	194.2	6.0
Florida	449.1	10.2
Georgia	256.5	11.7
Hawaii	64.1	7.2
Idaho	172.5	5.5
Illinois	209.0	9.9
Indiana	153.5	7.4
Iowa	89.8	2.3
Kansas	180.5	6.6
Kentucky	123.3	10.1
Louisiana	335.5	15.5
Maine	170.0	2.4
Maryland	358.9	8.0
Massachusetts	231.6	3.1
Michigan	274.6	9.3
Minnesota	85.8	2.7
Mississippi	189.1	14.3
Missouri	233.5	9.6
Montana	156.8	5.4
Nebraska	112.7	3.9
Nevada	355.0	15.8
New Hampshire	76.0	3.2
New Jersey	185.1	5.6
New Mexico	343.4	8.8
New York	319.1	10.7
North Carolina	318.3	10.6

Table 1. SAS/ASSIST Listing Output

The 'ADDITIONAL OPTIONS' button seen in Figures 3, 4, and 5 allows users to specify variable/labeling options, data subset options, and general options. By selecting *Locals->View Source* the user gets a display of the SAS code that generated the output based on their specifications (Figures 6 and 7).

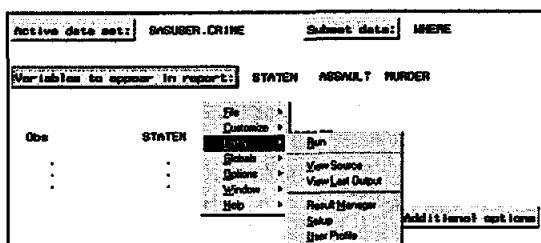


Figure 6. View Source

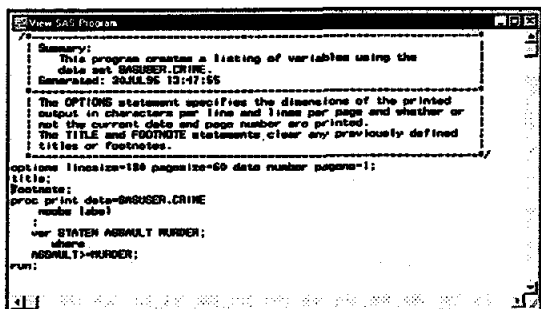


Figure 7. Source Code

The user can cut and paste the source code to the SAS program editor if they wish to do so.

Summary Report Writing

Interactively generating a Proc Tabulate summary table is another report used at STATPROBE, Inc. To define the tabulate table, the user clicks on the 'TABLES' button from the Report Writing Menu (Figure 2). Figure 8 displays the menu that appears once this action is taken.

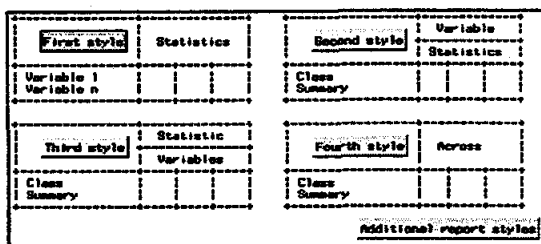


Figure 8. Table Menu

At this point, the user can select from the different summarization table styles. The 'ADDITIONAL REPORT STYLES' button displays additional summarization table styles.

Age in years	Height in hp	Heart rate while running	Age in years	Heart rate while running	Experimental group
27	72.87	54	12.50	174	2
27	72.86	52	11.17	152	2
32	76.32	48	9.83	184	2
39	79.37	48	9.32	144	2
31	87.25	48	10.88	172	1
34	81.23	44	12.88	168	2
31	73.71	53	10.47	186	2
32	59.88	49	9.33	148	2
48	76.22	54	9.49	186	2
48	81.29	52	11.59	179	2
32	82.78	52	10.54	174	2
44	73.82	45	10.13	182	1
45	87.35	56	14.12	182	1
45	66.45	51	11.12	170	1
47	73.35	47	10.09	182	1
54	82.12	59	10.33	185	1
49	86.15	44	9.17	190	1
51	85.69	57	10.35	188	1
51	77.31	48	10.80	182	1
48	81.23	48	10.25	182	1
49	72.37	70	10.88	182	1
44	85.47	52	11.87	179	1
48	75.87	52	10.07	185	1
44	85.34	45	9.85	154	1
42	86.15	44	9.17	182	1
38	83.82	55	9.22	178	1
47	77.45	58	11.83	175	1
48	75.38	74	11.35	175	1
48	81.19	54	10.82	182	1
44	81.42	53	13.08	174	1
38	81.87	48	9.83	179	1

Table 2. Fitness Data Set

If a user wanted to summarize the ages of the people in the fitness data set (Table 2) over their experimental groups they could use the SAS/ASSIST table button. The report style buttons preview the presentation of the report. The 'SECOND STYLE' button has a class variable as its rows, and variables as its columns. Under each of the variables defined by the user, statistics can be selected to be output. The definition of this report is displayed in Figure 9.

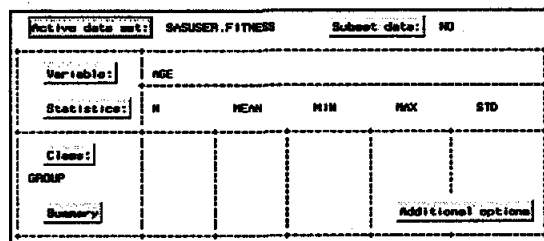


Figure 9. Fitness Report Definition

The user has defined SASUSER.FITNESS as the active data set to use, defined GROUP as the class variable to be displayed in the rows, and the variable AGE to be summarized with N, MEAN, MIN, MAX, and STD as the statistics. The resulting output is displayed in Figure 10.

Experimental group	Age in years				
	N	MEAN	MIN	MAX	STD
1	10.00	42.00	38.00	47.00	2.00
2	10.00	48.00	44.00	54.00	3.00
3	10.00	52.00	48.00	57.00	2.00

Figure 10. Fitness Report Output

As with the listing, SAS/ASSIST provides the source code that, when run, creates the fitness report output. *Locals->View Source* will open a window with the source code so

that the user can cut and paste it into the SAS program editor.

SAS/GRAPH Creation

Writing the code to create graphs in SAS can be frustrating unless the SAS programmer has the knowledge of the options available for the different graphing procedures. For the novice user, SAS/ASSIST provides an interactive graph development tool. To access the interactive graph development application, click on the 'GRAPHICS' button from the primary menu (Figure 1). SAS/ASSIST then prompts the user to select high or low resolution graphs. After selecting the presentation resolution, the screen in Figure 11 appears.

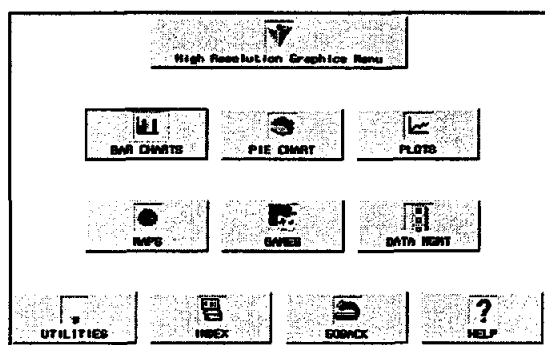


Figure 11. High Resolution Graphic Menu

The graphic menu allows the user to define bar charts, pie charts, plots, and other applications. As with the report writing applications, the user defines the graphing criteria interactively. SAS/ASSIST uses a combination of list boxes and selection lists to give the user flexibility in their definitions.

To produce a bar chart, the user would select the 'BAR CHARTS' button in the graphic menu screen. Once the selection has been made, the user is prompted as to whether they would like to define a horizontal bar chart or a vertical bar chart. Provided the user selects horizontal bar chart, the screen in Figure 12 appears.

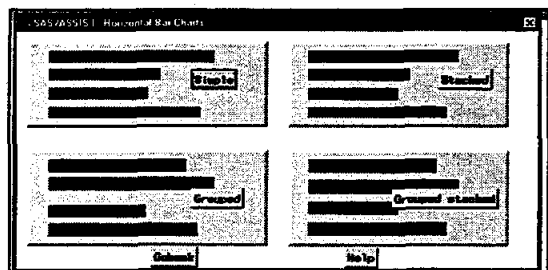


Figure 12. Horizontal Bar Chart Screen

This screen displays the different types of horizontal bar charts that can be presented. The choices are: simple, stacked, grouped, and grouped stacked. The simple graph creates a simple horizontal bar chart that charts a variable along the vertical axis. The stacked graph creates a simple horizontal bar chart that charts one variable along the vertical axis and subdivides the bars to display the value of a second variable as a stacking variable. Grouped graphs are simple horizontal bar charts that chart one variable along the vertical axis and group the bars in the chart according to the value of a second variable as a grouping variable. Group stacked graphs create simple horizontal bar charts that chart one variable along the vertical axis, subdivides the bars to display the values of a second variable as a stacking variable, and further groups the bars in the chart according to the value of a third variable as a grouping variable. With some knowledge of the type of graph the user would like to see, he/she selects the type of graph based on these definitions.

Let's assume the user wants to produce a grouped graph. Once the grouped graph icon is selected, a dialog box appears (Figure 13).

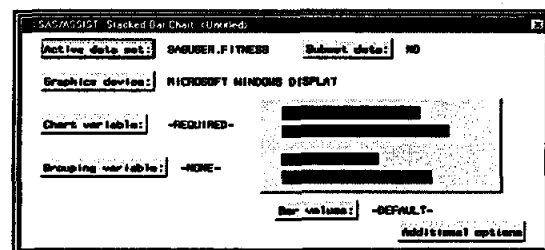


Figure 13. Grouped Bar Dialog Box

At this point, the selection process is similar to that described in the report writing section. The user selects the active data set to use, chart variable (y-axis), grouping variable, bar values (x-axis), data subset criteria, and additional options. The additional options include: general options, color and pattern options, number of bars, axis options, and legend options. Figure 14 shows a grouped bar dialog box with the criteria defined.

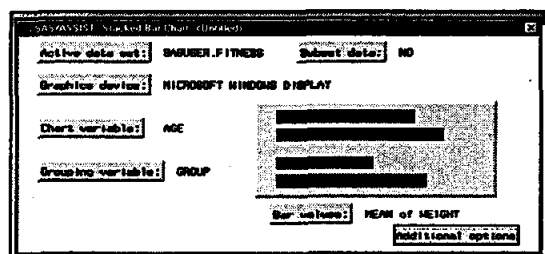


Figure 14. Defined Grouped Bar Dialog Box

For this example, the user is using the fitness data (Table 2). They are grouping by the experimental group, the vertical axis will be the ages, and the horizontal bars will be the mean of the weights. The resulting graph after a *Locals->Run* is shown in Table 3.

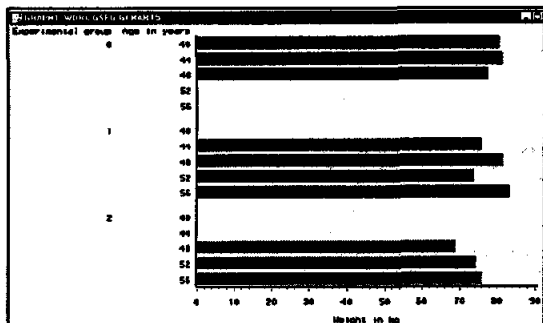


Table 3. Grouped Bar Horizontal Bar Chart

Using *Locals->View Source*, the user can get the code used to create the horizontal bar chart for the fitness data (Figure 15).

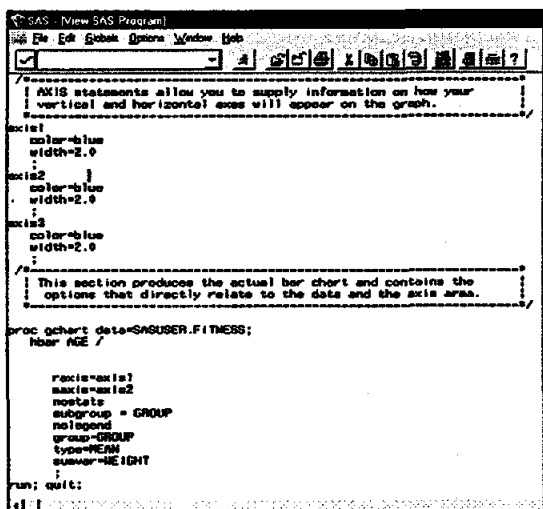


Figure 15. Source Code for Horizontal Bar Graph

Using this feature of SAS/ASSIST, novice users can interactively define the graph criteria and paste the code into their program.

Output Verification

The main use of SAS/ASSIST at STATPROBE, Inc. is for output verification. STATPROBE, Inc. produces listing and summary tables for its clients. It is of the utmost importance that the listings and summaries be correct. Previous methods of ensuring that the programs were correct involved spot-check and tedious hand-counts of the data. While this practice is still used, SAS/ASSIST provides

the tools for the novice user to produce output using the report writing and graphing to help facilitate the verification process.

To validate listings, Proc Prints of the data set(s) being used are compared to the STATPROBE, Inc. listing output. STATPROBE, Inc. uses a Data _null_ to produce its listing and summary output. Data _null_ allows the user more flexibility in the "look" of the output. SAS/ASSIST provides tools to interactively define the options to produce a nice print out of the data to check against the Data null report.

For example, a dummy data set created at STATPROBE, Inc. appears as a listing in Table 4. Table 5 displays the same data using a Proc Print defined through SAS/ASSIST, as well as the definition screen for the SAS/ASSIST print.

NUMBER	ADVERSE EVENT	DATE	DATE
1	FEVER	01/10/96	01/12/96
1	PAIN	02/01/96	02/10/96
1	TOOTHACHE	01/10/96	01/17/96
2	PAIN	02/15/96	03/11/96
2	PAIN	03/15/96	04/14/96
2	FEVER	03/01/96	03/05/96
3	FEVER	01/01/96	01/10/96
3	PAIN	03/10/96	03/12/96
4	TOOTHACHE	02/10/96	02/11/96

Table 4. Listing of the Temp Data Set

PATIENT NUMBER	ADVERSE EVENT	START DATE	STOP DATE
1	FEVER	01/10/96	01/12/96
1	PAIN	02/01/96	02/10/96
1	TOOTHACHE	01/10/96	01/17/96
2	PAIN	02/15/96	03/11/96
2	PAIN	03/15/96	04/14/96
2	FEVER	03/01/96	03/05/96
3	FEVER	01/01/96	01/10/96
3	PAIN	03/10/96	03/12/96
4	TOOTHACHE	02/10/96	02/11/96

```

PSAS/ASSIST Test Data Set (Unfilled)
Active data set: SWC.TEMP      Submit date: MD
-----
Variables to appear in report: PT      NETXT      STRTDT      STOPDT

Due      PT      NETXT      STRTDT      STOPDT
.         .         .           .           .
.         .         .           .           .
.         .         .           .           .

```

Table 5. SAS/ASSIST Listing of the Temp Data Set, Definition Screen

The Proc Print of the data in Table 5 would be used to verify that the data output in Table 4 using a Data _null_ was correct. This assurance would be done by spot-checking the Proc Print versus the Data _null_ output.

Similarly, summary tables can be verified using the interactive Proc Tabulate definition through SAS/ASSIST. For the TEMP data, the data summarized would be the number of patients with an adverse event and the number of adverse events reported. Example output from a Data _null_ is shown in Table 6, while Table 7 displays the number of adverse event reports using the interactive Proc Tabulate application. Novice users would produce two tabulates, one displaying the number of patients with an adverse event and the other displaying the number of reports. The distinct values of the adverse event text variable (AETXT) are what the user would like to summarize. Therefore, it is the class variable.

ADVERSE EVENT	NUMBER	
	PAT	REPT
FEVER	3	3
PAIN	3	4
TOOTHACHE	2	2
TOTAL	8	9

Table 6. Data _null_ TEMP Data Set Summary.

Number of Reports	
AETXT	Number of Reports
FEVER	3.00
PAIN	4.00
TOOTHACHE	2.00
TOTAL	9.00

SAS/ASSIST: Edit Report Style (Unlinked)	
Active data set:	SAVE.TEMP
Subset data:	NO
Across items:	-NONE-
Class:	AETXT
Summary	Additional options

Table 7. SAS/ASSIST Tables Summary and Definition (Number of Reports).

To produce the SAS/ASSIST table displaying the number of reports for each adverse event, the user defined the active data set (SAVE.TEMP), the class variable (AETXT) and the statistic desired (number of non-missing values). In addition, the user selected the summary button to get the TOTAL line to appear in the report.

Verifying the SAS/ASSIST output versus the Data _null_ output shows that the data points match. Table 8 contains the output for the number of distinct patients in each class value.

Number of Patients	
AETXT	Number of Patients
FEVER	3.00
PAIN	3.00
TOOTHACHE	2.00

SAS/ASSIST: Edit Report Style (Unlinked)	
Active data set:	WORK.TEMP
Subset data:	NO
Across items:	-NONE-
Class:	AETXT
Summary	Additional options

Table 8. SAS/ASSIST Number of Patients.

In comparing Table 8 versus Table 6 there are no differences. However, the total number of patients was not displayed in Table 8. Therefore, the user would generate another table to get the number of patients overall.

Notice that the Table 8 active data set is WORK.TEMP instead of SAVE.TEMP. This is done because the data contains multiple observations per patient. In counting the number of patients, they should only be counted once. Therefore, the SAVE.TEMP data set was sent through a Proc Sort with the nodupkey option on AETXT and PT. This provides the working data set with a patient having only one observation per distinct AETXT. Figure 16 displays the screen used in SAS/ASSIST to sort the data.

SAS/ASSIST: Sort a Data Set	
Active data set:	SAVE.TEMP
Output data set:	WORK.TEMP
Active data set currently sorted by:	
PT	
Sort by variables:	AETXT PT
Ordering of variables:	All variables in ascending order
Additional options	

Figure 16. SAS/ASSIST Sort Screen

The 'ADDITIONAL OPTIONS' button allows the user to specify the nodupkey option.

Table 9 displays the SAS/ASSIST table for the overall number of patients.

Number of Patients	
	Number of Patients
PT	4.00

SAS/ASSIST: Fast Report Style (Untitled)	
Active data set: WORK.TEMP	Subset data: NO
Statistics:	
Variables:	Additional options:
PT	

Table 9. SAS/ASSIST Number of Patients Table.

In comparing the value in Table 9 versus the total number of patients in Table 6 the user would notice a discrepancy. Table 9 says that there were 4 patients total, while table 6 says there were 8 patients total. At this point, the user looks back at the Proc Print in Table 5 and hand-counts the number of patients since they were so minimal. That would show that 4 is the correct answer and that the program that created Table 6 was incorrect. Figure 17 contains the program that generated Table 6.

```

Libname save 'c:\mydocu~1';
Proc sort data=save.temp out=temp nodupkey;
  By aetxt pt;
Proc summary data=temp;
  Class aetxt;
  Var pt;
  Output out=stat1 n=numpat;
Run;
Proc sort data=save.temp out=temp;
  By aetxt;
Proc summary data=temp;
  Class aetxt;
  Output out=stat2;
Run;
Proc sort data=stat1;
  By aetxt;
Proc sort data=stat2;
  By aetxt;
Data all;
  Merge stat1 stat2;
  By aetxt;
Run;
Proc sort data=all;
  By descending _type_ aetxt;
Data _null_;
  Set all end=eof;
  By descending _type_ aetxt;
  File print ps=65 ls=180 header=header;

```

```

If first.aetxt and _type_=0 then put / @1
"TOTAL" @;
Else if first.aetxt then put @1 aetxt $13. @;
Put @15 numpat 3. +3 _freq_ 3. @;
If last.aetxt then put;
If eof then link footer;
Return;
Header:
  Put /// @1 28*!';
  Put @15 'NUMBER NUMBER';
  Put @1 'ADVERSE EVENT'
    @15 'PAT REPT';
  Put @1 28*!';
Return;
Footer:
  Put @1 28*!';
Return;
Run;

```

Figure 17. Data _null_ Summary Program

The reason for the discrepancy is that the user is taking the Proc Summary output and keeping the summary statistics where `_type_=0`. In this case, this statistic is the sum of the number of observations where PT is not missing. Therefore, it is counting the number of observations and not the number of distinct patients for the total. With this information, the user can modify the program so that it outputs the correct summary data.

The 'TABLES' application is not the only SAS/ASSIST program that would provide this information. The 'COUNTS' button provides this information too, but 'TABLES' displays it in a nice format. Experimentation with the different modules in SAS/ASSIST will provide users with the information they need to decide which modules they prefer over others when doing verification.

Conclusion

SAS/ASSIST is a great tool for report generation, interactive graphic definition/presentation, and validation/verification. New SAS users and advanced SAS users can use SAS/ASSIST to assist them in a variety of ways. This paper scratched the surface of SAS/ASSIST with some examples of report writing, graphs, and validation/verification techniques used at STATPROBE, Inc. There are many more useful applications in SAS/ASSIST for programmers, data managers, and statisticians to implement in their daily job responsibilities.

Acknowledgments

SAS, SAS/GRAPH, and SAS/ASSIST are registered trademarks or trademarks of SAS Institute Inc. in the USA and other countries.

* indicates USA registration.

Author's Address

Matt Becker
Manager, RIS
3885 Research Park Drive
Ann Arbor, MI 48108
E-Mail: GIANT_ATX@MSN.COM