ABSTRACT

The PRINT procedure has been underutilized as a tool for generating tables. It can, however, be easier to use, more flexible, and more powerful than the TABULATE or REPORT procedures, or collections of PUT statements. This is achieved by printing a single character variable that has been manipulated to resemble a table. Statistics from procedures such as MEANS, SUMMARY, or UNIVARIATE are added to the character string by a macro, mainly by use of the SUBSTR function. If necessary, the table is built by concatenating the variable in a series of data steps, each preceded by a macro call that builds a row or several rows of the table. These data steps allow additional string manipulation. The final table is essentially a PROC PRINT of the final single-variable data set. Macro variable substitution in the TITLE and FOOTNOTE statements of PROC PRINT provides further flexibility. A final macro passes the output through a page numbering data step and does final customization.

INTRODUCTION

Have you ever struggled long hours with procedure like TABULATE only to find that there was no way, short of swearing and editing the output, to get the right percentages in certain cells? Statistical reporting in the pharmaceutical industry, particularly in small rapidly changing biotechnology companies, often requires ad hoc and customized report tables that cannot be generated by REPORT, TABULATE, or other SAS® software designed for tabular reporting. Limitations inherent in some of these SAS procedures, for example, do not allow mixing of discrete and continuous data in the same table. Yet such a table is very common. The usual demographic summary table with counts of patients and means of age is a good example. The alternative of only generating tables that can be produced by the standard SAS reporting procedures is clearly backwards – reports should be driven by analysis and presentation needs, not by the limitations of the software.

The alternative of editing the output is unacceptable. This only increases the potential for errors and creates revision nightmares. The traditional alternative for making completely customized tables is to use PUT statements to write a file. Such tables are almost entirely configurable to one’s needs. PUT statement code, however, can take a long time to build and can be difficult to maintain.

One of the approaches at LTI is to use the PRINT procedure to generate tables. We take advantage of the this procedure’s titling and footnoting facility and print a single character variable of length 200, modified to look like a table. Unlike the case when we use PROC PRINT to print several variables, and have little control over the placement of information on a page, in the case of one long character variable, we have complete control, both in format and content.

We use four main steps to create tables with the PRINT procedure.

- Make the content, such as counts or other summary statistics, and add it to the character variable in row and column format.
- For reporting of multiple variables, repeat the procedure and concatenate the new rows to the previous in a data step.
- Set up the print environment so that macro variables are automatically added to the titles and footnotes.
- Process the printed output further in order to add page numbers or any other formatting so that the table conforms to our standard reporting requirements.
MACRO MODULES

At the core of this process are macros that take a data set, process it with a SAS procedure, and add the necessary output to our character variable (hereafter referred to as LINE). The SUMTAB macro, for example, runs the SUMMARY procedure and adds the output to LINE with SUBSTR function statements.

To make columns of data such as the following:

<table>
<thead>
<tr>
<th>Mean (SD)</th>
<th>55.2 (69.2)</th>
<th>45.5 (50.1)</th>
<th>73.1 (65.7)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Range</td>
<td>10 to 102</td>
<td>12 to 95</td>
<td>19 to 146</td>
</tr>
</tbody>
</table>

the SUMTAB macro uses the code below.

```sas
*** mean;
  IF N > 1 THEN SUBSTR(LINE, XMEEN, &spacer) =
    PUT(MN, &fmt) II "&pcnt; II
    PUT(&svar, &sfmt) II "&pcnt";
  ELSE IF N = 1 THEN SUBSTR(LINE, XMEEN, &spacer) =
    PUT(MN, &fmt) II "&pcnt; II
    PUT(&svar) II "NA";
  ELSE IF N = 0 THEN SUBSTR(LINE, XMEEN, &spacer) =
    "NA" II
    "NA";

*** range;
  IF N > 0 THEN SUBSTR(LINE2, XMEEN, &spacer) =
    PUT(MINIM, &rangefmt) II "&pcnt to " II
    PUT(MAXIM, &rangefmt) II "&pcnt";
  IF N = 0 THEN SUBSTR(LINE2, XMEEN, &spacer) =
    "NA to NA";
```

In the local macro environment, &spacer controls the width of the column, &fmt controls the format of the mean statistic, &pcnt becomes a percent sign if the data are in percentages, &svar becomes SE or SD, and &sfmt specifies the format of &svar. A do loop repeats for each column in the table and calculates XMEEN, the starting position for each column. These and other variables and macro parameters control the output.

This macro allows five other standard layouts, which can be specified by the CELLMFT parameter. The above output was defined by CELLMFT=4. When CELLMFT=6, SUMTAB produces output such as that below.

```
Mean  39.12
Median 38
Range  23 to 56
N     127
```

Should we need a new kind of layout for the statistics provided by the SUMMARY procedure -- or statistics that can be derived from such output -- we can modify the macro and assign it as CELLMFT number seven.

With a second module, the FREQTAB macro, we process count and percentage statistics from the FREQ procedure. An example of this macro's output is as follows:

```
<100 40 (83.3%)
100 to <200 4 ( 8.3%)
200 to <800 2 ( 4.2%)
≥800 0 (0.0%)
Unavailable 2 (4.2%)
```

The usual FREQ procedure percentages are available, for column, row, or total. In addition, we have the option of specifying a different denominator data set for the percentages than that which PROC FREQ would ordinarily use.

The above two macros are designed with many parameters in common so that they can conveniently be used for the same table. Parameters which have the same function have the same name. Some of these parameters and their description are:

- `rowlen` = number of spaces allowed for the label or grouping variable on the left of the table.
- `spacer` = spaces allowed for a column.
- `colvar` = variable to define the column grouping.
- `colfmt` = format for the column variable.
- `coltot` = label for the "All" or totals column -- a blank here keeps this column from printing.
- `colskip` = number of a column which should be omitted from the table.
The following macro call, with numerous default macro parameters not shown, will produce a data set called TEMPSUM containing LINE with the CELLFMT 4 layout.

```
%sumtab(
  dsname = 'si!mtabl ..
  dsname = 'sgetab ..
  var = 'cofint
  rangefmt = 'cellfmt
  spacer = 'rowlen
  more = 'YES
);
```

When more is "NO," the macro will output a solid line that completes the data area of the table and any footnotes defined by the macro. SUMTAB and FREQTAB define f1 to f9 as footnotes that will print immediately after the data area (as opposed to the PRINT procedure footnotes that will print at the bottom of the page).

When we need other new statistics or layouts, we can use the standard parameters of our two macro modules as a model to design other macros. At the same time, we have extended our existing modules by adding specialized functions. A FREQTAB macro modification, for instance, creates our standard adverse event summary tables.

**DATA STEP MANIPULATION**

Each macro call creates a working data set that becomes rows of the final table. Since the data sets contain the same variable, LINE, we can easily concatenate or further manipulate the output in data steps. An example of this process is shown in the box below. A data set called "blankln," containing one observation of the variable LINE with no characters, provides a convenient blank line for filling the table.

```
[SUMTAB macro call that makes the data set "tempsum" with mean and range values]
```

```
data table;
  set tempsum blankln(ln=bb);
  output;
  if bb then do;
    line = " Maximum SGOT (AST) (IU/L)";
    output;
    line = "Mean (SD)";
    output;
    line = "Range";
    output;
    line = "Maximun Total Bilirubin (mg/dL)";
    output;
  end;
run;
```

The above sequence of macro calls and data steps produce a data set that looks like a table when printed (two columns are omitted in the example below):

```
<table>
<thead>
<tr>
<th>Maximum SGOT (AST) (IU/L)</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;100</td>
</tr>
<tr>
<td>100 to &lt;200</td>
</tr>
<tr>
<td>200 to &lt;300</td>
</tr>
<tr>
<td>≥300</td>
</tr>
<tr>
<td>Unavailable</td>
</tr>
<tr>
<td>Mean (SD)</td>
</tr>
<tr>
<td>Range</td>
</tr>
<tr>
<td>43 (86.0%)</td>
</tr>
<tr>
<td>3 (6.0%)</td>
</tr>
<tr>
<td>2 (4.0%)</td>
</tr>
<tr>
<td>0 (0.0%)</td>
</tr>
<tr>
<td>2 (4.0%)</td>
</tr>
<tr>
<td>54.5 (67.16)</td>
</tr>
<tr>
<td>9 to 438</td>
</tr>
<tr>
<td>Maximum Total Bilirubin (mg/dL)</td>
</tr>
</tbody>
</table>
```

Much more elaborate data step manipulation is possible. By keying on row or column labels with the INDEX function, we can find a specific line and add additional information. As an example, we can add a column containing p-values that were created in a previously called procedure and output as macro variables from a CALL SYMPUT function.
When we invoke the SAS System, it automatically creates macro variables, such as SYSDATE, that are available for our program. Similarly, our batch SAS shell command creates standard macro variables, such as our user name and program name. Additional macro variables, including clinical study number, come from initialization files in our local directories. We use these macro variables to automatically customize our titles and footnotes.

Before using a PROC PRINT, we invoke a macro that sets up our print environment. It defines standard PAGESIZE and LINESIZE options depending on whether we want portrait or landscape mode. It uses our job name (an automatic macro variable) to find the correct titles for our table in a lookup file that contains all titles and associated job names. This allows us to keep all titles and title numbers in one place for easy modification, and removes this specific code from the program itself. The titles become macro variables that are printed as TITLE1-TITLE6 or our table. A parallel process creates footnotes for our table.

After the PROC PRINT in our program, we invoke a final macro that searches our output for a footnote line containing PAGE XXX of YYY -- which has been added by our previous print setup macro -- and converts these by adding numbering, such as Page 1 of 35. It adds to the same line our standard source documentation of the form Source: nda/t_lmax run by jpy on 11MAR94:13:30. It does whatever final processing we want it to and provides us with our last chance to automatically process our PRINT procedure output.

Figure 1 shows a table produced by the entire process. The numbered explanations below explain how the pieces of the table were created. Each number refers to the subsequent rows in the table.

1. The first 6 lines are TITLE1 to TITLE6 of the PRINT procedure. The content is entirely from macro variables that have been defined by the study directory in which the job was run.

2. The heading of the table and the first line of data (Number of Patients) come from the SUMTAB macro. When the macro is the first one called in the program, it automatically creates a heading -- otherwise, it just creates data rows.

3. These rows are generated in the data step manipulation between macro calls.

4. Output from a FREQTAB macro call.

5. Output from a SUMTAB macro call.

6. SUMTAB output. Since this is the last macro call, it creates a line across the table and adds footnotes.

7. This is a PRINT procedure FOOTNOTE modified with macro variables to contain source documentation and table number. The pagination is added by the final macro call that comes after the PRINT procedure.

A system consisting of 1) some core macros for taming FREQ and SUMMARY output, 2) simple data step manipulation, 3) a generalized SAS shell environment, and 4) the PRINT procedure, provides a simple, flexible, and powerful method for creating customized report tables.
### TABLE

**MAXIMUM ALKALINE PHOSPHATASE, SGOT (AST), AND TOTAL BILIRUBIN BY DOSE GROUP**

<table>
<thead>
<tr>
<th>Dose Group</th>
<th>5 mg</th>
<th>10 mg</th>
<th>All</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of Patients</td>
<td>50</td>
<td>196</td>
<td>246</td>
</tr>
</tbody>
</table>

#### Maximum Alkaline Phosphatase (iu/L)

<table>
<thead>
<tr>
<th>Range</th>
<th>5 mg</th>
<th>10 mg</th>
<th>All</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;130</td>
<td>10 (20.0%)</td>
<td>27 (13.8%)</td>
<td>37 (15.0%)</td>
</tr>
<tr>
<td>130 to &lt;600</td>
<td>32 (64.0%)</td>
<td>109 (55.6%)</td>
<td>141 (57.3%)</td>
</tr>
<tr>
<td>600 to &lt;2300</td>
<td>6 (12.0%)</td>
<td>15 (7.7%)</td>
<td>21 (8.5%)</td>
</tr>
<tr>
<td>≥2300</td>
<td>0 (0.0%)</td>
<td>1 (0.5%)</td>
<td>1 (0.4%)</td>
</tr>
<tr>
<td>Unavailable</td>
<td>2 (4.0%)</td>
<td>44 (22.4%)</td>
<td>46 (18.7%)</td>
</tr>
</tbody>
</table>

**Mean (SD)**

| Mean (SD) | 331 (300.7) | 310 (312.8) | 315 (309.4) |
| Range | 81 to 1613 | 62 to 2514 | 62 to 2514 |

#### Maximum SGOT (AST) (iu/L)

<table>
<thead>
<tr>
<th>Range</th>
<th>5 mg</th>
<th>10 mg</th>
<th>All</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;100</td>
<td>43 (86.0%)</td>
<td>138 (70.4%)</td>
<td>181 (73.6%)</td>
</tr>
<tr>
<td>100 to &lt;200</td>
<td>3 (6.0%)</td>
<td>6 (3.1%)</td>
<td>9 (3.7%)</td>
</tr>
<tr>
<td>200 to &lt;800</td>
<td>2 (4.0%)</td>
<td>4 (2.0%)</td>
<td>6 (2.4%)</td>
</tr>
<tr>
<td>≥800</td>
<td>0 (0.0%)</td>
<td>0 (0.0%)</td>
<td>0 (0.0%)</td>
</tr>
<tr>
<td>Unavailable</td>
<td>2 (4.0%)</td>
<td>46 (24.5%)</td>
<td>50 (20.3%)</td>
</tr>
</tbody>
</table>

**Mean (SD)**

| Mean (SD) | 54.5 (67.16) | 39.4 (41.38) | 43.1 (49.20) |
| Range | 9 to 438 | 8 to 262 | 8 to 438 |

#### Maximum Total Bilirubin (mg/dL)

<table>
<thead>
<tr>
<th>Range</th>
<th>5 mg</th>
<th>10 mg</th>
<th>All</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;1.5</td>
<td>45 (90.0%)</td>
<td>142 (72.4%)</td>
<td>187 (76.0%)</td>
</tr>
<tr>
<td>1.5 to &lt;3.0</td>
<td>1 (2.0%)</td>
<td>2 (1.0%)</td>
<td>3 (1.2%)</td>
</tr>
<tr>
<td>≥3.0</td>
<td>2 (4.0%)</td>
<td>6 (3.1%)</td>
<td>8 (3.3%)</td>
</tr>
<tr>
<td>Unavailable</td>
<td>2 (4.0%)</td>
<td>46 (23.5%)</td>
<td>48 (19.5%)</td>
</tr>
</tbody>
</table>

**Mean (SD)**

| Mean (SD) | 0.9 (1.16) | 0.8 (1.15) | 0.9 (1.15) |
| Range | 0.0 to 6. | 0.0 to 8.8 | 0.0 to 8.8 |

---

**Note:** Within each patient, across all post-baseline laboratory evaluations, the maximum alkaline phosphatase, SGOT (AST), and total bilirubin are selected. The mean, SD, and range are computed from the patients' maxima.

**Source:** interim/timaxlrun by jpy on 12MAR94:13:30

---

Figure 1. A Table Produced with the PRINT procedure.
ACKNOWLEDGMENTS

I am grateful to Greg Silva for his exemplary and elaborate use of the SUBSTR function in macros, and to John Brega for providing specific examples of a generalized SAS batch processing environment.