Introduction

The advent of PostScript laser printers, easy 300 dpi (dots per inch), typesetting has become available to an increasing number of people. However, SAS users for the most part have been restricted to two methods for using this technology. 1) Using a system program that takes output from the SAS system and sends it to the printer. While this improves the output quality, it does not enable the use of the text typesetting capabilities of PostScript. 2) Importing the output generated by the SAS system into a word processing program, and using the typesetting capabilities of the word processor. For a one-time report, this approach is useful, but its utility diminishes with the number of reports, and the frequency of their generation. The macro library is designed to allow the creation of typeset reports from within a SAS program. It does not replace PROC PRINT or PROC TABULATE, nor does it replace PROC GPRINT, which has limited typesetting capabilities. It is for those who generate customized reports with the SAS data step. It is not a full-blown PostScript driver of the type that would be found in a word processing program. It is an add-on that makes it possible to do simple typesetting. The library has been tested on SAS versions 6.03 and 6.07 under UNIX, and versions 6.03 and 6.04 in MS-DOS without problem, using a variety of tasks ranging from tables to form letters.

What Is PostScript?

Simply put, it is a programming language. A document written for a PostScript printer is a program to be executed by the printer. A PostScript printer is equipped with an interpreter, which translates the commands in a file into printer actions. The language has variables, mathematical operators, and a set of commands. With this set of commands, you can specify lines to be drawn, or rotated. The lines can be any length and may be drawn in any number of ways. This facilitates the creation of what are called "font dictionaries." These font dictionaries are what gives PostScript its text typesetting capabilities. Each PostScript printer has a number of font dictionaries resident, which can be called by the appropriate PostScript command.

What Can The Library Do, and How Does It Work?

It allows up to 30 right-justified, center-justified, or left-justified columns of information. Font selection and scaling are handled, where the only limit is the number of fonts available on the printer and the physical dimensions of the page. Text may be centered within a line on the page. Title lines can be placed on each page, along with page numbering and one footnote. Customized spacing (single-, double-, or fractional) is available. Top and left margins may be defined, as well as the paper size and print orientation (portrait mode or landscape mode). With this, it is possible to create a neatly typeset report that does not require user intervention each time it is generated.

The library operates by writing the PostScript code necessary to move the "pen" around the page, and placing the output from SAS into the PostScript code. The files generated by the library are much larger than a non-PostScript SAS output file, since it contains PostScript instructions along with the output. As far as computing resources are concerned, virtually all the extra processing overhead is due to the increased number of I/O operations. As an example, here are the relevant (as far as comparing benchmarks) parts of two SAS logs.

What Can The Library Do, and How Does It Work?

1049

What Can The Library Do, and How Does It Work?

It allows up to 30 right-justified, center-justified, or left-justified columns of information. Font selection and scaling are handled, where the only limit is the number of fonts available on the printer and the physical dimensions of the page. Text may be centered within a line on the page. Title lines can be placed on each page, along with page numbering and one footnote. Customized spacing (single-, double-, or fractional) is available. Top and left margins may be defined, as well as the paper size and print orientation (portrait mode or landscape mode). With this, it is possible to create a neatly typeset report that does not require user intervention each time it is generated.

The library operates by writing the PostScript code necessary to move the "pen" around the page, and placing the output from SAS into the PostScript code. The files generated by the library are much larger than a non-PostScript SAS output file, since it contains PostScript instructions along with the output. As far as computing resources are concerned, virtually all the extra processing overhead is due to the increased number of I/O operations. As an example, here are the relevant (as far as comparing benchmarks) parts of two SAS logs.

369 %set(lines=17);
NOTE: DATA statement used:
real time 0.240 seconds
cpu time 0.170 seconds
370 %settab(1,85,2,35,2,85,3,35,3,85,4,35,4,85,4,85,5,35,
5,85,6,35,6,85,7,35,7,85,7,85);
NOTE: DATA statement used:
real time 1.660 seconds
cpu time 1.440 seconds
372 data _null_;
NOTE: DATA statement used:
real time 10.200 seconds
cpu time 2.990 seconds
373 %intt(ds=m1,by=descending yr site, file=sugilpt.ps);
397 do l=1 to 12;
398 %tabr(i,e{i},3.);
399 t{ij + e{ij;}
415 %term;
416 run;
NOTE: 1817 records were written to the file 'sugilpt.ps'.
The minimum record length was 3.
The maximum record length was 55.
NOTE: DATA statement used:
real time 10.200 seconds
cpu time 2.990 seconds

The total CPU time for this PostScript report (it is a four-page table) is 4.6 seconds, and it took 1817 records. The time to produce the same report without using the library was .83 seconds, and it occupied only 33 records.

NOTE: 33 records were written to the file 'sugitplasc'. The minimum record length was 52. The maximum record length was 92.

NOTE: DATA statement used:
- real time 1.800 seconds
- cpu time 0.830 seconds

What Good Is It, Really?

Given the larger amount of resources needed, it may be asked “Why bother?” The first reason for typesetting anything is publication. By using the macro, a publication-ready copy is created by the SAS system. Instead of using SAS to obtain the information, then importing the Information into a publication template, it is possible to use the SAS output itself. This can save a good deal of time over the course of a project if the reports are generated many times, or if the data changes often. A second reason for using typesetting is that proportional fonts are easier to read than fixed-pitch fonts. In addition, more information can be presented in the same amount of space if using a proportional instead of a fixed-pitch font. The following example will demonstrate.

CLEK: Screened Patients by Site and Month: Year 1991

<table>
<thead>
<tr>
<th></th>
<th>Jan</th>
<th>Feb</th>
<th>Mar</th>
<th>Apr</th>
<th>May</th>
<th>Jun</th>
<th>Jul</th>
<th>Aug</th>
<th>Sep</th>
<th>Oct</th>
<th>Nov</th>
<th>Dec</th>
</tr>
</thead>
<tbody>
<tr>
<td>SCC</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>.</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>3</td>
<td>1</td>
<td>.</td>
<td>3</td>
<td>1</td>
</tr>
<tr>
<td>SEL</td>
<td>.</td>
<td>15</td>
<td>8</td>
<td>1</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
</tr>
<tr>
<td>SHO</td>
<td>5</td>
<td>15</td>
<td>13</td>
<td>11</td>
<td>5</td>
<td>1</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
</tr>
<tr>
<td>SUN</td>
<td>.</td>
<td>1</td>
<td>1</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>1</td>
<td>1</td>
<td>6</td>
<td>7</td>
<td>.</td>
</tr>
<tr>
<td>UAB</td>
<td>.</td>
<td>.</td>
<td>3</td>
<td>4</td>
<td>4</td>
<td>7</td>
<td>12</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
</tr>
<tr>
<td>WEI</td>
<td>.</td>
<td>3</td>
<td>.</td>
<td>7</td>
<td>2</td>
<td>.</td>
<td>4</td>
<td>1</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
</tr>
<tr>
<td>YAG</td>
<td>6</td>
<td>2</td>
<td>4</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
</tr>
</tbody>
</table>

Monthly totals 12 38 28 22 12 7 14 17 2 6 10 1

CLEK: Screened Patients by Site and Month: Year 1991

<table>
<thead>
<tr>
<th></th>
<th>Jan</th>
<th>Feb</th>
<th>Mar</th>
<th>Apr</th>
<th>May</th>
<th>Jun</th>
<th>Jul</th>
<th>Aug</th>
<th>Sep</th>
<th>Oct</th>
<th>Nov</th>
<th>Dec</th>
</tr>
</thead>
<tbody>
<tr>
<td>SCC</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>.</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>3</td>
<td>1</td>
<td>.</td>
<td>3</td>
<td>1</td>
</tr>
<tr>
<td>SEL</td>
<td>.</td>
<td>15</td>
<td>8</td>
<td>1</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
</tr>
<tr>
<td>SHO</td>
<td>5</td>
<td>15</td>
<td>13</td>
<td>11</td>
<td>5</td>
<td>1</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
</tr>
<tr>
<td>SUN</td>
<td>.</td>
<td>1</td>
<td>1</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>1</td>
<td>1</td>
<td>6</td>
<td>7</td>
<td>.</td>
</tr>
<tr>
<td>UAB</td>
<td>.</td>
<td>.</td>
<td>3</td>
<td>4</td>
<td>4</td>
<td>7</td>
<td>12</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
</tr>
<tr>
<td>WEI</td>
<td>.</td>
<td>3</td>
<td>.</td>
<td>7</td>
<td>2</td>
<td>.</td>
<td>4</td>
<td>1</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
</tr>
<tr>
<td>YAG</td>
<td>6</td>
<td>2</td>
<td>4</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
</tr>
</tbody>
</table>

Monthly totals 12 38 28 22 12 7 14 17 2 6 10 1

1050
How Do I Use It?

The library has a macro for each typesetting function. The following table lists the macros, their functions, and their parameters.

SET-UP (Page description, Tab stops, Library usage)

%SET(w=8.5,h=11,lm=0.5,tm=0.5,lines=6,orient=S):
Describes the physical dimensions of the page. Size parameters are defined in inches.
- W: Paper width.
- LM: Left margin.
- TM: Top margin.
- LINES: Maximum number of lines to leave before going to a new page.
- ORIENT: Enter L for LANDSCAPE orientation.

%SETTAB(tab1,tab2,tab3,...tab30):
Defines tab settings in inches.

%INIT(DS=_LAST..BY=,FILE=,APPEND=N,PAGE=2):
Initializes library in report generation step.
- DS: SAS dataset to be used in the report.
- BY: Name(s) of any BY variables to be used.
- FILE: Name of the output file.
- APPEND: Append to an existing file.
- PAGE: The starting page number.

REPORT FORMATTING (Font selection, Tabs, and centering)

%STRT: Starts new page if necessary.

%FONT(type=Courier,size=10):
Selects font and size.
- TYPE: Name of PostScript font.
- SIZE: Size in points (1/72 inch units.)

%SHOW: Places contents of "PUT" statement on paper.

%CRLF(space):
Goes to a new line.
- SPACE: Number of lines to skip, default is 1.

%TERM: Ejects the last page from printer.

%PAGE: Starts a new page.

%TAB(n):
Moves "pen" to a tab stop "n", where n is the number of a tab stop defined in %SETTAB. N may also be a SAS variable.

%TABR(n,var,vfmt):
Right-justifies at a given tab stop.
- N: Tab stop number defined in %SETTAB.
- VAR: Value to be right-justified.
- Vfmt: The format to use for the value.

%TABC(n,var,vfmt):
Center-justifies at a given tab stop.
- N: Tab stop number defined in %SETTAB.
- VAR: Value to be right-justified.
- Vfmt: The format to use for the value.

%CTR(var,vfmt):
Centers text on a line.
- VAR: Value to be right-justified.
- Vfmt: The format to use for the value.

TITLES, FOOTNOTES, AND PAGE NUMBERING

%TITLE(n=1,string=,type=Bookman-Demi,size=16,loc=0,ctr=N):
Place text at top of each page.
- N: Number of title line.
- STRING: Text of title.
- TYPE: Font to use.
- SIZE: Size of the title font.
- CTR: Center title line, ignore LOC parameter.
- LOC: Horizontal location of title (inches from left edge).

%PAGNO(type=times-BoldItalic,size=10,loc=,justfy=R,strt=2,top=N,txt=Page,rtxt=):
Puts page numbers on each page.
- TYPE: Name of font to use.
- SIZE: Size of the title font.
- JUSTIFY: Specify left, center, right, or no justification, ignore LOC parameter.
- LOC: Horizontal location of title (inches from left edge).
- STRT: Physical page to start numbering.
- TOP: Put page number top or the bottom of page?
- TXT: The text to the lef of the page number.
- RTXT: The text to the right of the page number.

%FOOT(type=Times-BoldItalic,justfy=R,loc=,size=10,var=,vfmt=):
Puts 1 line of text at bottom of each page.
- TYPE: Name of font to use.
- SIZE: Size of the title font.
- JUSTIFY: Specify left, center, right, or no justification, ignore LOC parameter.
- LOC: Horizontal location of title (inches from left edge).
- VAR: Text to be right-justified.
- Vfmt: The format to use for the text.
Here is a simple example program showing use of the library, with commented code, and the output produced by the program. Some detailed program notes follow.

```sas
%include '/users/ficsit/pslib.mac';

%title(n=1,string=FICSIT Common Data Status Report,etr=Y,si=20);
%title(n=2,si=30);
%title(n=3,string=Baseline Data At A Glance,etr=Y,si=20);
%title(n=4,si=30);
%pagno(top=Y,si=1);

%init(ds=baseline,by=son,file='status.ps',page=1);

%set(UD=0.75,lm=.25,orient=L);
%settab(2.85,3.95,4.95,6.7,15.8,05.8,8.9,8);

data _null_;
/* Define dataset, sort order, output file */
%init(ds=baseline,by=son,file='status.ps',page=1);

/* Define title sequence */
%title(n=1,string=FICSIT Common Data Status Report,etr=Y,si=20);
%title(n=2,si=30);
%title(n=3,string=Baseline Data At A Glance,etr=Y,si=20);
%title(n=4,si=30);

/* Set page numbering system */
%pagno(top=Y,si=1);

/* This sequence defines footnote */
d = today();
foot = *This report was generated on * ||
left(put(d,mmdyy8));
%foot(var=foot);

/* Print column headers at each new page */
if (_newpage) then do;
  %font(type=Helvetica-Bold,si=12);
  /* Center headers at tab positions */
  do i = 1 to 8;
    %tabc(i,1,sitefmt);
  end;
  %crlf(2); /* Double-space */
end;

/* Print row headers */
if (first.son) then do;
  put '(' son sortlst. ')';
  %show;
end;

/* Right justify table values */
%tabr(site,count,4.);
if last.son then do;
  %crlf(1.5); /* One and a half space */
end;
%term; /* Eject last page at end of data */
run;
```

Program Notes

In the above program, note that parentheses surround ALL of the information to be printed. The PostScript language delimits character strings with parentheses. The information in the SAS output is treated as strings of characters (including numeric fields). Due to the varying contexts of the information (variables, constant text, constant text mixed with variables), it is virtually impossible to know where the parentheses should go ahead of time. Having the programmer put them in increases flexibility, with a minimum of extra programming effort. Also, every PUT statement is followed by the %SHOW macro. This is necessary for PostScript to actually put the "pen" on the paper. If omitted, whatever was to be written by the PUT statement won't show up on the page! In the above program, you will also see that whenever a macro is used conditionally, it is part of a DO group. Each macro contains more than one SAS statement, so the DO is necessary.

The above example shows how tables can be generated using the library with ease. By placing the column headers and the data inside DO loops, the index variable changes the tab position, while the %TABC and %TABR routines format and justify the values.

This program demonstrates the use of a multiple line title sequence, page numbering starting on the first page of output (%INIT macro), a compound footnote, and the method of printing a variable header on each page. The library has an internal variable (_NEWPAGE) which signals the beginning of a new page, and makes it possible to use headers without using macro variables. The use of less than full spacing (one-and-a-half instead of double) is also shown. The output from this code follows.
Baseline Data At A Glance

<table>
<thead>
<tr>
<th>DEMOGRAPHICS</th>
<th>PORTLAND</th>
<th>NEW HAVEN</th>
<th>SEATTLE</th>
<th>SAN ANTONIO</th>
<th>ATLANTA</th>
</tr>
</thead>
<tbody>
<tr>
<td>MINI-MENTAL</td>
<td>133</td>
<td>80</td>
<td>70</td>
<td>145</td>
<td>178</td>
</tr>
<tr>
<td>PHYSICAL FUNCTION</td>
<td>100</td>
<td>80</td>
<td>70</td>
<td>145</td>
<td>178</td>
</tr>
<tr>
<td>SIP/ADL</td>
<td>133</td>
<td>80</td>
<td>70</td>
<td>145</td>
<td>170</td>
</tr>
<tr>
<td>GAIT/BALANCE</td>
<td>100</td>
<td>80</td>
<td>70</td>
<td>145</td>
<td>170</td>
</tr>
<tr>
<td>CES-D</td>
<td>133</td>
<td>80</td>
<td>70</td>
<td>145</td>
<td>178</td>
</tr>
<tr>
<td>GDS</td>
<td>133</td>
<td>80</td>
<td>70</td>
<td>145</td>
<td>170</td>
</tr>
<tr>
<td>CHRONIC CONDITIONS</td>
<td>133</td>
<td>80</td>
<td>70</td>
<td>145</td>
<td>178</td>
</tr>
<tr>
<td>QUALITY OF LIFE</td>
<td>133</td>
<td>80</td>
<td>70</td>
<td>145</td>
<td>170</td>
</tr>
</tbody>
</table>

Typesetting Isn't Always The Answer

First, the macro library requires a certain amount of programming commitment. It is not for the casual SAS user, because problems can arise in many areas. 1) Improper use of the library. Examples of this include: not enclosing the macros in DO groups, and my personal favorite, forgetting to use the %SHOW macro. 2) Errors in the PostScript code generated. These are printer-dependent, and do NOT show up in the SAS log. The most frequent error of this type is incorrectly specifying a font. SAS has no way of knowing that "Corin-Bold" is not a valid PostScript font. Error checking could be built into the library, but there are so many PostScript fonts available that this would add an enormous amount to the processing overhead. If one of these errors does occur, it can be difficult to track down, especially if your printer doesn't produce error messages. 3) Last, and perhaps most important, there is the fact that typesetting requires some work in and of itself. Depending on how complex the report is, proper layout will take some time. The best tools for typesetting are a ruler, and a blank piece of paper. The macro library isn't WYSIWYG, so it generally takes a couple of printouts to make sure that the output looks the way that you want. Typesetting makes you think in terms of inches, not columns, because most PostScript fonts are proportional. This means that each character takes up a differing amount of space. If care is not taken when doing the page layout, report columns may run into each other because tabs are too closely spaced together, or the font is scaled too big. Margin problems can also arise because of this, and sometimes there isn't enough room to fit everything on one page.

Given those cautions, if the physical presentation of a report is as important as the content of that report, and until SAS/PUBLISH™ arrives, the macro library is a method to obtain PostScript typesetting from inside the SAS system. With a little creativity, as shown on the following page, the library can be a useful addition to the SAS programmer's tool kit. This work was partially supported by NIA/NCRR grant # IU01AG09098-01 and National Eye Institute grant # NEIHR21 EY08652.

Further inquiries welcome:
Derek Morgan
Washington University Division of Biostatistics
Box 8067, 660 S. Euclid Ave.
St. Louis, MO 63110
(314) - 362 - 3685

SAS and SAS/PUBLISH software are registered trademarks or trademarks of SAS Institute, Inc. in the USA and other countries. ® indicates USA registration. PostScript is a registered trademark of Adobe Systems, Inc.
Dear Kenneth,

Below you will find a list of the pending publications on which you are the contact person, with the list of people who have indicated interest in working on that publication. Please check these over and let Nancy Moore know of any changes that need to be made.

Thank you.

37 Compliance rates in frail elderly populations - Victor Stevens
38 Factors associated with compliance to health promotion interventions in frail elderly populations - Marcia Ory, Victor Stevens, Nancy Kutner
39 Subject recruitment in FICSIT - Patty Karlen, David Buchner, Marcia Ory
40 Cost of the individual FICSIT research projects - Mark Hornbrook, Victor Stevens

FICSIT Publications
Published or Presented Works

Multi-Site

2 Preplanned Meta Analyses In a clinical trial aimed at reducing frailty and Injuries In older persons
J.Philip Miller, Kenneth Schechtman, Michael Province, Cynthia Arfken, Deborah Smith, Cynthia Mulrow, Marcia Ory, Sharlene Weiss, Evan Hadley, The FICSIT Study Group
TYPE: Presentation
STATUS: Published
Journal/Conference: Society for Clinical Trials
Publication/Conference Date: July 5, 1991
Contact Person: J.Philip Miller

The NIH sponsored multicenter study, "Frailty and Injuries: Cooperative Studies of Intervention Techniques" (FICSIT), has a unique design which facilitates pre-planned meta analyses. Meta analyses generally involve the post-hoc selection of studies which may have different definitions of endpoints. Analyses are based on aggregate rather than raw data. The FICSIT design is such that there is a priori study selection, that endpoints have identical definitions, and that a common set of raw data will be analyzed. FICSIT consists of a coordinating center and eight "independent" randomized clinical trials, each of which is using a unique protocol to test its own intervention aimed at reducing frailty, falls, and/or fall-related injuries in older persons. Investigators collaborated to define a common data base consisting of a large set of common data items spanning the domains of demographics, cognition, gait, balance and strength, and a common protocol for prospectively documenting falls and injuries. Each site has agreed to collect all, or nearly all elements of the common data base in addition to trial/site specific items. Because of differences between trials, a simple pooled analysis is not justified. However, the use of identical measures and of raw rather than summary data means that the meta analysis will gain power and not be compromised by missing and incomplete data. Because the eight studies to be meta analyzed are pre-determined, selection and publication bias will not be present. Because individual trials that focus on risk factors for falling may not have the power to examine falls or fall-related injuries as endpoints, FICSIT will provide many traditional advantages of meta analyses while eliminating important pitfalls.