

A PostScript¹ Macro Library

Derek Morgan, Washington University Medical School

Introduction

With the advent of PostScript laser printers, easy 300-dpi (dots per inch) text 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 which takes output from SAS and sends it to the printer. While this improves the output quality, it does not allow the user to take advantage of the text typesetting capabilities of PostScript. 2) Importing the output generated by SAS 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 be an interface between PostScript and the SAS output. It writes SAS code to produce a PostScript program containing the SAS output. 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-fledged PostScript driver of the type that would be found in a word processing program. It is merely an add-on to enable the use of different fonts and scaling.

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 is 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 defined along with page numbering and one footnote, and placed on each page. Customized spacing (single-, double-, or one-and-a-half) 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, with boldface and italics to bring attention to the more important fields.

The first step in using the library is setting up the physical dimensions of the page to be printed. The macro used is %PS_SET. The next step is to define all of the tab positions being used in the report, with the macro %PS_STAB. Each tab position is defined in inches, and does not have to be in ascending or descending order. This also facilitates adding an extra column, or moving columns, since only the tab positions affected need to be changed, and it is all done in this statement. Both PS_STAB and PS_SET are executed outside of the DATA step in which the report is generated because they are complete DATA steps.

In the report-generation step, the first statement after the SET and any BY statements should be the %PS_INIT macro. This macro writes the SAS code that enables the use of the rest of the library. It also is where the numbering scheme for the page numbering facility is defined. Another important macro is %PS_STRT. Its main function is to make sure that there is enough room on the physical page to print following text. With this, it is possible to define where page breaks before and within observations belong.

Title, footnote, and page numbering information should follow, if they are desired. %PS_TITLE prints constant text at the top of each page of the document. You may use as many titles as will fit on the page, given the size and type of font you've selected. Each title may be centered, or placed anywhere within a horizontal line. Page numbering is also printed on each page if the %PS_PAGNO macro is invoked. The footnote macro, %PS_FOOT, is used to place text at the bottom of each page. The footnote may be any of the following: a string constant in quotes, variable, or array reference.

The remainder of the macros control the printing. %PS_TERM is used to eject the last page of the output when the end of the data set is reached. %PS_FONT controls font selection and scaling. %PS_CRLF provides the carriage return/line feeds in the PostScript code, %PS_PAGE causes a page eject, %PS_SHOW actually places the item on the page. Last, there are the tab stop macros, %PS_TAB, %PS_TABR, and %PS_TABC. PS_TAB sets the logical pen to one of the tab stops defined in PS_STAB. PS_TABR and PS_TABC are slightly different, since they must take an existing set of characters, and justify it at a given spot. Therefore, the actual starting location depends on the length of the characters to be

printed, which in turn depends on the characters, the font and the size. PS_TABR right-justifies the text (useful for numeric output), while PS_TABC center justifies the text (useful in printing column headings). When selecting tabs, any SAS expression which evaluates to a number may be used. This makes it convenient to use DO loops to print tables.

These are the macros in the library. The following program demonstrates some of the capabilities of the macro library. This example prints a phone book for the members of a national research project.

```
options missing='';
%include 'pslib.mac';

/* Define page layout for directory */
%ps_set(tm=1.0,lines=11,lm=1.7);
%ps_stab(5.2,5.5,2.06,2.);/* Define tab stops */
data _null_;
set a end=last;
by clinic;
array m1{5} addr1-addr5;
file 'ficsit.ps' noprint; /* Specify output file */
foot = 'Appendix B';
put(date(),mmdyy8.); /* Create footer line */
%ps_init(page=2); /* Initialize macros, first
page number is 2. */

/* Set up page break at each new clinic */
if (first.clinic and _n_ ne 1) then do;
%ps_page;
end;

/* Activate page numbering, starting on first
page, at top of page, right-justified */
%ps_pagno(strt=1,top=Y,lxt=B);
/* Put footing on page, starting at 4.25" from
left border */
%ps_foot(var=foot,justify=N,loc=4.25);

/* This section creates the header at the top of
each page and at the beginning of a clinic */
if (first.clinic or _newpage) then do;
/* Select font and size for header */
%ps_font(type=Times-Bold,size=16);
b = 'Clinic number: ' || put(clinic,2.) || ' - ' ||
put(clinic,locate.);/* Make header text */
b = trim(b);
%ps_strt;
%ps_ctr(b); /*Print header centered in line */
/* Double space after header */
%ps_crlf;
%ps_crlf;
end;

/* First line of directory entry in boldface; go to
next line */
%ps_font(type=Helvetica-Bold,size=9);
%ps_strt; /* Enough room on page for obs?
(>11 lines?) */

put '(' namestr ');
%ps_show;
%ps_crlf;

/* Reset font for remainder of obs */
%ps_font(type=Helvetica,size=9);
/* Second line of directory entry */
put '(' role ');
%ps_show;
%ps_tab(1); /* Go to first tab stop, same line */
put '(E-mail: ' email ');
%ps_show;
%ps_crlf; /* Go to next line */
/* Third line of directory entry */
put '(Phone: ' phone ');
%ps_show;
%ps_tab(1); /* Go to first tab stop, same line */
put '(FAX: ' fax ');
%ps_show;
%ps_crlf; /* Go to next line */
/* Fourth line of directory entry */
%ps_tab(3); /* Go to third tab stop, same line */
put '(MAIL)';
%ps_show;
%ps_crlf; /* Go to next line */

/* Address portion of directory */
do i = 1 to 5;
if m1{i} ne '' then do;
put '(' m1{i} ');
%ps_show;
%ps_crlf;
end;
end;
%ps_crlf; /* Skip a line */
%ps_term; /* Eject last page at EOF */
run;

/* Directory Index Code */

/* Define a new page layout */
%ps_set(tm=1.0,lines=7,lm=1.2);
%ps_stab(5.5); /* Reset tab stops */
data _null_;
set b end=last;
file 'ficsit.ps' noprint mod; /* Append to output file */

/* Initialize macros, first page number is next
```

```

number in sequence from last DATA step. */
%ps_init(page=&PAGENUM);

/* Define title sequence */
%ps_title(string=FICSIT Directory Index,ctr=Y,
type=Times-BoldItalic,size=20);
%ps_title(n=2,string=,size=29);

/* Assemble footer */
foot = 'Appendix B
put(date(),mmddyy8.);
/* Define placement of page numbers */
%ps_pagno(strt=1,top=Y,lxt=B);
/* Set up placement of footnote */
%ps_foot(var=foot,justify=N,loc=4.25);

```

```

%ps_font(type=Helvetica,size=9); /* Define font */
%ps_strt; /* Enough room on page for this
obs? */
put '(' namestr ',' locate ');
%ps_show;
%ps_tab(1); /* Go to tab stop, same line */
put '(Phone: ' phone ');
%ps_show;
%ps_font(size=14); /* Redefine line size. */
%ps_crlf; /* Go to next line. Next line is
14 points down, not 9.*/
%ps_term; /* Eject page at EOF */
run;

```

Clinic number: 0 - St. Louis, MO

ARFKEN, Cynthia, Ph.D.
Investigator
Phone: (314) 362-3594
MAIL
Washington University School of Medicine
Division of Biostatistics
Box 8067
660 South Euclid
St. Louis, MO, 63110

E-mail: cindi@wubios.WUstl.EDU
FAX: (314) 362-2693

MILLER, J. Philip, A.B.
Principal Investigator
Phone: (314) 362-3617
MAIL
Washington University School of Medicine
Division of Biostatistics
Box 8067
660 South Euclid
St. Louis, MO, 63110

E-mail: phil@wubios.WUstl.EDU
FAX: (314) 362-2693

MOORE, Nancy R.
Special Project Assistant
Phone: (314) 362-2612
MAIL
Washington University School of Medicine
Division of Biostatistics
Box 8067
660 South Euclid
St. Louis, MO, 63110

E-mail: nancy@wubios.wustl.edu
FAX: (314) 362-2693

MORGAN, Derek, A.B.
Data Coordinator
Phone: (314) 362-3685
MAIL
Washington University School of Medicine
Division of Biostatistics
Box 8067
660 South Euclid
St. Louis, MO, 63110

E-mail: derek@wubios.WUstl.EDU
FAX: (314) 362-2693

FICSIT Directory Index

ABRASS, Itamar B., M.D. , Seattle, WA
 AGUILAR, Christine, M.D.,M.P.H. , San Antonio, TX
 ALQUIST, Alan D., M.S. , Seattle, WA
 AMERMAN, Paula M., R.N., M.S.N. , Farmington, CT
 ARFKEN, Cynthia, Ph.D. , St. Louis, MO
 BAKER, Dorothy I., A.B.D.,M.N. , New Haven, CT
 BARR, Robin, Ph.D. , Bethesda, MD
 BIRD, Margie Britnell, B.A. , Bethesda, MD

Phone: (206) 223-3089
 Phone: (512) 694-5199
 Phone: (206) 685-0945
 Phone: (203) 679-3217
 Phone: (314) 362-3594
 Phone: (203) 785-5238
 Phone: (301) 496-3136
 Phone: (301) 496-1033

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 %PS_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! The FILE statement which defines the output file contains the NOPRINT option. This is critical, since SAS titles, footnotes, and carriage control will confuse the printer. 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. One more important item is the END=LAST option at the end of the SET statement. This insures that the last page of output will be ejected from the printer when %PS_TERM is called.

The output files are much larger than the comparable non-PostScript output file would be, since the PostScript code is generated as well as the SAS output. The size of the output file is related to the complexity of the page layout. Processing time also increases, but it is largely due to I/O operations. The amount of processing overhead required for execution of the library is small.

Two of the features demonstrated in the program are: 1) The ability to continue the page numbering sequence from one DATA step to the next. This is achieved by using the &PAGENUM macro variable, which is defined in the library. If there is more than one page layout in the same report, this removes the need to hard-code the page number. 2) Non-integral spacing. For the directory index, instead of double spacing, a little more than one-and-a-half spaces are used.

The last program to be considered is a tabular report. This will demonstrate the use of justified tabs, and DO loops to present tables.

```
%inc 'pslib.mac';
%ps_set(lines=17,lm=.9); /* Define page layout */

/* Define tab stops */

%ps_stab(2.05,2.5,3.00,3.5,4.4,4.8,5.25,5.7,6.1,6.5,7.0,
        7.2,7.55);
data _null_;
set m2 end=last;
by descending yr site;
```

```
elig13 = sum(of elig1-elig12);
array e{13} elig1-elig13;
array t{13} elgtot1-elgtot13;
array mon{12} $ _TEMPORARY_ ('Jan','Feb','Mar',
'Apr','May','Jun','Jul','Aug','Sep','Oct','Nov','Dec');
file 'mwsug.ps' noprint;
%ps_init;
d = date();
%ps_foot(type=NewCenturySchlbk-Italic,size=8,var=d,
        vfmt=mmddyy8.);
```

```

if _newpage then do;
  date = mdy(1,1,yr);
  %ps_font(type=Helvetica-Bold,size=12);
  z = 'CLEK: Eligible Patients by Site and Month: ' ||
    'Year ' || put(date,year4.);
  put (' z '); %ps_show;
  %ps_crlf; %ps_crlf; /* Double space */
/* This loop prints out the column headings */
  do i=1 to 12;
    %ps_tabr(i,mon(i),$.);
  end;
/* Total column is not right justified */
  %ps_tab(13); put '(Total)'; %ps_show;
  %ps_crlf; %ps_crlf; /* Double space */
end;
%ps_font(type=Helvetica,size=9);
put (' site '); %ps_show;
do i=1 to 12;
  %ps_tabr(i,e{i},3.);
  t{i} + e{i};

```

```

end;
%ps_tabr(14,e{13},5.); /* Use tab stop 14, and
right-justify. Looks better this way. */
%ps_crlf; %ps_crlf; /* double space */
t{13} + e{13};
if last.yr then do;
  %ps_crlf; %ps_crlf;
  put '(Monthly totals)';
  %ps_show;
  do i=1 to 12;
    %ps_tabr(i,t{i},3.);
    t{i} = 0;
  end;
  %ps_tabr(14,t{13},5.); %ps_crlf;
  t{13} = 0;
  if not last then do; %ps_page; end;
end;
%ps_term;
run;

```

CLEK: Eligible Patients by Site and Month: Year 1990

	Jan	Feb	Mar	Apr	May	Jun	Jul	Aug	Sep	Oct	Nov	Dec	Total
GUN	.	.	.	.	.	.	.	.	.	.	.	1	1
ILL	3	.	.	.	.	.	.	.	.	.	.	.	3
JUL	.	.	2	3	.	.	.	.	.	.	.	.	5
KPO	5	3	3	3	4	.	.	1	1	4	.	1	25
OSO	2	.	.	1	.	.	6	.	2	.	.	.	11
SCC	2	1	.	.	.	1	.	.	.	2	.	.	6
SUN	11	4	2	1	.	.	.	3	.	.	.	.	21
Monthly totals	23	8	7	8	4	1	6	4	3	6	.	2	72

Program Notes

The above example shows how tables can be generated using the library with ease.. Special attention should be paid to the *Total* column. Instead of being printed inside of the loop, the header was left-justified, and the actual data was printed right-justified at a different tab stop. As noted in the code, this is done to make the printout look nicer. Since the appearance of the output is of primary importance, sometimes it is necessary to fine-tune the typeset layout using little tricks such as this.

Typesetting Isn't Always The Answer

First of all, 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, forgetting to put the END=LAST option on the SET statement, and my personal favorite, forgetting to use the %PS_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 "Corier-Bold" is not a valid PostScript font. Error checking could be built into the library, but there are over 100 PostScript fonts available, and this would take a LOT of processing time. 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. This can cause report columns to 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.

Where does all of this lead? I have shown two of the report programs written using the library, and the resulting output. If the physical presentation of a report is as important as the content of that report, and until SAS/PUBLISH* arrives, the PostScript macro library is a means to that end. As a matter of contrast, a non-PostScript version of the tabular report is shown below. It was printed using the best output quality that could be obtained without using the library, and generated on the same printer as the other reports in this paper. It is for your clients to decide whether the extra effort is worth it. My clients have already made their decision, and the library will continue to be developed and refined to reflect their needs.

CLEK: Eligible Patients by Site and Month: Year 1990

	Jan	Feb	Mar	Apr	May	Jun	Jul	Aug	Sep	Oct	Nov	Dec	Total
GUN	.	.	.	.	.	.	.	.	.	.	.	1	1
ILL	3	.	.	.	.	.	.	.	.	.	.	.	3
JUL	.	.	2	3	.	.	.	.	.	.	.	.	5
KPO	5	3	3	3	4	.	.	1	1	4	.	1	25
OSO	2	.	.	1	.	.	6	.	2	.	.	.	11
SCC	2	1	.	.	.	1	.	.	.	2	.	.	6
SUN	11	4	2	1	.	.	.	3	.	.	.	.	21
Monthly totals	23	8	7	8	4	1	6	4	3	6	.	2	72

This work was supported by NIA/NCNR grant # IU01AG09098-01

Further inquiries welcome:

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

* SAS is a registered trademark of the SAS Institute, Inc., Cary, NC. USA. SAS/PUBLISH is a trademark of the SAS Institute, Inc. 1 - PostScript is a registered trademark of Adobe Systems, Inc.