MULTIPLE PLOTS PER PAGE? GRAPHICS AND TEMPLATE CATALOGS MAKE IT EASY

Peter Gaccione, Harvard University

ABSTRACT

This paper describes the accumulation and display of two system parameters for a typical Unix client-server based system: number of users per system and load per system. The presentation will describe how the raw data gets to the printed output page, in particular noting the creation of SAS/Graph catalogs and the greplaying of their entries. Emphasis will be placed on the setup needed to provide the desired look of multiple plots or graphs on one page of output.

INTRODUCTION

All computer operating systems provide utilities for acquiring, storing, and presenting data on system usage and performance. Frequently these measurements are compiled and tabulated so that only cumulative amounts are available for inspection and analysis. Many times administrators want to see the original data, inspect it for individual time points, match it with other system data, and watch the fluctuations of certain parameters over time. The SAS System's multitude of analytical and graphical capabilities makes it the perfect tool for manipulating, describing, and presenting these data.

Items in bold are Unix system commands, items in italics are DOS file or directory names, and underlined items are key SAS words or variable names from a SAS program. The data are gathered from Digital Equipment Corporation (DEC) DECSystem 5000 and Sun Sparc 1 workstations.

PROCESSING PROGRAMS

The SAS programs which will be discussed are those which process and graph output of the uptime Unix command. Other data included in the raw, system produced data, is from the ps -aux command (not analyzed or presented here). The program to read the file into a SAS data set is shown in Figure 1. The log of the program which creates the graphics template catalog and the entry for the user data is shown in Figure 2. Figure 3 shows the macro coded SAS program for creating, storing, and plotting graphs of the system information over time during a workday.

Figure 4 is an example of the graphics output.

Referring to Figure 1, the processing to read in the ascii data is set up within the macro psread, which has the parameter date. The program reads the ascii file ps&date.dat, which is where the Unix shell script writes the data captured throughout the day. (The data was subsequently moved to the Windows platform, upon which this work was performed. The script, not shown, automatically names the file for the day the statistics are being taken). It is comprised of blocks of 23 sets of data. The first part is the output from uptime, and the next 22 pieces are the ps-aux output from the 22 workstations of interest.

The variables of interest for this application are numusers (number of users on a particular machine) and load1 (average number of jobs in the run queue in the last one minute). Separate input statements exist for the DEC and Sun machines because the ascii output from the commands is slightly different. For possible future reference all data fields are read in, and certain care is taken to get all of the last field, the names of the processes themselves. It is of varying length and requires the use of the length option on the infile command. Hand calculations were done to determine approximately the time required for one sequence of data accumulation (about 6 minutes), after which the variable runtime, the number of minutes since 8 AM, was created. The character variable daytime is then created for plotting purposes, i.e., to show the times of the day on the horizontal axis of the plots. A SAS data set named ps&date.sd2 results.

The variables retained for the descriptive analysis are machine, numusers, and load1 from the output of the uptime command. Many of the daily data sets have more than 150,000 observations so it is important to shorten the lengths of variables, drop unnecessary ones, and compress files when processing is done.

GRAPHICS

Extensive use of template and graphics catalogs, set up by macros, allows for continuous storage and access to graphical data. The first task is to create the template catalog and its entries, so that PROC GREPLAY will have something into which to replay graphics catalog entries. The log of this
program (Figure 2) shows how the template entry user is created. The template catalog system.sct02 is created in the directory c:sugi21\catalog, and the template defined (by the includ option) is called user: It sets up a "page" of graphics output which will show 22 workstation graphs, and one title. This overall title must be put on the page also, and the easiest way to avoid its appearance over each graph instead of on the top of the page is to make it part of the template entry. The by...by keywords describe where on the page each panel will be placed. The numbers act like coordinates in the (x,y) plane, where a full page is 100 by 100. Note that the output from PROC GSLIDE, the title, will occupy the 23rd panel, though its delineation is the whole page. SAS will fit things correctly, depending on the text and character height definitions of the coder. This program is run independently, at the beginning of all data collection.

The program in Figure 3 creates the graphics files. There are three macros in the programs, two called from within a third called sortoral. This macro takes the parameters varname (numusers or load1 for the runtime data), type (user or load, which defines the template entry of system.sct2 to use), day, and data. The data set to be processed is sorted, the desired runtime data (where machine is not equal to missing) is chosen, and plots &vname by daytime are made. This occurs twice, first for the DEC machines and then for the Suns.

Two separate runs are done here in case one wishes only to view data from one type of workstation. This setup also shows how the psout option in PROC GGPLOT sends graphics output to a catalog. For the program in Figure 3 there are 11 DEC machines, thus by SAS system default the first 11 entries in graphics catalog user.sct02 (which is different from template entry user in template catalog system.sct2) are gplot, gplot1,..., gplot10. When the second PROC GGPLOT sequence runs for the Sun machines the entries become gplot11-gplot21. These are permanently saved to disk, as the two level SAS name has been used: &type as the libref referring to a directory (c:sugi21\catalog\user, e.g.) where the graphics catalog &data.sct2 (e.g., nov20.sct2) resides.

To make the output compatible with the desired output device and to create the proper print file macro plotarm is run. The postscript file, which is the final output, is defined by the filename statement. Owing to DOS restrictions the firstlet macro variable takes the first 3 letters of user or load, with the final 5 characters of the .ps filename holding the date. Various options statements massage the graphics output. Finally macro greplay is run, which renames the entries in the graphics catalog with the appropriate machine prefix and current date, and sends the output to the file to be printed. During this procedure the graphs have been set up in the desired template by the treplay command.

Figure 4 shows a page of 22 graphs depicting the number of users on each of the workstations during Monday, November 20. This is the result of running macro sortoral with the four parameters numusers, user, Mon, and nov20. We note the most activity in terms of the highest number of users on the server sda1. A quick glance at this type of output can give a system manager the needed information about overloaded, underutilized, or misbehaving computers in the network.

CONCLUSION

Some operating system caveats are in order. Programs and output for this presentation were done on a 133 MHz Pentium with 16 MB RAM and a 1.6 gigabyte hard disk, using SAS 6.10 for Windows (3.11). Care needs to be taken to adjust to DOS and SAS naming conventions when considering associations for macro variables (in particular, upper versus lower case, and eight character file and SAS variable/data set names). Repeated processing of the program in Figure 4 appends entries into the graphics catalogs, so that greplaying them as specified could cause problems and errors. A Unix platform, for example, allows for upper and lower case characters in filenames, and they can be longer. But the SAS restrictions still apply.

The SAS System provides the unique feature of enabling saved graphics as catalog entries files to be replayed at any time, modified in any way. Creation of the analogous entry/catalog system for templates directing how this output should be displayed increases the presentation choices exponentially. Once the SAS data sets are created the powerful graphical and statistical tools of the SAS system can be used to perform in-depth analysis in addition to presentation. In this application the results can be used in a comprehensive managing plan to keep a computer system running smoothly, quickly, and efficiently.
FIGURE 1

Posters
FIGURE 3

```plaintext
libname load "c:\sugi21\catalog\load\catalogs"; /*directory holding load catalogs*/
libname user "c:\sugi21\catalog\users"; /*directory holding user catalogs*/
libname system "c:\sugi21\catalog\system"; /*directory holding template catalogs*/
libname a "c:\sugi21\catalog\load\datasets"; /*directory holding SAS data sets*/

options prinfo:
   / * THE MACRO VARIABLES ARE:*
   varname - variable to be graphed (load or user info)
   type - type of graphs desired (load or user)
   date - date of information desired
   day - day of the week.
   */

************** SORTS AND GRAPHS ***************;
\macro sortgraf (varname, type, day, date);
\let firstlet=\substr(type,1,3); /*for filename fileref and postscript output */
plotparm(type, day, date, \firstlet);
   / * Title is graphics output to catalog \$(\&date) in user or load directory */
   / * By default the entry is called \$(\&date). There will be 23 entries in this */
   / * catalog; this one named \$(\&date) and the 22 plots */
proc galide gout=type..\&date;
title \&date pct "type state = \&date ";
   / * PROC gplot data by type date;*/
   / * and puts them into a graph consisting of 22 plots on a page.*/
*************** Replaying The Graphs ***************;
\macro replay(graph, date);
run;
\tand sortgraf;
   / * This graphly takes the graphics entries out of catalog type..\&date */
   / * and puts them into a graph consisting of 23 plots on a page. */
*************** Plotting Parameters ***************;
\macro plotparm (type, day, date, \firstlet);
filename \&data\&firstlet='\$(\&date)'; /*DDS file names of chars limited naming */
gplot1\&firstlet=\substr(type,1,3); /*for filename fileref and postscript output */
plotparm(type, day, date, \firstlet);
   / * Title is graphics output to catalog \$(\&date) in user or load directory */
   / * By default the entry is called \$(\&date). There will be 23 entries in this */
   / * catalog; this one named \$(\&date) and the 22 plots */
proc galide gout=type..\&date;
title \&date pct "type state = \&date ";
   / * PROC gplot data by type date; */
   / * and puts them into a graph consisting of 22 plots on a page. */
*************** Plotting Parameters ***************;
\macro plotparm (type, day, date, \firstlet);
filename \&data\&firstlet='\$(\&date)'; /*DDS file names of chars limited naming */
gplot1\&firstlet=\substr(type,1,3); /*for filename fileref and postscript output */
plotparm(type, day, date, \firstlet);
   / * Title is graphics output to catalog \$(\&date) in user or load directory */
   / * By default the entry is called \$(\&date). There will be 23 entries in this */
   / * catalog; this one named \$(\&date) and the 22 plots */
proc galide gout=type..\&date;
title \&date pct "type state = \&date ";
   / * PROC gplot data by type date; */
   / * and puts them into a graph consisting of 22 plots on a page. */
```

---

**Notes:**

- The text appears to be a set of SAS code snippets for generating graphics, particularly plots and graphs, using a macro system. The code includes various options for sorting and plotting data, with specific attention to the formatting and output of the graphs.
- The code snippet includes conditional logic for different types of machines (SGI, DEC, etc.) and directories, indicating a focus on data analysis and visualization within a SAS environment.
- The code uses macros for modularity, allowing for easier modification and reusability of the plotting functions.

---

**Visual Representation:**

The text does not seem to include a direct visual representation (such as a diagram or chart) that can be rendered as an image. It is text-based SAS code focused on generating graphs and plots for data analysis.
FIGURE 4

user stats - nov20