

Beyond Proc GLM

A Statistician's Perspective of (some of) The Rest of the SAS System

Clark K. Gaylord
Virginia Tech, Blacksburg, Va.

INTRODUCTION

In my experience using the SAS System™, I have met many statisticians who are very familiar with the statistical procedures and the statistically aligned SAS products, but I am often amazed at how few statisticians are familiar with the data management and application facilities the SAS system has to offer. There often seems to be a strong dichotomy between those who use SAS for statistics and those who use it for data processing and application development. I maintain that some of these data management products may occasionally be of use to us statisticians. In this article, I would like to give an overview of some of the data management oriented SAS products that statisticians may find useful, and to give a brief tutorial of some of these products.

Many statisticians are familiar with products such as SAS/STAT™, SAS/QC™, SAS/OR™, and SAS/IML™. These are some of the quantitative products available, and their utility to statisticians and high quality are generally well recognized. What makes all of these products so useful, are the powerful data analysis tools they make readily available to the analyst. But in order to analyze data, we must be able to access and manage it, so we will discuss two products that address these matters: SAS/CONNECT™ and SAS/ACCESS™. Another important class of

SAS products, the application development products, includes SAS/FSP™, SAS/AF™, and SAS/EIS™. While these products may be of value to certain statisticians, today we will focus on the above data management products.

SAS/CONNECT

The purpose of SAS/CONNECT is to help SAS users maximize their use of networked computer resources. Whether you use Big Iron, Vaxen, workstations, or PCs, you most likely are operating in some type of networked environment. SAS/CONNECT lets you access SAS data sets stored on remote computers or run SAS programs on higher powered CPUs, with the output coming back to your local workstation. While not all SAS products will operate seamlessly over a network in this fashion, it can be a very efficient way to divide the labor in a client-server type of distributed environment.

Client/Server environments

The first concepts to recognize when using SAS/CONNECT are the notions of client and server. At the risk of oversimplifying, let's refer to a "server" as an entity that performs some function at the request of another; the requester in this relationship will be called the "client". One of the difficulties in establishing the terminology is that these roles are often constantly shifting, and our servers are

sometimes clients of their own clients and our clients are sometimes servers of their own servers. Is that clear?

Let's take a couple of examples not directly related to SAS: the Network File System (NFS) designed by Sun and implemented as a common means of sharing file systems, and the X Window System, a method of displaying graphical programs over a network. Both of these protocols are often encountered in environments where Unix-type operating systems are found, though their use is not really operating system dependent.

With NFS, you have one computer "export", or make available, parts of its file system, usually a directory tree. Once this computer has exported its file system, other computers can attempt to "mount" this file system to make it part of their own. The computer that exports the file system is the server of this relationship; the computer that mounts the file system is the client. If I want to access files in my RT file system (running AIX) from my PC (running OS/2), I tell the RT to export /u/gaylord, and I tell my PC to mount RT:/u/gaylord as drive E: on the PC. But in many environments, we wish to share file systems between computers, so we have both machines set up with exported file systems and both machines mount file systems of the other. For example, I could also tell my PC to export d:\data, and the RT to mount PC:d:\data as /u/data. It is not so much that the RT or the PC is a server computer; it is that either can be a server in a given relationship. When we refer to one computer as being "a server" we just mean that for most (or even all) of the networking relationships that computer engages in, it plays the role of a server. Some networking protocols, such as Novell Netware, designate a given computer as "the server" though even there the roles can be somewhat fluid. With

NFS, you're new 10 megadollar (10M\$) Cray can have the disk throughput of an IBM XT -- is this a great time for computing or what?!

We might be tempted to think of our "big" computer as being the server in most situations. The X Window System readily proves this notion wrong. The purpose of X is to allow programs running on one computer to use the display of another computer. This notion is as old as timesharing itself, when we would use a small teletype terminal ("small" meaning we could, with some trouble, fit the thing into our office) to attach to our big central computer ("big" meaning we could, with some trouble, fit the thing into a warehouse roughly the size of Buffalo, New York). But the principle innovation of the X Window System is that you can be simultaneously displaying graphical output from multiple computers on your monitor. But with X, the *programs* request the services of the *display*, not the other way around. So, our PC is running the X server; our big central computer is running an X client program.

Establishing a SAS/CONNECT session

When you are using SAS/CONNECT, you establish a connection to another computer from your current SAS session. We will use this connection to transfer data or programs between SAS on the remote computer and SAS on our local computer. In this case, the "server" is the SAS session running on the computer we are connecting to; the "client" is our current local SAS session.

To establish the connection, we must first know our network protocol. The supported protocols include TCP/IP, APPC, DECNET, NETBIOS, and even a serial connection over a modem. The connection method is stored using the COMAMID= SAS system option. Once you know the connection method, you

locate the associated "script" for connecting to the server. SAS uses this file as SAS filename RLINK. SAS ships with a variety of scripts; these can easily be edited to suite your needs. The sample scripts are often found with the SAS/CONNECT software. On my OS/2 system, for example, SAS' sample scripts are in E:\SAS\CONNECT\SASLINK.

The information needed to actually connect to your SAS server varies from place to place, but it always includes some kind of network address or session name and often includes a username and password. The username and password are used after the actual connection is made, so this information is either written in your script or the script can prompt you for it. The network address or session name is needed in order to make the connection; you tell SAS this information with the REMOTE= SAS system options.

As an example, suppose I want to connect to vtaix using script VTAIX.SCR using a TCP connection. Then, I would use the following segment of SAS code:

```
OPTIONS COMAMID=TCP;
OPTIONS REMOTE=VTAIX;
FILENAME RLINK 'vtaix.scr';
```

We then issue the SIGNON display manager command to initiate our SAS/CONNECT session. We can also enter the COMAMID, REMOTE, and RLINK information in a dialog box, by selecting Locals->Signon from the menus.

One of the problems with this is that when using COMAMID=TCP, we usually have a full internet address, like vtaix.cc.vt.edu, but SAS limits us to eight characters for REMOTE=.

To address this we can either assign a macro variable to the full network address, for example:

```
%let vtaix=vtaix.cc.vt.edu
```

An unusual characteristic about using macros with the REMOTE= system option is that we do not use OPTIONS REMOTE=&VTAIX, as one might think, to evaluate the macro variable. Instead, we use OPTIONS REMOTE=VTAIX, and SAS will read from macro variable vtaix.

The other method for circumventing the eight character limit of REMOTE= is to use any aliasing your network system allows. In the case of TCP/IP suites, for example, there is often a file called "hosts" which allows you to specify a network address to a short name.

Using a SAS/CONNECT session

Once a SAS/CONNECT session is established, you send SAS code to the server SAS session for execution. This code might be self-contained analyses, manipulation of SAS datasets on the server's end, or transferring data between the client and server sessions. Two special procedures are used for transferring data: PROC UPLOAD and PROC DOWNLOAD. You then use the RSUBMIT and ENDRSUBMIT statements to denote the SAS code to send.

Let us suppose that we have a connection to a CMS system already established, and on this system resides a dataset SASDATA.IRIS. We could then perform our MANOVA by submitting the following program on our client computer:

```

rsubmit;
proc glm data=sasdata.iris;
class species;
model sl sw pl pw = species;
manova h=species;
run;
endrsubmit;

```

The great beauty of this is that our mainframe does the work of the analysis, but our PC displays the output and log. This division of labor can especially be useful for large computing tasks. Consider the following SAS/IML program:

```

proc iml;
x = j(1000,100,0);
x = rannor(x);
xpx = x` * x;
invxpx = inv(xpx);
einvxpx = eigval(invxpx)`;
print einvxpx;
quit;

```

On my PC with a 33 MHz Intel 80386 CPU and IIT coprocessor, under OS/2, this took approximately one and one-half minutes; when I RSUBMITTED to an RS/6000 model 550, it took approximately thirty seconds (and about ten seconds of CPU time). In both cases, I get the output in the output window of my PC.

The other principle use of SAS/CONNECT is transferring data between machines using PROC UPLOAD and PROC DOWNLOAD. The syntax of PROC UPLOAD and PROC DOWNLOAD is similar. The most rudimentary form is:

```

proc download data=sasdata.iris
out=mypclib.iris;

```

When RSUBMITTED, this would download the iris dataset from our server to our client. In addition to transferring SAS datasets, SAS catalogs and arbitrary files can be transferred. One could even use PROC DOWNLOAD as an alternative to ftp or kermit or similar general file transfer software.

SAS/ACCESS

Another important SAS offering for accessing data is SAS/ACCESS. This product provides a means of using data from other database software, including dBASE™, Oracle™, DB2™ and others. We will illustrate accessing dBASE files on an OS/2 system. The methods described are similar for other contexts.

Access descriptors

Before SAS/ACCESS can use data from a dBASE file, you have to catalog a description of the dBASE file with SAS. This description file is called an Access descriptor file, and it is stored in a SAS catalog. This descriptor is created by PROC ACCESS with the ACCDESC= option, for example, assuming you have SAS library defined called DBASE:

```

proc access accdesc=dbase.rental
function=c;
run;

```

You run this under the SAS display manager and fill in the information requested, especially the name of the dBASE file. In this case, we enter D:\DATA\RENTALS.DBF.

Once you have defined the dBASE file, you get a list of all the variables in the dBASE file. Note that dBASE allows more than eight character variable names, so ensure the SAS variable names are unique. You then select those variables you want SAS to be able to see in the descriptor. Usually, you will want to select all the variables at this point.

View descriptors

After defining the access descriptor, you create a SAS view to the dBASE file. This view can be accessed as if it were a SAS data set. In particular, any SAS procedure can refer to it with the DATA= option as if it were a SAS

data set (though it cannot be overwritten with the OUT= option).

To define the view, enter the following in a display manager:

```
proc access accdesc=dbase.rental;
```

You are then presented with another window similar to the access descriptor window; the most notable difference is that it is called the view descriptor window. In this window you select those variables from the access descriptor that you want to have available in the particular view.

It is also possible with some versions of the SAS System (e.g., Version 6.08 for Windows or OS/2 starting with TS407) to enter all the information for Access and View descriptors in batch mode. Also, these updates include the DBF procedure, a particularly easy way to access dBASE files. Using PROC ACCESS, our example above would then become:

```
proc access dbms=dbf;
  create dbase.rental.access;
  path='d:\data\rental.dbf';
  create dbase.rental.view;
  select all;
proc access viewdesc=dbase.rental
  out=sasuser.arc_data;
run;
```

Combining ACCESS and CONNECT

While the dBASE interface may be the most important for many statisticians, some others may find themselves occasionally handling Oracle or DB2 or some other popular data format. By combining SAS/ACCESS and SAS/CONNECT, these database files can be accessed easily on the server where they reside with the output from analyses being delivered to the statistician in their local SAS session. Since many of these database management systems lack the powerful statistical tools of

the SAS system, this provides an important vehicle for analyzing data from these systems.

REFERENCES

SAS Institute Inc. (1991), *SAS/ACCESS Interface to OS/2 Database Products: Usage and Reference, Version 6, First Edition*.

SAS Institute Inc. (1990), *SAS/CONNECT Software: Usage and Reference, Version 6, First Edition*.

SAS, SAS/ACCESS, SAS/AF, SAS/CONNECT, SAS/EIS, SAS/FSP, SAS/IML, SAS/OR, SAS/QC and SAS/STAT are registered trademarks or trademarks of SAS Institute Inc. OS/2 and DB2 are registered trademarks or trademarks of International Business Machines Corporation. Other brand and product names are registered trademarks or trademarks of their respective companies.