INTRODUCTION

I needed the power of a mainframe and the flexibility of a PC. With Version 6 SAS® software, I was able to get both.

I consult for a major electric and gas utility company. Last Fall, I was assigned to collect and display usage data on roughly 100 of our largest customers. For each customer they requested twelve monthly plots of load data, with the monthly plots scaled to the customer’s yearly maximum load. Since this project was last done three years ago, I expected several rounds of revisions, so I designed the process to be flexible rather than efficient.

I had to crunch 175 tracks of data, and I had to produce high resolution plots quickly. I needed a mainframe to process 875,000 observations, and a PC to quickly deliver 300 DPI plots. I was able to get both with Version 6 SAS software.

OVERVIEW OF THE PROBLEMS

There were three problem areas: generic data processing, SAS data processing, and cross-platform communications. The communications problems were the worst.

The generic data processing problem was identifying the relevant data.

The SAS problems were how to scale each monthly plot to the yearly maximum load, how to select the data for each plot, and how to name the plots for later replay. Indicating weekdays and weekends was also a bit of a problem.

The communications problems seem simple in hindsight, just as opening a safe seems simple once you figure out the combination.

DETAILS AND SOLUTIONS

Identifying the data was difficult because the analyst had three partially overlapping lists of the customers he wanted us to plot. None of the lists included the channel number under which we store usage data. Instead they had the control number under which we store billing data. I did not want to use these seven-digit control numbers in any of the code, especially since our plotters tend to eat the occasional plot and I expected to reprint several plots, so I decided to refer to the customers by their observation number in the file which would eventually contain all of them. (This proved a poor decision when we added five more customers after I started plotting, but since the analyst changed the plot specifications after that, it was a small loss.) I will return to these identification numbers when I discuss the plotting macros.

Customers differed enough from each other that a single vertical axis for all would reduce some plots to a few bumps on the horizontal axis. The plots differed enough across a year that automatic scaling of each month would conceal significant yearly patterns. I used PROC MEANS with a BY statement to get the yearly maximums by customer, and a series of macro variables (described later in this paper) to use the maximums in the HAXIS statement.

Developing a solution

I selected the data for each plot using a WHERE clause. The values were supplied in macro variables (explained below). There were some problems here which I never resolved. Even though the file was sorted by and indexed on the two variables in the WHERE clause, SAS read the entire file for each PROC GPLOT. I tried to figure out why, then I set CPU time to 300 minutes and got the job done.

I generated the plots as device-independent graphics in a graphics catalog by coding GOUT=DBN.GPLOTS in my PROC GPLOT statement. When you put graphics into a graphics catalog like this, you give each graphic a name (with the NAME option on the PLOT statement) or you accept the defaults.

There is a trick here which cost me dearly at first. If you give a new graphic an existing name, SAS will change the new name. If you generate eight test plots, modify the code, and regenerate the plots, the original names still refer to the original plots, and the new plots get default names. I never found a way to use wildcards in deletions. Since the graphics catalog was specific to this project, I simply PROC DELETE'd it each time I changed the plotting code.

My poor naming convention created another problem. I named each plot with the customer number I had assigned (from 1 to 112), followed by the month number (from 1 to 12). Later I tried to decide if graphic 111 was customer 11 month 1, or customer 111 month 1. Since customer 11 month 1 attempted to reuse an existing name, the graphic received a default name and got lost. I tried some fairly ugly things to force two digit month codes, then settled on putting a ‘C’ before the customer number and an ‘M’ before the month. Thus C11M1 is distinct from C11M1.

The next two pages present the SAS code, including a macro definition and invocation of that macro, which created the plots on the mainframe. Most of the macro code is straightforward, but a few points will benefit from clarification. Comments appear opposite relevant sections of code.
ANALYSIS OF THE MACRO

The **GOPTIONS** statement serves two purposes. The mainframe session had no capability to display graphics, and the **NODISPLAY** suppressed them. The **DEVICE=HPLJ300** specifies on which device my device-independent graphics should be dependent. This threw me several times. I knew I wanted my graphics device-independent, and I believed I was creating device-independent graphics, but if I did not specify a device, the jobs failed. SAS even verifies that the device is valid before ignoring it, so that when I coded **DEVICE=FAKE** the jobs bombed.

PROC MEANS finds the yearly maximum for each customer. **DBN** is our traditional DDNAME for a SAS library. Our batch PROC assigns it to the library we specify. **DBN.PLOTLIST** is the data to be plotted. The dataset contains the following variables:
- **CNTL** ID variable
- **ESCHED** descriptive variable
- **NAME** descriptive variable
- **DATETIME** series of SAS DATETIME values created with DHMS function
- **MONTH** selects individual months for plotting
- **LOAD** usage variable—what the analyst wanted to see.

The **null datastep** serves several purposes:
- Eliminates a few customers whose usage data is zero or missing.
- Flags a few customers with very high usage, so we can review them for accidental spikes.
- Creates the N macros. These are N1 to N112, and they contain the customer ID’s. These variables can translate a do loop from 1 to 112 into a do loop from the first ID to the last ID. The step also creates the C macros. These are C&N1 - C&N112, e.g. C00000004 or C7654321. These variables, one for each customer, contain the yearly maximum loads.
- Creates an extra N macro variable whose number is one higher than the last normal variable and whose value is "FINISHED". A macro which steps through the N macro variables stops when the current N variable resolves to "FINISHED". An early version of the macros omitted this, and it failed at the end with an unresolved macro variable message. It generated all of the plots, but it put an error message in the job log.

The macro CREATE generates the PROC GPLOT steps. By default, it generates plots for the first 9999 customers, and it plots month 1 to month 12. During testing we often coded values for fm and 1m. Their only use during production would be recreating lost or damaged plots. The macro variable DAYS is the number of seconds in a day. I did not want to hardcode the constant, and since it is defined outside the outermost do loop, it is evaluated only once. It is used to put daily tick marks on the horizontal axis. The first do loop will loop &cust from 1 to 113. When &N&CUST is evaluated for the 85th time, suppose macro variable N85 has value 27532. Then &N&CUST will resolve to &N85, and that will resolve to 27532. The point is, the double ampersand resolves into a single ampersand and a request for a second scan. The N is left alone but it is now preceded by a single ampersand, and the &CUST resolves into a number which combines with the N and the generated ampersand to form a macro variable reference (&N85). The second do loop, for month, is cleaner. Do not attempt to force two digit months by changing months less than ten to 0&month. The value will change, the loop will fail. &N&CUST is evaluated outside the month loop, to check for the flag value "FINISHED". It is evaluated inside the month loop, to put the control number (our ID variable) in the WHERE clause.

The HREF statements are rather crude. They draw vertical reference lines (references to the horizontal axis, hence the name) between Friday and Saturday, and between Sunday and Monday. The analyst wanted them to make it easy to detect weekday/weekend consumption patterns. I could have generated them with code which would work in any year, but it was not worth the time for a program which will not run again for three years, if then.

The VAXIS statement scales the monthly plots to the yearly maximum load for the customer. Assume &CUST is 85, &N85 is 27532 (the control number for the 85th customer), and &C27532 is 456789 (the yearly maximum usage for the 85th customer). Then on the first pass &&&C&&N&CUST resolves to &C&&N85, which resolves on the second pass to &C27532, which resolves on the second pass to 456789, the maximum load. The default BY value is one, which caused me a few errors at first. SAS could not fit 456789 tick marks on the vertical axis.

### Cust macro
<table>
<thead>
<tr>
<th>Cust number</th>
<th>CUST</th>
<th>(85)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Obs number</td>
<td>85</td>
<td></td>
</tr>
<tr>
<td>N macro</td>
<td>N85</td>
<td>(27532)</td>
</tr>
<tr>
<td>Our ID number</td>
<td>27532</td>
<td></td>
</tr>
<tr>
<td>C macro</td>
<td>C27532</td>
<td>(456789)</td>
</tr>
<tr>
<td>Customer's maximum load</td>
<td>456789</td>
<td></td>
</tr>
</tbody>
</table>
To this point, everything I have discussed ran in MVS batch on an IBM® 3090-400 under SAS 6.06. The input dataset was in a SAS data library on disk, and the output was written to the same library. At this point, things get trickier. Our mainframe plotter is awkward to use, and it is in another city, with two distribution runs each day. To get better-looking output, and to get faster turnaround during development, I wanted to plot locally.

I decided to connect PC and mainframe SAS, and route the graphs to a network-attached HP LaserJet® printer. Eventually I am going to descend into techie bit twisting for the benefit of others who may pioneer this route. For now, please assume that my PC and the mainframe are both running SAS, that they are communicating, and that I can select which one will execute a program or command I select. Then the following macro is remote submitted to run on the mainframe:

```sas
* macro does not assume a number of customers.
* the customers, using the obs numbers from the
* PROC MEANS. Fro and 1m are the first and last
* months to print (normally 1 and 12). The
* macro does not assume a number of customers.
* Instead it checks for a fake control number
* macro ('FINISHED') as its clue to quit.
* This method made more sense in the plotter
* macro, but works here.
* * * * * * * * * * * * * * * * * * * * * * *
%macro replay(first=1,last=9999, fm=1, lm=12);
  * The replay macro generates the plots on the
  * local output device. First and last select
  * the customers, using the obs numbers from the
  * PROC MEANS. Fm and lm are the first and last
  * months to print (normally 1 and 12). The
  * macro does not assume a number of customers.
  * Instead it checks for a fake control number
  * macro ('FINISHED') as its clue to quit.
  * This method made more sense in the plotter
  * macro, but works here.
  * * * * * * * * * * * * * * * * * * * * * * *
  %do cust = &first %to &last;
    %if &&n&cust = FINISHED %then %goto nomore;
    %do month = &fm %to &lm;
      PROC GREPLAY NOFS IOUT=REN.GPLOTS;
      %let cust = %j
      %let month = %j
      &cust.M&month:
      REPLAY C&cust.M&month;
    %end;
  %end; %* of cust;
  %end; %* of month;
%end;
%replay (first=75, last=75, fItFl.lITFl):
```

The local (PC) option DEVICE=VGA or HPLJ300 directed output to my VGA monitor for preview or to a LaserJet on (by default) LPTI for hardcopy. SAS wrote to the printer port through the BIOS like a good program should, so our local area network software could capture the output and direct it over the network to any printer I chose. Here comes the techie bit twisting I promised. I must warn you that much of this is specific to our hardware and software environment. I will try to point out what is unique and what is general.

My workstation for this project was a 25 megaHertz 386 running MS-DOS 3.30, and the Quarterdeck memory manager QEMM386 version 5.11. Both the 386 and QEMM are important. They allowed me to load 125K of Banyan® network drivers into high memory, leaving my conventional memory for PC-SAS. This is not important if your network drivers are smaller, or if you have a LaserJet printer physically attached to your PC.

I have an ancient Forte® terminal emulation board running PC789 version 3.02. The Forte card is important because SAS has custom drivers for it. My PC-SAS software is version 6.04, and our mainframe is running 6.06.0t. We do not have the SAS/CONNECT™ software. I did not need access to our mainframe SAS c-list, because our systems programmer wrote it to accept SAS invocation parameters in the c-list invocation. The sample c-list distributed by the SAS Institute also supports such options.

My PC-SAS files had a few quirks. During development, I coded -VERBOSE ON in CONFIG.SAS to get extensive progress messages as SAS loaded. I also included -EMS ALL to take advantage of the expanded memory provided by my DOS memory manager (QEMM386). I am not sure this bought me much, but I expected memory space problems, so I included this.

My AUTOEXEC.SAS had two lines which were specific to my environment. You will need equivalents, but they may be different. These were OPTION REMOTE=Forte, which describes the connection between the PC and the mainframe, and FILENAME RLINK 'IASROOT\SASLINK \LOGGEDON.SCR', which associates the default fileref for SIGNON and SIGNOFF with my TSO script. Valid values for REMOTE= on the PC include ASYNCh for comm port n, IRMA®, and CXITM®, among others.

The script invoked by SIGNON and SIGNOFF can be a powerful tool. In theory, you can script the entire mainframe logon, including prompting the user for their ID and password. For simplicity, I logged on outside PC-SAS, then exited to DOS leaving the terminal emulator resident. I still needed the script to invoke SAS on the mainframe. Our mainframe c-list is "RATESAS" and it supports an options specification, so I coded "RATESAS OPTIONS ("REMOTE=PCLink")." This allowed mainframe SAS to direct all output to the terminal emulation board in a form PC-SAS could intercept and process. The process of linking the PC and the mainframe, starting from turning on the PC was:

1) Invoke terminal emulation, log onto TSO, make terminal emulator resident and exit to DOS without logging off.
2) Invoke PC-SAS.
3) Enter SIGNON at the PC-SAS command line. The SIGNON command looks for the default fileref of RLINK, which I had associated with my TSO script in my AUTOEXEC.

Again, this can script the entire logon procedure. If your terminal emulator is too large for PC-SAS to load while the emulator is resident, you may have to script the entire
process. This is difficult at our site, as VTAM repeatedly leaves the terminal in an indeterminate state, and the message of the day often exceeds a full screen.

With PC-SAS running on the PC, and mainframe SAS running on the mainframe, and the two REMOTE= options to connect the two sessions, you are ready to run on either or both platforms. You can key SAS code into the PC program window, and enter SUB or RSUB in the command line to run that code locally (SUB) or remotely on the host (RSUB).

CONCLUSION

I found the combination of mainframe power and capacity with PC graphic resolution and local availability very useful. I cannot say it was easy. It is difficult to keep straight which options and commands apply to the mainframe and which apply to the PC. I did not mention what happens when the console operator broadcasts a message in the middle of your plot. There are too many possible combinations of hardware and software for the documentation to discuss the details of each configuration. It took four days and several calls to the Institute to get everything working together. Still, once it worked, it produced very good-looking graphs, and it produced them quickly. If you have enough time to develop tools, enough data to choke a PC, and never enough time to fine-tune graphics, this is the way to go.

SAS and SAS/CONNECT software are registered trademarks or trademarks of SAS Institute Inc. in the USA and other countries. ® indicates USA registration.

IBM is a registered trademark of International Business Machines Corporation.

Other brand and product names are registered trademarks or trademarks of their respective companies.