ABSTRACT

The SAS® Companion for the CMS Operating System (1986 Edition) states on page 1, "Generally speaking, SAS programs and their results are the same, regardless of the host operating system." The purpose of this paper is to determine the extent to which SAS Institute is generally speaking.

In the Department of Psychiatry at Washington University, we have developed a large SAS program of over 1,600 statements which scores a standardized psychiatric interview and creates a data set of over 1,700 variables. Other institutions often wish to run this scoring program on machines other than an IBM mainframe. In addition to aiding such users of our own program, it is hoped that this exercise of running large SAS code under different environments will be of more general help to those wanting to write SAS code that is independent of the host operating system.

HISTORICAL DEVELOPMENT OF SCORING PROGRAM

Our scoring program was written with the 1979 edition of SAS. At that time, the SAS User's Guide (1979 Edition) indicated that "SAS runs only on IBM 360/370 computers (and plug-compatible machines such as Amdahl, Itel, CDC Omega, Magnuson, Ryad, etc.) under OS or OS/VS (p.3)." As a result, there was no thought given to writing the program to run on a variety of machines with different operating systems.

The scoring program was designed in sections, since it is possible to limit the interview to selected sections. This was achieved by storing each section of the scoring program as a SAS MACRO. These refer to the old-style MACRO statements which begin with the word MACRO and end with a percent sign. These MACRO statements should not be confused with the MACRO language which was not available in the 1979 release.

We copied these MACROs to tape and then instructed users to concatenate this file to the front of the SYSIN file in their JCL. This was the method for simulating a MACRO library recommended in the 1979 Manual (p.12).

Another feature of SAS79 that later proved important relates to the ARRAY statement. The current Version 5 Manual distinguishes between explicitly and implicitly subscripted ARRAYS. In the 1979 edition there was no such distinction. The only type available in this earlier manual is now described as implicitly subscripted.

After our scoring program was initially developed, it was scheduled to be used in a large scale multi-centered research project. Each of the five centers in the project extensively reviewed and critiqued the code over the course of a year. After this review process was complete, the sentiment was to freeze the code. In this way other users could use the same exact program and compare their findings to those in this large scale project. Such standardization would greatly facilitate cross-study comparisons in a field plagued by inconsistent definitions and widely variant measures.

It was only after SAS Institute greatly expanded the types of machines and operating systems under which SAS software could run that we had to think of how the program could be used in these different environments. It was clear that parts of the program would have to change since some of its conventions are not supported under all operating systems. It was not clear, however, whether all operating systems could handle a program this size, whether performance would be acceptable, and whether the same results could be guaranteed.

TEST RESULTS

Whenever we distribute our scoring program, we include on the tape a file of 34 test cases stored according to the input format specified by the deck and column numbers contained in the interview. Along with the scoring MACROS, there is also a MACRO for the INPUT statement for the test data. Users are instructed to read the test cases with the INPUT MACRO, run the data set through the scoring MACROS, and do a FREQ and PRINT of forty-four variables created by the program. The output of these procedures should match the output we enclose with the tape. For the initial test we attempted to match these test frequencies across the four operating systems.

As an additional test, we ran the program with 3,400 observations to determine how performance under the operating systems would compare with a heavier load. These 3,400 observations were obtained by outputting each of the original cases one hundred times and passing this larger data set to the scoring program. The same PROC FREQ was then performed on the larger data set, but the PROC PRINT was omitted to avoid a lengthy listing.

Table 1 displays the four operating systems used in the testing, the versions of those operating systems, the hardware which was used, and the mode of SAS execution. The test results are described below.
I. AS BATCH
Since the scoring program was written to run under AS, it ran without complaint on an IBM 4341-2. A MACRO library was simulated by concatenating them to the front of the JCL SYSIN file, and the MACROS were invoked after the DATA and INFILE statements. The DATA step read in the 34 cases and created 1,743 variables - 1,074 variables coming from the interview’s INPUT statement and another 669 variables being created from the scoring program. This DATA statement took 52 seconds and used 868K. The PROC FREQ of the selected forty-four variables took 3 seconds and used 1140K. The PROC PRINT of these same variables also took 3 seconds and 996K.

The data steps necessary to create and score the 3,400 observations took a total of 5 minutes and 47 seconds. The PROC FREQ of this larger data set consumed 20 seconds. A comparison of these CPU times is displayed in Table 2.

II. CMS
The test on the CMS operating system was also run on an IBM-4341 mainframe. Perhaps it is not surprising then that only one modification was needed when changing to this environment. Since there is no JCL in CMS, the MACROS had to be identified with CMS FILEDEFs which could be %INCLUDEd into the job stream. Once this difference in referencing external files was made, the program ran identically to the OS version. The DATA statement took 48 seconds and used 916K. The PROC FREQ used 3 seconds and 1492K. The PROC PRINT used 2 seconds and used 1492K.

The data steps necessary to create and score the 3,400 observations took a total of 4 minutes and 42 seconds. The PROC FREQ of this larger data set consumed 22 seconds. A comparison of these CPU times is displayed in Table 3.

III. VMS™
The story becomes more complicated when switching operating systems to VMS and switching hardware to a VAX™ 11/785. Version 5.16 of SAS under VMS supports neither the old-style MACRO statement nor the MACRO language. Thus, a different approach was needed to make the INPUT statement and scoring statements available to the SAS job.

A. SYNTAX IN THE INPUT STATEMENT
The original INPUT statement made use of the n* modifier which specifies in format lists that the next format is to be repeated n times (p.137). The footnotes to the INPUT Statement section of the User’s Manual, however, indicate that the n* modifier is not available for AOS/VS, PRIMOS, and VMS. The original statement contained expressions such as (X Y Z) (@1 3*1.). These had to be changed to read @1 (X Y Z) (3.). There were 66 such occurrences in the INPUT statement.

B. IMPLICITLY SUBSCRIPTED ARRAYS
As mentioned previously, the only type of ARRAY statement available at the writing of the program is now described as implicitly subscripted. The original program contained the following type of construction:

```sas
ARRAY Z XI-X9;
DO OVER Z;
IF Z...1 THEN Y+l;
END;
```

Only explicitly subscripted ARRAYS can be processed under VMS. Thus, the above constructions had to be changed to:

```sas
ARRAY Z[*] XI-X9;
DO I=1 TO DIM(Z);
IF Z(I)=1 THEN Y+1;
END;
```

The program contained 39 ARRAY statements and 31 DO OVER statements which had to be changed in this manner.
C. SIZE OF THE PROGRAM

Once these syntax changes were made, the program was submitted in one data step as in the OS and CMS examples. The first such attempt resulted in an error message which read NOT ENOUGH MEMORY IN ROUTINE JSINST. DATA STEP MAY BE TOO LONG FOR SAS TO HANDLE. The VAX system manager increased my page file quota from 4MB to 8MB which doubled the size of the virtual machine available to me. This eliminated the error message of not enough memory, but another error message JUMP > 32767 GENERATED, PROGRAM TOO LARGE was issued.

We contacted SAS Institute about this error message. It turns out that the counter for the SAS compiler is a single precision number and reached its limit of 32,767. The program was indeed too large. The only recourse left was to divide the program into multiple data steps.

Since the program is written in sections, this division into multiple data steps was relatively straightforward. The first step simply read the data with the INPUT statement. Let's call this DATA ONE. The scoring program itself was then divided into three data steps by placing the statement DATA ONE; SET ONE; at logical break points. Such a strategy handled the problem that later sections of the program frequently depend on values generated in earlier sections.

These four data steps took a total of 6 minutes and 56 seconds of CPU while generating 47,817 pagefaults. The PROC FREQ used 28 seconds and 1,539 pagefaults. The PROC PRINT used 27 seconds and 1,986 pagefaults. The output produced by these procedures was identical to that from OS and CMS.

The data steps necessary to create and score the 3,400 observations took a total of 31 minutes and 21 seconds. The PROC FREQ ran in 28 seconds, the PROC PRINT took 31 seconds.

The data steps necessary to create and score the 3,400 observations took a total of 31 minutes and 21 seconds. The PROC FREQ of this larger data set consumed 13 minutes and 37 seconds. A comparison of these CPU times is displayed in Table 5.

IV. PC-DOS

Some of the changes in syntax necessary for the SAS System for Personal Computers (PC-SAS) resembled those needed for VMS. Although the n* modifier in the INPUT statement is allowed under PC-SAS, implicitly subscripted ARRAYS are not. Also, the old-style MACROS are not supported under PC-SAS, but parts of the MACRO language are. Thus, the MACRO-% statements could be changed to %MACRO;%MEND; constructions.

The real roadblock for PC-SAS, however, is the size of the program and the number of variables in the data set being created. Even when one limits the first data step to inputting the data, PC-SAS issues the error message UNABLE TO CREATE VARIABLES FOR DATASET. It is not clear how many variables and observations PC-SAS can handle operating with an IBM-AT with S12K, but 34 observations with 1,074 variables still exceeds the limit.

At this point, it became necessary to input only those variables needed for one section of the program and to score the data in this modular fashion. Each section could then be stored as a separate SAS data set. However, even this approach had to be broken down further when the number of variables in a given section grew too large or the number of programming statements were too many for the available memory. The largest data set we were able to create contained 361 variables. In all it took 49 data steps to run the program, and the data were stored in 24 separate data sets. The data steps took a total of 31 minutes and 21 seconds. The PROC FREQ ran in 28 seconds, and the PROC PRINT took 31 seconds.

The data steps necessary to create and score the 3,400 observations took a total of 6 hours 32 minutes and 29 seconds. The PROC FREQ of this larger data set consumed 13 minutes and 37 seconds. A comparison of these CPU times is displayed in Table 5.

COMPARISONS ACROSS OPERATING SYSTEMS

It is interesting to compare the relative performance of each of the operating systems when we move from scoring 34 observations to scoring 3,400 observations. Graph 1 shows that change in performance varied from the data steps to the procedure step. CMS showed the smallest relative CPU increase for the data steps when scoring the larger data set. For the PROC FREQ, however, it was VMS which showed the smallest relative CPU increase. Thus, the relative performance of the operating systems when processing larger data sets can vary widely depending on the application.

Consequently, performance comparisons across operating systems will depend upon the type of step (data vs procedure) and the number of observations being processed. Table 6 displays the ratio of CPU times to OS batch when scoring 34 observations. The times for CMS are closely comparable for the data step and the two procedures. VMS takes eight times as long as OS batch to process the data steps, but this greatly outperforms PC-DOS which takes over 36 times as long as OS batch. Of course, the difference undoubtedly relates to the fact that VMS can handle the program in four data steps, while PC-DOS needs 49 data steps. This VMS advantage over PC-DOS diminishes or disappears when using PROC FREQ and PROC PRINT with the 34 observations.
Table 7 shows these same CPU ratio comparisons when processing 3,400 observations. Here again, the CPU times required on CMS are very similar to those on OS. With these larger number of observations, however, PC-DOS shows a large degradation in performance and no longer rivals VMS in the procedure step.

LESSONS LEARNED

Despite the differences found across the operating systems in comparing 34 vs 3,400 observations and data vs procedure steps, there are general lessons that can be learned. When moving from one system to another, there are two considerations which will predominate: compatibility and performance. The lesson learned in each of these areas is described below.

1. COMPATIBILITY: IT'S ALL IN THE MANUALS - ALMOST

The differences in syntax across operating systems are well enough documented that the error messages produced are adequate guides to what needs adjusting. In fact, if this scoring program was to be written today, the User's Guide provides enough information that the syntax could be designed to be independent of the host operating system.

Of course, determining this type of syntax is not always easy. The section in the User's Guide dealing with the INPUT statement has ten footnotes covering a full page which explain the exceptions under the different operating systems. But this information does allows one to calculate the lowest common denominator.

In one instance, however, the User's Guide provided me with false hope. Chapter 19 on the SAS Macro Language indicates that it is available for all operating systems. A footnote on the first page does indicate that AOS/VS, PRIMOS, and VMS are limited to certain functions. Footnotes at the end of the chapter specify rules relating to specific operating systems, including VMS.

All of this led me to suspect some form of the MACRO language was available under VMS. However, the chapter on SAS System Options indicates that the MACRO option is not available under VMS. My experience verified this. MACRO on the OPTION statement produces a warning that MACRO IS NOT RECOGNIZED AS A VALID OPTION. A %MACRO statement produces the error message STATEMENT IS NOT VALID OR IT IS USED OUT OF PROPER ORDER.

More importantly, the User's Guide did not provide any guidance when the program proved too large for VMS. The Guide states that a VMS data set can contain 2,147,483,648 observations and the observations can be any length (p.586). The Guide is silent, however, on the size of the program that any operating system can handle. The manual for the SAS System for Personal Computers was also uninformative concerning the size of programs and data sets that it can handle.

2. PERFORMANCE: BIGGER MACHINES CAN HANDLE MORE THAN SMALLER MACHINES

Perhaps this is an obvious lesson, but it is one that should be kept in mind when running large SAS code. Operating systems on an IBM mainframe could handle this scoring program in one step and use under 1 minute of CPU. The operating system on a VAX mini-computer needed four data steps to do the same job and took eight times the amount of CPU. An IBM-AT was unable to even input the data set in one step, much less score it.

Thus, when it comes to issues of compatibility and performance, it is performance that is the key concern when assessing SAS software under different operating systems. It does appear generally true that SAS programs and their results are the same, regardless of the host operating system; however, the limitations of an operating system and its hardware can affect the way in which a program is run, the amount of resources it needs, and how the resultant data sets are stored and analyzed.
### TABLE 1
OPERATING SYSTEM INFORMATION

<table>
<thead>
<tr>
<th>SYSTEM</th>
<th>VERSION</th>
<th>HARDWARE</th>
<th>SAS VERSION</th>
<th>MOD OF EXECUTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>OS/VS2</td>
<td>Release 1.7</td>
<td>IBM 4341-2</td>
<td>5.16</td>
<td>Batch</td>
</tr>
<tr>
<td>VM/CMS</td>
<td>Release 3.1</td>
<td>IBM 4341-12</td>
<td>5.16</td>
<td>Non-interactive</td>
</tr>
<tr>
<td>VAX/VMS</td>
<td>V4.4</td>
<td>VAX 11/785</td>
<td>5.16</td>
<td>Batch</td>
</tr>
<tr>
<td>PC-DOS</td>
<td>3.20</td>
<td>IBM PC/AT</td>
<td>6.02</td>
<td>Non-interactive</td>
</tr>
<tr>
<td></td>
<td></td>
<td>512K - 8 mhz</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### TABLE 2
OS/VS2 CPU COMPARISONS
(TIME IN MINUTES:SECONDS)

<table>
<thead>
<tr>
<th></th>
<th>34 OBS</th>
<th>3400 OBS</th>
<th>RATIO</th>
</tr>
</thead>
<tbody>
<tr>
<td>DATA Steps</td>
<td>0:52</td>
<td>5:47</td>
<td>6.7</td>
</tr>
<tr>
<td>PROC FREQ</td>
<td>0:03</td>
<td>0:20</td>
<td>6.7</td>
</tr>
<tr>
<td>PROC PRINT</td>
<td>0:03</td>
<td>--</td>
<td>--*</td>
</tr>
</tbody>
</table>

* PROC PRINT was not run on 3400 observations

### TABLE 3
VM/CMS CPU COMPARISONS
(TIME IN MINUTES:SECONDS)

<table>
<thead>
<tr>
<th></th>
<th>34 OBS</th>
<th>3400 OBS</th>
<th>RATIO</th>
</tr>
</thead>
<tbody>
<tr>
<td>DATA Steps</td>
<td>0:48</td>
<td>4:42</td>
<td>5.9</td>
</tr>
<tr>
<td>PROC FREQ</td>
<td>0:03</td>
<td>0:22</td>
<td>7.3</td>
</tr>
<tr>
<td>PROC PRINT</td>
<td>0:02</td>
<td>--</td>
<td>--*</td>
</tr>
</tbody>
</table>

* PROC PRINT was not run on 3400 observations

### TABLE 4
VAX/VMS CPU COMPARISONS
(TIME IN MINUTES:SECONDS)

<table>
<thead>
<tr>
<th></th>
<th>34 OBS</th>
<th>3400 OBS</th>
<th>RATIO</th>
</tr>
</thead>
<tbody>
<tr>
<td>DATA Steps</td>
<td>6:56</td>
<td>57:16</td>
<td>8.3</td>
</tr>
<tr>
<td>PROC FREQ</td>
<td>0:28</td>
<td>1:38</td>
<td>3.5</td>
</tr>
<tr>
<td>PROC PRINT</td>
<td>0:27</td>
<td>--</td>
<td>--*</td>
</tr>
</tbody>
</table>

* PROC PRINT was not run on 3400 observations
TABLE 5

PC-DOS CPU COMPARISONS
(TIME IN HOURS:MINUTES:SECONDS)

<table>
<thead>
<tr>
<th></th>
<th>34 OBS</th>
<th>3400 OBS</th>
<th>RATIO</th>
</tr>
</thead>
<tbody>
<tr>
<td>DATA Steps</td>
<td>31:21</td>
<td>6:32:29</td>
<td>12.5</td>
</tr>
<tr>
<td>PROC FREQ</td>
<td>0:28</td>
<td>13:37</td>
<td>29.2</td>
</tr>
<tr>
<td>PROC PRINT</td>
<td>0:31</td>
<td>-- *</td>
<td>-- *</td>
</tr>
</tbody>
</table>

* PROC PRINT was not run on 3400 observations

GRAPH 1
RATIOS OF CPU TIMES:
3,400/34 OBSERVATIONS

LEGEND

<table>
<thead>
<tr>
<th></th>
<th>OS</th>
<th>CMS</th>
<th>VMS</th>
<th>PC-DOS</th>
</tr>
</thead>
<tbody>
<tr>
<td>VM/CMS</td>
<td>0.9</td>
<td>1.0</td>
<td>0.7</td>
<td>0.8</td>
</tr>
<tr>
<td>VAX/VMS</td>
<td>8.0</td>
<td>9.3</td>
<td>9.0</td>
<td>9.9</td>
</tr>
<tr>
<td>PC-DOS</td>
<td>36.2</td>
<td>9.3</td>
<td>10.3</td>
<td>67.9</td>
</tr>
</tbody>
</table>

TABLE 6

RATIOS OF CPU TIMES TO OS/VS2:
34 OBSERVATIONS

<table>
<thead>
<tr>
<th></th>
<th>VM/CMS</th>
<th>VAX/VMS</th>
<th>PC-DOS</th>
</tr>
</thead>
<tbody>
<tr>
<td>DATA Steps</td>
<td>0.9</td>
<td>8.0</td>
<td>36.2</td>
</tr>
<tr>
<td>PROC FREQ</td>
<td>1.0</td>
<td>9.3</td>
<td>9.3</td>
</tr>
<tr>
<td>PROC PRINT</td>
<td>0.7</td>
<td>9.0</td>
<td>10.3</td>
</tr>
</tbody>
</table>

TABLE 7

RATIOS OF CPU TIMES TO OS/VS2:
3400 OBSERVATIONS

<table>
<thead>
<tr>
<th></th>
<th>VM/CMS</th>
<th>VAX/VMS</th>
<th>PC-DOS</th>
</tr>
</thead>
<tbody>
<tr>
<td>DATA Steps</td>
<td>0.8</td>
<td>9.9</td>
<td>67.9</td>
</tr>
<tr>
<td>PROC FREQ</td>
<td>1.1</td>
<td>4.9</td>
<td>40.8</td>
</tr>
</tbody>
</table>

64