CLINICAL DATA MANAGEMENT AND SAS79
Michael S. Lajiness, The Upjohn Company

Introduction

Whenever a major release of SAS arrives, the user is faced with a common dilemma: Should he/she implement the new features into his/her new and older SAS programs? It's sometimes all too easy to leave well enough alone.

With the latest edition of SAS, SAS79, this dilemma is certainly evident. Several new statements and procedures are there for the using. Should we?

In this paper I will answer this question by showing some of the advantages of using the new commands and procedures of SAS79. I will contrast the performance of SAS79 and SAS76 programs in performing typical applications in the clinical area. This will include a comparison of the ease and efficiency of the coding as well as the associated CPU requirements.

Typical Data Management Functions and Examples

In the course of a clinical study, SAS is used to:

1) retrieve data from case report image (CRI) tapes containing the information collected during the clinical study;
2) back up SAS data files on disk (or tape) to tape and;
3) manipulate SAS data sets in preparation for statistical analysis or report writing.

Retrieving data from CRI tapes with SAS79 is basically the same as it was previously except that one can save CPU time and create a more structured program using SAS79. A typical SAS76 program to retrieve data from CRI tapes appears in Figure 1. The nested input structure is facilitated by the IF-THEN-GOTO-LABEL statements. However, one can improve the readability of the program and increase the computing efficiency by using the IF-THEN-DO-ELSE-END statements of SAS79 as shown in Figure 2. These SAS79 statements help reduce the computing (CPU) time by 15% as shown in Table 1.

Backing up SAS data sets was tedious and time consuming when using SAS76 but is greatly optimized in SAS79. Two new SAS79 procedures, COPY and TAPECOPY, reduce the coding and results in considerable CPU time. A SAS76 program to back up disk files to tape might look like the program in Figure 3. Each SAS data set must be selected and stored using the SET and DATA statements. 2 times N statements must be used to store N data sets. The SAS79 counterpart of this program is seen in Figure 4. This program takes just 2 statements, PROC COPY and the SELECT option, which requires just the one level SAS data set names. A 75% reduction in CPU time is realized by using the SAS79 program as shown in Table 2. This reduction is primarily the result of much fewer reads and writes to tape (TXCP). COPY and TAPECOPY are also very nice when used to copy data sets from one tape to another. Previously, tape rewinding was necessary to read multiple data sets from one tape to another. Now in SAS79 all is accomplished in one pass of the tape resulting in a tremendous reduction of real turnaround time (and CPU time).

Manipulation of SAS data sets prior to statistical analysis is often necessary and frequently messy. One of the more frequent requests is to combine several observations into one, possibly for an analysis of covariance or to take differences from initial.

While this is not especially difficult to do when the number of variables and observations to be combined is large, the program in SAS76 becomes cumbersome, to say the least. SAS79, however, makes life more simple and reduces the program to the point of being almost trivial.

Let us consider an example. In one study we had 50 variables and up to 19 follow-up reports (FU) on each patient. Each follow-up report comprised a single observation. It was desired to combine these multiple observations per patient into one observation per patient. One solution in SAS76 could be that in Figure 5. Here we had to initialize 950 new variables and have 950 assignment statements, the total number of statements being somewhere around 2,000. However, if one uses the ARRAY and DO statements of SAS79, we can reduce the number of statements to about 40! Quite a reduction of code and coding time, which makes it quite desirable for the quick and dirty application. Figure 6 shows the corresponding SAS79 program. 19 arrays represent the 19 groups of 50 new variables. One array represents the original 50 variables and another array represents the names of the 19 arrays of new variables. Initialiation is done in just 5 statements. Value assignment being done in 4 by taking advantage of the direct correspondence between follow-up report number and the numeric order of the arrays representing the new variables. The double hyphen, "--", also comes in handy and reduces the size of the RETAIN statement considerably. The FORMAT statement is unfortunately necessary because SAS insists on defining ARRAY's to be numeric which is inappropriate for this problem. Table 3 shows the computing requirements of programs similar to those in figures 5 and 6 in different test situations. It appears that the fewer the number of operations (variables, time periods) to be performed the less desirable it is to use ARRAY. Nevertheless, the difference between the two approaches is small in CPU terms.
but great in terms of program optimization (code reduction).

ARRAY's are also useful in report generator programs. For example, translating drug codes into drug names used to be code consuming. This was especially true when working with data for many drugs in each observation. Using ARRAY's it is possible to set up one group of IF-THEN assignments to handle all the translating as shown in Figure 7. In this example, data was generated by the FREQ procedure, passed to MATRIX and then input to a program similar to Figure 7.

Two of the new SAS79 procedures UNIVARIATE and SUMMARY are very useful to obtain many different descriptive statistics, sometimes in a more optimal way. UNIVARIATE provides certain statistical measures in an output data set that were previously messy to obtain at best. These include the range, median and the mode. The availability of these statistics are especially helpful in my report generation programs.

SUMMARY creates a SAS data set containing the more common statistical measures for different subgroups of the observations in the input data set. These subgroups represent all possible combinations of the levels of the variables in the CLASSES statement removing the need for a SORT and BY in most situations. The fact that you can use summary for several variables and several subgroup (class) variables makes it especially useful in generating summary reports. However, if one does not desire summary statistics for all combinations of subgroups, SUMMARY becomes more cumbersome to use. In creating 2-way tables of statistics across time and treatment given the cell, row, column, and total statistics one needed to run the MEANS procedure several times (Lajiness, Greenberg, Johnson, Schooley, 1977). With SAS79 one need run SUMMARY but once to get the same information. This reduces coding and the computing requirements.

Conclusion

SAS79 appears to be a significant advance in many respects. The coding of programs may be minimized and structured better resulting in increased programmer productivity and a reduction of certain error rates. The new commands and procedures of SAS79 further lead to more efficient use of computer resources in most cases. One should consider rewriting old SAS production programs in view of these new powerful features of SAS79.

References


// EXEC SAS76
// /CRI DD DSN='CR(001111)', VOL=SER='001111', DISP=OLD,UNIT=2400
DATA DATA1(KEEP=PROT INV PAT T MC VARI-VAR10) DATA(KEEP=PROT INV PAT T MC VAR11-VAR20)

DATA DATA(KEEP=PROT INV PAT T MC VARN1-VARN10)
INFILE CRI;
INPUT REC 1-2 PROT 19-22 INV 23-26 PAT 27-30 MC 31-32 T 33;
IF REC=10 THEN GO TO R11;
INPUT(VAR1-VARI0)(@50 10*1.);
OUTPUT DATA1;
RETURN;
R11: IF REC=11 THEN GO TO R12;
R99: IF REC=99 THEN GO TO END;
INPUT(VARN1-VARN10)(@50 10*1.);
OUTPUT DATAN;
RETURN;
END: DELETE;

Figure 1. Case Report Retrieval Program in SAS76.

// EXEC SAS79
// /CRI DD DSN='CR(0011111)', VOL=SER='001111', DISP=OLD,UNIT=2400
DATA DATA1(KEEP=PROT INV PAT T MC VARI-VAR10) DATA(KEEP=PROT INV PAT T MC VAR11-VAR20)

DATA DATA(KEEP=PROT INV PAT T MC VARN1-VARN10)
INFILE CRI;
INPUT REC 1-2 PROT 19-22 INV 23-26 PAT 27-30 MC 31-32 T 33;
IF REC=20 THEN DO;
INPUT(VAR1-VARI0)(@50 10*1.);
OUTPUT DATA1;
RETURN;
ELSE IF REC=11 THEN DO;
ELSE IF REC=99 THEN DO;
INPUT(VARN1-VARN10)(@50 10*1.);
OUTPUT DATAN;
RETURN;
END;
DELETE;

Figure 2. Case Report Retrieval Program in SAS79.
### COMPUTING EFFICIENCY COMPARISON OF RETRIEVING CR INFORMATION

<table>
<thead>
<tr>
<th>ITEM</th>
<th>SAS76</th>
<th>SAS79</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPU TIME</td>
<td>294.12</td>
<td>248.55</td>
</tr>
<tr>
<td>OXCP</td>
<td>341</td>
<td>343</td>
</tr>
<tr>
<td>TXCP</td>
<td>469</td>
<td>469</td>
</tr>
<tr>
<td>PAGEINS</td>
<td>37</td>
<td>28</td>
</tr>
<tr>
<td>PAGEOUTS</td>
<td>10</td>
<td>31</td>
</tr>
</tbody>
</table>

Table 1. Computing Efficiency Comparisons of Retrieving Case Report (CR) Information.

```
// EXEC SAS76
// OLD DD DSN=MSL.SAS,DISP=SHR
// NEW DD DSN=MSSL.SAS,DISP=(NEW,PASS),UNIT=2400,
// VOL=SER=007286
DATA NEW.DATA1; SET OLD.DATAl;
DATA NEW.DATAN; SET OLD.DATAN;
```

Figure 3. Disk to Tape Backup in SAS76.

```
// EXEC SAS79
// OLD DD DSN=MSL.SAS,DISP=SHR
// NEW DD DSN=MSSL.SAS,DISP=(NEW,PASS),UNIT=2400,VOL=SER=007286
PROC COPY IN=OLD OUT=NEW;
SELECT DATAl..... DATAN;
```

Figure 4. Disk to Tape Backup in SAS79.

### COMPUTING EFFICIENCY COMPARISON OF DISK BACKUP

<table>
<thead>
<tr>
<th>ITEM</th>
<th>SAS76</th>
<th>SAS79</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPU TIME</td>
<td>54.96</td>
<td>13.00</td>
</tr>
<tr>
<td>OXCP</td>
<td>1025</td>
<td>240.00</td>
</tr>
<tr>
<td>TXCP</td>
<td>12112</td>
<td>1553</td>
</tr>
<tr>
<td>PAGEINS</td>
<td>143</td>
<td>10</td>
</tr>
<tr>
<td>PAGEOUTS</td>
<td>43</td>
<td>10</td>
</tr>
</tbody>
</table>

Table 2. Computing Efficiency Comparison of Disk Backup.

```
DATA A; INPUT DRUG FU (V1-V50)(06 50*$1.);
CARDS:

*data here;
PROC SORT; BY DRUG;
DATA A; SET A; BY DRUG;
IF NOT FIRST.DRUG THEN GO TO FU1;

R1V1='';..... R1V50='';
R2V1='';..... R2V50='';
initialization (950 statements)
FU1: IF FV NE 1 THEN GO TO FU2;
R1V1=R1; R1V2=R2;..... R1V50=R50;
FU2: IF FV NE 2 THEN GO TO FU3;

value assignment (950 statements)
FU19: IF FV NE 19 THEN GO TO END;
R19V1=R1; R19V2=R2;..... R19V50=R50;
END: IF LAST.DRUG THEN OUTPUT;
RETURN;
RETAiN R1V1-R1V50;

R19V1-R19V50;
```

Figure 5. Combining Multiple Observations in SAS76.

```
DATA A; INPUT DRUG FU (V1-V50)(06 50*$1.);
CARDS:

*data here;
PROC SORT; BY DRUG;
DATA A; SET A; BY DRUG;
FORMAT R1V1-R1V50;
```

Figure 6. Combining Multiple Observations in SAS79.

### COMPUTING EFFICIENCY COMPARISON OF DISK BACKUP

<table>
<thead>
<tr>
<th>ITEM</th>
<th>SAS76</th>
<th>SAS79</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPU TIME</td>
<td>54.96</td>
<td>13.00</td>
</tr>
<tr>
<td>OXCP</td>
<td>1025</td>
<td>240.00</td>
</tr>
<tr>
<td>TXCP</td>
<td>12112</td>
<td>1553</td>
</tr>
<tr>
<td>PAGEINS</td>
<td>143</td>
<td>10</td>
</tr>
<tr>
<td>PAGEOUTS</td>
<td>43</td>
<td>10</td>
</tr>
</tbody>
</table>

Table 2. Computing Efficiency Comparison of Disk Backup.

DATA A; INPUT DRUG FU (V1-V50)(06 50*$1.);
CARDS:

*data here;
PROC SORT; BY DRUG;
DATA A; SET A; BY DRUG;
IF NOT FIRST.DRUG THEN GO TO FU1;

R1V1='';..... R1V50='';
R2V1='';..... R2V50='';
initialization (950 statements)
FU1: IF FV NE 1 THEN GO TO FU2;
R1V1=R1; R1V2=R2;..... R1V50=R50;
FU2: IF FV NE 2 THEN GO TO FU3;

value assignment (950 statements)
FU19: IF FV NE 19 THEN GO TO END;
R19V1=R1; R19V2=R2;..... R19V50=R50;
END: IF LAST.DRUG THEN OUTPUT;
RETURN;
RETAiN R1V1-R1V50;

R19V1-R19V50;
```

Figure 5. Combining Multiple Observations in SAS76.

```
DATA A; INPUT DRUG FU (V1-V50)(06 50*$1.);
CARDS:

*data here;
PROC SORT; BY DRUG;
DATA A; SET A; BY DRUG;
FORMAT R1V1-R1V50;
```

Figure 6. Combining Multiple Observations in SAS79.
Table 3. Computing Efficiency Comparison of Combining Multiple Observations.

<table>
<thead>
<tr>
<th>Test Situation</th>
<th>SAS76</th>
<th>SAS79</th>
</tr>
</thead>
<tbody>
<tr>
<td>10 Variables &amp; 10 FU's</td>
<td></td>
<td></td>
</tr>
<tr>
<td>-37 obs. CPU TIME</td>
<td>3.65</td>
<td>4.26</td>
</tr>
<tr>
<td>DXCP</td>
<td>87</td>
<td>87</td>
</tr>
<tr>
<td>-1000 obs. CPU TIME</td>
<td>11.19</td>
<td>12.55</td>
</tr>
<tr>
<td>DXCP</td>
<td>91</td>
<td>97</td>
</tr>
<tr>
<td>50 Variables &amp; 19 FU's</td>
<td></td>
<td></td>
</tr>
<tr>
<td>-37 obs. CPU TIME</td>
<td>29.88</td>
<td>24.00</td>
</tr>
<tr>
<td>DXCP</td>
<td>371</td>
<td>369</td>
</tr>
<tr>
<td>-1000 obs. CPU TIME</td>
<td>36.96</td>
<td>34.64</td>
</tr>
<tr>
<td>DXCP</td>
<td>398</td>
<td>396</td>
</tr>
</tbody>
</table>

* Assume NC=# of drugs in study;  
* The N drug codes are stored in the variables;  
* COL1-COLn;  
ARRAY MC COL1-COLn;  
ARRAY DRUG DRUG1-DRUGn;  
DO OVER MC;  
  IF MC=1 THEN DRUG='DRUG A';  
  IF MC=2 THEN DRUG='DRUG B';  
  IF MC=n THEN DRUG='DRUG n';  
END;  
FILE PRINT;  
PUT &40 DRUG1 @;  
PUT &55 DRUG2 @; IF NC=2 THEN GO TO END;  
PUT &70 DRUG3 @; IF NC=3 THEN GO TO END;  
  
PUT &110 DRUGn;  
  
Figure 7. Use of ARRAY in Report Writing Program.