Abstract

Computers are becoming an every day part of everyone's lives and SAS® has made life in the Clinical Data Management world easier. Utilizing SAS BASICS, Clinical Data Specialists can program simple SAS queries in order to more effectively perform quality assurance tasks on clinical data. Developing a basic programming skills class reviewing such things as DATA steps, IF-THEN-ELSE logic, SET, MERGE, SAS dates, PROC SORTS, and PROC PRINT logic, we in COM have turned our non-programmers into efficient basic SAS programmers which has resulted in quicker and more effective data quality assurance methods.

Introduction

Clinical Data Management (CDM) is primarily responsible for the computerization of data from pharmaceutical clinical trials and for ensuring that databases from these clinical trials are complete and accurate. This is accomplished through careful case report form (CRF) design, accurate processing and entry of data, and extensive data quality assurance measures. It is in the area of data quality assurance that CDM can utilize SAS. Data quality assurance is a two level process. Level one, data quality control, includes all procedures used to find and correct discrepancies across the CRFs or in the database on an ongoing basis. Level two, data quality assurance, consists of measures taken to assure that level one was done consistently and according to pre-defined procedures.

Clinical Data Specialists have the responsibility for preparing CRFs for data entry, comparing data on CRFs within patients across visits for validity and completeness, editing the database, and verifying that the CRF matches the database. The two largest of these tasks are ensuring that the data are valid and complete and verifying the CRFs to the database.

Computer Environment

In CDM, SAS processing occurs in an MVS environment on an IBM 3090 model 180J. SAS is utilized in interactive mode under TSO. CDM staff access TSO through a PC using a mainframe link.

Getting Started

Due to departmental procedures there were several dependencies on the programming department regarding database development, programming verification listings and quality assurance edits, and preparing formal tables and listings. Because these tasks are not completed concurrently during the processing of a clinical trial and priorities of the programming department (the programming department supports other departments as well) and CDM were not always the same the programming needs of CDM could not always be met in a timely fashion.

One of the responsibilities of senior staff in CDM is to write simple SAS programs to query the database. Therefore, some staff in CDM were already familiar with base SAS software and writing simple programs prior to any formal training. Since we possessed some of the basic skills we thought we could provide our own programming to help relieve some of the dependencies we had on the programming department.

We then asked what kinds of programming we needed, what kinds of programming we could do, what resources we needed, and how we could improve the skills of the CDM staff in order to accomplish the task. We decided that we needed standard efficient programs to edit data and programs to list data. With improved skills we could accomplish the programming on our own.

Edit programs are used to help CDM spot errors or outliers in the data and listings are used to verify that the data on the CRF matches what is in the database. The formal listings and tables to be included in the research reports would remain the responsibility of the programming staff.

Course Development

First Attempt

Classes were set up on-site at Parke-Davis where hands-on instruction could be used. The first instructor, who had several years Clinical Data Applications Programming, offered the classes on a first-come basis. Memos were sent out to all CDM staff giving the dates and times, and little preparation was given regarding course materials or content. These initial classes were held to primarily answer questions about simple programs people had written, offer help about getting started, and get general feedback about what CDM would like to see if we began offering SAS classes.

After several of these three hour trial sessions were held, feedback was requested from staff. It was the general feeling that the "improptu" approach was not meeting the needs of CDM. It was discovered that a more structured approach was needed. Many people needed very basic information before they could begin programming on their own, and information needed to be presented in some kind of order. It was also felt that the length of each class should be limited since three hour sessions created information overload, and that classes should be limited to one or two topics. Supervisory staff thought that an assignment should be given so Clinical Data Specialists could work on their own project. "Homework" would also provide supervisors with feedback on their staff's capabilities.

Second Attempt

We next tried a more modular approach to training. Several modules were developed and another instructor was brought in. Sessions were limited to two hours and the sessions always ended with actual hands-on assignments and a question and answer time.

With each different session we built upon the last session and used data from the same database to illustrate programming techniques. Once some of the basics were understood, class topics moved into other areas in which staff had expressed interest and were applicable to the job. These
topics included merging, arrays, accessing sequential data files, and functions.

In time, these modules were pared down and sessions were limited to one or two topics. See Figure 1 for a list of training modules. Modules are continuously changing as skill levels have increased and demand for new topics has risen.

<table>
<thead>
<tr>
<th>Training Modules</th>
</tr>
</thead>
<tbody>
<tr>
<td>basic programming techniques</td>
</tr>
<tr>
<td>allocations</td>
</tr>
<tr>
<td>uses for simple SAS programs</td>
</tr>
<tr>
<td>proc contents, what it is and how to use it</td>
</tr>
<tr>
<td>intro to the data step</td>
</tr>
<tr>
<td>set statement</td>
</tr>
<tr>
<td>merge statement</td>
</tr>
<tr>
<td>sample errors and how to correct them</td>
</tr>
<tr>
<td>handling dates in SAS, SAS dates</td>
</tr>
<tr>
<td>using external files as input</td>
</tr>
<tr>
<td>printing output from SAS</td>
</tr>
<tr>
<td>file maintenance</td>
</tr>
<tr>
<td>arrays</td>
</tr>
</tbody>
</table>

![Figure 1](image)

Course Materials

Prior to attending these training classes, CDM staff were asked to view the SAS Computer Based Training courses offered on the mainframe so they would be familiar with terminology and programming concepts. These CBT courses could then be reviewed after taking classes to better grasp the programming concepts.

Copies of production SAS databases for clinical trials were used during classes, and base SAS software documentation was shown and explained. During class everyone sat at their own terminal and examples were shown using a projector screen PC. Assignments were given and the instructor was available for questions and answers. Actual programs were written and run against the SAS databases. See the sample datasets in figure 4 and a sample program in figure 5.

Training Modules

The following sections define the information contained in the current set of modules, although more modules are being developed, and training will continue.

Basic Programming Techniques

Since the participants were non-programmers, special attention was given to good programming techniques, style, and documentation. Good programming techniques, when taught early, become second nature. Documentation and indentation are two of the most powerful tools to good programming. Examples of simple SAS programs using these techniques were shown and explained. Everyone was encouraged to get into the habit of using these.

The SAS Display Manager System was explained and a demonstration of each of the parts - program statements, log, and procedure output was given.

Allocating Files

Allocating files is a hard concept for non-programmers to grasp. One training session is dedicated to this topic as well as including allocation information quickly at the beginning of several subsequent sessions. There are two kinds of allocations that are necessary in the CDM environment, creating and accessing files. Basically, the participants are taught how to create files in which to store their programs and how to associate these files with a libref for use within SAS.

Uses for Simple SAS Programs

Examples of uses for SAS programs were given and discussed. Sample programs were shown and explained, with emphasis placed on programming listings of the data, writing programs to query the database, and programs to locate errors in the data. Some examples of SAS programs used in data quality assurance are shown in Figure 2.

<table>
<thead>
<tr>
<th>Uses for SAS Programs</th>
</tr>
</thead>
<tbody>
<tr>
<td>LISTINGS</td>
</tr>
<tr>
<td>adverse events</td>
</tr>
<tr>
<td>blood pressures</td>
</tr>
<tr>
<td>concurrent medications</td>
</tr>
<tr>
<td>dosing information</td>
</tr>
<tr>
<td>EDITS/OUTLIERS/etc.</td>
</tr>
<tr>
<td>outliers of height and/or weight</td>
</tr>
<tr>
<td>blood pressures over a certain range</td>
</tr>
<tr>
<td>terminations due to adverse events</td>
</tr>
<tr>
<td>serious adverse events</td>
</tr>
<tr>
<td>male patients who were taking Drug A</td>
</tr>
<tr>
<td>male patients who also have had a hysterectomy</td>
</tr>
<tr>
<td>differentials that are over 100%</td>
</tr>
</tbody>
</table>

MISCELLANEOUS OTHERS:

- count the number of patients in the database
- check all adverse events/meds that started in a certain phase, or at a certain visit

The List of Possibilities is endless.

![Figure 2](image)

Proc Contents: What It is and How to Use It

Proc contents: the road map to a SAS data library. An actual proc contents of a SAS database was distributed, and each section was discussed. Emphasis was given to the location of basic information such as the libname, the name of the data set, variable names, type of variable, and the format/informat of the variable.

Introduction to the Data Step

This section described what a data step is, why it’s used, and how it’s used. Although there are many statements available for use in the data step, the list was limited for the class discussion. Each statement was described and its usage shown and explained. Some of the statements discussed were: data, set, delete, output, subsetting if, do, if-then-else, by, drop, keep, and rename. Data step flow, data step execution, and data from two sources were also discussed.

Set Statement

For the set and merge statement modules separate test data sets were created to use instead of the production copy. This was done in order to limit the number and kind of data in the test data sets. It is easier to explain set and merge if it is easy to spot from which data set the data came. See figure 4 for sample data sets.
The set statement session described the set statement in great detail. It may seem excessive to have an entire class period dedicated to one SAS statement, but in order to have staff understand how and why a statement is used, it is necessary to be very simplistic in explanations. This was accomplished by breaking the uses of the set statement into individual explanations. Several different ways of using set statements were discussed, among these were concatenating two data sets, using by with set, using keep statements alone or with the set, using rename and drop, using multiple set statements, and conditionally using set.

Merge Statement
As most people who are familiar with base SAS software know, the merge statement is a little more complex than the set statement. Unless you fully understand how the statement executes, you may get results you did not expect. Several simple examples were given and results of merging several different ways were discussed. Increasingly more complex merge examples were then given. For example, first a simple merge was shown, without using options, then using a by statement, then renaming variables, then using in variables. One-to-one, one-to-many, and many-to-many merging were explained and examples were given. Figure 3 contains sample merge statements.

Sample Merge Statements

1. data allmeds;
  merge dbmeds olmeds;
2. data allmeds;
  merge dbmeds olmeds;
  by protocol trial patient med ;
3. data allmeds;
  merge dbmeds(indb=indb) olmeds(in=inol);
  by protocol trial patient med ;
  if indb and not inol;

Figure 3

Sample Errors and How to Correct Them
Beginning programmers need to know how to find and fix common errors - fast. This session was devoted to demonstrating what happens when you make common mistakes - where SAS gives you a warning or error message, how to find your errors, and how to correct them. The types of errors/mistakes demonstrated were leaving off a semi-colon, having an uninitialized variable, using a variable name not in the data set, leaving off an end statement, using an improper if-then-else statement, sorting without a by, and merging without first sorting. Showing the use of the log in locating and correcting errors was a big help for first-time programmers. Now they could begin to debug their own programs.

Handling Dates in SAS, SAS Dates
Have you ever formatted a numeric field with a date format? What an wild and crazy date! Definition of a SAS date, recognizing that 041592 is not a SAS date, that 11032 could be a SAS date, and how to format dates were some of the topics covered in this session.

Informats, formats, and some date functions were also briefly explained in this session. This was a small module, since once you've got it, you've got it!

Using External Files as Input
Although the majority of our programming work in CDM is limited to working with SAS databases, sometimes it is useful to work with sequential files prior to a SAS database being created. This session focused on the use of infile and input statements. How to read external sequential files into SAS was demonstrated and examples were given. Once other basic SAS statements are understood, it is relatively easy to explain the concept of reading external files into SAS data sets.

Printing Output from SAS
Due to the nature of our system, printing output is easier said than done. Production jobs run in batch mode have output routed to an assigned "bin". In interactive SAS, output has to be saved (or printed) to a file, and then the file printed. Output can be routed to the mainframe printer or to a laser printer dedicated to CDM. Instructions on routing output to the two different printers was given in this session.

File Maintenance
This module demonstrated our Computer Center's archival procedures, pointers for storing and accessing programs, and disk/file cleanup (archiving and deleting files). Although production files are maintained through the Computer Center, the care of personal files is left up to the individual. Through this session good file maintenance skills were taught to our future programmers.

Arrays
This session was limited to CDM staff who were more advanced in their programming. In this module, very simplistic explicit arrays were explained. Examples of different arrays to process data were presented and explained. Information like this must be carefully explained and understood by non-programmers before they can properly execute the necessary code.

Conclusions and Future Directions
Through the use of modularized training classes, base SAS programming skills have effectively been taught to Clinical Data Management staff who had little or no programming background. At the present time:

- thirteen different modules have been developed
- thirty-eight non-managerial staff have attended one or more sessions
- the average class size was sixteen
- and one instructor was used for each session.

The feedback we have received has helped us to focus the training modules toward the needs of our staff and to improve in areas where improvement was needed. The high demand for these training modules also proves that this is a much needed and appreciated service that we are offering. We have also learned that few resources were necessary when we began assessing our needs and discovered that we could offer this service to our staff without using outside resources.

We hope to continue to offer these classes in the future. We are bringing a formal base SAS software training session to Clinical Data Management this year. As you can see, we are dedicated to offering training to our Clinical Data Management staff to help make our department independent and our product accurate and quality assured.
Acknowledgements

The author would like to thank Kit Howard for her help in teaching and developing some of the training modules, to H. Jean Lieverman for her advice and comments during the writing of this paper, and to Parke-Davis for allowing the presentation at SUGI 17.

SAS and the SAS Display Manager System are registered trademarks of SAS Institute, Inc. in the USA and other countries. © indicates USA registration.

IBM is a registered trademark of International Business Machines Corporation.

Sample Datasets

<table>
<thead>
<tr>
<th>OBS</th>
<th>PROTOCOL</th>
<th>TRIAL</th>
<th>PATIENT</th>
<th>AGE</th>
<th>SEX</th>
<th>HEIGHT</th>
<th>WEIGHT</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>001</td>
<td>1</td>
<td>1</td>
<td>50</td>
<td>1</td>
<td>73.0</td>
<td>178.5</td>
</tr>
<tr>
<td>2</td>
<td>001</td>
<td>1</td>
<td>2</td>
<td>65</td>
<td>2</td>
<td>65.3</td>
<td>125.0</td>
</tr>
<tr>
<td>3</td>
<td>001</td>
<td>1</td>
<td>3</td>
<td>71</td>
<td>1</td>
<td>75.5</td>
<td>185.3</td>
</tr>
<tr>
<td>4</td>
<td>001</td>
<td>1</td>
<td>4</td>
<td>68</td>
<td>2</td>
<td>66.3</td>
<td>118.0</td>
</tr>
<tr>
<td>5</td>
<td>001</td>
<td>1</td>
<td>5</td>
<td>76</td>
<td>2</td>
<td>63.0</td>
<td>104.5</td>
</tr>
</tbody>
</table>

DBMEDS

<table>
<thead>
<tr>
<th>OBS</th>
<th>PROTOCOL</th>
<th>TRIAL</th>
<th>PATIENT</th>
<th>PHASE</th>
<th>MED</th>
<th>JSTDAY</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>999</td>
<td>5</td>
<td>1</td>
<td>db</td>
<td>333</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>999</td>
<td>5</td>
<td>2</td>
<td>db</td>
<td>334</td>
<td>2</td>
</tr>
<tr>
<td>3</td>
<td>999</td>
<td>5</td>
<td>3</td>
<td>db</td>
<td>335</td>
<td>3</td>
</tr>
<tr>
<td>4</td>
<td>999</td>
<td>5</td>
<td>4</td>
<td>db</td>
<td>336</td>
<td>4</td>
</tr>
<tr>
<td>5</td>
<td>999</td>
<td>5</td>
<td>4</td>
<td>db</td>
<td>337</td>
<td>4</td>
</tr>
</tbody>
</table>

OLMEDS

<table>
<thead>
<tr>
<th>OBS</th>
<th>PROTOCOL</th>
<th>TRIAL</th>
<th>PATIENT</th>
<th>PHASE</th>
<th>MED</th>
<th>STDRUG</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>999</td>
<td>5</td>
<td>1</td>
<td>ol</td>
<td>333</td>
<td>aspirin</td>
</tr>
<tr>
<td>2</td>
<td>999</td>
<td>5</td>
<td>2</td>
<td>ol</td>
<td>334</td>
<td>aspirin</td>
</tr>
<tr>
<td>3</td>
<td>999</td>
<td>5</td>
<td>3</td>
<td>ol</td>
<td>335</td>
<td>aspirin</td>
</tr>
<tr>
<td>4</td>
<td>999</td>
<td>5</td>
<td>5</td>
<td>ol</td>
<td>444</td>
<td>aspirin</td>
</tr>
<tr>
<td>5</td>
<td>999</td>
<td>5</td>
<td>5</td>
<td>ol</td>
<td>445</td>
<td>aspirin</td>
</tr>
<tr>
<td>6</td>
<td>999</td>
<td>5</td>
<td>5</td>
<td>ol</td>
<td>445</td>
<td>aspirin</td>
</tr>
</tbody>
</table>

Sample Program

```sas
data sex(keep=protocol trial patient age sex height weight);
set save.sub;
if protocol=1 and trial=1;
proc sort data=sex; by protocol trial patient;
data female male;
set sex;
if sex=1 then do;
output male;
end;
if sex=2 then output female;
proc print data=female split='*'; by protocol; id patient;
var trial age sex weight height;
title 'Female Patients Subject Card Information';
label age = 'Patient\*age at*Screening'
weight = 'Patient\*Weight'
height = 'Patient\*Height';
proc print data=male split='*'; by protocol; id patient;
var trial age sex weight height;
title 'Female Patients Subject Card Information';
label age = 'Patient\*age at*Screening'
weight = 'Patient\*Weight'
height = 'Patient\*Height';
run;
```

Figure 4

```sas
data sex(keep=protocol trial patient age sex height weight);
set save.sub;
if protocol=1 and trial=1;
proc sort data=sex; by protocol trial patient;
data female male;
set sex;
if sex=1 then do;
output male;
end;
if sex=2 then output female;
proc print data=sex; by protocol; id patient;
var trial age sex weight height;
title 'Patient Subject Card Information';
```

Figure 5

1507