How to use CD-ROM as a Simplified CANDA

Steve Wong, IBAH, Inc., Emeryville, CA

ABSTRACT

An application for approval to market a drug requires extensive regulatory agency review from different scientific disciplines, one of which is statistics. Clinical data and statistical software must therefore be transferred to regulatory agencies. The transfer media could vary from the high-tech sophisticated CANDAs (computer-assisted new drug applications) to simple floppy diskettes, tapes, or CD-ROMs containing data and program files. This paper discusses SAS programming techniques that enhance the transfer, especially if a CD-ROM is the desired media. The strategy developed by Matrix Pharmaceutical, Inc. to resolve these issues and some of the problems encountered along the way will be addressed.

INTRODUCTION

Before a regulatory agency will grant approval for a drug to be marketed to the public, statistical and medical reviewers must assess the overall safety and effectiveness of the drug. The assessment is based on evidence from the review of all pivotal and supportive secondary studies. During the review process, reviewers often conduct re-analysis or alternative analysis on the same or modified clinical data sets submitted by the sponsor. Statistical software is provided by the sponsor. Large pharmaceutical companies with resources and experience often submit a user-friendly, enhanced versions of CANDA which integrate statistical, textual and archive information. New companies may submit data and program files on CD-ROMs.

There are two different approaches to using CD-ROMs as the transfer media. The first method would be to download all of the data and software (SAS programs) from the CD.-ROM to the hard drive and run SAS programs and data from the hard drive. The second method is to leave both the data and the SAS programs on the CD-ROMs and run the them from the CD-ROMs. The advantages and the disadvantages of running SAS programs and data from CD-ROMs are as follows:

ADVANTAGES	*DISADVANTAGES*
Saves disk Space: SAS programs and data remain on CD-ROMs (FDA hard drives may not have enough space since they are constantly working on multiple submissions.)	WORM: A CD-ROM can only be written once. Output files must be directed to the hard drive and can not be stored on the CD-ROM.
Confidentiality: Data are managed.	Special Hardware is needed to transfer files to CD-ROMs.
File Protection: Files can mot be accidentally overwritten because of the CD-ROM -specific feature “WORM”.	
Capacity: Each CD-ROM holds about 640MB of data.	

Since the advantages clearly outweigh the disadvantages, running SAS programs off CD-ROMs seems like a logical choice.

REAL WORLD EXPERIENCE

At Matrix Pharmaceutical, we filed our first NDA using CD-ROMs. The division of Dermatologic Drug Products at FDA told us that we were the first at their division to use CD-ROMs as a simplified CANDA. The CANDA was “Simplified” in the sense that only statistical materials were included and no interactive query tools or document management systems were provided. For each study in the NDA, we copied all SAS programs, CRF (raw) data sets, user-created data sets, tables, graphs, data listings and CRF tabulations to the CD-ROMs.

We placed both the CRF data sets and user-created data sets together in the same directory. Unfortunately, this combination created difficulty for reviewers, who had trouble distinguishing the two types of data sets. Also, since a CD-ROM could only be written once, it was necessary to direct newly created output files to a local hard drive. We received a request from the FDA reviewers to change the paths in all of our programs so that the source data were read from the CD-ROM, but temporary and new data sets were written to the hard drive. This request demanded revisions to more than 1000 programs because the external paths were defined explicitly in each program. Providing programmatic systematic changes to our SAS code using “sed” and “shell” scripts was not practical since differences existed among programs that precluded system changes. FDA reviewers also told us that they were not interested in any “fancy” operations but a simple process to replicate our results and further investigate the data. Again, we did not have an easy solution for them because of the lack of modular programming. Furthermore, some programs were used only to create data sets for other programs; some were used only to create summary tables and others were only used to create graphs. Hence, it was crucial to keep track of the order of jobs being run. However, because of the lack of previous experience, no schema were incorporated in to each program to provide the necessary documentation in describing the order of jobs run and no mechanism (batch file or script) was provided for the reviewers to rerun all programs together.

As a result of the painful experience encountered at Matrix Pharmaceutical, Inc., several key issues were identified as critical to the success of a simplified CANDA. The key areas include: directory structure, autoexec.sas, modular programming, auto documentation, table numbering and executable scripts.

DIRECTORY STRUCTURE

A well-defined directory structure provides a necessary environment to assign librefs and filerefs in SAS. To create external files or to access SAS data sets and format catalogs,

filerefs and librefs must be created in a SAS program to specify the physical locations (preferably, filerefs and librefs are defined in autoexec.sas - this subject is discussed below).

One important technique is to create separate directories for the CRF data sets and for the user-created data sets. Unlike the user-created analysis data sets, CRF data sets should never be modified; by separating CRF data sets and user-created data sets, no CRF data sets could be overwritten accidentally or intentionally. Also, when running a SAS job, CRF data sets only serve as input. The directory separation allows reviewers to distinguish the two different dataset types much more easily.

At Matrix Pharmaceutical, we now define our production directory using the following levels:

System Level
Therapeutic Area
Indication
Study/ Project
Analysis
Function
Files

e.g.: /u/onc/i37192/s19/fin0596/program/demog.sas

/u:	system level
/onc:	therapeutic area - oncology
/I37192:	IND number - liver cancer
/s19:	study number
/fin0596:	analysis - final analysis started in May 96
/program:	subdirectory to store SAS programs
/demog.sas:	files - SAS program to generate summary of demographics

Under each analysis, there are subdirectories for different functions:

format:	Study-specific format catalog
output:	Summary tables, data listings and CRF tabulations
program:	SAS programs, logs and lst
rawdata:	CRF data sets in ASCII format
rawsas:	SAS version of CRF data sets
sasdata:	User-created SAS data sets
text:	Protocol, study report, and other text files.

We created separate subdirectories ("RAWSAS" and "SASDATA") to store two types of SAS data sets. The original CRF SAS data sets were stored in "RAWSAS" and the user-created data sets were stored in "SASDATA". User-created data sets often contain derived variables that are not explicitly coded in any CRFs. We also created a subdirectory "FORMAT" to store the study-specific format catalog. It is especially useful to separate the study format catalog from the global format catalog.

AUTOEXEC.SAS

Autoexec files contain SAS statements that are executed immediately after the SAS System is initialized but before it processes any source statements. An autoexec file is usually used to specify SAS system options and librefs and filerefs that are commonly used.

When running a SAS job without using the host-specific option AUTOEXEC (which specifies the autoexec file to be used), the SAS System searches three directories for the autoexec.sas file, in the following order:

your current directory
your home directory
the sasroot directory

The SAS System uses the first autoexec file it finds to initialize the session. The ECHOAUTO system option can be used to view the contents of the autoexec file when the SAS System is invoked.

At Matrix Pharmaceutical, Inc., as a result of our CD-ROM experience, we have created a specific autoexec file for each study or project. In each autoexec file, we have defined SAS system options, created macro variables to define paths for libnames and filerefs and we log all of the generic macros being used in a particular study/project. Thus, no path has to be coded in a SAS program to access external files.

Another technique is to create two macro variables as input and output librefs, pointing to the same directory in which user-created data sets reside. This will allow easy changes to the output location. A sample autoexec file are as follows:

General information for source line

```
%let user     =NLW;          ** user initial
%let company  =MATRIX;       ** company name
%let study    =#19-96-3;     ** study number
```

Macro variables to specific different paths

```
**efficacy SAS programs
%let s19p     =/u/onc/i37192/s19/fin0596/program

** efficacy SAS output tables
%let s19r     =/u/onc/i37192/s19/fin0596/output

** study-specific format catalog
%let s19f     =/u/onc/i37192/s19/fin0596/format

** CRF SAS data sets
%let s19d     =/u/onc/i37192/s19/fin0596/rawsas

** user-created SAS data sets
%let s19in    =/u/onc/i37192/s19/fin0596/sasdata

** user-created SAS data sets

 %let s19out  =/u/onc/i37192/s19/fin0596/sasdata
```

Libname definitions

```
libname library   "/u/format";      ** global format catalog
libname library1  "/u/format/onc";  ** project format catalog
libname library2  "&s19f";          ** study-specific format
                                       catalog
libname dmsasin   "&s19d";
libname sasin     "&s19in";
libname sasout    "&s19out";
```

SAS system options

```
options ps=50 ls=130 mprint fmtsearch=(library2 library1
library) nofmterr nodate nocenter nonumber;
```

External SAS macros
%inc "/u/share/goprint.sas";
%inc "/u/share/gopage.sas";

MODULAR PROGRAMMING

It is highly desirable, for maintenance purposes, to decompose a program into modules. A popular technique is the top-down design. It is characterized by moving from a general statement of the program to detailed statements of specific tasks. Using a top-down design allows you to list all of the possible tasks required to perform a particular analysis. Each job on the list represents an individual module (or macro). The bottom line has one macro for each distinct job. By using this programming style, reviewers can select specific module(s) to run. Common code can be shared within a program or among programs and the logic flow of a program can be easily understood by examining the main "module".

Consider a program that has been designed to analyze patient response in a clinical trial, comparing three treatments for condylomate acuminate (genital warts). The program generates patient response rates by treatment group and wart location. A three-way crosstabulation table with Cochran-Mantel-Haenszel statistics is required. The program is as follows:

```
%macro autodoc;
.....
%mend ;

%macro getnum;
.....
%mend ;

%macro sample;
.....
%mend ;
.....
```

***** macro calls;

%autodoc	for automatic documentation
%getnum	to generate table number
%sample	to determine the analysis sample
%predata	to carry out any necessary pre-analysis data manipulation
%cmh	to perform a categorical analysis by running PROC FREQ to get CMH p-value
%postdata	to restructure data into presentation format
%report	to present results using either DATA NULL or PROC REPORT

At Matrix Pharmaceutical, Inc., most programs consist of six major modules: auto documentation, sample determination, pre-analysis data manipulations, statistical hypothesis testing, post-analysis data manipulations, and report writing. If reviewers prefer to run a subset of modules, this can be done by commenting out the unnecessary macro calls.

AUTO DOCUMENTATION

As in the above example, the first module to execute is the auto documentation, which provides important information about the program. It gives the name of a program, a detailed description of the program, the assumptions built into the program, the input data files required, the output data files created, the name and location of the output tables/graphs/listings, the print command to be used, the author of a program, the date a program was created, the last user to make revisions and the last revision date.

The auto documentation serves as the audit trail for a program. It can also serve as the table of contents for all programs in a study or project and it is remarkably useful to the reviewer in identifying different programs. The autodoc module is a "must" for any kind of CANDA system.

TABLE NUMBERING

Another valuable tool is a module that manages the table numbering of each table, graph or listing, since the order of the final presentation in a study report often changes depending on the internal reviewers' comments. For any study with a large number of tables, a mechanism to update the table number within each program is critical to minimize the turnaround time following text revisions. One way to manage this process is to create a database to keep track of all the table numbers in a study. The database has three key pieces of information: table number, program name and table title. Internal and external reviewers may also use this database as a cross-reference for any study with a large number of tables.

At Matrix Pharmaceutical, Inc., we created a "SAS Table Database" for each study. The database is a SAS dataset that may be read by all of the programs in each study, to identify and assign the appropriate table number.

SCRIPTS

During the development stage, running SAS jobs interactively may help developers to debug a program. However, in production mode, batch processing is the preferred method. In UNIX, using a script is best for batch processing. A script is a tool that automates a repetitive chore. In some systems, scripts are called batch files or macros. Since end-users should be given a choice of running a specific program or running all programs in one study, two scripts should be made: the first prompts end-users for the program name and runs that program. The second specifies all SAS programs in the script and executes all of them. The script file is also a useful place to store print commands for the output produced by each program.

When using a CD-ROM, output files (including log files, listing files and summary tables/graphs) can not be written back to the CD-ROM. By incorporating paths into autoexec files, one can easily direct output files to any external location in a hard drive, using the script.

The following is a typical set-up used at Matrix Pharmaceutical, Inc.. All of the SAS programs in s19 are in the directory /u/onc/i37192/s19/fin0596/program and the output

files are stored in /u/onc/i37192/s19/fin0596/output. If a reviewer wants to redirect output to a testing output

directory***(/u/fda/matrix/nda20981/s19/output***), the script should look like this:

```
setenv PROGPATH /u/onc/i37192/s19/fin0596/program
setenv OUTPATH   /u/fda/matrix/nda20981/s19/output

sas $PROGPATH/sastabno.sas -log $OUTPATH/sastabno.log
-print $OUTPATH/sastabno.lst

sas $PROGPATH/demog.sas  -log $OUTPATH/demog.log -print
$OUTPATH/demog.lst

sas $PROGPATH/f2way.sas   -log $OUTPATH/f2way.log -print
$OUTPATH/f2way.lst
```

CONCLUSION

If development cost in time and resources precludes a fully integrated CANDA in your organization, CD-ROM might be an appropriate choice for you. Using a CD-ROM does not require a large investment or long commitment, it only requires thorough upfront planning. Depending on the degree of complexity desired, the application can be developed in as little as a week if the techniques described above are used. Even if CD-ROM is not the ultimate approach you choose, the concepts discussed still apply to any kind of CANDA.

REFERENCES

SAS Institute Inc., *SAS® Companion for UNIX Environments: Language, Version 6, First Edition*, Cary, NC: SAS Institute Inc. , 1993. 256 pp.

CANDA Guidance Manual. U.S. Department of Health and Human Services, Public Health Service, Food and Drug Administration.

McConnell, S. (1993). *Code Complete*. Microsoft Press.

ACKNOWLEDGMENTS

I wish to thank Dr. Morgan Stewart, Bryan Selby and Noreen Layden for their contributions to this paper.

AUTHOR CONTACT INFORMATION

Steve Wong

IBAH, Inc.
5801 Christie Avenue, Suite 355, Emeryville, CA 94608-1928

Telephone (510) 597-7200; Fax (520)420 0651
E-mail address: wong@resbiom.com