Building An Online Information Processing System Using The SAS® System And UNIX® Shell Script

Annie Guo, Ischemia Research and Education Foundation, San Francisco, California
Long Ngo, Ischemia Research and Education Foundation, San Francisco, California

Abstract

At Ischemia Research and Education Foundation, the data stored in the SAS® System for UNIX® of more than fifteen clinical trials and epidemiology studies have been collected from various sources and systems over the past ten years. The Online Information Processing System is built by integrating base SAS® software and SAS/STAT® software with UNIX C shell script to automate the centralization and documentation of the data in a standard structure, and help non-statisticians and non-SAS users like physicians and principal investigators perform exploratory data analysis. It satisfies our major concerns in the design phase: data confidentiality, transmission speed and cost.

1.0 Background

At the Ischemia Research and Education Foundation, massive clinical research data have been collected in the last ten years from more than fifteen clinical trials and epidemiology studies. In the beginning the data were stored in CMS and in the SAS System for PC, and in the last few years the UNIX operating system has been put in place to centralize all the data sets. Since the data were obtained from many different sources and a variety of systems, there was a lack of a standard structure for storing these data. Furthermore, there are always new programmers or database managers managing different parts of different clinical studies. The need for a smooth transition of the knowledge of the data to the new staff is crucial. Thus it is critical to implement a standard structure to store the data. Additionally, there is now a demand from the organization's staff that an online analysis system be established so that new programmers, non-statisticians or non-SAS users can also access the database, search for information, and be able to perform exploratory and standard statistical analyses through a menu-driven system. In addition to the confidentiality issue of the data, a large number of our staff also wants to have the option of accessing the system without having to rely on the graphical interface tools since if accessed through a standard modem of 14.4 bps or 28.8 bps, transmission of graphic screen can be quite slow. This paper demonstrates how base SAS software, SAS/STAT software and the UNIX shell script can be used together to develop an application that satisfies our organizational needs.

2.0 Objective

The above problems in short demonstrate that the following set of objectives has to be achieved :

-Select the appropriate tools (software). Determine the resources needed to acquire proficiency in using the selected software.

-Examine the structures of all clinical studies to identify the common characteristics, and design and automate the standard structure to centralize all the data. Also apply this automated procedure to all the new studies in the future. This procedure is called Information Organizational Procedure (IOP).

-Design and implement the Information Analysis System (IAS) which allows immediate access to the data set up by IOP. This menu-driven system also allows different types of search and retrieval options. Basic PROC steps for data exploration from base SAS software and SAS/STAT software will be available.

3.0 Selection of Software

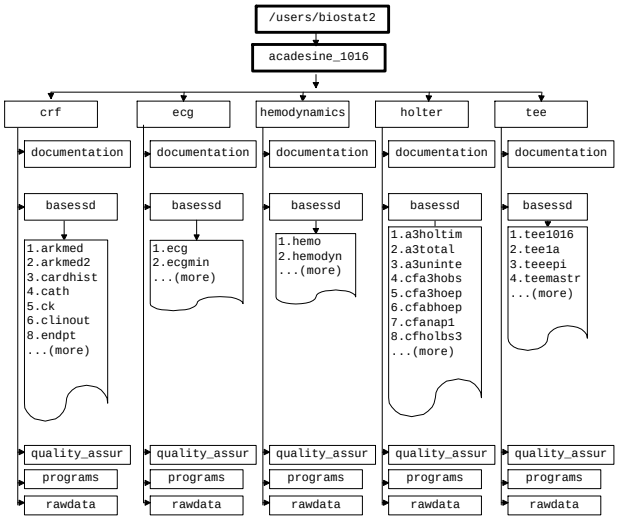
There are two important factors in our decision of selecting the appropriate software. First, due to the confidential nature of our clinical data, modem access is the only authorized mode of data transmission for off-site users. Secondly, the transmission speed is a major factor and the amount of graphic screens should be as minimal as possible. Going over a 14.4 bps modem, a full text-based screen (about 2000 bytes) can be transmitted almost instantaneously, whereas in a richer graphical environment such as that of the SAS Display Manager System, the transmission is much slower. Furthermore, a fast search routine is needed so that users, without the knowledge of the database structure, can quickly search for a particular string of information in a data set, variable or label.

The UNIX shell script is the appropriate tool that allows a smooth integration with the SAS System. Its utility tools such as SED and AWK provide efficient string manipulation. They are basically available as part of the UNIX operating system so the cost is practically nil. There are also books available so that one can acquire the proficiency in relatively short time. The UNIX C shell script will be used to drive the menu-driven system as well as executing SAS code embedded within the shell script.

4.0 Implementation of Information Organizational Procedure (IOP)

After the structures of all the past clinical studies were examined, a common set of characteristics was identified (e.g. all studies must have a case report form, CRF, component). The common structure is used as a standard structure for all the past, current and future studies. Figure 1 shows an example of the proposed standard structure used for a clinical study.

Figure 1 : Standard Data Structure for a Clinical Study



The standard structure has at the root the central location of all SAS clinical data (e.g. /users/biostat2). Below that is the name of the study (e.g. acadesine_1016). The next level shows five types of data, each of which has a different subdirectory. Below the data type subdirectory is the documentation subdirectory and the location for the SAS data sets. There are also subdirectories for quality assurance, raw data, and SAS programs. This directory structure is applied to all other clinical studies.

To automate the standard structure, the UNIX C shell script *study_tree* (Appendix) is created. When executed under the study directory (e.g. /users/biostat2/acadesine_1016), this script creates automatically the desired standard structure shown in figure 1. After the standard directory structure is in place, the data sets can then be created or moved to the designated subdirectory (e.g. in figure 1 the data set *endpt* is moved to /users/biostat2/acadesine_1016/crf/basessd).

Once the data are in the appropriate locations, the next step is to ensure as much as possible that the data sets and the variables are labeled completely. This is useful from the viewpoint of documentation and information retrieval. The label modification process for variables and data sets is achieved by executing the SAS program *labels.sas* (Appendix) for each type of data (e.g. crf). This program creates three files. The first file called *addlabel.datetime.inc* lists all the variables of missing labels from all the data sets under ../crf/basessd subdirectory, where *datetime* refers to the system date and time when the SAS program was executed. A field of 40 blank spaces is reserved for each variable. Table 4-1 below shows an example of such a file. The programmer then edits this file to fill in the appropriate label description. Note that this file is then included into a SAS program (Table 4-2).

Table 4-1: ADDLABEL.DATETIME.INC

```
label
  AGE            =  "                                        "
  AUC            =  "                                        "
  CENTER         =  "                                        "
  DRUGDATE       =  "                                        "
  EVAL           =  "                                        "
  FLAG           =  "                                        "
  PATID          =  "                                        "
  PROTOCOL       =  "                                        "
  RACE           =  "                                        "
  SITEID         =  "                                        "
  UNIT           =  "                                        " ;
```

Table 4-2 shows *variables.datetime.sas*, the second file produced by the program labels.sas which when executed will update all the SAS data sets with the modified or new variable labels. Note the %INCLUDE statement.

Table 4-2: VARLABELS.DATETIME.SAS

```
%macro addlabel(libname=,dsname=);
libname sasdata "&libname";
libname library "&libname";
data sasdata.&dsname;
  set sasdata.&dsname;
  %include "addlabel.datetime.inc"/source2;
  run;
libname sasdata clear;
libname library clear;
%mend addlabel;

%addlabel(dsname=CARDHIST,
 libname=/users/biostat2/acadesine_1016/crf/basessd);
%addlabel(dsname=ENDPT,
 libname=/users/biostat2/acadesine_1016/crf/basessd);
... (more macro calls for additional data sets)
```

Subsequently, to update the labels for the SAS data sets, the third file *memlabels.datetime.sas* is generated by labels.sas and the correct or new labels can be inserted into this file. Table 4-3 shows the file. When executed, this program will update all the data set labels.

Table 4-3: MEMLABELS.DATETIME.SAS

```
libname sasdata "/users/biostat2/acadesine_1016/crf/basessd";
libname library "/users/biostat2/acadesine_1016/crf/basessd";

proc datasets library=sasdata memtype=data;
  modify CARDHIST  (label="                              ");
  modify ENDPT     (label="                              ");
  ... (more data sets)
  run;
```

At this point all the SAS data sets are in the desired structure and all the labels for data sets and variables have been updated. This is the objective of the Information Organizational Procedure (IOP).

The program *rebuild.sas* (Appendix) then generates three documentation files which will be used later on by the Information Analysis System in helping users search for the desired variables, data sets, information in the labels, or the user-defined format values. Table 4-4, 4-5, and 4-6 show these files *dslabels.doc*, *variables.doc*, and *formats.doc*, respectively.

Table 4-4: DSLABELS.DOC

```
Data Set                          # of    # of
Name      Data set Label          Obs     Vars

CARDHIST  CARDIAC HISTORY         7608    17
ENDPT     ENDPOINT EVALUATION     633     37
... (more data sets)
```

Table 4-5: VARIABLES.DOC

```
---------- Library Member Name = ENDPT ----------
DATA SET
NAME      NAME      TYPE LENGTH FORMAT   LABEL

ENDPT     PATID     CHAR 11              PATIENT ID
ENDPT     PROTOCOL  CHAR 4               PROTOCOL
ENDPT     CENTER    CHAR 2               CENTER
ENDPT     RANDNUM   NUM  8               RANDOMIZATION #
ENDPT     DEATH     CHAR 1               CARDIAC DEATH
... (more data sets and variables)
```

Table 4-6: FORMATS.DOC

```
FORMAT
NAME         TYPE     START               LABEL

CODE         CHAR     0                   PLACEBO
CODE         CHAR     1                   0.05MG/KG/MIN
CODE         CHAR     2                   0.10MG/KG/MIN
CODE         CHAR     3                   0.19MG/KG/MIN
CODE         CHAR     4                   0.38MG/KG/MIN
... (more format values)
```

5.0 Implementation of Information Analysis System (IAS)

The Information Analysis System is an UNIX C Shell script designed to interact with users, and help users search for information and perform exploratory analysis. Because all the SAS data sets have been centralized in the standard structure and summarized in the three documentation files (dslabels.doc, variables.doc and formats.doc) by the IOP, the IAS script can locate a particular study and data type, and query the relevant SAS data sets by looking into the three documentation files. It prompts the user with questions step by step, reads lines of the user's input as UNIX variables, generates a SAS program automatically based on the defined UNIX variables, and displays the SAS LST file if no errors are found in the SAS LOG file.

The IAS script has minimal run time because of the 3 text-based documentation files. It can be executed by more than one user simultaneously without having one's SAS program be overwritten by another's, because each user's SAS program is named by an unique UNIX process id. Only standard options of SAS procedures are provided to avoid the violation of statistical assumption and invalid testing results, because non-statisticians such as physicians and principal investigators are part of the target users.

There are three parts of the information analysis system at our organization : *Basic Statistical Tools*, *Data Exploration* and *WEB Interface*. The Basic Statistical Tools and WEB Interface are still under development, and the Data Exploration is being released to the staff. Table 5-1, 5-2 and 5-3 show the first three screens resulted from executing the IAS script, where the user can enter appropriate numbers to select a desired type of data processing (e.g. Data Exploration), study (e.g. acadesine_1016) and data type (e.g. crf), respectively.

Table 5-1: Main Screen

```
          ONLINE INFORMATION PROCESSING SYSTEM

1. Introduction
2. Basic Statistical Tools
3. Data Exploration
4. WEB Interface
5. Exit

Please enter a number: 3
```

Table 5-2: Selecting a Study

```
Data Exploration

List of Studies
---------------------------------------------------------
1 ACADESINE_1013
2 ACADESINE_1016
... (more studies)

#  : Previous screen
## : Main screen

Please enter a number: 2
```

Table 5-3: Selecting a Data Type

```
Data Exploration

List of Data Types for ACADESINE_1016
---------------------------------------------------------
1 CRF
2 ECG
3 HEMODYNAMICS
... (more data types)

#  : Previous screen
## : Main screen

Please enter a number:  1
```

All the valid choices such as the integers, '**#**' and '**##**' are displayed on the screen. Each valid choice entered is used as a value assigned to an UNIX variable which will later be linked with the directory path to locate the desired SAS data information. For example, in Table 5-3, choosing option 1 creates an UNIX variable of the value *crf* which is then combined with the previously created UNIX variables storing the two values */users/biostat2* and *acadesine_1016* (Table 5-2) to create the physical location of the CRF data (e.g. /users/biostat2/acadesine_1016/crf). Besides, the selections from previous screens are displayed in the current screen, so that the user can make sure the right choices have been made, and if necessary, erase and redo the selection(s) by typing '#' to go back to the previous screen or '##' to go back to the main screen.

Once the study and data type are selected, the user can begin to search for a particular variable, data set, format, or program. In Table 5-4, selection 1, 2 and 3 (e.g. Datasets, Variables, and Formats) utilize the three documentation files in /users/biostat2/acadesine_1016/crf/documentation. Selection 4 (e.g. Programs) issues UNIX commands to display the files stored in /users/biostat2/acadesine_1016/crf/programs. For example, if the user is interested in retrieving a list of the first fifty patients and their cardiac death status, and also the distribution of all patients by center, he first has to select the option number 2 (Table 5-4) and is then prompted to enter the search string. In Table 5-5, for example, a possible search string *card* is entered to search for any variable or label which contains *card*. The UNIX script searches the documentation file variables.doc and displays all 17 lines, each of which contains at least one occurrence of *card*. The user then identifies the data set which contains the desired variable name. In this case, the data set *endpt* is identified because the desired variable is *death.* In the situation where the search is not successful the user can try other search strings until he can locate the desired information.

Note that the pattern matching is always done in uppercase because the three documentation files are written in uppercase. If the user's input is in lowercase or a mix of both, it will be converted to uppercase and then the pattern matching is performed.

Table 5-4: Selecting a Search Option

```
Data Exploration

Study    : ACADESINE_1016
Data Type: CRF

Search for:
---------------------------------------------------------
1 DATASETS
2 VARIABLES
3 FORMATS
4 PROGRAMS

#  : Previous screen
## : Main screen

Please enter a number:  2
```

Table 5-5: Data Query on Variables

```
DATA EXPLORATION

Study    : ACADESINE_1016
Data Type: CRF
Searching: VARIABLES
---------------------------------------------------------

#          : Previous screen
##         : Main screen
q          : To quit listing
Space Bar  : To display by page
Return Key : To display by line

Please enter a keyword to search names and labels: card

List of variables matching keyword: CARD,
total 17 hit(s). Hit RETURN to continue.

DATA SET
NAME     NAME    TYPE LENGTH FORMAT  LABEL

CLINOVER CARD1DX NUM  8      ECGDXS  CARDIOLOGIST 1 DX
CLINOVER CARD2DX NUM  8      ECGDXS  CARDIOLOGIST 2 DX
ENDPT    DEATH   CHAR 1              CARDIAC DEATH
... (more variables)
```

Table 5-6 shows the screen for selecting the desired data set. In our example, the data set is *endpt* and the selected variable is *death*. Table 5-7 shows the seven SAS procedures currently in use. The user is then guided step by step to enter necessary information such as variable names and options necessary for the IAS script to automatically construct the SAS program.

Table 5-6: Data Query on Data Sets

```
DATA EXPLORATION

Study    : ACADESINE_1016
Data Type: CRF
Searching: DATASETS

List of DATASETS
-----------------------------------------------------
Dataset                          # of    # of
Name      Dataset Label          Obs     Vars

CARDHIST  CARDIAC HISTORY        7608    17
ENDPT     ENDPOINT EVALUATION    633     37
... (more data sets)

#  : Previous screen
## : Main screen
q  : To quit listing

Please enter a data set name: endpt
```

Table 5-7: SAS Procedures Available

```
DATA EXPLORATION

Study    : ACADESINE_1016
Data Type: CRF
Searching: DATASETS
Dataset  : ENDPT

List of SAS Procedures
-----------------------------------------------------
1 PRINT
2 CONTENTS
3 UNIVARIATE
4 FREQ
5 CORR
6 PLOT
7 REG

#  : Previous screen
## : Main screen

Please enter a number: 1
```

In our example, PROC PRINT will be invoked (options 1) to print the three variables (*patid*, *center*, and *death)* of the first fifty patients. Table 5-8 shows that the user is prompted to enter the number of observations (50) and the desired variable names. All the selections entered are assigned as UNIX variables ($*varname*) to create SAS code as in Table 5-9. Similarly, for the distribution of patients by *center*, option 4 (PROC FREQ) is invoked. Table 5-10 shows that the user is prompted to enter the TABLE statement, and then a SAS program is generated automatically and the frequency table of *death* by *center* is retrieved.

Table 5-8: Procedures to Automate PROC PRINT

```
DATA EXPLORATION

Study    : ACADESINE_1016
Data Type: CRF
Searching: DATASETS
Dataset  : ENDPT
SAS Proc : PRINT

#  : Previous screen
## : Main screen
q  : To stop listing at anytime
```

(Continued)

```
List of Variables
-----------------------------------------------------------
NAME     TYPE LENGTH FORMAT   LABEL

ENDPT    PATID    CHAR 11              PATIENT ID
ENDPT    PROTOCOL CHAR 4               PROTOCOL
ENDPT    CENTER   CHAR 2               CENTER
ENDPT    RANDNUM  NUM  8               RANDOMIZATION #
ENDPT    DEATH    CHAR 1               CARDIAC DEATH
... (more variables)

Please enter the number of observations,
or hit RETURN to choose 100: 50

Please enter variable name(s) or hit RETURN to print
all variables: patid center death
```

Table 5-9: UNIX C Shell script to Generate a SAS Program Automatically

```
# NOTE: UNIX variables have been assigned as follows:
# $datadir  -  /users/biostat2  (e.g. root directory)
# $study    -  acadesine_1016
# $modality -  crf  (e.g. data type)
# $proc     -  print
# $obs      -  50
# $var      -  patid center death
# $$        -  UNIX process id allowing creation of
#              unique file name

echo "libname data'$datadir/$study/$modality/basessd';"
                                      >> /tmp/temp$$.sas
echo "options nocenter ls=80 ps=59 fmtsearch = (data);"
                                      >> /tmp/temp$$.sas
echo "proc $proc data= data.$updata (obs=$obs);"
                                      >> /tmp/temp$$.sas
echo "   var $var;"                   >> /tmp/temp$$.sas
echo "   run;"                        >> /tmp/temp$$.sas
```

Table 5-10: Procedures to Automate PROC FREQ

```
DATA EXPLORATION

Study    : ACADESINE_1016
Data Type: CRF
Searching: DATASETS
Dataset  : ENDPT
SAS Proc : FREQ

List of Variables
-----------------------------------------------------------
NAME     TYPE LENGTH FORMAT   LABEL

ENDPT    PATID    CHAR 11              PATIENT ID
ENDPT    VISIT    NUM  8               VISIT NUMBER
ENDPT    PROTOCOL CHAR 4               PROTOCOL
ENDPT    CENTER   CHAR 2               CENTER
ENDPT    SCRNUM   NUM  8               SCREENING #
ENDPT    RANDNUM  NUM  8               RANDOMIZATION #
ENDPT    DTVIS_A  CHAR 11              VISIT DATE
ENDPT    DEATH    CHAR 1               CARDIAC DEATH
... (more variables)

#  : Previous screen
## : Main screen
q  : To quit listing

Please enter variable name(s) separated by *
(for example, center * gender):
death * center
```

Upon the entry of a data set name or variable names, the IAS script will check the validity of the names entered against the documentation files using AWK language. If any invalid data set or variable names are found, the user will be prompted to re-enter. Errors such as a character variable instead of a numeric variable used in PROC UNIVARIATE will cause the script to display the message "ERRORS found in LOG file. Check with a statistician." If no errors are found in the SAS LOG file, the output LST file will be displayed on the screen and then the user can choose to print or save the file in a desired location.

6.0 Conclusion

The implementation of the IOP and IAS meets our organizational needs of providing a data centralization and data processing solution for all of our SAS clinical data sets. It also demonstrates that an user-friendly menu-driven application, using existing resources such as the C shell script in UNIX operating system and basic SAS modules, satisfies our major considerations in the design phase: security, speed, and cost.

7.0 Reference

Anderson G, Anderson P: *The UNIX C Shell Field Guide*, Prentice Hall, NJ 1986.

SAS Companion for the UNIX Environment and Derivatives, Version 6, First Edition, SAS Institute Inc., NC 1990.

SAS Guide to Macro Processing, Version 6, Second Edition, SAS Institute, Inc., NC 1990.

SAS Language: Reference, Version 6, First Edition, SAS Institute Inc., NC 1990.

SAS Procedures Guide, Version 6, Third Edition, SAS Institute Inc., NC 1990.

SAS/STAT User's Guide, Version 6, Fourth Edition, SAS Institute, Inc., NC 1990.

8.0 Acknowledgments

Author

Annie Guo asg@orion.iref.org
Long Ngo lhn@orion.iref.org

Ischemia Research and Education Foundation
250 Executive Park Blvd. #3400, San Francisco, CA 94134
(415) 715-2300

Appendix

STUDY_TREE

```
#      The first line must contain # indicating a C shell script.
onintr error_exit #go to error_exit upon CTRL-C

if ($#argv == 0) then
    echo 'No directory path given. Usage: study_tree `pwd` '
    goto error_exit
else
    set studydir = $1 #/users/biostat2/acadesine_1016 by `pwd`
endif

cd $1                       # go to /users/biostat2/acadesine_1016
umask 002                    # set access permissions to rwxrwxr-x

mkdir crf
mkdir crf/documentation
mkdir crf/basessd
mkdir crf/quality_assur
mkdir crf/programs
mkdir crf/rawdata
#... (more directories to make according to Figure 1)

foreach modality (crf ecg hemodynamics holter tee)

set targetdir = "$studydir/$modality" # set to
/users/biostat2/acadesine_1016/crf
set technol = `echo "$targetdir" |cut -f5 -d "/"`    # set to crf
set study = `echo "$targetdir" | cut -f4 -d "/" ` #acadesine_1016

clear
echo "TO INSTALL TOOLS IN: $targetdir/documentation ..."
echo "    Press ENTER to continue, or CTR-C to exit   "
set response = `line`
sleep  1
echo "Installing labels.sas ..."
sed -e "s/%%STUDY%%/$study/"           \
    -e "s|%%TARGETDIR%%|$targetdir|"    \
    -e "s/%%TECHNOL%%/$technol/"       \
    /users/biostat2/tools/labels.sas  > \
    $targetdir/documentation/labels.sas
chmod 775 $targetdir/documentation/rebuild.sas
echo "                                  DONE. "

echo "Installing rebuild.sas ..."
sed -e "s/%%STUDY%%/$study/"           \
    -e "s|%%TARGETDIR%%|$targetdir|"    \
    -e "s/%%TECHNOL%%/$technol/"       \
    /users/biostat2/tools/rebuild.sas > \
    $targetdir/documentation/rebuild.sas
chmod 775 $targetdir/documentation/rebuild.sas
echo "                                  DONE. "

echo " MODALITY $modality - PRELIMINARY  INSTALLATION COMPLETE"
sleep 1
end
exit 0 #normal exit, status 0
error_exit:
exit 1 #error exit, status 1
```

LABELS.SAS

```
%macro labels(study=, libname= , technol=);
libname sasdata "&libname";
libname library "&libname";

proc contents data=sasdata._all_ noprint out=contents;
  run;

**** Generating 2 files: addlabel.&sysdate..&systime..inc and
**** varlabels.&sysdate..&systime..sas to fix missing var labels;

data contents;
  set contents(keep=memlabel memname name format length label);
  if memname in ('CONTENTS','DATASETS','FMTLIB') then delete;
  run;                                *keep all the SAS data sets;


proc sort data=contents;                      *sorted by var label;
  by label;                          *blank var labels go to top;
  run;

data _null_;
  set contents;    *&var=missing if at least one var of no label;
  if _n_=1;              *else, &var is blank if all have labels;
  if label=' ' then call symput('var','missing');
  else call symput('var',' ');
  run;

%if &var=missing %then %do;     *if at least one var of no label;

  data labels (keep=line);
    length line $80.;
    set contents;
    if label=' ' then do;
      label= "                                ";
      line= name||" =  """||label||" "" ";
      output;          *40 blank spaces for each var of no label;
      end;                          *output all vars of no labels;
    run;

  proc sort data=labels nodupkey;
    by line;                                  *sorted by var name;
    run;

  data labels;
    file "addlabel.&sysdate..&systime..inc";
    set labels end=eof ;          *if at least 1 var of no label;
    if _n_ eq 1 then put "label";  *starting the label statement;
    put line;                 * followed by the lines of var names;
    if eof then put ";";             *ending the label statement;
    run;

  proc sort data=contents;              *sorted by data set name;
    by memname;
    run;

  data _null_;
    set contents;
    by memname;
    if first.memname;         *only one occurrence each data set;
    if _n_ eq 1 then do;      *if at least 1 SAS data set exists;
      file "varlabels.&sysdate..&systime..sas";    *do this once;
      put '%macro addlabel(libname=,dsname=);'
        /'libname sasdata "&libname";'
        /'libname library "&libname";'
        /'data sasdata.&dsname;'
        /'set sasdata.&dsname;'
        /"%include ""addlabel.&sysdate..&systime..inc""/source2;"
        /'run;'
        /'libname sasdata clear;'
        /'libname library clear; '
```

```
       /'%mend addlabel;';
      end;                    *do the following for each data set;
    file "varlabels.&sysdate..&systime..sas";
    put '%addlabel(dsname=' memname
      ',libname=/users/biostat2/acadesine_1016/crf/basessd);';
    run;

%end;

%else %if &var= %then
   %put NOTE(LABELS): No missing variable labels found in
/users/biostat2/acadesine_1016/crf/basessd;
*note in LOG file if all vars have labels;

**** Generating memlabels.&sysdate..&systime..sas
**** to fix missing data set labels;

proc sort data=contents;
  by memlabel;                          *sorted by data set label;
  run;                                *blank labels move to the top;

data _null_;
  set contents;                              *take the 1st data set;
  if _n_=1;     *&var=missing if at least 1 data set of no label;
  if memlabel=' ' then call symput('mem','missing');
  else call symput('mem',' ');               *else, &var is blank;
  run;

%if &mem=missing %then %do;   *if at least 1 data set of no
label;

  data datasets(keep=memlabel memname name format length label);
    length memname name $8. format $8. length 8. label $40.;
    set contents;
    if memlabel=' ' then output;  *output data sets of no labels;
    run;

  proc sort data=datasets nodupkey;
    by memname;                           *sorted by data set name;
    run;

  data _null_;
    set datasets end=eof;
    by memname;                            *if at least 1 data set;
    if _n_ eq 1 then do;                    *do the following once;
      file "memlabels.&sysdate..&systime..sas";
      put 'libname sasdata '
          '/users/biostat2/acadesine_1016/crf/basessd";'
         /'libname library '
          '"/users/biostat2/acadesine_1016/crf/basessd";'
         /'proc datasets library=sasdata memtype=data;';
    end;          *do the following statement for each data set;
    file "memlabels.&sysdate..&systime..sas";
    put  "   modify " memname '(label="          ");' ;
    if eof then put "run;";                 *ending proc datasets;
    run;
%end;
%else %if &mem= %then
  %put NOTE(LABELS): No missing data set labels found in
/users/biostat2/acadesine_1016/crf/basessd;
*note in the LOG file if all data sets have labels;
%mend labels;

options ps=59 ls=80 nonumber nodate;
%labels(study=acadesine_1016, technol=crf,
   libname=/users/biostat2/acadesine_1016/crf/basessd);
```

REBUILD.SAS

```
**** Data set listing by data set name ****;
%macro datasets(study=, libname=, technol=);
libname sasdata "&libname.";
libname library "&libname.";

proc contents data=sasdata._all_ noprint out=contents;
  run;

proc sort data=contents; *sorted by data set name & var number;
  by memname varnum;
  run;

data contents;
  set contents;
  by memname varnum;               *one occurrence each data set;
  if last.memname=1;     *obtain the # of vars for each data set;
  run;

data sasdata.datasets          /* source file of dslabels.doc */
    (label="DATASETS: &study. &technol."
    keep=memname memlabel nobs varnum);
  length memname $8. memlabel $40. nobs 8. ;
  set contents;
  by memname;     *40 blank spaces for each data set of no label;
  if memlabel=' ' then memlabel="<________________________________>";
  memlabel=upcase(memlabel);
  run;                      *uppercase for easy pattern matching;

data _null_;
  file 'dslabels.doc' print n=ps notitles header=head;
  set sasdata.datasets;
  put @1 memname @13 memlabel @58 nobs @74 varnum;
  return;
  head:
    put #3 @1 'Data Set                    # of    # of'
        #4 @1 'Name      Dataset Label     Obs     Vars'
        #5 @1 ' ';
    return;
  run;
%mend datasets;

**** Variable listing by data set and var number ****;
proc format;        *format used for the header of variables.doc;
   value type
      1='NUM'
      2='CHAR';
   run;

%macro variable(study=, libname=, Technol=);
libname sasdata "&libname";
libname library "&libname";

proc contents data=sasdata._all_ noprint out=contents;
  run;

proc sort data=contents;
   by memname varnum;           *by data set name and var number;
   run;

data sasdata.contents         /* source file for variables.doc */
    (label="CONTENTS: &study. &technol."
    keep=memname name format length label type);
  length memname name $8. format $8. length 8. label $40.;
  set contents;    *40 blank spaces for each variable of no
label;
  if label=' ' then label="<________________________________>";
  label=upcase(label);
  run;                     *uppercase for each pattern matching;

data _null_;
  file 'variables.doc' print n=ps notitles ;
  set sasdata.contents;
  by memname;
  if first.memname then do;     *page break & header by data set;
    put _page_;
    put #3 @1 10*'-' 'Library Member Name =' memname 10*'-'
        #5 @1 'DATA SET'
        #6 @1 'NAME     NAME     TYPE LENGTH FORMAT   LABEL'
        #7 @1 ' ';
    end;
  put @1 memname @10 name @19 type type. @24 length
      @31 format @40 label;
  return;
  run;
%mend variable;

**** Format listing ****;
proc format;
  value $type
    'N'='NUM'
    'C'='CHAR';
  run;

%macro formats(study=, libname=, technol=);
%sysexec ls &libname./formats.sct01 > temp.err;
%if &sysrc ne 0 %then %do;                *if no format library;
  %put ERROR(REBUILD): Format library not found in &libname;
  %goto quit;
  %end;

libname sasdata "&libname.";
libname library "&libname.";

proc format library=sasdata cntlout=fmtout;
  run;

data sasdata.fmtlib            /* source file  for formats.doc */
    (label="FMTLIB: &study. &technol."
    keep=type fmtname start label);
  length type $1. fmtname $8. start $20. label $40.;
  set fmtout;
  label=upcase(label);     *uppercase for easy pattern matching;
  run;

data _null_;
  file 'formats.doc' print n=ps notitles header=head ;
  set sasdata.fmtlib;
  put @1 fmtname @13 type $type. @21 start @41 label;
  return;
  head:
    put #3 @1 'FORMAT'
        #4 @1 'NAME        TYPE    START         LABEL'
        #5 ' ';
    return;
  run;

%quit:
   %sysexec rm temp.err;
%mend formats;

options ps=59 ls=80 nonumber nodate;
%datasets(study=acadesine_1016, technol=crf,
  libname=/users/biostat2/acadesine_1016/crf/basessd);
%variable(study=acadesine_1016, technol=crf,
  libname=/users/biostat2/acadesine_1016/crf/basessd);
%formats(study=acadesine_1016, technol=crf,
  libname=/users/biostat2/acadesine_1016/crf/basessd);
```