SAS MACROS FOR (CLINICAL) DATABASE CHECKING AND LISTING

Sam X. Ye
Whitehall-Robins, A.H.P. Co.

ABSTRACT

A clinical trial usually has dozens of structurally different SAS data sets capturing the information on the Case Report Form. These data sets are kept in a separate directory (SAS data library) and are sometimes called a clinical study database. Despite the variety of the nature of studies, a number of tasks which are performed on every study are basically the same. These routine jobs include obtaining crucial information about variables, data error checking, and sample data listings. It is desirable to design a set of macros with minimum of parameters and with the ability to handle any clinical database. This paper presents 8 macros (called the DB macros) which will greatly reduce laborious programming for the routine tasks. Some of the DB macros can be applied directly to other SAS databases (data libraries). The usage examples, the concept, and the programs of the DB macros are provided.

INTRODUCTION

SAS programmers in the pharmaceutical industry often work on dozens even hundreds of clinical trial studies (databases), and each study consists of several SAS data sets. A number of jobs have to be done to all data sets in every database. It would be much more efficient to have macros that can recognize automatically all data sets one by one in any database without mentioning the data set names, and can perform a specific routine job. Eight macros presented in this paper are: DB, DBSORT, DBCONT, DBMEAN, DBVAR, DBPRT, DBDUPL and DBERR. I refer to them as the DB macros. These macros can process all SAS data sets or a group of SAS data sets in a directory (SAS library) for sorting the data sets, getting important information about variables, checking data errors, and listing sample data. No data set names are needed as parameters. Some of the DB macros have no parameters at all. Therefore, it is very easy to invoke the macros and implement the jobs. A few simple examples will be given in the next section. Detailed explanations and more examples will be discussed in the section "THE CONCEPT OF THE DB MACROS". The last section provides the programs for all of the DB macros. Throughout this paper the UNIX system is assumed, but the programs can be modified for other systems.

THE USAGE EXAMPLES

For an illustration, suppose a study database consists of following three data sets named ADR, DEMO, and PE. They are kept in a directory named '/db001' (in UNIX system).

The data list of ADR:

<table>
<thead>
<tr>
<th>ID</th>
<th>CODE</th>
<th>TERM</th>
</tr>
</thead>
<tbody>
<tr>
<td>11</td>
<td>111</td>
<td>abdominal</td>
</tr>
<tr>
<td>11</td>
<td>212</td>
<td>diarrhea</td>
</tr>
<tr>
<td>13</td>
<td>335</td>
<td>nausea</td>
</tr>
<tr>
<td>12</td>
<td></td>
<td>headache</td>
</tr>
</tbody>
</table>

The data list of DEMO:

<table>
<thead>
<tr>
<th>ID</th>
<th>SEX</th>
<th>AGE</th>
<th>HEIGHT</th>
<th>WEIGHT</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>2</td>
<td>35</td>
<td>74</td>
<td>200</td>
</tr>
<tr>
<td>11</td>
<td>1</td>
<td>25</td>
<td>65</td>
<td>150</td>
</tr>
<tr>
<td>12</td>
<td>1</td>
<td>11</td>
<td>68</td>
<td>140</td>
</tr>
<tr>
<td>13</td>
<td>2</td>
<td>48</td>
<td>58</td>
<td>200</td>
</tr>
</tbody>
</table>

The data list of PE:

<table>
<thead>
<tr>
<th>ID</th>
<th>PE1</th>
<th>PE2</th>
<th>PECOM</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>0</td>
<td>1</td>
<td>abnormal</td>
</tr>
<tr>
<td>11</td>
<td>1</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>1</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>13</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
</tbody>
</table>

If the way of using SAS autocall facility is set up properly (see NESUG '94 paper "THE CREATIVE USE OF PUT STATEMENT TO IMPROVE PRODUCTIVITY AND FLEXIBILITY IN TABLE GENERATION" by SAM X. YE), the user can simply submit the DB macros working on any directory (contains SAS data sets) as
The results of the invocations of the above DB macros are shown in Table 1, Table 2, and Table 6 respectively. Table 1 shows the data set names in the directory /db001 and the global macro-variables. Table 2 contains the simple data listing for ID=10, 12, and 13; one data set follows another in the order of the data set name. Table 6 (at the bottom of this section) collects very useful information about variables of each data set in the database.

**TABLE 1**

<table>
<thead>
<tr>
<th>DATABASE</th>
<th>***** /db001 *****</th>
</tr>
</thead>
<tbody>
<tr>
<td>%let DN1   = ADR  ;</td>
<td></td>
</tr>
<tr>
<td>%let DN2   = DEMO ;</td>
<td></td>
</tr>
<tr>
<td>%let DN3   = PE   ;</td>
<td></td>
</tr>
<tr>
<td>%let START = 1   ;</td>
<td></td>
</tr>
<tr>
<td>%let END   = 3   ;</td>
<td></td>
</tr>
</tbody>
</table>

**TABLE 2**

<table>
<thead>
<tr>
<th>ID #: 10 12 13, DATABASE * /db001 *</th>
</tr>
</thead>
<tbody>
<tr>
<td>ID CODE TERM</td>
</tr>
<tr>
<td>10 101 headache</td>
</tr>
<tr>
<td>12</td>
</tr>
<tr>
<td>13 335 Nausea</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>DATA SET NAME: DEMO</th>
</tr>
</thead>
<tbody>
<tr>
<td>ID SEX AGE HEIGHT WEIGHT</td>
</tr>
<tr>
<td>10 2 35 74 200</td>
</tr>
<tr>
<td>12 1 11 68 140</td>
</tr>
<tr>
<td>13 2 48 58 200</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>DATA SET NAME: PE</th>
</tr>
</thead>
<tbody>
<tr>
<td>ID PE1 PE2 PSCOM</td>
</tr>
<tr>
<td>10 0 1 abnormal</td>
</tr>
<tr>
<td>12 1 1</td>
</tr>
<tr>
<td>13 0 0</td>
</tr>
</tbody>
</table>

**THE CONCEPT OF THE DB MACROS**

1. The first DB macro is the DB(DBDIR=./) macro. It extracts the names of data sets in the desired directory and assigns the data set names to a series of global macro-variables. Any subsequent DB macros can get the data set names from these global macro-variables. This macro also connects a SAS library reference with a directory which is provided by the user through the keyword parameter DBDIR=. The default directory is ./, meaning current directory in UNIX system. All subsequent DB macros use this library reference. Therefore, the DB(DBDIR=./) macro should be the first macro to be called before any other DB macros can be invoked.

For the purpose of easier reference and to allow flexibility when modifying the range or the order of data sets to be treated in subsequent DB macros, the DB(DBDIR=./) macro also outputs all codes of assigning the global macro-variables (as shown in table 1). In the case that the number of data sets is large and modifications are needed, the output can be copied into the macro invoking program, the values of the macro-variables then can be modified for subsequent DB macros to use (see the example in the DBPRT macro below).

In the framework indicated above, any job which is considered to be a common process for the data sets in any study database can be programmed into a specialized macro and be invoked when the job needs to be done. This paper provides seven macros which can be viewed as just a few examples.

2. The DBSORT(BY=id) macro is for sorting all data sets by variables given by the keyword parameter BY=. The default variable is id.

3. The DBCONT macro is for obtaining contents of all data sets.

4. The DBMEAN macro is for obtaining n, maximum, minimum, mean, and standard deviation of numerical variables of all data sets. These macros are fairly simple and straightforward.

5. The DBVAR macro is more complicated yet powerful. It combines the outputs of PROC CONTENTS, the outputs of PROC MEANS, and the actual length information of character variables. The macro produces a table for every data set in the database. Each table provides following information about the variables in a data set (see Table 6 at the bottom of this section).

For both character and numerical variables:

**TYPE** indicates numerical or character variable.
NOBS the numbers of observations.
N the numbers of non-missing (or non-blank) observations.
LENGTH the lengths of variables defined in a data set.
FORMAT the formats of variables.
INFORMAT the informats of variables.
LABEL the labels of variables.

For numerical variables:
MIN the minimum values of variables.
MAX the maximum values of variables.
MEAN the means of variables.
NMISS the numbers of observations with missing value for numerical variables.

For character variables:
MAXLEN the actual maximum lengths of variables.
NBLANK the numbers of observations with blank (missing) for character variables.

This table is very useful for validating data sets. By scanning NMISS, NBLANK or the extreme values (MIN and MAX), the user can quickly choose problematic variables from all data sets, and then investigate these variables to locate the individual cases (the DBERR macro can be used for this purpose). The DBVAR table also is very useful for formal data listings. The user can use the maximum length information (MAXLEN) to determine the appropriate space or a wrapping method for displaying a character variable, and use the MAX and MIN to estimate an appropriate format for displaying a numerical variable.

6. The DBDUPL(DSUST, VARllST) macro checks duplicate values of variables listed by position parameter VARllST in the data sets listed by position parameter DSUST.

For example, the following codes output duplicate id's for the 1st and the 2nd data sets in the directory /db001. The results are shown in Table 3. Note that the checking serve only as an example. In some data sets it is not necessarily wrong to have observations with duplicated id.

%DB(DBDIR=/db001);  
%DBDUPL(1 2, id)

Take another example. The invocation of the macro %DBDUPL(2 5 10, ID VISIT) will check the 2nd, 5th, and 10th data sets in the database respectively to see if there is any two observations which have the same values of ID and VISIT (the key variables). If there are duplicated cases, in terms of variables ID and VISIT, the macro will output all cases with the values of ID and VISIT. If there is no duplicate key variable, the output will say so. Note that both DSLIST and VARllST parameters allow unlimited listing of the items (data sets and variable names). The items should be separated by spaces.

### Table 3

**DATABASE ***** /db001 *****

---ADR: These are duplicate with the same id

<table>
<thead>
<tr>
<th>OBS</th>
<th>OUTCOME</th>
<th>ID</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>DUPLICATE</td>
<td>11</td>
</tr>
</tbody>
</table>

**DATABASE ***** /db001 *****

---DEMO: NO duplicate case with the same id

| OBS | OUTCOME |  
|-----|---------|-----|
| 1   | NO DUPLICATE CASE |  

7. The DBPRT(IDLIST, IDNAME=id) macro prints database for subjects listed in the position parameter IDLIST. The variable name for identifying the subjects is given by keyword parameter IDNAME=(id is the default variable name). The simple invocation of DBPRT will print data sets one by one in the order of the data set names (alphabetic order). Sometimes this order is not desirable, and or only part of the data sets need to be printed. By modifying the values of the macro-variables, the user can select the data sets to be printed and set the printing order of the data sets. The values of macro-variables also can include more title descriptions for each data set in addition to the default setting (the subject listing, the database and the data set name). For example, the page numbers of the Case Report Form are very helpful to understand the data set name and are desirable to be put into the title. The additional title can be captured by adding a ! sign and the text in the macro-variable assignment. The following is an example of the modification of the macro-variables:

%DB(DBDIR=/db001);  
%let _DN1=DEMO! Page 1;  
%let _DN2=PE! Page 2;  
%let _DN3=ADR! Page 3;  
%let _START=1;  
%let _END=2;  
%DBPRT(10 12 13);

The macro-variables _DN1, _DN2 and _DN3 rearrange the order of data sets to be printed which is DEMO, PE, and ADR. These macro-variables also carry the additional titles for the data sets (Page 1, 2, and 3). The
Posters

START=1 indicates the starting data set number is 1 (i.e. the 1st data set DEMO) and _END=2 indicates the ending data set number is 2 (i.e. the 2nd data set PE). The Page 1, Page 2 will appear in the outputs as shown on Table 4.

<table>
<thead>
<tr>
<th>ID</th>
<th>SEX</th>
<th>AGE</th>
<th>HEIGHT</th>
<th>WEIGHT</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>2</td>
<td>35</td>
<td>74</td>
<td>200</td>
</tr>
<tr>
<td>12</td>
<td>1</td>
<td>11</td>
<td>68</td>
<td>140</td>
</tr>
<tr>
<td>13</td>
<td>2</td>
<td>46</td>
<td>58</td>
<td>200</td>
</tr>
</tbody>
</table>

**----------------------------------------**

--- Page 2, DATA SET NAME: PE

<table>
<thead>
<tr>
<th>ID</th>
<th>PE1</th>
<th>PE2</th>
<th>PECOM</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>0</td>
<td>1</td>
<td>abnormal</td>
</tr>
<tr>
<td>12</td>
<td>1</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>13</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
</tbody>
</table>

---

8. The DBERR(COND=, DSNAME=, ERRTYPE=, PAGE=, VAR1=, VAR2=, VAR3=, VAR4=, IDNAME=id, OUTPUT=no) macro can generate a error report for a database. The report includes ID (the subject identification), ERRTYPE, PAGE, V1_VALUE to V4_VALUE. This macro is designed for checking a dirty database with complicity. In other words, it is very efficient for the situation when many variables in the database need to be checked, and the checking could involve any relation expression among variables cross over the database. Among 9 keyword parameters, only COND= is required for each call of the macro, and OUTPUT=yes is required for the last call of the macro to print out the checking report.

The COND= describes the condition to be checked out. It can be any SAS expression (the part after IF statement). The IDNAME= specifies the variable name for subject identification. The default is ID. The DSNAME= specifies the data set name(s) containing the variables to be checked. If more than one data set name are given, they will be merged by the variable from IDNAME. If DSNAME= is omitted, the whole database (or the data sets redefined by the macro-variables as described in the macro DBPR1) will be merged by the variable from IDNAME.

The ERRTYPE= is optional for specifying the error type that is the verbal description of the checking condition (SAS expression). This description is printed out under ERRTYPE. If ERRTYPE is omitted, the checking condition will be printed out.

The PAGE= is optional for specifying the page information of the Case Report Form containing those variables being checked. This information is printed out under PAGE. If PAGE= is omitted and DSNAME= is specified, the value of DSNAME= will be printed out.

The VAR1=, VAR2=, VAR3=, and VAR4= are optional for specifying the variable names. The values of these variables will be printed out under V1_VALUE, V2_VALUE, V3_VALUE and V4_VALUE respectively. They are designed for showing the error values of variables satisfying the checking condition(COND=).

The following is an example of using the DBERR macro:

```sas
%DB(DBDIR=ldb001);
%DBERR(COND=height > 72, DSNAMES=demo, PAGE= Page 1, VAR1=height);
%DBERR(COND=(sex=1 and pe1=0), DSNAMES=demo pe, VAR1=sex, VAR2=pe1);
%DBERR(COND=height < 60 and weight > 180, PAGE=Page 1, VAR1=height, VAR2=weight, OUTPUT=YES);
```

The results of the invocations of this macro are shown in Table 5. Note that the table is ordered by ID for easy comparing with the Case Report Form.

<table>
<thead>
<tr>
<th>OBS</th>
<th>ID</th>
<th>ERRTYPE</th>
<th>PAGE</th>
<th>V1_VALUE</th>
<th>V2_VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>10</td>
<td>height &gt; 72</td>
<td>Page 1</td>
<td>74</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>13</td>
<td>height &lt; 60 and weight &gt; 180</td>
<td>Page 1</td>
<td>58</td>
<td>200</td>
</tr>
<tr>
<td>3</td>
<td>None</td>
<td>(sex=1 and pe1=0)</td>
<td>demo pe</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
TABLE 6

DATABASE ***** /DB001 *****
INFORMATION OF VARIABLES --- ADR

<table>
<thead>
<tr>
<th>VARNUM</th>
<th>VARIABLE</th>
<th>TYPE</th>
<th>NOBS</th>
<th>N</th>
<th>NBLANK</th>
<th>MAXLEN</th>
<th>NMISS</th>
<th>MIN</th>
<th>MAX</th>
<th>MEAN</th>
<th>LENGTH</th>
<th>FORMAT</th>
<th>INFORMAT</th>
<th>LABEL</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>ID</td>
<td>Num</td>
<td>5</td>
<td>5</td>
<td>0</td>
<td>10</td>
<td>13</td>
<td>11.40</td>
<td>8</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>CODE</td>
<td>Num</td>
<td>5</td>
<td>4</td>
<td>1</td>
<td>1 101</td>
<td>335</td>
<td>189.75</td>
<td>8</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>TERM</td>
<td>Char</td>
<td>5</td>
<td>4</td>
<td>1</td>
<td>14</td>
<td></td>
<td></td>
<td>20</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

DATABASE ***** /DB001 *****
INFORMATION OF VARIABLES --- DEMO

<table>
<thead>
<tr>
<th>VARNUM</th>
<th>VARIABLE</th>
<th>TYPE</th>
<th>NOBS</th>
<th>N</th>
<th>NMISS</th>
<th>MIN</th>
<th>MAX</th>
<th>MEAN</th>
<th>LENGTH</th>
<th>FORMAT</th>
<th>INFORMAT</th>
<th>LABEL</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>ID</td>
<td>Num</td>
<td>4</td>
<td>4</td>
<td>0</td>
<td>10</td>
<td>13</td>
<td>11.50</td>
<td>8</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>SEX</td>
<td>Num</td>
<td>4</td>
<td>4</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>1.50</td>
<td>8</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>AGE</td>
<td>Num</td>
<td>4</td>
<td>4</td>
<td>0</td>
<td>11</td>
<td>48</td>
<td>29.75</td>
<td>8</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>HEIGHT</td>
<td>Num</td>
<td>4</td>
<td>4</td>
<td>0</td>
<td>58</td>
<td>74</td>
<td>66.25</td>
<td>8</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>WEIGHT</td>
<td>Num</td>
<td>4</td>
<td>4</td>
<td>0</td>
<td>140</td>
<td>200</td>
<td>172.50</td>
<td>8</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

DATABASE ***** /DB001 *****
INFORMATION OF VARIABLES --- PE

<table>
<thead>
<tr>
<th>VARNUM</th>
<th>VARIABLE</th>
<th>TYPE</th>
<th>NOBS</th>
<th>N</th>
<th>NMISS</th>
<th>MIN</th>
<th>MAX</th>
<th>MEAN</th>
<th>LENGTH</th>
<th>FORMAT</th>
<th>INFORMAT</th>
<th>LABEL</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>ID</td>
<td>Num</td>
<td>4</td>
<td>4</td>
<td>0</td>
<td>10</td>
<td>13</td>
<td>11.5</td>
<td>8</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>PE1</td>
<td>Num</td>
<td>4</td>
<td>4</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0.5</td>
<td>8</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>PE2</td>
<td>Num</td>
<td>4</td>
<td>4</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0.5</td>
<td>8</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>PECCM</td>
<td>Char</td>
<td>4</td>
<td>1</td>
<td>3</td>
<td></td>
<td></td>
<td>8</td>
<td>25</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

THE PROGRAMS OF THE DB MACROS

(1) The DB macro

```plaintext
%MACRO DB(DBDIR=./); ** assign the data set names in the directory to macro-variables; LITERAL IN "&DBDIR" ;

** define global macro-variables;
%GLOBAL_DIR ;
%LET_DIR=&DBDIR; ** for output title in other macros;
%GLOBAL_CNT ;
%LET_CNT=0; ** for creating working data set names in DBERR macro;
%GLOBAL_START END DN1 DN2 DN3 DN4 DN5 DN6 DN7 DN8 DN9 DN10 ;
%GLOBAL_DN11 DN12 DN13 DN14 DN15 DN16 DN17 DN18 DN19 DN20 ;
%GLOBAL_DN21 DN22 DN23 DN24 DN25 DN26 DN27 DN28 DN29 DN30 ;
%GLOBAL_DN31 DN32 DN33 DN34 DN35 DN36 DN37 DN38 DN39 DN40 ;
%GLOBAL_DN41 DN42 DN43 DN44 DN45 DN46 DN47 DN48 DN49 DN50 ;
%LET_START=1 ;

** get data set names of the database;
PROC CONTENTS DATA=IN , ALL NOPRINT OUT=DNAM(E(KEEP=MEMNAME)) ;
PROC SORT DATA=DNAM(E)_NODUP ;
BY MEMNAME ;
DATA NULL ;
SET DNAM(E) END=LAST ;
CALL SYMPUT('DN', LEFT(N, MEMNAME)) ;
IF LAST THEN CALL SYMPUT('END', LEFT(N_)) ;

** output the data set names and the macro-variables;
OPTION NODATE NONUMBER ;
TITLE ;
```

683
(2) The DBSORT macro

```plaintext
%MACRO DBSORT (BY=ID);  
** sort by specified variable for the database;  
%DO I=& start &to &end;  
  PROC SORT DATA=IN_&& DN%;  
  BY &BY;  
%END;  
%MEND DBSORT;  
```

(3) The DBCONT macro

```plaintext
%MACRO DBCONT;  
** print CONTENTS for the database;  
%DO I=& start &to &end;  
  TITLES "--- &DN&, THE CONTENTS OF DATA SET ";  
  PROC CONTENTS DATA=IN_&& DN% POSITION;  
%END;  
%MEND DBCONT;  
```

(4) The DMEAN macro

```plaintext
%MACRO DMEAN;  
** print MEANS for the database;  
%DO I=& start &to &end;  
  TITLES "--- &dn%, THE DESCRIPTIVE STATISTICS OF NUMERICAL VARS";  
  PROC MEANS N MIN MAX MEAN  
  DATA=IN_&& DN% MAXDEC=2 ;  
%END;  
%MEND DMEAN;  
```

(5) The DBVAR macro

```plaintext
%MACRO DBVAR;  
** output variable information of the database;  
OPTION MISSING=' ';  
PROC FORMAT;  
  VALUE TYPE 1='Num' 2='Char' ;  
** a macro to transpose a working data set from the output of PROC MEANS;  
%Macro TRAN(WDS);  
DATA &WDS;  
  SET &WDS ;  
  DROP _FREQ, _TYPE ;  
  PROC TRANSPOSE DATA=&WDS OUT=T&WDS ;  
  PROC SORT DATA=T&WDS(RENAME=(_NAME_=NAME));  
  BY NAME ;  
%MEND TRAN;  
** do loop for each data set in the database;  
%DO I=& start &to &end;  
  PROC CONTENTS DATA=IN_&& DN%  
  OUT=CONT__ NOPRINT ;  
  %LET NCHAR=0 ;  
  %LET NNUM=0 ;  
** count number of character variables -- NCHAR;  
** count number of numerical variables -- NNUM ;  
** get a data set CHAR__ for character variables;  
DATA CONT CHAR__;  
  SET CONT _END-LAST;  
  RETAIN NCHAR 0 ;  
```

684
IF TYPE = '1' THEN DO;
  NN = NN + 1;
END;
IF TYPE = '2' THEN DO;
  NC = NC + 1;
  OUTPUT CHAR_
END;
IF LAST THEN DO;
  CALL SYMPUT('NCHAR', LEFT(NC));
  CALL SYMPUT('NNUM', LEFT(NN));
** the LEFT function here is necessary for later program;  
END;
KEEP VARNUM NAME TYPE LENGTH FORMAT INFORMAT LABEL NOBS;
RUN;

** calculate maximum length and number of observations with blank
** for character variables. if no character variable then skip;
%IF &NCHAR=0 %THEN %GOTO TOMEAN ;
DATA DDD;
  RETAIN NBLK{&NCHAR} MAXL{&NCHAR} 0 ;
  SET IN (&NCHAR END=LAST);
  ARRAY A{&NCHAR} CHARACTER ;
** character variables are in the order of VARNUM;
  ARRAY NBLK{&NCHAR} NBLK{&NCHAR} ;
  ARRAY MAXL{&NCHAR} MAXL{&NCHAR} ;
  DO J=1 TO &NCHAR ;
    LEN=LENGTH(A{J}) ;
    IF A{J}=' ' THEN DO ;
      LEN=0 ;
      NBLK{J}=NBLK{J}+1 ; %* count blank for each char var;
    END ;
    IF LEN > MAXL{J} THEN DO ;
      MAXL{J}=LEN ;
    END ;
  END ;
  IF LAST THEN OUTPUT ;
** after transpose, the TBLKS____ is in the order of VARNUM;
  PROC TRANSPOSE DATA=DDD (KEEP=NBLK{&NCHAR} MAXL{&NCHAR}) OUT=TBLKS ;
  PROC TRANSPOSE DATA=DAA (KEEP=NBLK{&NCHAR} MAXL{&NCHAR}) OUT=TMAXL ;
** make data CHAR___ in the same order as TBLKS____ and then merge;
  PROC SORT DATA=CHAR ;
  BY VARNUM ;
DATA CHAR ;
  MERGE CHAR (KEEP=NAME NOBS)
  TBLKS  (RENAME=(COL1=NBLANK))
  TMAXL (RENAME=(COL1=MARGIN)) ;
  N=NOBS-NBLANK ;
  KEEP NAME N NBLANK MAXLEN ;
  PROC SORT DATA=CHAR ;
  BY NAME ;
  PROC SORT DATA=CONT ;
  BY NAME ;
%TOMEAN: ** get statistics for numerical variables. if no numerical variable then skip;
%IF &NNUM=0 %THEN %GOTO TOMEAN ;
PROC MEANS DATA=IN (KEEP=NAME NOBS) NO PRINT;
  OUTPUT OUT=N N=;
  OUTPUT OUT=MEAN MEAN ;
  OUTPUT OUT=MIN MIN =;
  OUTPUT OUT=MAX MAX =;
  OUTPUT OUT=NMIS NMIS =;
%TRAN(N);
%TRAN(MEAN);
%TRAN(MIN);
%TRAN(MAX);
%TRAN(NMIS);

%FINAL:    ** combine all information, merge by variable names;
  DATA DDD
  MERGE
  %IF &NCHAR > 0 %THEN
    CHAR;
  %IF &NNUM > 0 %THEN %DO;
    %IN (RENAME=(COL1=N))
    %TMEAN (RENAME=(COL1=MEAN))
    %TMIN (RENAME=(COL1=MIN))
    %TMAX (RENAME=(COL1=MAX))
    %TNMISS (RENAME=(COL1=NNMISS))
  %END;
  CONT____;
  BY NAME;
  RENAME NAME=VARIABLE;
  PROC SORT;
  BY VARNUM;

** print the combined data;
  PROC PRINT NOOBS;
  TITLE4 "INFORMATIONS OF VARIABLES --- &DDD&I";
  FORMAT TYPE. ;
  VAR VARNUM VARNUM TYPE NOOBS N
  %IF &NCHAR > 0 %THEN NBLANK MAXLEN ;
  %IF &NNUM > 0 %THEN NMISS MIN MAX MEAN ;
  LENGTH FORMAT INFORMAT LABEL ;
  %END;
  OPTION MISSING='.' ; ** recover the default setting;
  %END DBVAR ;

(6) The DBDUPL macro

%MACRO DBDUPL(DSLIST, VARLIST);
  ** find duplicate cases according to key variables listed in VARLIST for
  ** data sets listed in DSLIST;
  %LOCAL FLAG I J K L D;
  %LET FLAG=1;
  ** count K, the number of items in DSLIST;
  %LET I=1;
  %DO %WHILE (%LENGTH(%SCAN(&DSLIST, &I)) NE 0 );
    %LET I=%EVAL(&I+1);
  %END;
  %LET K=%EVAL(&I-1) ;
  ** count L, the number of items in VARLIST, get the key variable names;
  %LET I=1;
  %DO %WHILE (%LENGTH(%SCAN(&VARLIST, &I)) NE 0 );
    %LET KEY&I=%SCAN(&VARLIST, &I);
    %LET I=%EVAL(&I+1);
  %END;
  %LET I=%EVAL(&I-1) ;
  ** do loop for each data set in DSLIST;
  %DO I=1 %TO &K ;
    %LET D=%SCAN(&DSLIST, &I) ;
    PROC SORT DATA=IN___&DDD&I OUT=DDD___;
    BY &VARLIST ;
    DATA DDD__;
      SET DDD___ END=LAST;
      LENGTH _OUTCOME $20 ;
      RETAIN _OUTCOME ' ' ; ** this line is needed for no duplicate cases;
  ** set the condition for finding out duplicate cases, use the LAG function;
    IF &KEY1=LAG(&KEY1)
      %IF EL > 1 %THEN %DO;
        %DO J=2 %TO EL ;
          AND &KEY&J=LAG(&KEY&J)
          AND &KEY&J=LAG(&KEY&J)
%END;
%END;
THEN DO;
   OUTCOME='DUPLICATE';
   OUTPUT;
END;
   IF LAST AND OUTCOME=' ' THEN DO;
   OUTCOME='NO DUPLICATE CASE';
   OUTPUT;
   CALL SYMPUT('FLAG', 'NODUP');
   END;
%IF &FLAG=NODUP THEN
   TITLE4 "-- &_DN&N: NO duplicate case with same &VNU.IST" %STR(;);
%ELSE
   TITLE4 "-- &_DN&N: These cases are duplicate with same &VNU.IST" %STR(;) ;
** print out the values of key variables of duplicate cases;
PROC PRINT;
   VAR QUTC.'CIIIE
   %IF-&FLAG NE NODUP %THEN 'oDO;
   &VARLIST
   %-END;
%-END;
%-END DBDPT;

(7) The DBPRT macro
%Macro DBPRT(IDLIST, IDNAME=); %** print data with specified id(s) for the database;
OPTIONS NODATE NONUMBER;
%** count J, the number of items in IDLIST;
%LET I=1;
%DO %WHILE (%SCAN(&IDLIST, &I) NE );
   %LET I=%EVAL(&I+1);
%END;
%LET J=%EVAL(&I-1);
%** do loop for each data set;
%DO I=&_START %TO &_END ;
%** separate the data set name and the descriptive title;
   %LET_DNN&I=%SCAN(&& DN&I, 1, 1)
   %LET _PAGE=%SCAN(&& DN&I, 2, 1)
   %IF &J=1 %THEN %DO;
      TITLE2 "&IDNAME #: &IDLIST, DATABASE * &DIR *" %STR(;) ;
   %END;
   %ELSE TITLE2 %STR(;)
   TITLE3 "---- _PAGE, DATA SET NAME: && DNN&I ";
%** choose id(s) to be printed out;
   PROC PRINT DATA=IN____.&& DNN&I UNIFORM DOUBLE
   ID &IDNAME;
   WHERE &IDNAME=%SCAN(&IDLIST, 1)
   %IF &J > 1 %THEN %DO;
      K=2 %TO &J;
   &IDNAME=%SCAN(&IDLIST, &K)
   %END;
%END;
%END DBPRT;

(8) The DBERR macro
%Macro DBERR(COND=, DSNAMES=, ERRYPE=, PAGE=, VAR1=, VAR2=, VAR3=, VAR4=, IDME=id, OOTPUI'=no);** generate an error report for a database;
%LET _CNT=%EVAL(&CNT+1);
** _cnt is a global macro-variable used here to name working data sets for
** every call of the macro. the initial value is 0 set by the DB macro;
DATA &_CNT;
  LENGTH ID $6 ERRTYPE $40 PAGE $20 ;
  RETAIN PAGE ERRTYPE ID _ID ;

** merge the data sets for checking error;
MERGE
  ** if _DSNAMES" NE " THEN DO;
  %LET I=1 ;
  %DO %WHILE(%LENGTH(%SCAN(&_DSNAMES, &I)) NE 0);
    %LET DS&I=%SCAN(&_DSNAMES, &I);
    IN . &DS&I
    %LET I=%EVAL(&I+1);
  %END;
  %END;
%ELSE %DO;
  %DO I=&_START %TO &_END ;
    IN . &_NAME &I
  %END;
  %END;
  END=LAST ;
  BY &_NAME;

** assign value for variables: page and errtype;
  ** if _PAGE" = " THEN page=" &PAGE" %STR();
  %ELSE PAGE="&_DSNAMES" || " &STR();
  IF _ERRTYPE" = " THEN ERRTYPE="&_ERRTYPE" %STR();
  %ELSE ERRTYPE="&COND" %STR();

** if the condition is true then output id and values;
IF &COND THEN DO ;
  %IF &VAR1 = %THEN V1.VALUE=&VAR1 %STR();
  %IF &VAR2 = %THEN V2.VALUE=&VAR2 %STR();
  %IF &VAR3 = %THEN V3.VALUE=&VAR3 %STR();
  %IF &VAR4 = %THEN V4.VALUE=&VAR4 %STR();
  ID = &_NAME;
  OUTPUT ;
END;

** if no case is found then output final line;
IF LAST AND _ID EQ ' ' THEN DO ;
  ID = 'NONE'
  OUTPUT ;
END;
KEEP _ID PAGE ERRTYPE
  %IF &VAR1 = %THEN V1.VALUE ;
  %IF &VAR2 = %THEN V2.VALUE ;
  %IF &VAR3 = %THEN V3.VALUE ;
  %IF &VAR4 = %THEN V4.VALUE ;

** if output is required then merge data and print out the report;
%IF %UPCASE(&OUTPUT)=YES %THEN %DO ;
DATA SHOW ;
  %DO I=1 %TO &_CNT ;
    &I
  %END ;
  OPTION MISSING = ' ';
  PROC SORT ; BY _ID ;
  %LET CNT=0; /** after printing out, reset _CNT=0 for possibly more call of DBERR;
  %END;
  OPTION MISSING='.';
  %END DBERR

688
CONCLUSION

It is desirable to design a set of macros with minimum of parameters and with the ability to handle any clinical database. The DB macros presented in this paper can greatly reduce laborious programming for routine tasks such as obtaining crucial information about variables, checking data error, and listing sample data for databases. Some of the DB macros can be applied directly to other SAS databases (SAS data libraries).

REFERENCES


ACKNOWLEDGEMENTS

Special Thanks to Yingcong Zheng, Lou Anne Amerman, and Sinh Luu for valuable comments and suggestions.

SAS is a registered trademark of SAS Institute Inc. Cary, NC, USA. Other brand and product names are registered trademarks or trademarks of their respective companies.

AUTHOR CONTACT

Sam X. Ye
Whitehall-Robins Inc. AHP Corp.
Dept. of Clinical Research
5 Giralda Farms
Madison, NJ 07940