Paper 104-2011

Who Do You Have? Where Are They?

William C. Murphy
Howard M. Proskin & Associates, Inc., Rochester, NY

ABSTRACT

Questions like this are often asked when dealing with a database consisting of more than one SAS® data set: What subjects do you have? In what data sets do they provide data? Is the data in the different data sets consistent? If you had only 2 or 3 small data sets, you might just open them up with the SAS viewer and resolve these questions by inspection. However, when your database consists of dozens, if not hundreds, of data sets, your approach must be different. In our analyses, we use the SAS DICTIONARY tables to obtain lists of the relevant data sets in our database. Using this information, along with a macro DO-loop, we are able to create cross tabulations of the database. These tabulations not only provide lists of the subjects that are present in the different data sets but also can be used to find inconsistencies among the common variables. A simple macro program that produces these tabulations will be presented.

BACKGROUND

In our company, we build databases and perform analyses of data obtained from clinical trials. Sometimes this work involves only a handful of data sets, but usually there are dozens or more. As part of our work, we report on the content of this database. Furthermore, we must be sure that the data is relatively 'clean' and consistent.

INTRODUCTION

In the larger clinical trial studies, large volumes of data are generated from different source. Each source usually maintains its data in a separate database. To do analyses of these data, they are combined into one large SAS database with data sets representing the individual components. These component data sets may include such things as demographics, physical characteristics, physical exams, adverse events, medications, lab results, and a variety of study specific data sets. When such a database is created, inevitably someone will ask which subjects are represented in the study and in which data sets do they have data. Furthermore, the consistency of such things as dates and gender across the data sets is questioned. In a small database with few data sets, these issues can usually be resolved by inspecting the data sets in the SAS viewer. Alternately, you might use a DATA step with MERGE to combine the data and check for subject occurrence and consistency. However, for really large data sets, another approach must be used.

DICTIONARY

The first thing in checking your database is to determine exactly what data sets it is composed of. To do this, the SAS system provides programmers with a group of DICTIONARY tables which list all sorts of information about your database. For example, if we wanted to know the components of our database we could write

```
proc sql noprint;
       create table Datasets as
              select memname
              from dictionary.Members
              where upcase(libname)="LEVEL1";
       quit;
```

where we have accessed the SAS DICTIONARY table Members which contains information about the members in active libraries. We have restricted ourselves to the library LEVEL1 with the WHERE statement. For a different database, you would use the WHERE to point to the appropriate library.

Now this list of library members provides us with a starting point for creating a check list of the subjects present in our database, but two things must be taken into consideration. First, the SAS DICTIONARY tables contain information on all members, including such thing as format and macro catalogs. For the present, we are only interested in data sets. Secondly, we are only interested in data sets containing references to the subject identification variable. This

means we should ignore data sets containing such information as visit timing, investigators' names and addresses, and various accounting details.

To do this we turn to another DICTIONARY member called Columns which contains not only a list of the members, but also a list of each variable contained in the various members:

```
proc sql noprint;
	create table Datasets as
		select memname
		from dictionary.Columns
		where upcase(libname)="LEVEL1" and memtype='DATA'
				and upcase(name)='SUBJID'
		;
	quit;
```

where we have now generated a data set which contains a list of the component data sets (memtyper='DATA') that contain the subject identification (upcase(name)='SUBJID') from our database (upcase(libname)="LEVEL1").

With this list in hand we are ready to check for subject occurrence across the database.

SUBJECT CHECKLIST

Each data set in the Datasets table will be converted to macro variable and processed to generate a table in a manner that has been previously specified (Murphy 2007). Within a macro we now open our list of data sets and save the number of observations into a macro variable:

```
%let id=%sysfunc(open(work.Datasets));
%let nDataSets=%sysfunc(attrn(&id,NOBS));
```

Now we initialize the variables in Datasets as macro variables:

```
%syscall set(id);
```

Now we can loop over each data set represented in our list:

```
%do i=1 %to &nDataSets;

	%let rc=%sysfunc(fetchobs(&id,&i));

	data _1;
		length Source $32;
		set LEVEL1.&memname(keep=SubjID);
		Source="&memname";
		run;

	proc sort data=_1 nodupkey;
		by SubjId Source;
		run;

	proc append force base=_Accum data=_1;
		run;

	%end;
```

We open each selected data set in LEVEL1 into the data set _1, keeping only the identification information for the subject and the name of the data set from whence it came (Source). Data set _1 is then sorted to get rid of duplicates. This information is then appended into the data set _Accum. The FORCE option is used in the APPEND procedure to assure successful appending even if the attributes of the variable SubjID vary among the LEVEL1 data sets

The data set _Accum has everything we need to generate a subject check list. It has all the subjects in our database and an entry for ever data set that they occur in. We could now easily generate a report:

```
proc tabulate data=_Accum;
	class subjid source/style=[just=center];
```

```
        table subjid=' ', source=' '*n=' '*f=1.
              /rts=14 box=subjid misstext=' ';
        run;
```

The output from the TABULATE procedure would look something like

```
------------------------------
|Subject     | | | | |P|P| | |
|Number      | | | | |H|H| |M|
|            | | | | |Y|Y|S|E|
|            | |D| | |S|S|T|D|
|            | |E| |M|C|E|A|H|
|            | |M|L|E|H|X|T|I|
|            |A|O|A|D|A|A|U|S|
|            |E|G|B|S|R|M|S|T|
|------------+-+-+-+-+-+-+-+-|
|753-003     | |1| | |1|1|1| |
|------------+-+-+-+-+-+-+-+-|
|753-005     |1|1|1|1|1|1|1|1|
|------------+-+-+-+-+-+-+-+-|
|903-002     |1|1|1|1|1|1|1|1|
|------------+-+-+-+-+-+-+-+-|
|903-007     | |1| | | | | | |
------------------------------
```

where the data set names from LEVEL1 are written vertically as column headers and the subject identifications are the row labels. A one is displayed if information on the subject is contained in the data set and a blank is displayed otherwise. Of course in a large study, this would have several more columns and span many pages.

ERROR CHECKING

The same procedure for generating a subject checklist can also be used to check for consistency across the database. For instance, we came up with a gender discrepancy in a recent study. The subjects' sex in the demographics file was different from the lab file. In turn, there were more differences with the physical characteristics file. Since there were a lot more data sets where this problem might have occurred, we turned to the method given above.

First we generate a list of the data sets in our database that may be involved:

```
proc sql noprint;
      create table Datasets0 as
            select memname, name
            from dictionary.Columns
            where upcase(libname)="LEVEL1" and memtype='DATA'
                        and upcase(name) in ('SUBJID',GENDER');
      quit;
```

You will notice that this is almost identical to the code given above. We have only demanded that the selected data sets now contain the variable SUBJID or GENDER and we retain the variables names in our output (name), and we have also named the table created Datasets0. We now must make sure that both SUBJID and GENDER occur together:

```
proc summary nway data=Datasets0;
      class memname ;
      output out=Datasets (keep=memname _freq_ where=(_freq_=2));
      run;
```

where our desired list of data sets must contains both variables of interest (*i.e.* _freq_=2). The output is now presented in Datasets.

We can now write a macro loop similar to the above. We merely replace DATA step _1 with

```
data _1;
       length Source $32;
       set LEVEL1.&memname(keep=SubjID Gender);
       Source="&memname";
       run;
```

You could than run a PROC TABULATE, as above, taking into account the variable gender or you could do further processing to find the inconsistencies:

```
proc sort data=_Accum;
       by SubjID Gender;
       run;

data Lists;
       set _Accum;
       by Subjid Gender;
       length DataList $1000;
       if first.Subjid then DataList=memname;
       else DataList=catx(',',memname,DataList);
       if last.SubjID then output;
       run;
```

where the data set Lists contains an entry for each unique combination of the subject identification and any associated genders, along with a list (DataList) of the data sets that contain this gender. If you wanted to isolate the problems you could write:

```
proc sort data=Lists;
       by SubjID;
       run;

data Problems;
       set Lists;
       by SubjID;
       if not (First.SubjID and Last.SubjID);
       run;
```

Subjects with the correct gender will only have one entry in Lists. Consequently, by choosing observations that are not both first and last, the data set Problems will only contain subjects with gender differences across the database, along with the associated data sets. The resulting data set would look like

	Subjid	Gender	DataList
1	753-003	F	DEMOG,LABS,MEDS,MEDHIST,PHYSEXAM,STATUS
2	753-003	M	PHYSCHAR
3	903-002	F	LABS,MEDHIST
4	903-002	M	DEMOG,MEDS,PHYSCHAR,PHYSEXAM,STATUS

As you can see two subjects are presented with different genders in a variety of data sets. Subjects that did not appear in this data set would have a consistent gender across the components of the database. Hopefully the data set Problems would remain small even if the study involved thousands of subjects.

AND MORE ERROR CHECKING

Of course gender is not the only thing that may cause problems, but this method can be use to check for the consistency of any value. For example, if you replaced the variable Gender in the above code with BirthDate, you could come up with a list of subject with birth date issues. In a similar vein, you could replace the gender variables with variables containing the subjects name, examination site, race, or any other person-specific information, and come up with a list of the problem subjects.

You need not consider just subject information; you could use the method above to examine such things as visit date discrepancies Merely replace the subject identification with both the subject identification and the visit number, and

replace the gender variable with the visit date variable. Then run through the delineated procedure to generate a Problems data set with data issues.

CONCLUSIONS

The information that the SAS system provides in the DICTIONARY tables is a value resource for the programmer. This information along with some macro programming can be used to create check lists that display both the content and the inconsistencies of databases composed of numerous data sets.

REFERENCES

Murphy, W.C. (2007), "Changing Data Set Variables into Macro Variables", *Proceedings of the SAS® Global Forum 2007 Conference*, SAS Institute Inc., Cary, NC, paper 050-2007. (url: http://www2.sas.com/proceedings/forum2007/050-2007.pdf).

APPENDIX

The following code is the complete macro program for generating a table of subject identifications by data set names. In this code we have assumed that the subject is identified by SUBJID and the database resides in the library LEVEL1. These variables should be changed to reflect the user's circumstances. The reader could also generalize the macro by introducing 2 macro parameters that point to these variables.

```
%macro CheckID;

	%local id nDataSets i rc memname;

	%*** Make List of Data Sets ***;
	proc sql noprint;
		create table Datasets as
			select memname
			from dictionary.Columns
			where upcase(libname)="LEVEL1" and memtype='DATA'
					and upcase(name)='SUBJID'
			;
		quit;

	%*** Get Rid of Accumulation Data Set (_Accum) ***;
	%if %sysfunc(exist(work._Accum)) %then %do;
		proc datasets nolist library=work;
			delete _Accum;
			quit;
		%end;

	%*** Loop Through Each Data Set in Datasets ***;
	%let id=%sysfunc(open(work.Datasets));
	%let nDataSets=%sysfunc(attrn(&id,NOBS));

	%syscall set(id);

	%do i=1 %to &nDataSets;

		%let rc=%sysfunc(fetchobs(&id,&i));
		%put ..... Processing &memname;
		%put ..... (&i of &nDataSets Data Sets);

		%*** Make List of IDs in Each Data Set ***;
		data _1;
			length Source $32;
			set LEVEL1.&memname(keep=SubjID);
			Source="&memname";
			run;

		%*** Get Rid of Duplicates ***;
```

```
        proc sort data=_1 nodupkey;
               by SubjId Source;
               run;

        %*** Accumulate Results in _Accum ***;
        proc append force base=_Accum data=_1;
               run;

        %end;

%*** Make Table of ID Versus Data Set Name ***;
proc tabulate data=_Accum;
        class subjid source/style=[just=center];
        table subjid=' ', source=' '*n=' '*f=1.
                     /rts=14 box=subjid misstext=' ';
        run;

%mend;
```

After loading the macro, just submit the following code to execute:

```
%CheckID;
```

CONTACT INFORMATION

Your comments and questions are values and encouraged. Contact the author at

William C. Murphy
Howard M. Proskin & Associates, Inc.
300 Red Creek Dr., Suite 220
Rochester, NY 14623
Phone 585-359-2420
FAX 585-359-0465
Email wmurphy@hmproskin.com or wcmurphy@usa.net
Web www.hmproskin.com