Creating and Maintaining a Central SAS® Library for Health Care Management

William C. Murphy
Rochester Community Individual Practice Association, Rochester, NY

ABSTRACT

In our organization, several programmers are working on the same data sets simultaneously. These data sets contain the health care information for over a half a million individuals and can be quite large. In order to prevent the data sets from becoming even larger, simple codes are used to identify the physician or other health service provider, the patients' diagnoses, and the medical procedures undergone. However, in the generation of various reports, more descriptive information is needed about these parameters. This supplemental information is contained in separate data sets. To make this information available, a format library is created from these data sets which maps the codes to the pertinent information. In order to maintain consistency, the library is kept in a central location on our network and is accessible to all users. Furthermore, certain repetitive chores that are done by all the programmers are maintained in a central macro library. The construction of such a library takes careful planning but can save an enormous amount of disk space and programmer effort. Errors are also minimized and updating the data is easy when the library information is centralized.

INTRODUCTION

Our organization is an association of over two thousand physicians and other related professionals who provide health care for over half a million individuals. To help manage the care for our members, several programmers are constantly analyzing and studying the insurance claims data. Their work involves utilization and quality reporting and the examination of health trends. The claims data set is quite large and contains most of the information on the health care providers, the patients' diagnoses, and the medical procedures in terms of short codes. In the generation of reports, however, it is often necessary to know such things as the providers' specialty or insurance plan participation status. Furthermore, a report looks far more professional with the name of the provider and the medical procedures utilized rather than code numbers. To accomplish this, we must access the databases containing the provider information, and the diagnosis and procedure descriptions. We could do this by merging the necessary data for each report when we needed it. However, sorting and merging require extra processing time and the resulting data sets require more storage space. Also any changes in this information could be a nightmare to upgrade on each user's computer and reports. To solve these problems, we use the provider information and diagnosis and procedure descriptions to create format libraries. These libraries are maintained in a central location on our network.

Furthermore, there are a number of routines that are used often and by all the programmers. These programs include such things as encoding patient identification for confidentiality purposes, listing out the claims files, and subsetting the claims by age criteria. Because they are used so often, they were written as macros and, as in the case of formats, stored in a central library.

COMPUTER SYSTEM BACKGROUND

All of our analysis work is done on Pentium Pro PC's with at least 20 gigabytes of hard disk space operating under Windows/NT 4.0. The PC's are connected to a Novell network. The central file server where we currently maintain our library is also a PC with a 3 gigabyte hard disk. Plans are being made to upgrade the network to a more powerful Windows/NT server with faster data lines. The library will be moved to the new server when it is online.

FORMAT FILE CREATION

To create our format file, we first define our library:

```
libname provs 'e:\input' ;
```

We designated this library on our local computer. When we try to create a format library over a Novell network, our program generated various error statements. This condition only seems to show up when the contents of the format library passes a few hundred values. Apparently there is some incompatibility between the SAS system and Novell. Once the library is created locally, it is moved to the network.

Next we read the provider data into a SAS data set:

```
data one;
  set provs.provref;

  type='C';
  start=provid;

  array xname{7} $ _temporary_ ('fulname' 'p1spec'
             'p2spec' 'prvdea' 'prvname' 'prvstat' 'prvtype');
  array xlabel{7} $  fullname  pspec1   pspec2   dea
           provname  pstatus   provtype;

do i=1 to 7;
  fmtname=xname{i};
  label=xlabel{i};
  if label='' then hlo='O';
    else hlo='';
  output;
  end;

keep fmtname type start label hlo;
```

The provider reference file, provref, contains all of the information on our health care providers, listed by the provider identification number, provid. The data set created is a control file for creating formats. All of our format types are character and all of them map to the same start variable, i.e. the identification code for the provider contained in the variable provid. The format names are created using a temporary array statement with contents set to the format names. An arrray of variables containing the format labels is assigned to the appropriate variables in the provider data. Seven different formats are thus created: the providers full name (fulname), primary speciality (p1spec), secondary specialty (p2spec), drug number (prvdea), the lastname first (prvname), status (prvstat), and type (prvtype). The logical statement in the do-loop is to make sure that any variables with a missing entry are grouped together and mapped to an 'Other' format (hlo='O'). The data set is then sorted by format name:

```
proc sort data=one;
  by fmtname;
```

This is done to assure that the format names are grouped together. This is what the SAS system expects when creating a format library. Finally, we write

```
proc format cntlin=one library=prov;
  run;
```

This creates a permanent format library from the control data contained in data set one.

A similar procedure is used to create a format library containing the descriptions of the diagnoses and procedures. We map the diagnosis code to the diagdes format and the procedure code to the procdes format. This is done in a different format library from the provider formats.

NETWORKING

Once the format libraries are created, they are ready to be positioned on the network drive. However, this is done in several different areas. The codes for procedures and diagnoses do not change very often and are consequently grouped together and put in their own directory. The provider information can change quarterly or even more often.

On the programmers' computers, we define the libraries as

```
libname procdiag 'r:\library\procdiag'
          access=readonly;
libname provlib   'r:\library\prod'
          access=readonly;
```

where r: is mapped to our network drive and the directory library is our shared information. The procedures and diagnoses code formats are placed in procdiag and the provider information formats are placed in the production library, prod. For those unfamiliar with SAS on a network, you must place the restriction access=readonly on the libname statements of all the people who will access the libraries. This allows simultaneous multiple access of the libraries by several programmers. If so much as one user has other than readonly access, no one will be able to open the libraries. In addition to the libname statements, the programmer must also change the option so that the system knows to search those libraries for formats:

```
option fmtsearch=(procdiag,provlib);
```

All of these SAS statements are placed in the autoexec.sas program on the programmers' computers. Consequently, accessing this information in a SAS program is nearly transparent.

Since the provider library can change quarterly and we sometime are required to do historical analysis, we must maintain the old format libraries. Before a new production library is installed, the old format library catalog file is renamed, for example, to fmtq197 which would contain the formats used in the first quarter of 1997. To use these old formats, we rewrite the option statement as

```
option fmtsearch=(procdiag,provlib.fmtq197);
```

This can also be made transparent by placing this statement in the autoexec.sas file, however, it is probably best to maintain the original autoexec.sas in your production folder and a different autoexec.sas for historical research in its own folder.

MACROS

In addition to using the same format information on providers and procedures, the programmers share a variety of standard programs. These include such things as id scramblers/descramblers to assure patient confidentiality, reformatting tools to make the diagnosis codes comply to different standards, and various reporting tools. All of these codes are written in the form of macros and stored in the central library.

Storing macros is fairly simple. First you must define your macro library in a libname statement and then change your system options to allow stored macros:

```
libname  macs ‘e:\input’;
option mstored sasmstore=macs ;
```

You then write your macro:

```
%macro encode/store;
     .
     .
     .
%mend encode;
```

When you run this macro, the store option on the macro line will cause a compiled version of the macro to be saved in the library indicated by the sasmstore system option. Unlike the creation of the format library, we experienced no problems on writing directly to the network. However, it is probably best for data testing and updating to create the macros locally and then copy them to the network. On the network, we usually store the macro library with the procedure and diagnosis codes, since they also do not change very often. Furthermore, you can use the same libname, procdiag, in the sasmstore option. It should be noted that the macro system option must also be installed on all of the systems accessing the macros. This can also be put in your autoexec.sas file.

CONCLUSIONS

Using a central library for SAS formats and macros can be both a time and space saver. It avoids the unnecessary waste of maintaining additional information in data sets beyond the codes for the providers, the diagnoses, and the procedures. It makes sorting and merging unnecessary. Programmers do not have to rewrite standard code that is contained in the macros. Consistency is maintained across all programmers and updating the information is easy since it is centrally located. Creating descriptive health care management reports under this system is easy and transparent to the user.

ACKNOWLEDGEMENTS

The author would like to thank Mary MacDougall for her comments and suggestions during the preparation of this paper.

CONTACT

William C. Murphy
Rochester Community Individual Practice Association
16 Main Street West
Suite 800
Rochester, NY 14614
Phone 716-454-1490 ext 257
E-mail bmurphy@rcipa.com