Cross-referencing Data Contents and Case Report Form
Sam Mao, Quintiles, Inc., Kansas City, MO

ABSTRACT
Annotated case report forms (CRF), data contents, and data formats are the core documents for statistical programmers to understand the data collection, the relationship between variables, and variable properties and coding. CRF and data contents are often cross-referenced. Output from PROC CONTENTS does not facilitate quick and easy cross-reference between CRF pages and data contents. This paper introduces the macro %CNTFMT to create customized data contents, which is more compacted but information richer, and allow quick cross-referencing between CRF pages and data contents. Other features of the macro, which can meet a programmer’s reference needs, will also be presented. Those features include getting study data-specific format decodes from multiple levels of format libraries, contents of partial library and so on. Macro-programming tips will also be shared. The macro has been tested and used with both SAS Version 6 and 8. The paper is suitable for all levels of audience.

Key words: data contents, format decodes, CRF, system options

INTRODUCTION
To create tables, listings and figures for clinical study, statistical programmers reference a variety of documents including both general references and project/study specific references. Examples of general references are FDA, CDISC, ICH, and company wide guidelines. Project/study specific references include protocol, case report forms (CRF), data analysis and report plan (DARP), analysis file specification, table shells, contents of SAS datasets, and format decodes etc. Among these references, data contents and format decodes are critical for programmers to get to know the data. To understand how the data was collected and in which way the variables and datasets is related, the CRF often needs to be referenced together with data contents, i.e. cross referencing of data contents and CRF. While the PROC CONTENTS procedure of SAS allows the programmers to get data contents, the output of the procedure is somewhat redundant, and sometimes doesn’t meet the needs in a fast paced environment. For example, 1) the dataset name is somewhat redundant, and sometimes doesn’t meet the needs in a fast paced environment. For example, 1) the dataset name is embedded in the text in upper left corner of each contents output. This makes it difficult to find the contents page of a particular dataset one wants to reference. 2) The contents does not have information about CRF pages in which the data was collected, so one cannot quickly go to the CRF pages corresponding to the datasets. 3) Though the name of format associated with a variable is available from data contents, the format decodes are not. Though PROC FORMAT procedure can be used to get decodes from format library, using PROC FORMAT procedure to access format library in a system established for years can face challenges. These challenges may include huge output, listing all the format names manually in SELECT statement and one procedure call for one format library reporting only. These challenges will be detailed later. Macro %CNTFMT is developed to allow a user to get both customized contents, which facilitates efficient cross referencing, and format decodes with little effort.

FUNCTIONALITIES OF THE MACRO AND PARAMETERS:
Initially, the macro was developed to facilitate cross-referencing of data contents and CRF. After a period of use in the production environment, the macro has evolved to meet various needs. Currently, the macro has the following functionality:
1. Create a Summary of Contents page with CRF page numbers listed for each datasets.
2. Contents of each dataset facilitate quick cross-referencing of contents ↔ CRF pages. To achieve this, the contents output has CRF page number(s) for each dataset in the title line, and dataset name(s) on the 4 corners of each contents pages.
3. Option to generate contents of one or more selected datasets instead of whole library of datasets.
4. Options to generate sample data listing following contents of each dataset.
5. Drop database system variables to make sample data listing more concise.
6. Searches through the format path to get study, data-specific format decodes.
7. Option to tailor large formats to get study, data-value specific format decodes.

To achieve the above functionality, the macro is designed to have the following parameters:

%cntfmt(datalib=, data=, pgvar=crfpgno, contype=BR, numobs=, dropvar=, schfmt=libsys1 libsys2, lgfmt=lbcd lbunits)

![Figure 1: Layout of Summary of Contents from macro %CNTFMT](image)
IMPLEMENTATION
1. Summary of Contents
The first page of the output from PROC CONTENTS is a column of member names in a library and a column indicating the type of each member. This information is too brief. When using the DETAIL option, then three more columns (dataset label, No. of variables and No. of observations in each dataset) will be added to the first page. This gives a good overview or summary of the datasets in a library. For simplicity, that first page is named as Summary of Contents in this paper. Macro %CNTFMT adds to the Summary of Contents two more columns (Figure 1). These are dataset creation time and CRF page numbers from which the data of each dataset is collected. Thus, the Summary of Contents can also serve as the detailed index of CRF. Dataset creation time can be obtained from dictionary.tables, while getting CRF page number will be explained in next section.

2. Concise contents for quick cross-referencing
The output of PROC CONTENTS for each member dataset has two parts. Part one is meta data about dataset and host/engine information which contains dataset name, label, creation date and time, no. of variables and observations, and file location, etc. Part two is meta data about variables which contains alphabetic list of variables and their attributes. While part two may be referenced frequently, most information in part one is rarely referenced for programming purpose except those mentioned above. Further, the information in part one mentioned above could be listed in the Summary of Contents page, listing them again in the page for each dataset can be redundant. The macro %CNTFMT provides an option for either keeping part one or eliminating part one to have concise contents. By eliminating the part one, contents of each dataset can usually fit in one page, which is convenient for referencing.

<table>
<thead>
<tr>
<th>Dataset Name</th>
<th>CRF Page No.</th>
<th>Dataset Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>Library location</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Body of Contents

<table>
<thead>
<tr>
<th>Dataset Name</th>
<th>Dataset Name</th>
</tr>
</thead>
</table>

Figure 2: Layout of customized contents output from macro %CNTFMT.

Since the name of the dataset is embedded in the text of the contents, locating the contents of a particular dataset is not easy. On the other hand, the CRF page number from which the data is collected is not shown in the dataset contents from the PROC CONTENTS procedure. The macro %CNTFMT will create contents output in the layout as shown in Figure 2. With this output layout, one can quickly go to the contents page of a particular dataset by looking for the dataset name in any one of the four corners of the contents output while referencing a CRF page. Using the CRF page number in the contents, one can also quickly locate the CRF page when referencing the contents of a dataset. So the output of %CNTFMT facilitates quick cross-referencing between CRF and data contents.

The macro reads the CRF page information from a database field in each data module. Simply specify the name of database field, which contains the CRF page number, in the macro parameter PGVAR. The macro will then extract the distinct page numbers for each data module.

%*For each dataset(&&file&i), get the CRF page number;
 proc sql noprint;
 select distinct &pgvar into: pgno
 separated by ' ' from &datalib..&&file&i;
 quit;

Program code for dealing with dataset having empty CRF page variable or having no CRF page variable at all is needed to go with the above code.

Getting the dataset name in the four corner of contents output is easy. Simply adding the dataset name in title and footnote statements will do the job. Since dataset names vary between 2 and 8 (v6 SAS) or more (v8 SAS) characters, a little calculation for spacing between left corner and right corner may be needed. This is so that long dataset names will not be cut off on the right corner and short dataset names will not be too far away from right corner.

3. Contents of partial data library
Occasionally, one may need to print the contents of several datasets in a library, instead of contents of all members in the library. With PROC CONTENTS procedure, one needs to make multiple procedure calls to get the contents of several member datasets in a library with one call for each dataset. Macro %CNTFMT can print the contents of several datasets in a library in a single macro call, and present the output in alphabetical order of selected datasets. To do this, simply specify a list of datasets in the DATA parameter.

%* process the DATA parameter **;
%if %length(&data) ^=0 %then %do;
 %let j=1;
 %do %while (%length(%scan(&data, &j, ' ')))>
 %let temp=%scan(&data, &j, ' ');
 %if &j=1 %then %let datalist="&temp";
 %else %if &j>1 %then
 %let datalist=datalist,"&temp";
 %let j=%eval(&j + 1);
 %end;
 %end;

%* Get the list of dataset names;
 proc sql noprint;
 create table _cont as
 select memname format=$20. , memlabel,
 nvar, nobs, crdate
 from dictionary.tables
 where libname = "&datalib"
 %if %length(&data) > 0 %then %str(and memname in (&datalist));
 order by memname;
 quit;

4. Interleaving sample data listing with data contents
Some SAS programmers prefers to print out sample data listing following the contents of each datasets, especially when working in UNIX environment. This feature is also needed for purpose of documenting database as reported by W. Murphy (2002). Macro %CNTFMT achieves this feature through an optional parameter NUMOBS. To activate the feature, assign a numeric value to the parameter NUMOBS, then a sample data listing containing the specified number of observations will be interleaved with the data contents output. To deactivate the feature, leave the parameter NUMOBS blank.

5. Dropping database system variables
It is not uncommon to see SAS datasets that contain some database system variables. The database system variables make
sense to database programmers but not useful to report programmers, except adding confusion. Macro \%CNTFMT has another optional parameter DROPVAR, which allows the user to specify a list of variables to be dropped. The specified list of variables will be dropped from sample data listings, but not the data contents.

6. Getting study data-specific format decode from format libraries

Contents output has the format name with which a variable is associated, but not the format decodes. Getting format decodes can be quite a challenge in an established system since one may need to deal with format libraries of different levels, such as study level, project level, and one or more system levels. The author works in a system with two system levels of format libraries, which contain formats for many years of clinical studies of multiple compounds. One of the format libraries contains about 300,000 codes. Using the PROC FORMAT procedure, the following program will generate about 5,000 pages of format decodes; the output is for the whole format library regardless whether or not a format is associated with study data. Such large decodes output is useless.

```
proc format lib=libsys1 fmtlib noprint
   out=temp;
run;
proc print data=temp;
   var fmtname start end label;
run;
```

One may suggest using the SELECT statement to make the output much smaller. So the above program can be modified as following.

```
proc format lib=libsys1 fmtlib noprint
   out=temp;
   select race aedrea unitss lbvannms ;
run;
```

Sounds great, but there are two more problems. a) The list of format names needs to be found and added manually. Data of one study may have several dozen formats associated, and finding the list of formats can be time consuming. b) It is not known which format is in which library, some may be in libsys1, and some may be in libsys2. Those challenges make format referencing difficult. Macro \%CNTFMT deals with those challenges easily with one parameter SCHFMT. This order of the libraries specified in the SCHFMT should be consistent with the order specified in the SAS option FMTSEARCH. The macro will first get a list of format names associated with study data from PROC CONTENTS output, then search through the format libraries in order specified until all formats are found or to the last format library is searched. The output from such search is about 20-30 pages in the author’s system, which can be more efficiently referenced.

7. Getting study data-value specific format decodes

Parameter SCHFMT helps to get study data-specific format decodes from multiple-levels of libraries and the output is in a much more controllable size. There is room to make further improvement. For example, the format for lab test code in the author’s system has several hundreds of codes, and a study typically has only about 20-30 lab tests. So it is desirable to tailor large formats to get study data-value specific format decodes. To do this, simply list the names of large formats in parameter LGFMT in macro \%CNTFMT. With the addition of this parameter, the format decodes for a typical study in the author’s system takes only about 5 pages. Referencing such a document is much easier.

8. Tips: Setting up options for macro running environment

The outputs for contents and format decodes are generated with PROC REPORT. To get consistent looking outputs for contents and formats when running in different environments, options such as linesize and pagesize are specified in the macro. However, options such as linesize and pagesize in the macro may be different from those in the environment where the macro is to run. On one hand, the macro needs to be portable to run in any environment and give output of consistent looking, and on the other hand the macro should not change the environment options. To solve the dilemma, the following SAS code can be used in the macro.

```
%let_macrog=%sysfunc(getoption(macrogen));
%let _mlogic=%sysfunc(getoption(mlogic));
%let _mprint=%sysfunc(getoption(mprint));
%let _ps=%sysfunc(getoption(ps));
%let _ls=%sysfunc(getoption(ls));
%let _date=%sysfunc(getoption(date));
%*Save the environment options in macro
%*specify options for the macro;
options nodate nocenter ls=120 ps=55 nomprint nomlogic nomacrogen;
%*At the end of macro;
%*restore the environment options;
options &_date &_center ls=&_ls ps=&_ps &_mprint &_mlogic &_macrog;
```

CONCLUSION

The macro can meet various needs for understanding clinical data; the feature for cross-referencing of data contents and CRF is especially useful in a fast paced environment. The challenges of getting format decodes from multiple levels of format libraries has been solved by the macro with the use of a single parameter. By using a single macro call, one can get all data related reference at once; this is especially helpful for a person rushing into a new project and needing to know data quickly. The macro has been tested and used in both V6 and V8, and works well.

REFERENCES


ACKNOWLEDGMENTS

The author would like to thank Jim Edgington for reviewing the manuscript and providing valuable comments.

CONTACT INFORMATION

Your comments and questions are valued and encouraged. Contact the author at:

Sam Mao
Quintiles, Inc.
P.O. Box 9708
Kansas City, MO 64137-9708
Work Phone: (816) 767-3783
Fax: (816) 767-7372
Email: sam.mao@quintles.com