Paper 99-27

A Quick and Easy Data Dictionary Macro

Pete Lund, Northwest Crime and Social Research, Olympia, WA

ABTRACT

A good data dictionary is key to managing even a modest-sized project. There are limited tools available in SAS to give a comprehensive, project- wide look at the contents of your datasets.

This paper presents a macro that generates a browser-based data dictionary that goes beyond PROC CONTENTS in a number of ways. Any number of datasets can be passed to the macro and the results displayed in a table of contents-driven web page. Additionally, all user-written formats associated with dataset variables are hyperlinked, allowing the user to click on a format name and view the format values and labels.

THE CALL

The call to the macro is relatively simple. There are only four required parameters:
PROJECT: a name for the dictionary that will be displayed as the Table of Contents header. This is simply a text string, for example, “Jail Data”.
HTMLPATH: the directory where the generated HTML code will be stored. This is just the directory name, “d:\projects\jail”.
FRAMENAME: the name of the HTML file that “drives” the process (the one to open). This is just the file name, with an .HTM or .HTML extension, “jail_data.htm”.
DSLIST: the list of datasets to be included in the dictionary. Datasets in the list are separated by spaces. Entire libraries can be included with a *libref.** reference. For example, “jail.* gen.sample”.

Using the examples above a call to the macro would be:

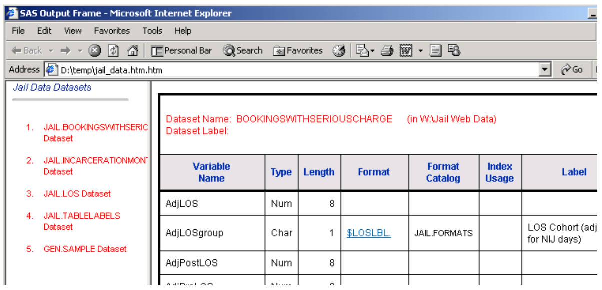
```
%DataDictionary(project=Jail Data,
                htmlpath=d:\projects\jail,
                framename=jail_data.htm,
                dslist=jail.* gen.sample)
```

Additionally, dataset inclusion in the dictionary can be filtered based on the creation date of the dataset with following parameters:
  DATEMIN: begin date on which to keep datasets.
  DATEMAX: end date on which to keep datasets.

THE OUTPUT

The output of the macro is a number of HTML files. The file that will be viewed is referenced in the *framename* parameter and is stored in the directory referenced in the *htmlpath* parameter.

Opening this file will display a two-frame file with a table of contents in the left frame and information about a dataset in the right frame.


SAS Output Frame - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Personal Bar Search Favorites
Address D:\temp\jail_data.htm.htm
Go

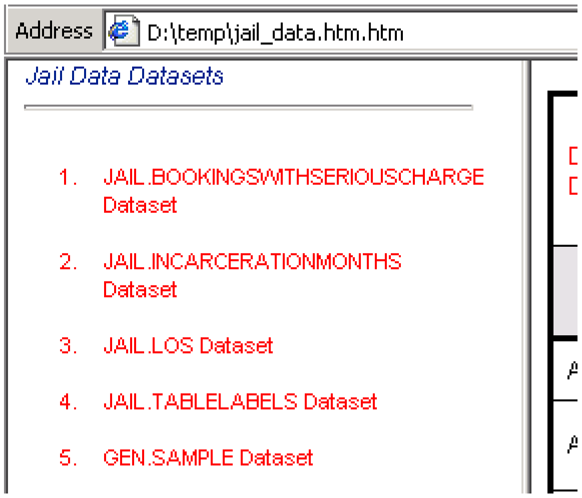
Jail Data Datasets

1. JAIL.BOOKINGSWITHSERIC Dataset
2. JAIL.INCARCERATIONMONT Dataset
3. JAIL.LOS Dataset
4. JAIL.TABLELABELS Dataset
5. GEN.SAMPLE Dataset

Dataset Name: BOOKINGSWITHSERIOUSCHARGE (in W:\Jail Web Data)
Dataset Label:

Variable Name	Type	Length	Format	Format Catalog	Index Usage	Label
AdjLOS	Num	8				
AdjLOSgroup	Char	1	$LOSLBL.	JAIL.FORMATS		LOS Cohort (adj for NIJ days)
AdjPostLOS	Num	8				

There is one addional file stored in the *htmlpath* directory. This file contains the table of contents information and is named using the *framename* name with "_contents" added. In the example above, this file would be named, "jail_data_contents.htm". The table of contents contains an entry for each dataset referenced in the *dslist* parameter. Clicking on a dataset name will display information about that dataset.

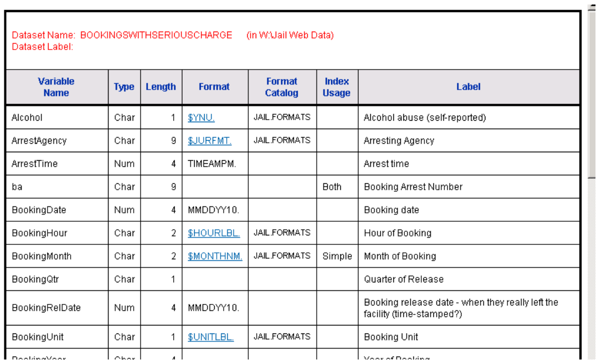


Address D:\temp\jail_data.htm.htm

Jail Data Datasets

1. JAIL.BOOKINGSWITHSERIOUSCHARGE Dataset
2. JAIL.INCARCERATIONMONTHS Dataset
3. JAIL.LOS Dataset
4. JAIL.TABLELABELS Dataset
5. GEN.SAMPLE Dataset

There is an HTML file created for each dataset stored in a directory called "datasets", under the *htmlpath* directory. The dataset HTML files are called "<libref>.<dataset>.htm". The dataset information is stored in an HTML table and contains dataset-level information: dataset name and label, physical location, creation date, size, number of variables, etc. It also contains variable-level information: variable name, type, length, format, index usage and label.

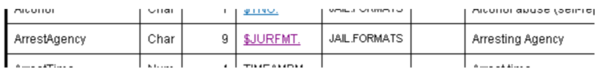


Dataset Name: BOOKINGSWITHSERIOUSCHARGE (in W:\Jail Web Data)
Dataset Label:

Variable Name	Type	Length	Format	Format Catalog	Index Usage	Label
Alcohol	Char	1	$YNU.	JAIL.FORMATS		Alcohol abuse (self-reported)
ArrestAgency	Char	9	$JURFMT.	JAIL.FORMATS		Arresting Agency
ArrestTime	Num	4	TIMEAMPM.			Arrest time
ba	Char	9			Both	Booking Arrest Number
BookingDate	Num	4	MMDDYY10.			Booking date
BookingHour	Char	2	$HOURLBL.	JAIL.FORMATS		Hour of Booking
BookingMonth	Char	2	$MONTHNM.	JAIL.FORMATS	Simple	Month of Booking
BookingQtr	Char	1				Quarter of Release
BookingRelDate	Num	4	MMDDYY10.			Booking release date - when they really left the facility (time-stamped?)
BookingUnit	Char	1	$UNITLBL.	JAIL.FORMATS		Booking Unit

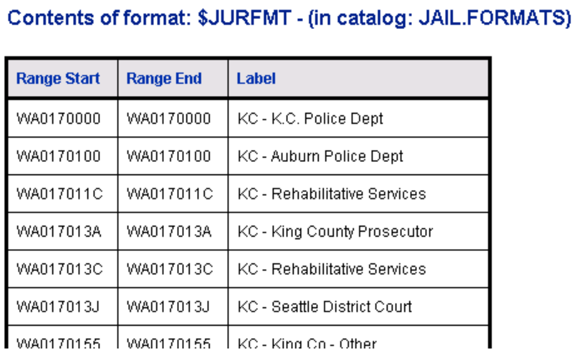
Additionally, if a variable has an associated user-written format, the format name is hyperlinked in the HTML output. Clicking on the format name will display the format values and labels. There is an HTML file for each format found in the datasets referenced in the *dslist*. They are stored in a directory called "formats", under the *htmlpath* directory. The format HTML files are called "<libref>.<catalog>.<format>.htm".

Clicking on the $JurFmt. Format link below



ArrestAgency	Char	9	$JURFMT.	JAIL.FORMATS		Arresting Agency

would display the following web page:



Contents of format: $JURFMT - (in catalog: JAIL.FORMATS)

Range Start	Range End	Label
WA0170000	WA0170000	KC - K.C. Police Dept
WA0170100	WA0170100	KC - Auburn Police Dept
WA017011C	WA017011C	KC - Rehabilitative Services
WA017013A	WA017013A	KC - King County Prosecutor
WA017013C	WA017013C	KC - Rehabilitative Services
WA017013J	WA017013J	KC - Seattle District Court
WA0170155	WA0170155	KC - King Co - Other

In order for the format linking to work correctly, the SAS FMTSEARCH= option must be set with the same values that would be used if the datasets were being used. If not, formats may not be found or a format from the wrong catalog may be linked.

Output in this form allows the user quick access to information about all the datasets in a project. Access to formatted values of variables is also a significant enhancement over standard PROC CONTENTS output.

HOW IT WORKS

The process for creating the data dictionary files can be broken down into the following steps.

Determine if Paths Exist

The first step is to determine if the paths required for the process exist – this includes the path referenced in the *htmlpath* parameter and the /datasets and /formats subdirectories. If the directories do not exist they are created. This is done via a subordinate macro that is run for each of the three paths. There is a bunch of error handling in that macro (does the drive exist, is the fileref eight characters or less), but the crux of it comes down to a single line that is executed if the path does not exist:

```
%let rc = %sysfunc(system(md "&dname"));
```

The path (&dname) is passed to the operating system command MD (make directory) and the job is done.

Flesh out the dataset list

The dataset list that is passed in the &Dslist parameter may contain one-level names or wildcards (*). The process loops through all the datasets and checks for completeness of the name, i.e. a two-level name. If any one-level names are passed a WORK. prefix is added.

```
%do i = 1 %to &_NumDS;
   %let _piece = %scan(&DSlist,&i,%str( ));
    %if %index(&_piece,.) eq 0 %then
      %let _piece = WORK.&_piece;
                  :
```

If any library-level wildcards are passed a list of all the datasets in the library is created by concatenating the libname and member names together in space-separated list.

```
%if %index(&_piece,*) ne 0 %then
  %do;
    proc sql noprint;
       select compress(libname||'.'||memname)
                into :BigList separated by ' '
       from sashelp.vstable
        where libname eq
                 upcase("%scan(&_piece,1,.)");
     quit;
   %end;
```

When finished with this process a complete list of two-level dataset names is available.

Get Variable Information

From the COLUMNS and TABLES DICTIONARY tables information about the dataset variables is loaded into a dataset. The macro variable &NewDS contains a quoted, comma-separated list of datasets in the dataset list.

```
proc sql noprint;
  create table DSinfo as
  select c.*,crdate
  from sashelp.vcolumn c,
       sashelp.vtable t
 where c.memname eq t.memname and
       c.libname eq t.libname
       &DateRange and
       c.libname||'.'||c.memname in (&NewDS)
 order by upcase(c.name);
quit;
```

A variable containing the path to a format HTML file is also created here. This variable will have the structure noted above in *THE CALL* section and will be used to create the hyperlink to the format file. Additional dataset information is gathered from the MEMBERS DICTIONARY table.

Get a List of the Available Formats

The FMTSEARCH= value is grabbed with the GETOPTION function and the result is parsed into catalog names.

```
%let FmtList=%sysfunc(getoption(fmtsearch));
```

Note: the value returned by the GETOPTION function contains parentheses around the FMTSEARCH value – these are stripped off in the actual code.

Those catalog names are compared to the CATALOGS DICTIONARY table. The default catalog type .FORMATS is added to the catalog name if there is no type in the catalog list.

```
data _FmtList;
 length FormatEntry $100;
 %do i = 1 %to &NumCats;
  %let CatName = %scan(&FmtList,&i,%str( ));
    %if %index(&CatName,.) eq 0 %then
      %do;
        FormatEntry = &CatName"||".FORMATS";
      %end;
    %else
      %do;
        FormatEntry = "&CatName";
      %end;
     ListPosition = &i;
     output;
  %end;
run;
```

This dataset is then unduplicated by format name and position in the FMTSEARCH list. This will assure that if there are formats of the same name in multiple catalogs the correct one is pulled. This dataset is called *_FmtHits*.

Match Formats to Variable List

This list of all formats is compared to the list of formats associated with the variables in the datasets listed in the *dslist* parameter. Only requested formats are kept.

When the variable information was gathered earlier a dataset of format information was created, *_FormatsToFind*. This dataset contains, among other things, the format name (ObjName) and the format type (ObyType). The type will be either character or numeric.

```
proc sql;
  create table _FormatsFound as
  select h.*,Format
  from _FmtHits h,
       _FormatsToFind f
  where h.ObjName eq f.ObjName and
        h.ObjType eq f.ObjType
  order by libname,memname;
quit;
```

A macro variable “array” is created with all the format names. This list is looped through and a CNTLOUT dataset created from each format. The datasets are output with PROC SQL, using ODS to create HTML files.

The ODS BODY parameter will create a file that has the name of the format catalog and format. For example, JAIL.FORMATS.JURFMT.HTM.

```
ods html
    body="formats\&&lib&i...&FmtName..htm"
     path="&HTMLpath" (url=none)
    style=datadictionary;


proc format cntlout=_temp library=&&lib&i;
  select &FmtName;
run;

title "Contents of format: &FmtName - (in
       catalog: &&lib&i)";
```

```
proc sql;
  select Start
  %if &Ranges gt 0 %then
    %str(label='Range Start',
         End  label='Range End',);
  %else %str(label='Value',);
    Label label='Label'
  from _temp;
quit;
```

The SQL generates a file with the start and end range values and the formatted value for the format. Note: an earlier step had determined whether the format contained any ranges or if all start and end values were the same. If there were no ranges (&Ranges eq 0) the output contains only the start and formatted values.

Generate Dataset Output

Finally, a PROC REPORT is run for each dataset in the *dslist* parameter. ODS is also used here to generate HTML files.

```
proc report data=DSinfo
        nowd headline headskip split='*'
        contents='Variable List';
  column name fmtlink type length format
         FormatLib informat idxusage label;

  define name / order order=data
              'Variable*Name' width=&maxlen;
 define type / display 'Type' width=6;
 define length / display 'Length' width=8;
 define format / display 'Format' width=10;
 define FormatLib / display 'Format*Catalog'
                    width=15;
 define informat / display noprint;
 define idxusage / display 'Index*Usage'
                   width=11;
 define label / display 'Label';
 define fmtlink / display noprint;

 compute before _page_ /
         style=[just=left font_face=Arial
                font_size=3
                background=white
                foreground=red];
  DSN = "Dataset Name:  &memname")||
        "(in &path)";
    DSL = "Dataset Label: "&memlabel";
    line ' ';
    line @1 DSN $120.;
    line @1 DSL $120.;
    line ' ';
  endcomp;

  compute fmtlink;
    if fmtlink ne '' then
     call define('format',"URL",fmtlink);
  endcomp;
run;
```

The new “URL” method in PROC REPORTs CALL DEFINE is used to create the link to the format HTML files. In the variable dataset the observation was flagged if it contained a user defined format – this was determined if the variable matched to a record in the list of formats created above. If so, a link is created using the value of the FmtLink variable as the link location. This variable is created using the catalog and format name as was noted above.

WHAT THE FUTURE HOLDS

When working on a project like this, there are a number of enhancements that come to mind during the development process.

First, if a format contains an embedded format as a label all that displays is the embedded format reference, for example, “[NewFmt.]”. It would be preferable if that embedded link was also hyperlinked to the format referenced. This would require an iterative process for generating the format files, looking for embedded format names.

Second, allow linking of additional metadata. If a user wants to maintain a separate file (or files) of information about datasets or variables there could be a mechanism for linking that information and displaying it in the data dictionary system.

AUTHOR CONTACT INFORMATION

Pete Lund
Northwest Crime & Social Research, Inc.
215 Legion Way SW
Olympia, WA 98501
(360) 528-8970
pete.lund@nwcsr.com
www.nwcsr.com

Copies of the macros needed for this process are available electronically upon request.

TRADEMARK INFORMATION