Paper 45-28

Building Metadata Repository for Data Sets

Haining Luo, Medical Star Hospital, Beijing, China
Haiping Luo, Dept of Veterans Affairs, Washington, DC

ABSTRACT

Metadata describe SAS dataset features, such as dataset name, physical location, creation date, last modified date, number of observations, variable names, and so on. A metadata repository gives users quick access to many datasets' features without opening those datasets. This paper demonstrates macros to automate the process of building and reporting metadata repositories. Macro **%getmeta** allows a programmer to document metadata for a dataset and append these metadata to a metadata repository of other datasets. Macro **%viewmeta** generates a metadata report from a metadata repository as a comma-separated file. Users may view the report through MS Excel. The metadata repository includes over 50 variables, 42 of them come from 'Proc Contents' output, and the rest come from a macro **%getfreq** and programmer's input. The output from 'Proc Contents' describes the features of datasets. The output from **%getfreq** lists the most frequent values or value ranges of all variables of the dataset. The programmer can add project name, code name, and the physical file name of the dataset to the repository. This repository experiment has been tested in Windows environment using PC SAS 8.2. The macros run with Base SAS. Any programmer with minimal SAS skills can run these macros.

DATASET DOCUMENTATION PRINCIPLES

Documentation is a tedious but extremely important part of any programming and application development process. Besides good commenting practice during programming, a developer should document all datasets s/he creates. Documenting datasets should follow these basic principles:

1. A dataset should be documented at the time when the dataset is generated.
2. The dataset documentation should have a reference to the code which generated this dataset.
3. The documentation for all datasets of a project should be kept together within the project file directory.
4. The dataset documentation should be accessible without invoking SAS.
5. The metadata being captured should describe variable values in addition to dataset features.

To implement these principles, a developer often needs to use several SAS procedures and to write many lines of code. This paper demonstrates macros to automate the process of documenting datasets. With the help of these macros, a developer can build and report metadata repositories on the fly. The resulting metadata repositories satisfy these documentation principles.

USING MACROS TO BUILD METADATA REPOSITORY

Before we introduce the contents of the macros, let's see how we can use the macros to build a metadata repository. Three macros will be used:

1. **%getmeta**
2. **%getfreq**
3. **%viewmeta**

All of these macros can be downloaded from:
http://chinafoundation1.org/hp/sugipapers.htm.

Among these three macros, only two, **%getmeta** and **%viewmeta**, will be called directly. **%getmeta** is used to generate metadata from a dataset then append these metadata to a metadata repository for a specific project. **%getmeta** calls another macro, **%getfreq**, to obtain the frequent values, the frequency count, and the mean & maximum & minimum of all variables. The output of **%getmeta** include: 1) a repository dataset which contains all metadata about all datasets of a project and 2) a comma-separated (csv) metadata file which contains brief dataset information and variable's value range information. The csv file can be viewed through MS Excel to get a quick look of the datasets documented in the repository.

%viewmeta is used to generate a complete report of all metadata of all datasets in the repository. This report is a csv file in a viewer-friendly format for being viewed through MS Excel.

Now assume this scenario. We have a project called "SUGI 28 Paper". This project contains two SAS programs 'shoes.sas' and 'retail.sas'. These programs created two permanent datasets, 'shoes' and 'retail', respectively. We first call **%getmeta** at the end of the program code 'shoes.sas' to generate metadata for dataset 'shoes':

```
* point to the macro library which contains
%getmeta and %getfreq;
options sasautos="d:\home\sasmacrolib";
* call %getmeta, which will call %getfreq
internally;
%getmeta
(datapath=d:\home\paper\sas\sugi28\,
 datalib=SUGI28,
 dataname=SHOES,
codename=d:\home\paper\sas\sugi28\shoes.sas,
macrpath=d:\home\sasmacrolib);
```

This macro call will generate a repository dataset 'meta_SUGI28.sas7bdat' and a brief csv file 'meta_SUGI28.csv' in the data path 'd:\home\paper\sas\sugi28\'. At this moment, the repository dataset has the entire set of metadata for the dataset 'shoes', while the csv file is a brief version of the repository.

We then call **%getmeta** at the end of the program code 'retail.sas' to generate metadata for dataset 'retail':

```
options sasautos="d:\home\sasmacrolib";
%getmeta
(datapath=d:\home\paper\sas\sugi28\,
 datalib=SUGI28,
 dataname=RETAIL,
codename=d:\home\paper\sas\sugi28\retail.sas,
macrpath=d:\home\sasmacrolib);
```

This macro call will append metadata for dataset 'retail' to the repository dataset 'meta_SUGI28.sas7bdat' and to the csv file 'meta_SUGI28.csv'. Now the repository dataset has the complete metadata for both 'shoes' and 'retail', while the csv file is a brief version of the repository. Such calls to **%getmeta** for a same dataset can be repeated as many times as needed. Each new call will delete the existing metadata for this same dataset and append the new result. The brief csv file contains the following metadata:

Data Set Label, Dataset Name, Path of Dataset Library,
The Code Which Created Dataset
Observations in Data Set, Total Number of Variables
Variable Name, Variable Label

Most Frequent Values of Character Variables
Most Frequent Values of Numeric Variables
Frequency Count, Percent of Total Frequency
Variable Mean, Maximum, Minimum
Dataset Created Date, Last Modified Date
Date & Time Metadata Were Generated

To get a complete view of all metadata of all datasets in the repository, we will write a small program to call **%viewmeta**:

```
* point to the macro library which contains
%viewmeta;
options sasautos="d:\home\sasmacrolib";
* call %viewmeta;
%viewmeta
(datalib=SUGI28,
 metapath=d:\home\paper\sas\sugi28\);
```

This macro call will generate a viewer-friendly csv file 'viewmeta.csv' in the same path of the repository. This csv file displays all metadata contained in the repository file 'meta_sugi28.sas7bdat'. For each dataset, the csv file has two blocks to display the metadata: a 'Dataset Block' which displays the metadata for the dataset and a 'Variable Block' which displays variable names, labels, formats, frequent values, frequency count, mean, maximum, minimum and other information about all variables in the dataset. This **%viewmeta** can also be called as many time as needed, each call will generate the current view of the repository and replace the previous 'viewmeta.csv' file.

The tables below show a portion of the 'Dataset Block' and 'Variable Block' in 'viewmeta.csv' file:

Meta Variable Name	Meta Variable Description	Value
LIBNAME	Library Name	SUGI28
MEMNAME	Dataset Name	SHOES
MEMLABEL	Data Set Label	Fictitious Shoe Company Data
ENGINE	Engine Name	V8
created	Date & Time Dataset Were Created	10AUG1999:15:34:42
modified	Date & Time Dataset Were Last Modified	10AUG1999:15:34:42
metadate	Date & Time Metadata Were Generated	12/15/2002 21:09
libpath	Path of Dataset Library	d:\home\paper\sas\sugi28
codepath	The Code Which Created Dataset	d:\home\paper\sas\sugi28\shoes.sas
NOBS	Observations in Data Set	395
SORTED	Sorted and/or Validated	.

Dataset Name	Variable Name	Variable Type	Variable Length	Variable Label	Most Frequent Values of Numeric Variables	Most Frequent Values of Character Variables	Frequency Count	Percent of Total Frequency	Mean
SHOES	Product	Character	14		.	Boot	52	13.2	.
SHOES	Product	Character	14		.	Slipper	52	13.2	.
SHOES	Product	Character	14		.	Sport Shoe	51	12.9	.
SHOES	Product	Character	14			Women's Dress	51	12.9	.
SHOES	Stores	Numeric	8	Number of Stores	1		37	9.4	11.65
SHOES	Stores	Numeric	8	Number of Stores	3		33	8.4	11.65
SHOES	Stores	Numeric	8	Number of Stores	2		29	7.3	11.65
SHOES	Stores	Numeric	8	Number of Stores	4		20	5.1	11.65

In short, to build a metadata repository for a project, all we need to do is to call **%getmeta** for each dataset. To view all metadata in MS Excel, a call to **%viewmeta** will prepare the metadata file in csv format.

DESIGN OF %GETMETA

2. USER INPUT

%Getmeta collects user input through five parameters:
Datapath: physical path for the dataset directory; the meta repository will be stored in the same directory.
Datalib: library name for the dataset directory. It should represent the project the data belong to and be consistent each time %getmeta is run for the same dataset. It is recommended using the same datalib name for all %getmeta calls for all datasets in the same project.
Dataname: dataset name from which the metadata will be generated.
Codename: physical path and file name for the SAS program which created this dataset.
Macrpath: physical path for the macro library directory where %getfreq is stored.

3. STEPS

There are six steps in macro **%getmeta**:
1) Get dataset physical path.
Although a parameter is set for a user to input the path to the data directory, the physical path to the data directory is captured through querying dictionary.members. The capture can guarantee the format of the path. Once the path is obtained, it is assigned to a macro variable 'libpath' to be stored later in metadata.

```
proc sql noprint;
create table dict as
     (select path
        from dictionary.members
        where libname="&datalib");
quit;
data _null_;
    set dict(keep=path);
** note: libpath will not have \ at the end;
    call symput('libpath',trim(left(path)));
```

2) Get dataset contents table.
The metadata for the dataset mainly come from 'PROC contents' output. The output of 'PROC contents details' has 48 variables. All of them are kept as metadata for this dataset. But the output table does not contain total number of variables, the physical path to the dataset directory, and the program code which created the dataset. Since the directory path has been captured in step 1), and the code path and name have been collected through parameter values, only the total number of variables needs to be obtained. A piece of code is written to calculated the number of variables:

```
proc means max data=acont noprint;
    var varnum;
    output out=vn max=vnum;
data _null_;
    set vn;
    call symput('vnumb',vnum);
```

where 'acont' is the output table from 'PROC contents', 'varnum' contains the sequential number assigned to a variable in the original dataset, the maximum of 'varnum' is the total number of the variables in the dataset. This total number is then assigned to a macro variable 'vnumb' to be put into the metadata later.

3) Obtain variable value range, most frequent values, and frequency distribution.
The next step is to call **%getfreq** (see SUGI 26 Paper 81-26 by Haiping Luo for a detail explanation of **%getfreq**) to get variable value range, most frequent values, frequency distribution and other variable information. To control the size of frequency table returned by **%getfreq**, the variable values which have 1 occurrence in the entire dataset or which's frequency is below 5 percent of the total observations are dropped from the returned frequency table.

4) Merge content table and frequency table.
The output of 'PROC contents' and the frequency table created by %getfreq are merged together to form the metadata table. The dataset path, number of variables, and the date and time when the metadata were created are added into this table.

5) Append the metadata table to the existing metadata repository.
The following code checks the existence of metadata repository. If the repository exists, the new metadata will be appended to the repository. Otherwise, a new repository will be created. Also if the dataset has already had metadata stored in the repository, the existing metadata for the dataset will be deleted and the new capture will be stored into the repository.

```
%if %sysfunc(exist(&&datalib..meta_&datalib))
%then %do;

   data &datalib..meta_&datalib;
        set &datalib..meta_&datalib;
        if
            memname="&dataname" and
            codepath = "&codename" and
            libpath = "&libpath"
            then delete;

    proc datasets force;
       append base=&datalib..meta_&datalib
       data=meta_&dataname;
    run;
    quit;
    %end;
%else %do;
    data &datalib..meta_&datalib;
        set meta_&dataname;
    run;
    %end;
```

6) Create a comma-separated file to demonstrate a brief version of the repository.
At the end of **%getmeta**, a print out is posted to a csv file through invoking a 'ODS csv' statement.

3. OUTPUT

The output of **%getmeta** are a repository dataset 'meta_&datalib..sas7bdat' and a brief csv file 'meta_&datalib..csv'. Both of them are stored in the same directory where the dataset is located. The repository has 53 metadata variables, while the csv file has 18 metadata variables.

DESIGN OF %VIEWMETA

1. USER INPUT

%Viewmeta collects user input through two parameters:
Datalib: this libname is used to identify the metadata repository dataset created by **%getmeta**, i.e., 'meta_&datalib..sas7bdat'.
Metapath: physical path for the directory where the metadata repository is located.

2. STEPS

There are two steps in macro **%getmeta**:
1) Get the total number of datasets stored in the repository.
For the purpose of displaying metadata one dataset at a time in the output file, a dataset id is assigned to each dataset in the metadata repository. The total number of datasets is captured and stored in a macro variable 'dsetnumb'.

```
proc sort data=in.meta_&datalib out=s1;
    by libname MEMNAME;
*** get number of datasets;
data meta_&datalib;
    retain dsetid 0;
    set s1;
    by libname MEMNAME;
    label dsetid = "Dataset Order in
Repository meta_&datalib";
    if first.memname then dsetid + 1;
    if last.libname then do;
        call symput('dsetnumb',dsetid);
    end;
```

2) Loop through datasets to post to Dataset Block and Variable Block to the output csv file.
At this step, 'ODS csv' is first invoked to allow posting printout to the output file 'viewmeta.csv', dataset by dataset along the loop. Then the loop will go through each dataset until the last dataset defined by the macro variable &dsetnumb.,

There are two levels of metadata in the repository, those applied to the dataset as a whole and those applied to specific variables. During the loop, the dataset level metadata, such as dataset name, number of variables, etc., are putting into the 'Dataset Block' from top down in the csv file; the variable level metadata, such as variable name, value, mean, etc., are putting into the 'Variable Block' from left to right in the csv file (see a previous section for a demo).

To generate the dataset level list from top down, the metadata are transposed by character values and by numeric values. After the transpose, the dataset level list and the variable level list are printed to the csv file. Then the loop continues to the next dataset in the repository.

3. OUTPUT

The output of **%viewmeta** is a cvs file in the same directory where the metadata were stored: 'viewmeta.csv'. All 53 metadata variables for all captured datasets are displayed in the file. Each time when **%viewmeta** is run, the existing 'viewmeta.csv' will be replaced by the new output.

TEST

You can follow these steps to build a test metadata repository:
1. Assume you have a directory to store SAS macros 'd:\home\sasmacrolib'. Download the three macros, %getfreq, %getmeta, and %viewmeta, from:
http://chinafoundation1.org/hp/sugipapers.htm
Store the macros into the macro directory.
2. Create a project directory 'd:\home\paper\sas\sugi28\'. Copy two sample datasets 'shoes.sas7bdat' and 'retail.sasd7dat' from the '[SAS root]\core\sashelp' directory to the project directory.
3. Create a small SAS program, type the three macro calls (two %getmeta calls and one %viewmeta call in this order) as described in the previous section 'Using Macros to Build Metadata Repository'.
4. Run this program and you will see the output in the directory 'd:\home\paper\sas\sugi28\':
'meta_SUGI28.sas7bdat'
'meta_SUGI28.sas7bdat'
'viewmeta.csv'
5. Open the csv files to view the metadata captured.

SUMMARY

This paper introduced the designs and usage of two macros, **%getmeta** and **%viewmeta**. These two macros plus another macro **%getfreq** (see SUGI 26 Paper 81-26 by Haiping Luo) can be used together to automate the process of building and reporting metadata repositories for datasets. A repository stores over 50 variables to describe the datasets and the variables in the datasets. These metadata can provide handy documentation for SAS projects.

NOTE

The updated paper and macros can be found in this link:
http://chinafoundation1.org/hp/sugipapers.htm

CONTACT INFORMATION

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

Name	Haiping Luo
Company	Dept of Veterans Affairs
Email:	haiping.luo@mail.va.gov
