Paper SA17-2012

Creating a SAS Frequency Data Set of Valid Categorical Variables for Export to Microsoft Excel or Access

Neil Kamdar
University of Michigan, Department of Biostatistics
Kidney Epidemiology and Cost Center (KECC), Ann Arbor, MI

ABSTRACT

This paper describes a project that required the creation of a data dictionary with frequencies of categorical variables with appropriate formats to be stored for a variety of SAS Medicare Claims data sets. The implementation of CALL SYMPUT, PROC SQL, and the VNAME function are utilized to ensure lists of frequencies of categorical variables can be collapsed into a single data set for exportation to Microsoft Excel, Access, or other data processing and reporting software packages. After multiple macro calls and PROC APPEND with the FORCE option, final recommendations are made for post-macro post-processing to ensure readability of the dataset with associated formats is feasible. All analysis and programming was conducted on SAS Version 9.2 on a Windows XP Professional Workstation.

INTRODUCTION

Creating data libraries and repositories is useful to compare frequencies of variables over periods of time. In the Medicare realm, volumes of claims data exists; thus, this implies a need to define expected counts of claims as well as other key measures with every update. Creating a data dictionary with expected counts, frequencies, and univariate statistics would be essential for testing and validation of claims processing. As part of the effort to further assist testing and validation protocol, code was written to produce a data dictionary containing actual frequencies within the target data set. The macros were written with the notion that SAS data set repositories may exist in different libnames and years. The scope of this paper will be relegated to the frequencies generation of variables with known formats and informats and the challenges presented, and a working assumption would involve a lookup data set with a variable list generated from a PROC CONTENTS procedure.

EXAMPLE DATA SET

Preparation of a simple data set that can be generated using the following code would assist for future macro use.

```
proc contents data = my_lib.my_dataset out = my_lib.test;run;

proc sql;
create table my_lib.test as
select MEMNAME, VARNUM, NAME,
   case
     when TYPE = 2 then 'CHAR'
     when TYPE = 1 then 'NUM'
   end as TYPE,
   LENGTH, FORMAT, INFORMAT, LABEL, LIBNAME
from
   my_lib.test;
quit;

data my_lib.test1;
   set my_lib.test;
   DATASET_NAME = upcase(MEMNAME);
   DATASET_ID = 1;
   Library_name1 = upcase(LIBNAME);
run;
```

The code above can be used with any existing data set for which you want to generate frequencies. For simplicity, DATASET_NAME and Library_name1 are derived copies of MEMNAME and LIBNAME from the PROC CONTENTS output data set since they fit within the macro to be discussed later. PROC CONTENTS generates a TYPE variable, where TYPE = 1 is a numeric variable, and TYPE = 2 is a character variable. For frequencies, we are only interested in character variables; therefore, recoding TYPE into ‘CHAR’ or ‘NUM’ would be more meaningful in a data dictionary. PROC CONTENTS produces an output data set containing MEMNAME as the data set name, VARNUM as the SAS system variable number, NAME as the explicit variable name in the data set, length, format, informat, and labels, if they exist. If any of these do not exist, they appear as blank or missing. DATASET_ID is defined because Microsoft Access requires a unique identifier; therefore, for simplicity, we include a user-defined DATASET_ID = 1.

The first five records outputted from a given source data set would look like this:

MEMNAME	VARNUM	NAME	TYPE	LENGTH	FORMAT	INFORMAT	LABEL	LIBNAME	DATASET_NAME	DATASET_ID	Library_name1
MY_DATASET	43	CAN_FIRST_LISTING_DT	NUM	8	DATE		First Date patient is ever waitlisted	MY_LIB	MY_DATASET	1	MY_LIB
MY_DATASET	45	CAN_REM_CD	NUM	8	REMCD		Reason why candidate was removed (removal code)	MY_LIB	MY_DATASET	1	MY_LIB
MY_DATASET	44	CAN_REM_DT	NUM	8	DATE		rem_dt /remreg, censored by CAN_DEATH_DT, tx_dt	MY_LIB	MY_DATASET	1	MY_LIB
MY_DATASET	19	DEATH_SOURCE	CHAR	8			Source of death date	MY_LIB	MY_DATASET	1	MY_LIB
MY_DATASET	18	DIED	NUM	8	DATE		Death date selected from multiple sources, usually the median	MY_LIB	MY_DATASET	1	MY_LIB

THE PROCESSING MACRO

The code that would be used to create the frequencies data set is contained within two macros. The %processing macro with the calling definition as:

```
%processing (library_name, dataset_name1, iterator, library_name2, dataset_name2);
```

The %processing macro takes the library name where the data set contents data set produced in the previous section resides, the data set name, a data set identifier which we define to 1 for the first instance of the macro call, the true source data set library name, and the data set name for the source data. In our example, the specific macro call is denoted as:

```
%processing (my_lib, test, 1, my_lib, my_dataset);
```

The code for the %processing macro is below, with an explanation to follow:

```
%macro processing (library_name, dataset_name1, iterator, library_name2,
dataset_name2);
   %let new_dataset_name = &dataset_name1;
   data character_w_formats;
      set &library_name..&dataset_name1.&iterator.;
      if type = 'CHAR' and format not in ('','$') then output;
   run;
       /*get observation count for the conditional logic of producing frequencies*/
   proc sql noprint;
      select count(*)
      into :OBSCOUNT
      from character_w_formats;
   quit;

   %put Count=&OBSCOUNT.;
   %if &OBSCOUNT > 0 %then %do;
      proc sort data = character_w_formats;    by varnum name; run;

     data character_w_formats
     (keep = newstring varnum DATASET_NAME NAME FORMAT library_name1 dataset_id);
        set character_w_formats;
        length newstring $2000;
        by varnum name;
        retain newstring;
        newstring = trim(newstring)||' '|| trim(NAME);
        if type = 'CHAR' and format ne '' then output;
      run;

      data character_w_formats;
```

```
         set character_w_formats;
         i+1;
         call symput ("VARIABLE_LIST2", newstring);
         call symput ("OBSERVATION_COUNT", i);
      run;

      data &dataset_name1._2;
         set &library_name2..&dataset_name2.;
      run;

      %put &VARIABLE_LIST2; %put &OBSERVATION_COUNT; %put &new_dataset_name._2;

      data _null_;
         array varlist(&OBSERVATION_COUNT) &VARIABLE_LIST2 ;
         do i = 1 to dim(varlist);
            %put &new_dataset_name;
            call execute( '%diafreq(' || vname(varlist(i)) || ')' );
         end;
      run;
   %end;
%mend;
```

Assign variable dataset_name1 to a global variable name new_dataset_name to be used in further macro processing across data steps. In the initial data step that creates character_w_formats, the contents of the example data set, test1, outputs only those rows containing character variable names whose format is not blank or a $. This simply means that their format as stored in PROC CONTENTS should be populated with a known character format type with a defined length. Explicitly declared user-defined formats in these data sets would normally be outputted. The next step uses PROC SQL with the NOPRINT option to create an &OBSCOUNT variable that counts the number of character variables that exist in the character_w_formats data set. If character_w_formats generates 0 observations, the PROC CONTENTS for the source data set has no character variables. The macro will not proceed in this case and no data set is created for append to a base data set of frequencies.

However, in our example, assuming there is at least one character variable with a defined format, the macro will perform a PROC SORT to be used in a subsequent data step for BY variable processing with VARNUM and NAME (variable name).

```
      data character_w_formats
      (keep = newstring varnum DATASET_NAME NAME FORMAT library_name1 dataset_id);
         set character_w_formats;
         length newstring $2000;
         by varnum name;
         retain newstring;
         newstring = trim(newstring)||' '|| trim(NAME);
         if type = 'CHAR' and format ne '' then output;
      run;
```

The creation of the NEWSTRING variable creates a list of all character variable names. For example, if there were five valid character variables (from VAR1 to VAR5) to conduct frequency analysis, the final observation in this data set will contain:

```
        newstring='VAR1 VAR2 VAR3 VAR4 VAR5'
```

The subsequent data step uses CALL SYMPUT to assign a global scope variable, VARIABLE_LIST2, to represent the contents of the NEWSTRING variable.

```
      data character_w_formats;
         set character_w_formats;
         i+1;
         call symput ("VARIABLE_LIST2", newstring);
         call symput ("OBSERVATION_COUNT", i);
      run;
```

We further define the dimension of the future array to be used by creating a global variable called OBSERVATION_COUNT from the assignment of a row counter variable, i, to ensure the array has proper dimensionality.

```
   data _null_;
         array varlist(&OBSERVATION_COUNT) &VARIABLE_LIST2 ;
         do i = 1 to dim(varlist);
            %put &new_dataset_name;
            call execute( '%diafreq(' || vname(varlist(i)) || ')' );
         end;
      run;
   %end;
```

The key step in the macro involves the creation of the VARLIST array, whose dimensions are dynamically defined by global variable OBSERVATION_COUNT, and whose contents are defined by VARIABLE_LIST2, which is derived from a previously created local variable, NEWSTRING. VARIABLE_LIST2 is merely a space-delimited list of all character variable names to be used for array processing. The DO-LOOP will execute the %diafreq macro using the CALL EXECUTE procedure with the macro call defined dynamically.

```
       call execute( '%diafreq(' || vname(varlist(i)) || ')' );
```

CALL EXECUTE will execute the string within parentheses as a command. The VNAME function will take the i-th element of the VARLIST array and process frequencies using the %diafreq macro. For example, if VAR1 is the first variable name and first element of the VARLIST array, then the explicit macro call would resolve to:

```
       call execute( '%diafreq(VAR1)');
```

THE FREQUENCIES GENERATING MACRO

The %diafreq macro is included below with attention to specific coding changes:

```
%macro diafreq(varname);
   %put &new_dataset_name._2;
   proc freq data= &dataset_name1._2;
      tables &varname/ missing out = &varname;
      *create the output dataset for the frequencies;
    run;

   data &varname (rename = (&varname = Category));
      set &varname;
      format NAME $200.;
      NAME = left(trim("&varname"));
      label &varname = "Category";
      if &varname = '' then do;
         &varname = 'M'; *if the category is missing, assign a dummy value;
      end;
    run;

    proc sort data = &varname; by  NAME; run;
    proc sort data = character_w_formats; by  NAME; run;

    data &varname;
       merge &varname (in=a)
       character_w_formats (in=b keep = varnum NAME DATASET_NAME FORMAT dataset_id);
          by  NAME;
          if a;
    run;

   data &varname (drop = category);
       length format $200.;
       length category2 $200.;
       set &varname;
       format = left(trim(format)) || '.';
       category2 = put(left(trim(category)), $200.);
    run;

    proc append data = &varname base = final force; run;
%mend diafreq;
```

The PROC FREQ calculates frequencies only for the macro parameter VARNAME that was passed from the %processing macro. This implies the %diafreq macro gets called for each character variable in the array VARLIST.

The first data step provides some formatting and adjustments with the creation of the NAME variable that fills down the VARNAME for each row in the temporary data set. If a category happens to be missing among a categorical character variable, then a dummy value is assigned as an "M." In the subsequent data step, CATEGORY2 variable is created with a defined length and format. This is to prevent issues concerning multiple formats occurring in a single column as a result of using the FORCE option with PROC APPEND. This technique is necessary to remove warnings in the SAS log that indicate different format types that are used. If one does not use the PUT statement to specify the CATEGORY2 variable, the initial format on the base data set would be used instead of the new format being appended with the new set of frequencies. Therefore, when the VARNAME data set appends to the existing FINAL data set, the format that exists for CATEGORY2 would be used instead of using the new formats. This will lead to redundancy of the same formats; therefore, ultimately leading to erroneous results.

PROC SORT and basic merge of the character_w_formats data set with the VARNAME data set with final adjustment of the FORMAT variable with the concatenation of the period yields a data set that is appended to the data set named FINAL. This data set will become the main driver for the post-macro processing data steps to prepare for export to Excel or Access.

If there is a need for all upper case variable names, which is sometimes the case for ease of analysis and export, invoke the OPTIONS statement:

```
options validvarname = upcase;
```

Any data steps or PROC SQL that generates new data sets will create column names that are all capitalized. This overrides the need to specify the UPCASE function in SAS for specific variables in data steps or SELECT statements in PROC SQL. After the frequencies processing, the resultant data set has the following records for categorical variables with associated formats.

Format	Category	COUNT	PERCENT	NAME	VARNUM	DATASET_NAME	DATASET_ID
$SEXFMT	0	6	0.06	SEX	3	MY_DATASET	1
$SEXFMT	F	9919	99.19	SEX	3	MY_DATASET	1
$SEXFMT	M	75	0.75	SEX	3	MY_DATASET	1
$RACEEDB	M	8772	87.72	EDB_RACE	4	MY_DATASET	1
$RACEEDB	0	1228	12.28	EDB_RACE	4	MY_DATASET	1
$RACEFMT	M	8772	87.72	PMMIS_RACE	5	MY_DATASET	1
$RACEFMT	5	1228	12.28	PMMIS_RACE	5	MY_DATASET	1
$RACEFMT	1	64	0.64	race	14	MY_DATASET	1
$RACEFMT	2	125	1.25	race	14	MY_DATASET	1
$RACEFMT	3	2726	27.26	race	14	MY_DATASET	1

For example, the SEX variable has three categories (0, M, F), with varying counts and percentages with an associated format called SEXFMT. If more interested in the underlying formatted values, perform a PROC SORT and merge against the FORMAT field with user-defined format libraries. Frequencies for these categorical variables in a single data set can be used to import to Microsoft Access or another RDMS to aid in the creation of a data dictionary application.

MACRO POST-PROCESSING AND CONCLUDING REMARKS

Given the data set above, this format is very useful merging with format libraries in SAS to understand formatted values for the CATEGORY variable. In many data repositories with a myriad of categorical variables, formatted values for these categories may be necessary to understand the physical meaning of the variable.

CONCLUSION:

SAS has the power to aggregate attributes of archived data sets for users in a development or analysis team to understand the contents of all these data sets as well as the expected frequencies for key categorical variables. Although there are alternatives to generate these summarizations, this is one method being implemented that involves generating frequencies for categorical variables containing both numeric and character formats. The necessary input parameters with a SAS data set repository involve knowing the library and data set name to generate aggregated frequencies and associated formats.

CONTACT:

Neil Kamdar
University of Michigan

School of Public Health, Department of Biostatistics
Kidney Epidemiology and Cost Center (KECC)
1415 Washington Heights, Suite 3645
Ann Arbor, MI 48109-2029
Phone: 734-358-6759
Email: neilseal@umich.edu

REFERENCES:

Slaughter, Susan & Lora D. Delwiche (2003). *The Little SAS Book: A Primer.* Cary, NC: SAS Publishing.

CODE APPENDIX:

```
proc contents data = my_lib.my_dataset out = my_lib.test;run;

proc sql;
create table my_lib.test as
select MEMNAME, VARNUM, NAME,
   case
     when TYPE = 2 then 'CHAR'
     when TYPE = 1 then 'NUM'
   end as TYPE,
   LENGTH, FORMAT, INFORMAT, LABEL, LIBNAME
from
   my_lib.test;
quit;

data my_lib.test1;
   set my_lib.test;
   DATASET_NAME = upcase(MEMNAME);
   DATASET_ID = 1;
   Library_name1 = upcase(LIBNAME);
run;

%macro processing (library_name, dataset_name1, iterator, library_name2,
dataset_name2);
   %let new_dataset_name = &dataset_name1;
   data character_w_formats;
      set &library_name..&new_dataset_name.&iterator.;
      if type = 'CHAR' and format not in ('','$') then output;
   run;
       /*get observation count for the conditional logic of producing frequencies*/
   proc sql noprint;
      select count(*)
      into :OBSCOUNT
      from character_w_formats;
   quit;

   %put Count=&OBSCOUNT.;
   %if &OBSCOUNT > 0 %then %do;
      proc sort data = character_w_formats;    by varnum name; run;

data character_w_formats
(keep = newstring varnum DATASET_NAME NAME FORMAT library_name1 dataset_id);
         set character_w_formats;
         length newstring $2000;
         by varnum name;
         retain newstring;
         newstring = trim(newstring)||' '|| trim(NAME);
         if type = 'CHAR' and format ne '' then output;
      run;

      data character_w_formats;
         set character_w_formats;
         i+1;
         call symput ("VARIABLE_LIST2", newstring);
         call symput ("OBSERVATION_COUNT", i);
      run;

      data &dataset_name1._2;
         set &library_name2..&dataset_name2.;
      run;

      %put &VARIABLE_LIST2; %put &OBSERVATION_COUNT; %put &new_dataset_name._2;

      data _null_;
         array varlist(&OBSERVATION_COUNT) &VARIABLE_LIST2 ;
         do i = 1 to dim(varlist);
            %put &new_dataset_name;
            call execute( '%diafreq(' || vname(varlist(i)) || ')' );
         end;
      run;
   %end;
%mend;
%macro diafreq(varname);
   %put &new_dataset_name._2;
   proc freq data= &dataset_name1._2;
      tables &varname/ missing out = &varname; *create the output dataset for the
frequencies;
    run;

   data &varname (rename = (&varname = Category));
      set &varname;
      format NAME $200.;
      NAME = left(trim("&varname"));
      label &varname = "Category";
      if &varname = '' then do;
         &varname = 'M'; *if the category is missing, assign a dummy value;
      end;
    run;

    proc sort data = &varname; by  NAME; run;
    proc sort data = character_w_formats; by  NAME; run;

    data &varname;
       merge &varname (in=a)
       character_w_formats (in=b keep = varnum NAME DATASET_NAME FORMAT dataset_id);
          by  NAME;
          if a;
    run;

   data &varname (drop = category);
       length format $200.;
       length category2 $200.;
       set &varname;
       format = left(trim(format)) || '.';
       category2 = put(left(trim(category)), $200.);
    run;

    proc append data = &varname base = final force; run;
%mend diafreq;
%processing (my_lib, test, 1, my_lib, my_dataset);
```

TRADEMARK CITATION: