Paper BB13-2013

Creating Formats on the Fly

Suzanne M. Dorinski, U.S. Census Bureau, Washington DC

ABSTRACT

The Census Bureau conducts the Common Core of Data surveys for the National Center for Education Statistics annually. We have written SAS® programs to automate the database documentation. We try to avoid including hard-coded values in the programs. Thanks to a record layout spreadsheet, the analysts can quickly update the survey metadata outside the SAS programs. This paper explains how SAS can read the record layout spreadsheet to create formats on the fly. The analysts can update the values as changes occur over time without having to worry about writing correct SAS syntax. Behind the scenes, SAS is using dictionary views, macros, ODS OUTPUT, PROC TEMPLATE, PROC FORMAT, the ODS Report Writing Interface, and RTF to create the desired results. This paper uses syntax for SAS 9.2, written for programmers at the intermediate level.

DISCLAIMER

This report is released to inform interested parties of ongoing research and to encourage discussion of work in progress. Any views expressed are those of the author and not necessarily those of the U.S. Census Bureau.

BACKGROUND

The Common Core of Data consists of a series of annual surveys about public elementary and secondary education in the United States. This paper focuses on the Local Education Agency (School District) Universe. Information collected for the school districts includes phone number, location and type of agency, current number of students, number of high school graduates and completers in the previous year.

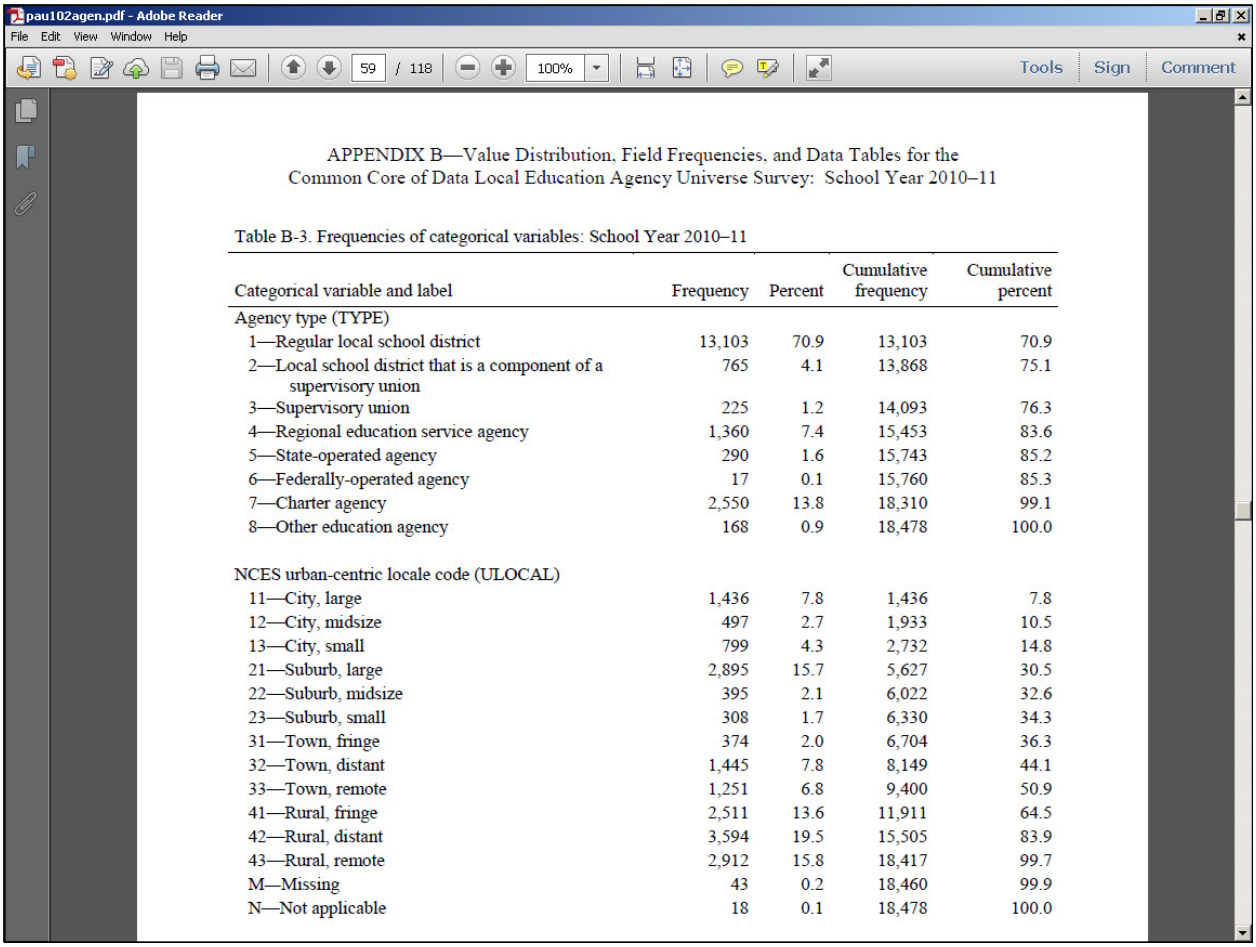
The National Center of Education Statistics (NCES) makes the data files available to the public on their web site. http://nces.ed.gov/ccd/pubagency.asp is the link for the school district universe data. NCES provides the data as both a flat file and a SAS data set. They provide both SAS and SPSS code so that users can read the flat file into the desired statistical package. NCES provides a record layout for the flat file. NCES also publishes database documentation that explains the methodology and the guidelines for using and processing the data. Appendix B of the database documentation includes tables that show the distributions of character variable responses (missing / not applicable / reported); the minimum, maximum, and mean values for continuous variables, and frequencies of categorical variables.

This paper explains how we create formats on the fly for the frequencies of categorical variables. We will use the Local Education Agency Universe Survey for School Year 2010-11, version provisional 2a as the example. http://nces.ed.gov/ccd/Data/zip/ag102a_sas7bdat.zip is the link for the public-use version of this data set, while http://nces.ed.gov/ccd/pdf/pau102agen.pdf is the link for the database documentation. Table B-3 in the database documentation lists the frequencies of the categorical variables.

The complete code for the example discussed in this paper is located at http://www.sascommunity.org/wiki/Creating_Formats_on_the_Fly.

DESIRED OUTPUT

Figure 1 shows the desired output for the first two categorical variables on the file. The variables are TYPE and ULOCAL. While PROC FREQ will produce the counts and percentages easily, the default output will not be close to the desired output. By default, PROC FREQ shows the percentages to two decimal places, and the counts do not use a comma format. PROC FREQ centers the variable's label above the results, and uses the variable name as the column header in the column that shows the values. Figure 2 shows what PROC FREQ produces by default for the variables TYPE and ULOCAL.



APPENDIX B—Value Distribution, Field Frequencies, and Data Tables for the Common Core of Data Local Education Agency Universe Survey: School Year 2010–11

Table B-3. Frequencies of categorical variables: School Year 2010–11

Categorical variable and label	Frequency	Percent	Cumulative frequency	Cumulative percent
Agency type (TYPE)				
1—Regular local school district	13,103	70.9	13,103	70.9
2—Local school district that is a component of a supervisory union	765	4.1	13,868	75.1
3—Supervisory union	225	1.2	14,093	76.3
4—Regional education service agency	1,360	7.4	15,453	83.6
5—State-operated agency	290	1.6	15,743	85.2
6—Federally-operated agency	17	0.1	15,760	85.3
7—Charter agency	2,550	13.8	18,310	99.1
8—Other education agency	168	0.9	18,478	100.0
NCES urban-centric locale code (ULOCAL)				
11—City, large	1,436	7.8	1,436	7.8
12—City, midsize	497	2.7	1,933	10.5
13—City, small	799	4.3	2,732	14.8
21—Suburb, large	2,895	15.7	5,627	30.5
22—Suburb, midsize	395	2.1	6,022	32.6
23—Suburb, small	308	1.7	6,330	34.3
31—Town, fringe	374	2.0	6,704	36.3
32—Town, distant	1,445	7.8	8,149	44.1
33—Town, remote	1,251	6.8	9,400	50.9
41—Rural, fringe	2,511	13.6	11,911	64.5
42—Rural, distant	3,594	19.5	15,505	83.9
43—Rural, remote	2,912	15.8	18,417	99.7
M—Missing	43	0.2	18,460	99.9
N—Not applicable	18	0.1	18,478	100.0

Figure 1. Screen shot of first page of Table B-3 in database documentation

Education Agency Type Code

TYPE	Frequency	Percent	Cumulative Frequency	Cumulative Percent
1	13103	70.91	13103	70.91
2	765	4.14	13868	75.05
3	225	1.22	14093	76.27
4	1360	7.36	15453	83.63
5	290	1.57	15743	85.20
6	17	0.09	15760	85.29
7	2550	13.80	18310	99.09
8	168	0.91	18478	100.00

NCES urban-centric locale code

ULOCAL	Frequency	Percent	Cumulative Frequency	Cumulative Percent
11	1436	7.77	1436	7.77
12	497	2.69	1933	10.46
13	799	4.32	2732	14.79
21	2895	15.67	5627	30.45
22	395	2.14	6022	32.59
23	308	1.67	6330	34.26
31	374	2.02	6704	36.28
32	1445	7.82	8149	44.10
33	1251	6.77	9400	50.87
41	2511	13.59	11911	64.46
42	3594	19.45	15505	83.91
43	2912	15.76	18417	99.67
M	43	0.23	18460	99.90
N	18	0.10	18478	100.00

Figure 2. PROC FREQ default output for TYPE and ULOCAL

MOTIVATION FOR ELIMINATING HARD CODED PROC FORMAT WITHIN SAS PROGRAM

While it is possible to use PROC FORMAT within the SAS program to make the value 1 show up as "1—Regular local school district" for the TYPE variable, the analysts who run the SAS programs may not be familiar with the PROC FORMAT syntax. The Common Core of Data is an annual data collection, and the values for variables may change over time, or the descriptive text for an existing value may change.

It was simple to add a few more variables to the metadata spreadsheet that was being prepared. We have agreed on a standard way of entering the information that SAS needs to create the formats. The metadata spreadsheet can easily be updated as changes occur. The only part of the SAS program that needs to change each year is the values of the macro variables at the top of the program, where the input and output directories and the file names are specified.

We were influenced on earlier projects by two papers (Dilorio and Abolafia, 2004), (Dilorio and Abolafia, 2006).

USE OF DOC_FORMATS VARIABLE IN METADATA SPREADSHEET

SAS uses the variable DOC_FORMATS in the metadata spreadsheet to build the formats on the fly. We use the pipe character, denoted |, as the delimiter in DOC_FORMATS. The first word in DOC_FORMATS is whatever descriptive text shows up in Table B-3 about the variable, while the remaining words are the formatted values for the PROC FREQ output. Table 1 shows DOC_FORMATS for the variables TYPE and ULOCAL.

Table 1. DOC_FORMATS values for TYPE and ULOCAL

Variable	DOC_FORMATS
TYPE	Agency type | 1\emdash Regular local school district |2\emdash Local school district that is a component of a supervisory union |3\emdash Supervisory union | 4\emdash Regional education service agency |5\emdash State-operated agency |6\emdash Federally-operated agency | 7\emdash Charter agency |8\emdash Other education agency
ULOCAL	NCES urban-centric locale code|11\emdash City, large |12\emdash City, midsize |13\emdash City, small |21\emdash Suburb, large|22\emdash Suburb, midsize|23\emdash Suburb, small |31\emdash Town, fringe |32\emdash Town, distant |33\emdash Town, remote |41\emdash Rural, fringe |42\emdash Rural, distant |43\emdash Rural, remote |M\emdash Missing |N\emdash Not applicable

\emdash is an RTF control word, which creates — in the RTF output. Figure 3 shows the relationship between the DOC_FORMATS values and the desired output.

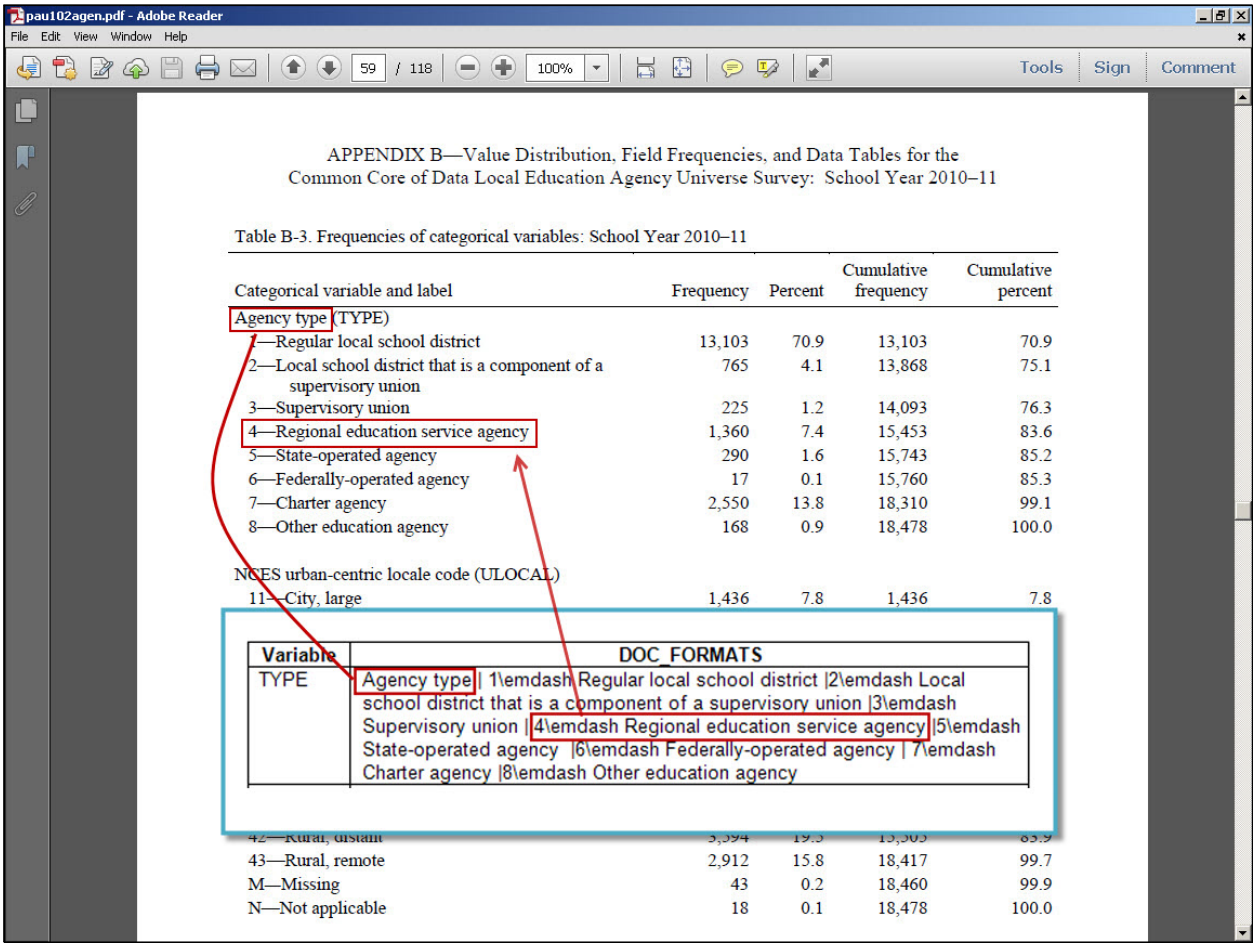


Figure 3. The relationship between DOC_FORMATS values and the desired output

USE COUNTW AND SCAN FUNCTIONS TO PROCESS DOC_FORMATS

In the SAS code below, we use PROC SQL to create the data set var_list. PROC SQL is reading the variable name and type from the dictionary view VCOLUMN, located in the SASHELP library. The variable name on the metadata spreadsheet is DELIVNAME, while the variable type on the metadata spreadsheet is FIELD_TYPE, so we rename the variable from the dictionary view in PROC SQL.

The PROC IMPORT step reads the metadata spreadsheet. The subsetting IF in the freq_list data step is selecting the variables used on the school district universe part of the Common Core of Data (survey='AGN') and only those variables that will be needed for Table B-3 (database_doc_table='FREQ').

We merge the two data sets in the updated_freq_list data step. We use the variable list from the dictionary view as a check that the metadata spreadsheet is current for new variables. New variables will show up in Table B-3 without formatted values if the metadata spreadsheet is not current.

We use the COUNTW function to count how many words DOC_FORMATS contains. Then we use the SCAN function to assign the descriptive text to the variable named LABEL_PART. Since we specified the pipe delimiter in both functions, SAS will only recognize that symbol as the delimiter between words.

```
proc sql;
       create table var_list as
       select name as delivname,
                     type as field_type
              from sashelp.vcolumn
              where libname="LEA" AND
                     memname="%upcase(&dsn)";
quit;

PROC IMPORT OUT= WORK.meta_spreadsheet
              DATAFILE= "&spreadsheet_dir\&spreadsheet_FULL_NAME"
              DBMS=XLS REPLACE;
        textsize=32000;
        guessingrows=500;
RUN;

data freq_list;
       set meta_spreadsheet;
       if survey='AGN' and upcase(database_doc_table)='FREQ';
run;

proc sort data=freq_list;
       by delivname;
run;

proc sort data=var_list;
       by delivname;
run;

data updated_freq_list;
       merge freq_list(in=f)
              var_list;
       by delivname;
       if f;
       words_to_process=countw(doc_formats,'|');
       label_part=scan(doc_formats,1,'|');
run;
```

PROCESSING THE REST OF DOC_FORMATS

The first PROC SQL below is a quick bit of housekeeping. We find the maximum length of DOC_FORMATS and store it in the macro variable BIG_NUMBER_FOR_LENGTH_STATEMENT. That value comes in handy later in the program, to ensure that variable lengths are uniform across data sets when using PROC APPEND.

To finish processing the rest of the DOC_FORMATS text, we need to know how many words are possible. We use the second PROC SQL below to assign the maximum number of words to the macro variable WORD_COUNT, and then create another macro variable MAX where we subtract 1, since the first word is always the descriptive text on the left side of the variable name in the desired output.

The get_word macro deals with the second through last words in DOC_FORMATS. It pulls each word off and assigns it to FOOTER_1 through FOOTER_&MAX (max is 16 in this example). [A note about variable names: In an earlier version of this program, the explanatory text for each value was listed below the PROC FREQ output. That is why the variables were named FOOTER_1 through FOOTER_&MAX. We kept the variable names when we were asked to provide the explanatory text within the PROC FREQ output, rather than below it.]

```
proc sql noprint;
      select max(length(doc_formats)) into :big_number_for_length_statement
            from updated_freq_list;
quit;

proc sql noprint;
      select max(words_to_process) into :word_count
            from updated_freq_list;
quit;

%let max=%eval(&word_count-1);

proc sort data=updated_freq_list;
      by delivorder;
run;

* get_word macro is pulling 2nd through last words in doc_formats variable,
* assigning to footer_ variables, so that we can build formats on the fly. ;

%macro get_word(upper);
      %do i=1 %to &upper;
            temp=left(scan(doc_formats,&i,'|'));
            %let j=%eval(&i-1);
            footer_&j=temp;
      %end;
%mend get_word;
```

CREATING THE FORMATS ON THE FLY

The format_info_to_transpose data set calls the get_word macro and creates the footer variables (FOOTER_1 through FOOTER_16 for this example). We then transpose the data to get it into the shape needed for a control data set for PROC FORMAT.

1\emdash Regular local school district is the value of footer_1 for TYPE. In the data step after the PROC TRANSPOSE, we exclude rows where the footer value is blank. The label variable contains the formatted value we want displayed in the PROC FREQ output. We use the resolve function to assign the value of COL1 to the label variable. Here we use the scan function to pull off the first word, but this time we use \ as the delimiter. We use \ as the delimiter here because every footer variable consists of the value followed by \emdash and then the explanatory text. PROC FORMAT needs to know what the variable value is, which is assigned to the start variable. FMTNAME is the name of the format, and here we use the value of DELIVNAME.

We are creating a format for each variable in the output, and we are naming that format with the same name as the variable name. That makes programming the macro a bit easier. All but one of the variables in the desired output is a character variable. By setting type='c', we do not have to use $ as the first character of the format name, because PROC FORMAT will handle that for us. For more details on creating a format from a SAS data set, see Wright (2007).

Once we have the control data set, we use PROC FORMAT to create the formats in a temporary catalog.

```
data format_info_to_transpose(keep=delivname field_type
                                        footer_1--footer_&max );
      set updated_freq_list(keep=delivname field_type doc_formats);
      %get_word(&word_count)
run;

proc transpose data=format_info_to_transpose
               out=transposed_format_info;
      var footer_1-footer_&max;
      by delivname field_type notsorted;
run;

* \ is delimiter in footer_ values, since value is ALWAYS
* followed by \emdash.
* first word is value needed to build format. ;

data format_control_data_set(keep=fmtname start label type);
      set transposed_format_info;
      if col1 ne '';  *exclude rows where footer value was blank ;
      label=resolve(col1);
      start=scan(label,1,'\');
      fmtname=delivname;
      if field_type='char' then type='c';
*setting type='c' means i don't have to add $ in front of fmtname –
*proc format will do that. ;
run;

proc format library=work cntlin=format_control_data_set;
run;
```

Figure 4 shows the formats created for TYPE and ULOCAL.

FORMAT NAME: $TYPE LENGTH: 73 NUMBER OF VALUES: 8
MIN LENGTH: 1 MAX LENGTH: 73 DEFAULT LENGTH 73 FUZZ: 0

START	END	LABEL (VER. V7|V8 10JUN2013:20:01:46)
1	1	1\emdash Regular local school district
2	2	2\emdash Local school district that is a
3	3	3\emdash Supervisory union
4	4	4\emdash Regional education service agen
5	5	5\emdash State-operated agency
6	6	6\emdash Federally-operated agency
7	7	7\emdash Charter agency
8	8	8\emdash Other education agency

FORMAT NAME: $ULOCAL LENGTH: 25 NUMBER OF VALUES: 14
MIN LENGTH: 1 MAX LENGTH: 40 DEFAULT LENGTH 25 FUZZ: 0

START	END	LABEL (VER. V7|V8 10JUN2013:20:01:46)
11	11	11\emdash City, large
12	12	12\emdash City, midsize
13	13	13\emdash City, small
21	21	21\emdash Suburb, large
22	22	22\emdash Suburb, midsize
23	23	23\emdash Suburb, small
31	31	31\emdash Town, fringe
32	32	32\emdash Town, distant
33	33	33\emdash Town, remote
41	41	41\emdash Rural, fringe
42	42	42\emdash Rural, distant
43	43	43\emdash Rural, remote
M	M	M\emdash Missing
N	N	N\emdash Not applicable

Figure 4. Format information for TYPE and ULOCAL

THE FREQ_TABLE MACRO DOES THE PROC FREQ FOR EACH VARIABLE

In this section, we will discuss parts of the macro. Dorinski (2007) explains in more detail this processing approach. The macro handles one variable at a time, and stores the results in the categorical_report data set. We append the information for each variable to the categorical_report data set.

The _null_ data step is reading the information for a variable to display in the desired output. DELIVNAME is the name of the variable, LABEL_PART is the descriptive text, and DOC_ORDER is a variable from the metadata spreadsheet that tells us what order to list the variables in the desired output. For the TYPE variable, DELIVNAME is TYPE, while LABEL_PART is “Agency type”.

The ODS OUTPUT statement is selecting the output to keep from the PROC FREQ. For this example, we are only interested in the one-way frequencies. We rename the variables so that the PROC APPEND at the bottom of the loop works properly.

The table variable tells us which table statement generated the ODS OUTPUT observation. Since this program uses a table statement with only one variable, we do not need the table variable, so we drop it. F_<variable name> is the formatted value of the variable.

In the PROC FREQ, we check for the existence of a character or numeric format for the variable. If a format exists, we use it. The RACECAT variable on the school universe data set is numeric, while all the other variables in the desired output are character.

```
data _null_;
      obsnum=&i;
      set updated_freq_list point=obsnum;
      if _error_ then abort;
      call symputx('field',delivname);
      call symputx('label_part',label_part);
      call symputx('doc_order',doc_order);
      stop;
run;

ODS OUTPUT OneWayFreqs=&field._freq_output(drop=table
                                           rename=(f_&field=f_field
                                            &field=temp_value));

proc freq data=lea.&dsn;
      tables &field / list missing;
      %if %sysfunc(cexist(work.formats.&field..formatc))
            %then format &field $&field.. ;;
      %if %sysfunc(cexist(work.formats.&field..format))
            %then format &field &field.. ;;
run;
```

We use the PROC SQL below to test if we have numeric data. All the information stored in the categorical_report data set needs to be character, so the data step after the PROC SQL is converting the results for RACECAT from the PROC FREQ. We use the macro variable in the length statement in that data set. If we do not specify the lengths of those variables, they will vary across survey items, and the PROC APPEND will fail when it tries to append variables with different lengths. The macro variable's value of 527 here is overly generous, but the program will automatically adjust as the survey metadata is changed.

```
/* need to know if value variable in &field._freq_output is
 numeric or character.  if numeric, need to convert it to
 character.  */

proc sql noprint;
      select type into :var_type
            from sashelp.vcolumn
                  where libname="WORK" and
                        memname="%upcase(&field._freq_output)" AND
                        name="temp_value";
quit;

/* use macro variable for length statement in data step so
that PROC APPEND will work properly.  length of field,
value, f_field, and label_part vary between survey items.
&big_number_for_length_statement is always large enough
to accomodate all survey items.  */

data &field._freq_output(drop=temp_value);
      length field value f_field label_part $
            &big_number_for_length_statement ;
      label f_field='formatted value';
      set &field._freq_output;
      field="&field";
      label_part="&label_part";
      doc_order=&doc_order;
      %if &var_type=char %then
            %do;
                  value=temp_value;
            %end;
      %else
            %do;
                  value=put(temp_value,best10.);
            %end;
run;

proc append base=categorical_report data=&field._freq_output;
run;
```

MODIFY TEMPLATES FOR DESIRED RESULTS

We modify the BASE.FREQ.ONEWAYLIST template so that the PROC FREQ output will display as desired. By default, PROC FREQ uses two decimal places for percentages. You can modify the template to show percentages to one decimal place instead.

Previous versions of the program modified the BASE.FREQ.ONEWAYFREQS template. However, there were syntax changes beginning in SAS 9.2. See SAS Usage Note 37442 at http://support.sas.com/kb/37/442.html for more details. As suggested in the Usage Note, the program now modifies the BASE.FREQ.ONEWAYLIST template instead.

We modify the RTF style template to control the overall formatting of the output document. Dorinski (2006) explains the techniques in more detail.

ODS REPORT WRITING INTERFACE USED TO ARRANGE DATA IN TABLE

We use the ODS Report Writing Interface to display the categorical_report data set as desired. The table in the output has five columns, but we want to have the label followed by the variable name in parentheses on the first row of the output for the variable. The ODS Report Writing Interface allows us that kind of control over the output. Dorinski (2008) explains an example in more detail.

The ODS Report Writing Interface was originally called ODS Object Oriented Features. SAS has published a tip sheet for the ODS Report Writing Interface. You can download it from http://support.sas.com/rnd/base/ods/Tipsheet_RWI.pdf. The syntax and Daniel O'Connor's SAS Global Forum 2009 paper about the ODS Report Writing Interface are available at http://support.sas.com/rnd/base/datastep/dsobject/index.html.

UNANTICIPATED BENEFIT FROM THIS APPROACH

We wrote this program (and others for other pieces of database documentation) to document the final data set created for the survey year. Because we worked to eliminate as much of the hard coding as possible, users of the program discovered that the programs were more robust than originally anticipated. They now run these programs on the edited but not yet imputed files, looking for anomalies in the data.

REFERENCES

Dilorio, Frank and Abolafia, Jeff. 2004. "Dictionary Tables and Views: Essential Tools for Serious Applications", *Proceedings of the Twenty-Ninth Annual SAS® Users Group International Conference*, available online at http://www2.sas.com/proceedings/sugi29/237-29.pdf.

Dilorio, Frank and Abolafia, Jeff. 2006. "The Design and Use of Metadata: Part Fine Art, Part Black Art", *Proceedings of the Thirty-first Annual SAS® Users Group International Conference*, available online at http://www2.sas.com/proceedings/sugi31/104-31.pdf.

Dorinski, Suzanne M. 2006. "How To Produce Almost Perfect RTF Output", *Proceedings of the Nineteenth Annual NorthEast SAS® Users Group Conference*, available online at http://www.nesug.org/proceedings/nesug06/io/io12.pdf.

Dorinski, Suzanne M. 2007. "A Lazy Programmer Case Study: Dynamic Macro Code To Deal With Changing Number of Variables Over Time", *Proceedings of the Twentieth Annual NorthEast SAS® Users Group Conference*, available online at http://www.nesug.org/proceedings/nesug07/ap/ap08.pdf.

Dorinski, Suzanne M. 2008. "Using ODS Object Oriented Features To Produce A Formatted Record Layout", *Proceedings of the Twenty-first Annual NorthEast SAS® Users Group Conference*, available online at http://www.nesug.org/proceedings/nesug08/bb/bb02.pdf.

Wright, Wendy L. 2007. "Creating a Format from Raw Data or a SAS® Dataset", *Proceedings of the SAS® Global Forum 2007 Conference*, available online at http://www2.sas.com/proceedings/forum2007/068-2007.pdf.

ACKNOWLEDGEMENTS

The author thanks Cindy Sheckells for suggesting that we extend the methods we originally developed for the State Library Agency Survey to the surveys that make up the Common Core of Data. The author thanks Suzanne McArdle for teaching her how to edit SharePoint wikis, which made it easier to get started on sasCommunity.org. The author thanks John Barrow, Mary Ann Koller, and Carma Hogue for reading the draft of this paper and providing helpful comments.

CONTACT INFORMATION

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

Suzanne M. Dorinski
US Census Bureau
GOVS HQ-6K062E
4600 Silver Hill Road
Washington DC 20233
301-763-4869
Suzanne.Marie.Dorinski@census.gov