A Comprehensive Codebook Generator

Naoko S. Stearns, Research Triangle Institute; RTP, NC
John R. Gerlach, IMS America; Plymouth Meeting, PA

Abstract

A recent project at Research Triangle Institute (RTI) involved a large transportation survey that required validating and packaging of the data, in the form of ASCII text files and SAS data sets, for use by the public domain. In addition, a detailed codebook was produced for each edited file, which included a frequency distribution for categorical data and descriptive statistics for continuous data, along with the usual information concerning data attributes.

In order to facilitate the process of generating a codebook for each edited file, RTI developed a SAS application that allowed those people involved with the project to produce the codebook independently. Upon creating the codebook listing, the member of the project team, or end-user, would export the listing from SAS creating a RTF (Rich Text Format) file, then, import the RTF file into a Word document producing the finished deliverable. This paper explains the SAS solution that generates the codebooks for a large survey project.

The Process

The transportation survey consists of many pages of raw data that produces different files consisting of 3,000 to 400,000 records and 30 to over 200 variables. Each file requires many validation checks that begin the rigorous task of cleaning the data. Once the data have been validated, two deliverable files are produced: an ASCII text file and a SAS data set. Finally, an in-house SAS application creates a detailed codebook representing the deliverable files.

The In-house Application

The codebook generator developed for the transportation survey project consists of several SAS programs: a driver program that allows the end-user to specify options and user-specified information stored in a CARDS statement and a format library; a program that creates the SAS data set representing the codebook; and, a report generator that produces the final product.

Typical of most projects, the in-house application contains many unique specifications that might complicate the scope of this paper, as well as exceed its preferred format. Consequently, the options and driver programs are not discussed. Rather, this paper expounds on the idea of including actual values or ranges with respective frequencies along with the usual information found in codebooks, namely the data attributes (e.g., type and length).

Several Requisites

The main program requires two end-user sources, as well as the edited (validated) file. One source indicates the type of analysis for each variable, that is, whether the application computes ranges as well as a frequency, depending on the nature of the variable, such as categorical and continuous data, rather than nominal data such as an ID number or zip code. The codebook generator at RTI obtains this information via a CARDS statement located in the driver program. The in-stream data consists of four fields denoting type of analysis (1,2, or 3), the variable identifier, the associated format, and a possible special code. Consider the following illustration.

```
cards;
1 WKBYAUTO $YORN.
1 WKBYVAN  $YORN.
2 HOUSEID  HOUSEID.
2 TIMETOWK TIMET.    994
2 PARKAMNT PRKAMT.   9994
3 STFIPS
3 MSA

     < more data >
;
```

Also, the end-user needs to define a format library, as appropriate, which the application uses in order to list discrete and range values, along with their respective meaning. Consider the following example.

```
proc format cntlout=fmtdata(keep=
   fmtname start label);

   value $yorn    ‘01’ = ‘01 = Yes’
                  ‘02’ = ‘02 = No’
                  ‘94’ = ‘94 = Legitimate Skip’
                  ‘98’ = ‘98 = Not Ascertained’
                  ‘99’ = ‘99 = Refused’;

   value houseid other = ‘ ‘;

   value timet   other = ‘ ‘
                   994 = ‘994 = Legitimate Skip’
                   998 = ‘998 = Not Ascertained’
                   999 = ‘999 = Refused’;

   value prkamt other = ‘ ‘
                 9994 = ‘9994 = Legitimate Skip’
                 9998 = ‘9998 = Not Ascertained’
                 9999 = ‘9999 = Refused’;
   run;
```

Given both end-user specified sources and the edited data file, the codebook program produces a very comprehensive codebook that exceeds the usual listing of data attributes.

The Report

The codebook report consists of the variable name, the data type, the length of its internal value, the label, and the values along with respective frequencies. The values listed in the report depends on how the user defines the variable, either as discrete, continuous, or as nominal data. Consequently, the codebook generator uses one of three computing methods for each variable, depending on what the user specified as appropriate. Also, the macros %freq, %range, and %charange perform the specified task.

Cetegorical Variables

Categorical variables denote distinct groups or levels that can be stored as either numeric or character variables. Regardless of data type, the RTI codebook generator produces a frequency distribution based on the formatted values of the variable. The macro %freqs performs this task.

Continuous Numeric Variables

Age, time, and income illustrate continuous data often found in surveys. For these variables, the codebook generator computes frequencies based on a format that denotes special values (e.g., 998, 999) or other, which represents nonmissing values. Then, it computes the minimum and maximum range of the variable, replacing the other category with that range. For this scenario, the macros %freq and %ranges accomplish the task.

Nominal Data

Consider the numeric variable denoting a household identifier, which emulates a continuous numeric variable. In this case, the codebook generator treats the variable as it would any other such variable. Obviously, however, you do not want to see every unique value listed, having a frequency of one occurrence. To avoid that from happening, the user specifies a format so that there is only one level of other category, if there are no missing values. Then, the %ranges macro computes the minimum and maximum values, superseding the nonmissing value with that range. The process requires a different approach for character data, hence, the macro %charange.

The Macro %freqs

Initially, the FREQ procedure creates an output data set that contains those values stored by the variable along with the respective frequencies. Then, utilizing the respective format for each variable, the CNTLOUT option of the FORMAT procedure creates the data set **temp**, which ensures that all possible values are represented, not only those found in the data. A subsequent Data step creates the so-called filler data set depending on the type of format, either character or numeric.

```
%macro freqs;
   proc freq data=&dslib..&indsn.;
      %do i = 1 %to &fn.;
         tables &&fv&i. / list missing noprint
            out=frq&i.(keep=&&fv&i. count);
         format &&fv&i. &&ff&i.;
         %end;
   run;

   %do i = 1 %to &fn.;
      proc format cntlout=temp(keep=start
         rename=(start=&&fv&i.));
         select &&fe&i.;
      run;
      %if “%substr(&&fe&i.,1,1)” eq “$”
         %then %do;
            data temp;
               retain count 0;
               set temp;
            run;
            %end;
         %else %do;
            data temp;
               retain count 0;
               set temp(rename=(&&fv&i.=dummy));
               &&fv&i. = input(dummy,??
                  best16.);
               if &&fv.&i. eq .
                  then delete;
               drop dummy;
            run;
            %end;
      data frq&i.;
         set frq&i.;
         contodr = _n_;
      run;
      data temp;
         set temp;
         contodr = _n_;
      run;
      proc sort data=frq&i.;
         by &&fv&i. contodr;
      run;
      proc sort data=temp;
         by &&fv&i. contodr;
      run;
```

The following merge creates a data set representing all values, even those having a frequency of zero. Thus, each variable has a respective data set that contains formatted values with it frequency of occurrence.

```
      data frq&i.;
         merge temp frq&i.(drop=contodr);
            by &&fv&i.;
      run;
      data frq&i.(drop=&&fv&i.);
         retain name ‘        ‘;
         set frq&i.;
         call vname(&&fv&i.,name);
      run;
   %end;
```

The following code concatenates the frequency data sets into a single data set. Then, the data set cbvlist that contains the user-specified information for each variable contributes the format name of each variable required for an equi-join using the SQL procedure. A final merge of the data sets **freqdata** and **fmtdata** produces the desired comprehensive data set.

```
   data freqdata;
      set %do i = 1 %to &fn.; frq&i. %end; ;
   run;
   proc sort data=cbvlist;
      by name;
   run;
   proc sort data=freqdata;
      by name contodr;
   run;
   data freqdata;
      merge freqdata(in=a) cbvlist;
         by name;
      if a;
   run;
   proc sort data=cbvlist;
      by fmtname;
   run;
   proc sort data=fmtdata;
      by fmtname labodr;
   run;
   proc sql;
      create table fmtdata as
      select cb.name, fm.label, fm.lablodr,
         cb.fmtname
         from cbvlist cb, fmtdata fm
         where cb.fmtname eq fm.fmtname;
   quit;
   data freqdata;
      merge freqdata(in=a) fmtdata;
         by name fmtname;
      if a;
   run;
%mend freqs;
```

The Macro %ranges

For each continuous numeric variable, the MEANS procedure creates an output data set that contains its minimum and maximum value, and, the subsequent Data step creates the variable **name** that identifies the analysis variable. Then, another Data step concatenates those data sets and creates the variable **label** that denotes the range of each analysis variable. Finally, the data set is sorted by the variable **name**.

```
%macro ranges;
   %do i = 1 %to &mn.;
      proc means data=&dslib..&indsn.(keep=
         &&mv.&i. %if &&scv&i. ne . %then
         where=(&&mv&i. lt &&scv&i.); )
         noprint;
         var &&mv&i.;
         output out=rang&i.(drop=_type_
            _freq_ min=min max=max;
       run;
       data rang&i.;
         length name $8;
         set rang&i.;
         name = “%upcase(&&mv&i.)”;
      run;
      %end;
   data rangdata;
      set %do i = 1 %to &mn.; rang&i. %end;
      label = ‘(‘ || trim(left(put(min,13.))) ||
        ‘-‘ || trim(left(put(max,13.))) || ‘)’;
   run;
   proc sort data=rangdata;
      by name;
   run;
%mend ranges;
```

The Macro %charange

The macro %charange deals with nominal, character type data. First, the macro sorts the data set by the variable of interest. Then, the following Data step obtains the first and last values, storing them in the variables *min* and *max*, respectively. Similar to the %range macro, another Data step concatenates the several data sets and creates the variable **label** that denotes the ‘range’ of each variable. Finally, the data set is sorted by the variable **name**.

```
%macro charange;
   %do i = 1 %to &cn.;
      proc sort data=&section..&indsn.(keep=
         &&cv&i. where=((&&cv&i. ne ‘’) and
         &&cv&i. ne ‘none’))) out=chrange&i.;
         by &&cv&i.;
      run;
      data chrang&i.;
         length name $8 min max $13;
         set chrang&i. end=eof;
            by &&cv&i.;
         retain min;
         name = “%upcase(&&cv&i.)”;
         if _n_ eq 1 then min = &&cv&i.;
         if eof then do;
            max = &&cv&i.;
            output;
            end;
      run;
      %end;
   data chrang;
      drop min max;
      set %do = 1 %to &cn.;
         chrang&i. %end; ;
      label = ‘(‘ || min || ‘ - ‘ ||
         max || ‘)’;
   run;
   proc sort data=chrang;
      by name;
   run;
%mend charange;
```

The Methods Combined

Given the validated data, the format library, and the user-specified method for computing frequencies, the macros %freq, %range, and %charange produce a data set containing the formatted values or ranges, along with the respective frequencies. Then, the several data sets are merged together by the common variable **name** to produce the data set used for reporting, as follows.

```
data dict.&outdsn.;
   merge freqdata rangdata chrang;
      by name;
run;
```

The Result

The REPORT procedure produces the actual codebook, which is part of the driver portion of the RTI codebook generator. The column attributes include: the variable name, label, code denoting value, and the frequency of occurrence. The partial output below lists the frequencies in juxtaposition with the formatted values, not usually found in most codebooks.

```
Target
Variable  Value Range                Freqs
_____________________________________________________

WKBYAUTO  01 = Yes                   8,772
          02 = No                    3,152
          94 = Legitimate Skip      13,048
          98 = Not Ascertained           9
          99 = Refused                   0

WKBYVAN   01 = Yes                     378
          02 = No                   11,544
          94 = Legitimate Skip      13,048
          98 = Not Ascertained          11
          99 = Refused                   0

HOUSEID   (1200302 - 1280295)       24,981

TIMETOWK  (0 - 600)                 11,751
          994 = Legitimate Skip     13,048
          998 = Not Ascertained        172
          999 = Refused                 10

PARKAMNT  (0 – 850)                    766
          9994 = Legitimate Skip    24,211
          9998 = Not Ascertained         4
          9999 = Refused                 0

STFIPS    (05      - 51     )            .

MSA       (0000    - 8880   )            .
```

Caveat

The label portion of a user-specified format includes the value being formatted. For example, the numeric format GENDF might be defined as follows:

```
proc format;
   value gendf  1 = ‘1 = Male’
                2 = ‘2 = Female’
               94 = ‘94 = Legitimate Skip’
               95 = ‘95 = Not Ascertained’
               99 = ‘99 = Refused’;
run;
```

Thus, the application utilizes tailor made formats. One alternative would be to export an existing format, modify the label to include the actual value, then create a revised format that is more suitable to the application. Consider the following SAS code that modifies the format GENDF.

```
proc format cntlout=gendf;
   select gendf;
run;

data gendf;
   length label $50;
   set gendf;
   if start eq end
      then label=trim(start) || ‘ = ’ || label;
      else label=trim(start) || ‘-‘ ||
         trim(end) || ‘ = ’ || label;
run;

proc format cntlin=gendf;
run;
```

Conclusion

The following code produces a typical codebook using the SQL procedure and Dictionary tables. However, the report does not contain information made available using the RTI codebook generator, which includes actual frequencies of formatted values or range of values, as specified by the user, depending on the nature of the data.

```
%macro codebook(lib=WORK,memtype=DATA);
   options number pageno=1;
   proc sql;
      create table _codebk_ as
      select memname label='Data Set', name
         label='Variable', upcase(type) as type
         label='Type', length label='Length',
         label label='Label', format
         label='Format', informat
         label='Informat'
         from dictionary.columns
         where libname = "%upcase(&lib.)" and
            memtype = "%upcase(&memtype.)";
   quit;
   proc print data=_codebk_ label;
      id memname;
      by memname;
      where memname ne '_CODEBK_';
      title "Codebook of %upcase(&lib.)”;
   run;
%mend codebook;
```

Author Information

Naoko S. Stearns
Research Triangle Institute
Research Triangle Park, NC 27709-2194
919.541.7369

John R. Gerlach
IMS America
600 West Germantown Pike
Plymouth Meeting, PA 19462-1048
610.832.5493