Paper 37-25

CODEBOOK: Taking Another Look At Your Data

Greg Silva, PPD, Wilmington, NC

ABSTRACT

You should have a good understanding of your data before you begin writing a program. A PROC CONTENTS of the data provides information about the data, but no examples of the data. A PROC PRINT shows the data, but in the case of large datasets, is impractical. Using PROC UNIVARIATE you can get a lot of information about the data, but it is not always helpful for the naive user.

For these reasons, it became clear that another tool that would combine some of the best features of these procedures was needed. The codebook program provides a variety of information about the actual data in datasets, and presents the information in a user friendly manner.

INTRODUCTION

A good understanding of the data you are working with will help in coding and analysis. By knowing the types of variables, as well as some idea of the ranges of variable values, the programmer and analyst will have an easier time understanding the underlying data. This can speed up analysis and programming.

The output from PROC CONTENTS provides the programmer with information about a variable: name, type, format, etc. It also shows the number of variables in the dataset, as well as the number of records in the dataset. PROC PRINT allows the analyst to see the actual data. With this information, it is easier to see how the output from any report generated from the data will look. The output from PROC UNIVARIATE shows the ranges of data values, as well as descriptive statistics that give a better perspective on the data.

The Codebook macro combines parts of all of these procedure outputs to produce a concise and useful report. Without having to manipulate datasets, you can use the Codebook macro to generate a reasonable snapshot of the data that you will be working with.

THE CODEBOOK PARAMETER CALL

The call to the Codebook macro requires little user input. The *libname* parameter to define the library and the *dsname* parameter to specify the dataset name are the only parameters that you need to run the Codebook macro. All others have defaults. The following is a sample call with some of the other parameters defined.

```
%codebook(LIBNAME = TEST,
          DSNAME  = ADVERSE,
          OUTTYPE = TEXT,
          FORMATS = NO,
          OUTDSN  = HERE.ADVERSE ) ;
```

Example of a call to codebook

The *outtype* parameter allows the user to specify the format of the output. Currently the options are TEXT to produce plain text output, and HTML to produce a Web page for the dataset specified. The Codebook macro is flexible enough to allow for any type of output, with appropriate programming code incorporated into the macro.

The *formats* parameter defaults to YES, allowing the Codebook macro to use defined formats. To use the default parameter for formats, you need to define a format library with the libname statement. If there are formats, the formatted and unformatted values are presented in the generated report. If there are no formats, you can specify that the report should be generated without formatted values.

The default output for the Codebook macro is a text file that can be printed on any printer. The only special character used is a page break, which occurs at the start of each dataset's output to break up the report. The other output option is HTML. When this is specified, an HTML output of the file is generated, with links to the top and bottom of the HTML file.

The *outdsn* parameter allows you to specify a dataset where the information from the report can be stored. This allows for manipulation of data normally generated by the Codebook macro. This allows you to generate reports other than those built into the macro.

WHAT'S HAPPENING INSIDE

The macro splits up the numeric and character variables. It produces a report for each, and combines them at the end. By using procedures that are efficient for each type of variable, the program can quickly collect information about the variables without having to make sure that the procedure works with that type of data. The following is a brief look at the processing used by the codebook.

The macro starts by identifying the types of variables. This includes generating a list of the numeric and character variables, as well as the counts for each, and a total count. Since certain procedures can only use numeric variables, the identification process is necessary.

Processing starts with using PROC UNIVARIATE on the numeric variables. Macro loops are used to generate unique variable names for the statistics that are generated and stored in the output dataset. This dataset is then transposed and sorted for merging.

Frequencies are generated with PROC FREQ. This generates a truer view of the unique variables than what PROC UNIVARIATE provides. The two datasets are then merged, with the counts of uniques generated, as well as all of the standard output for the numeric variables.

For character variables, a transpose is used to prepare the data for merging with the output of the PROC FREQ of character variables. The DATA step that merges the character data is similar to the DATA step that merged the numeric variables together: the count of uniques is calculated, and the output format for the report is defined.

Since minimum and maximum cannot be calculated on character variables, the DATA step also finds the "largest" and "smallest" values by taking the first and last in the sorted dataset. This will give some odd results in the final report, but it at least provides a feel for the data.

The final step is to set the numeric and character datasets together, and generate the report. Based on the *outtype* parameter, this is either straight text, or text wrapped in HTML. By using the *outdsn* parameter, you can also save the resulting dataset.

THE REPORT FROM CODEBOOK

The codebook macro provides certain dataset information in a single report. Information about the number of records in a dataset, the number of variables in a dataset, and the date that the dataset was last modified are provided in the header of the report. Example 1 is the header for a codebook output file.

This is information that is available from PROC CONTENTS. In fact, any information that can be captured by PROC CONTENTS output could be presented here.

The section of the output from Codebook macro that deals with variables combines information from PROC CONTENTS, PROC PRINT, and PROC UNIVARIATE. See example 2 for a report on a character variable in a dataset.

For each character variable in the dataset, the following information is provided:

- Name of variable
- Type of variable
- Length of the variable
- Format associated with the variable
- The minimum value of the variable
- The maximum value of the variable
- The number of blank values of a variable
- The number of non-blank values of a variable
- The number of unique values of a variable
- The five smallest unique values of a variable (using the collating sequence) with their count and as a percent of total.
- The five largest unique values of a variable (using the collating sequence) with their count and as a percent of total.

For numeric variables, extra statistics can be generated. The current version of the Codebook macro provides the following statistics for numeric variables, along with those specified above:

- The mean of numeric variables
- The median of numeric variables

Example 3 shows the information presented for numeric variables.

The program is currently set to print all unique values if there are less that 11, or the five highest and the five lowest values if there are more than 10 unique values. This is something that can be easily modified in the code.

A WRAPPER TO REPORT ON LIBRARIES

Codebook is a very useful tool for reporting on individual datasets. However, it could be inconvenient if you had to run the macro for every dataset that you were interested in. To remedy this situation, a wrapper program has been written that allows you to generate codebooks for all datasets in a given library.

By defining a couple of parameters, a single codebook file will be generated with all of the datasets within a single library. The resulting report is a single file with all dataset results if the output is text. For HTML output, a Web page is generated for each dataset.

The following is an example of a call to the wrapper program.

```
* Make sure to define a format library. ;
libname  library "mystudy:[format]";
libname  here '[]' ;

options fmtsearch=(library.formats) ;

* Point to the codebook macros.  ;
options sasautos=("tool:[tools.codebook]" ) ;

%allbooks(library  = MYSTUDY:[DATA.DERIVED],
          study    = Protocol X,
          dirtype  = Derived ,
          outtype  = HTML,
          alldsn   = HERE.MYDATA) ;
```

Call to wrapper program for Codebook

The *library* parameter specifies the physical location of the datasets. The *study* parameter allows the user to specify a more “user friendly” description of the data. The *dirtype* parameter is used to identify whether the data is RAW or DERIVED. If the *outtype* parameter is set to HTML, then the output will be an HTML file for each dataset, as well as an index file with links to each of the HTML files. The final parameter - *alldsn*, allows the user to specify a permanent dataset to store the data derived by the Codebook macro. This can be used for further data analysis.

METADATA GENERATION FOR OTHER ANALYSIS

A future direction of the Codebook macro is to generate metadata: data about data. A parameter in the Codebook macro allows the user to specify a dataset to store all of the information from the codebook. This dataset can be used to compare a variable across datasets, or to compare changes in data from one run to the next.

By providing a dataset to store the data collected by Codebook, you can use your own tools to further analyze the data. The current output is a dataset that contains the text that is printed from Codebook, as well as variables containing all of the information used to generate the report.

By checking on a variable across datasets, you can get a better picture about the data as a whole. For example, by comparing key fields across datasets, you can determine if merges are being done incorrectly.

With ODS capabilities expanding into Word documents and PDF files, a number of new formatting options will soon be available. By having the data in a dataset and using standard procedures like PROC REPORT, the output can be tailored to match the needs of nearly any user.

CONCLUSION

The Codebook macro has been useful for development and debugging. By providing an extensible list of information about datasets and variables, the Codebook macro can reduce the amount of ad hoc programming needed to learn more about the data.

With a standard output format, you can understand and discuss data issues more clearly. And, with a dataset containing the information from the report, you can generate custom reports using any number of tools, both home grown and commercial.

CONTACT INFORMATION

Greg Silva
PPD
3151 South 17th Street
Wilmington, NC 28412
Phone: (910) 772-6986
E-mail: greg.silva@wilm.ppdi.com

EXAMPLE OUTPUT FROM THE CODEBOOK MACRO

```
Codebook for  ABC123

Codebook For: ABC123    Directory: BIOSTAT  Dataset: LABS

Dataset: LABS ,              Observations in dataset: 2,769
Number of variables in data set: 56
Dataset last modified:     Tuesday, August 24, 1999 14:37:16
This listing generated on: Monday, August 30, 1999 11:51:26
```

Example 1 - Header Information from Codebook

```
AENAME                              Adverse Event Inv Term

   Character Variable, Length= 60
   Minimum .................. BRONCHITIS
   Maximum .................. TORN ARCHILLE TENDON (1995)
   Number unique ............      20
   Number missing ...........       1
   Number non-missing ........     62
   Unique value                                         Count     Percent
   -blank- ........................................         1        1.6%
   BRONCHITIS .....................................         1        1.6%
   BURNING ON FACE ................................         1        1.6%
   CARCINOMA - ENDOMETRIUM ........................         1        1.6%
   COLD SYMPTOMS ..................................         1        1.6%
   ---- 5 highest values ----
   SINUS INFECTION ................................         3        4.8%
   SINUSITIS ......................................         1        1.6%
   SORENESS IN FACE ...............................         1        1.6%
   TIGHTNESS OF SKIN ..............................         1        1.6%
   TORN ARCHILLE TENDON (1995) ....................         1        1.6%
```

Example 2 - Character variable output

```
AEEMON                             Adverse Event End Month

   Numeric Variable, Length=8, Format=MONTH.
   Minimum ....................        1  (JAN)
   Maximum ....................       12  (DEC)
   Mean ....................... 4.533333  (4.5)
   Median .....................        3  (MAR)
   Number unique ..............        7
   Number missing .............       48
   Number non-missing .........       15
   Unique value                    Count     Percent
   -missing- ...................      48       76.2%
   JAN .........................       2        3.2%
   FEB .........................       5        7.9%
   MAY .........................       3        4.8%
   JUL .........................       1        1.6%
   SEP .........................       1        1.6%
   DEC .........................       3        4.8%
```

Example 3 - Numeric variable output