SESUG 2023 Paper 154

Utilizing SAS® to Create HTML Codebooks

William Zachary Smith, RTI International

ABSTRACT

Creating codebooks for datasets is an important part of delivering a high-quality data product package to a client or stakeholder. Most times, codebooks for datasets can be delivered in an Excel spreadsheet, but other times clients or stakeholders may ask for codebooks in a more elegant and presentable format. However, that process can often become cumbersome when dealing with large datasets with hundreds of variables and is often prone to human error when manually transferring the information out of an Excel spreadsheet and into another format. To reduce the amount of effort required and room for error, a SAS program was created that reads in all the variable information from an Excel spreadsheet and transforms it into a formatted and easily navigable HTML codebook. This paper will demonstrate how to integrate HTML code into your SAS programs to create such codebooks, making the process of creating custom codebooks seamless, and sometimes even fun.

INTRODUCTION

RTI International is responsible for many large-scale surveys across the country, and delivering high quality and easily navigable codebooks is often a requirement when providing large and complex datasets to clients. Datasets with many variables are hard to navigate and cumbersome to deal with. In numerous studies at RTI, variable metadata is stored in an Excel spreadsheet and can include things such as variable labels, variable type, variable position, valid values, variable descriptions, etc. While excel spreadsheets are helpful for storing this information, they are often not the best format to present them in as the spreadsheets can be hard to navigate, tedious to read through, and at times an eye sore.

One of the many surveys that RTI conducts and processes requires delivering a codebook to a client that contains over 200 variables. Presenting the vast amounts of metadata on these 200 variables as a codebook via an Excel spreadsheet is not viable, nor preferred by the client. Therefore, the metadata on these variables has been presented and delivered in a Word document by manually transferring the information from the Excel spreadsheets containing the metadata to the Word document. However, this has proved cumbersome and prone to human error due to the vast amount of effort involved.

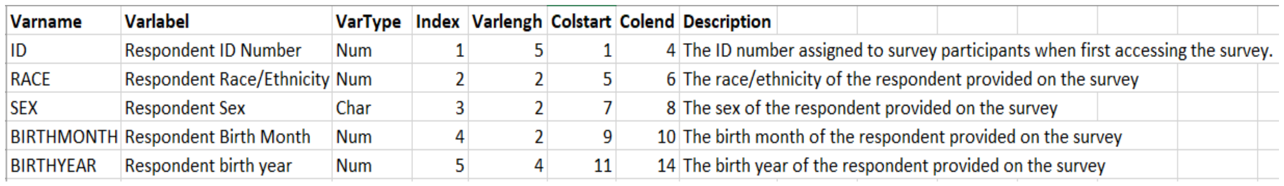
To reduce the amount of manual labor and potential room for human error, a SAS program was created that reads in all the variable information from the Excel spreadsheet and transforms it into a formatted and easily navigable HTML codebook. This paper will demonstrate how to integrate HTML code into your SAS programs to create these codebooks by providing an example Excel spreadsheet containing synthetic variable information. From there the paper will show steps on how to present your variable information in an easy and navigable format via a variable listing. Finally, the paper will then show how to create your actual variable entries based on the metadata in your Excel Spreadsheets. Techniques for implementing HTML code within SAS to produce this codebook is demonstrated throughout these steps. The methods used in this paper are simple, easy to learn, and relevant to SAS users of all levels. Readers should note that this paper assumes readers understand how to read and write basic HTML code. The purpose of this paper is not to teach HTML code to readers, but rather how to integrate HTML code into their SAS programs.

EXCEL METADATA AND VALUE LABELS LAYOUT

For this paper, 5 synthetic variables are created with their metadata loaded into an Excel file. This Excel file has two sheets. The first sheet, titled “Variables”, contains basic metadata information about the 5 synthetic variables and includes the following fields:

- Varname: Variable Name
- Varlabel: Variable Label
- VarType: Variable Type (Numeric or Character)
- Index: What order the variable appears in the dataset
- Varlength: Length of the variable
- Colstart: Starting position of the variable in the dataset
- Colend: Ending position of the variable in the dataset
- Description: Detailed variable description

Using these fields, the “Variables” sheet for our 5 synthetic variables is as follows:



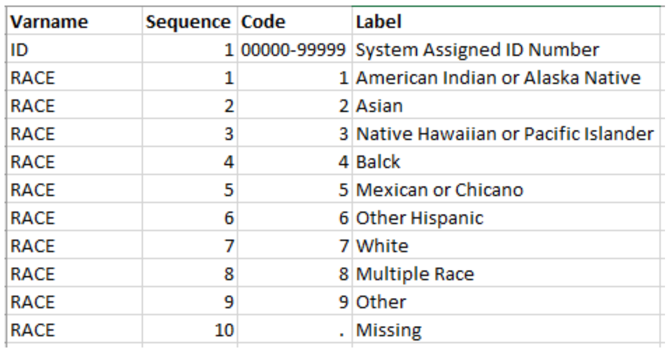
Varname	Varlabel	VarType	Index	Varlengh	Colstart	Colend	Description
ID	Respondent ID Number	Num	1	5	1	4	The ID number assigned to survey participants when first accessing the survey.
RACE	Respondent Race/Ethnicity	Num	2	2	5	6	The race/ethnicity of the respondent provided on the survey
SEX	Respondent Sex	Char	3	2	7	8	The sex of the respondent provided on the survey
BIRTHMONTH	Respondent Birth Month	Num	4	2	9	10	The birth month of the respondent provided on the survey
BIRTHYEAR	Respondent birth year	Num	5	4	11	14	The birth year of the respondent provided on the survey

Figure 1. Excel Spreadsheet Metadata Example

The second sheet in the Excel file, titled “Variable Codes and Labels” contains the variable code values and associated labels for each of the 5 synthetic variables and includes the following fields:

- Varname: Variable Name
- Sequence: Order the variable value code and label should appear in the codebook
- Code: The value that appears in your dataset for the variable
- Label: Label for the variable value

Using these fields, a snapshot of the “Variable Codes and Labels” sheet is as follows:



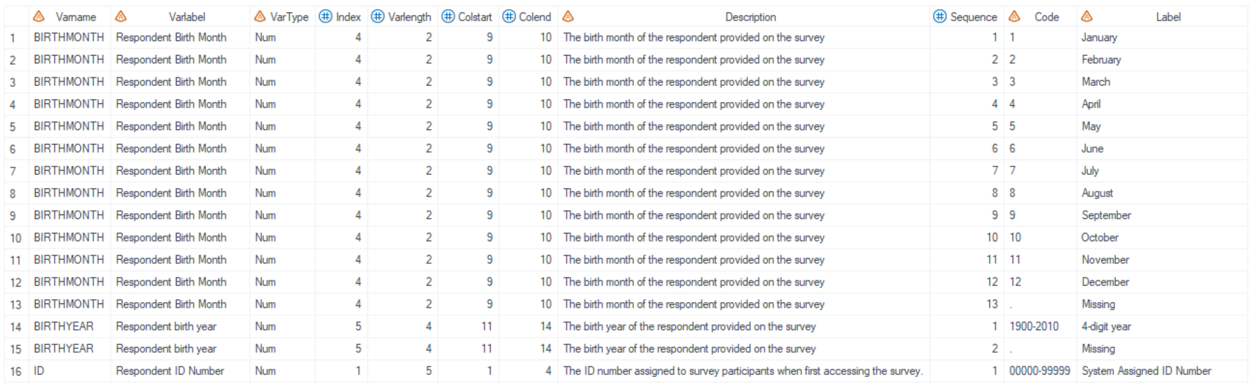
Varname	Sequence	Code	Label
ID	1	00000-99999	System Assigned ID Number
RACE	1	1	American Indian or Alaska Native
RACE	2	2	Asian
RACE	3	3	Native Hawaiian or Pacific Islander
RACE	4	4	Balck
RACE	5	5	Mexican or Chicano
RACE	6	6	Other Hispanic
RACE	7	7	White
RACE	8	8	Multiple Race
RACE	9	9	Other
RACE	10	.	Missing

Figure 2. Excel Spreadsheet Codes and Labels Example

Once the variable metadata and value labels are formatted in Excel as shown above, the two sheets will be read into SAS using PROC IMPORT. Afterwards, the program will merge the two sheets on variable name. This is shown in the following code snippet below:

```
Proc import datafile="C:/Users/wzsmith/SESUG
2023/SESUG_Codebook_Spreadsheet.xlsx"
        out=variables
        dbms=xlsx
        replace;
      sheet='Variables';
        run;

Proc import datafile="C:/Users/Users/wzsmith/SESUG
2023/SESUG_Codebook_Spreadsheet.xlsx"
        out=variable_values
        dbms=xlsx
        replace;
        sheet='Variable Codes and Labels';
        run;

proc sort data=variables; by varname; run;
proc sort data=variable_values; by varname; run;

data combined;
merge variables variable_values;
by varname;
run;
```

This will create the following dataset in SAS:



	Varname	Varlabel	VarType	Index	Varlength	Colstart	Colend	Description	Sequence	Code	Label
1	BIRTHMONTH	Respondent Birth Month	Num	4	2	9	10	The birth month of the respondent provided on the survey	1	1	January
2	BIRTHMONTH	Respondent Birth Month	Num	4	2	9	10	The birth month of the respondent provided on the survey	2	2	February
3	BIRTHMONTH	Respondent Birth Month	Num	4	2	9	10	The birth month of the respondent provided on the survey	3	3	March
4	BIRTHMONTH	Respondent Birth Month	Num	4	2	9	10	The birth month of the respondent provided on the survey	4	4	April
5	BIRTHMONTH	Respondent Birth Month	Num	4	2	9	10	The birth month of the respondent provided on the survey	5	5	May
6	BIRTHMONTH	Respondent Birth Month	Num	4	2	9	10	The birth month of the respondent provided on the survey	6	6	June
7	BIRTHMONTH	Respondent Birth Month	Num	4	2	9	10	The birth month of the respondent provided on the survey	7	7	July
8	BIRTHMONTH	Respondent Birth Month	Num	4	2	9	10	The birth month of the respondent provided on the survey	8	8	August
9	BIRTHMONTH	Respondent Birth Month	Num	4	2	9	10	The birth month of the respondent provided on the survey	9	9	September
10	BIRTHMONTH	Respondent Birth Month	Num	4	2	9	10	The birth month of the respondent provided on the survey	10	10	October
11	BIRTHMONTH	Respondent Birth Month	Num	4	2	9	10	The birth month of the respondent provided on the survey	11	11	November
12	BIRTHMONTH	Respondent Birth Month	Num	4	2	9	10	The birth month of the respondent provided on the survey	12	12	December
13	BIRTHMONTH	Respondent Birth Month	Num	4	2	9	10	The birth month of the respondent provided on the survey	13	.	Missing
14	BIRTHYEAR	Respondent birth year	Num	5	4	11	14	The birth year of the respondent provided on the survey	1	1900-2010	4-digit year
15	BIRTHYEAR	Respondent birth year	Num	5	4	11	14	The birth year of the respondent provided on the survey	2	.	Missing
16	ID	Respondent ID Number	Num	1	5	1	4	The ID number assigned to survey participants when first accessing the survey.	1	00000-99999	System Assigned ID Number

Figure 3. Snapshot of Merged Variable Metadata and Variable Label SAS Dataset

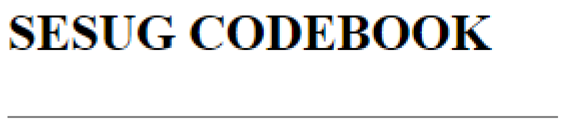
From there, a codebook title and a variable listing will be created and output to the HTML codebook file.

CODEBOOK TITLE AND VARIABLE LAYOUT LISTING

Once reading in the variable metadata information, the first step in creating your HTML codebook is creating the HTML file and using HTML code within SAS to output a codebook title to the HTML file. This is accomplished with the following SAS code within a DATA step:

```
data _NULL_;
file 'C:/Users/Users/Users/wzsmith/SESUG
2023/SESUG_CODEBOOK.html';
put '<p align="left" style="font-size:1.6em">
      <br>
      <strong> SESUG CODEBOOK </strong>
       <br>
       </p>'
;
put '<hr size="1" width="100%" noshade style="color:#000000"/>';
run;
```

The FILE statement points SAS to where the HTML file should be output, creates the initial starting HTML file, and names the file (named "SESUG CODEBOOK.HTML" in this example). Afterwards, HTML code is written in quotations within PUT statements to add a title and a line under that title. This produces the following output in an HTML file:



SESUG CODEBOOK

Figure 4. Initial HTML Codebook Title Output

From here, a variable listing is created under the title that lists out each variable in the imported Excel file, along with the file layout information provided in the "Variables" sheet of the Excel File. This is accomplished with the following SAS code:

```
data _NULL_;
file '/rtpnfil02/rtpnfil02_vol7/sed17/Users/wzsmith/SESUG
2023/SESUG_CODEBOOK.html' mod;
set variables;
by index;
if index=1 then do;
put '<table align=left cellspacing=8 cellpadding=1
style="width:35%; font_size:1.0em">
     <tr>
      <td><b>Variable Name</b></td>
      <td><b>Type</b></td>
      <td><b>Variable Label</b></td>
      <td><b>Index Number</b></td>
      <td><b>Variable Length</b></td>
      <td><b>Column Start</b></td>
      <td><b>Column End</b></td>
```

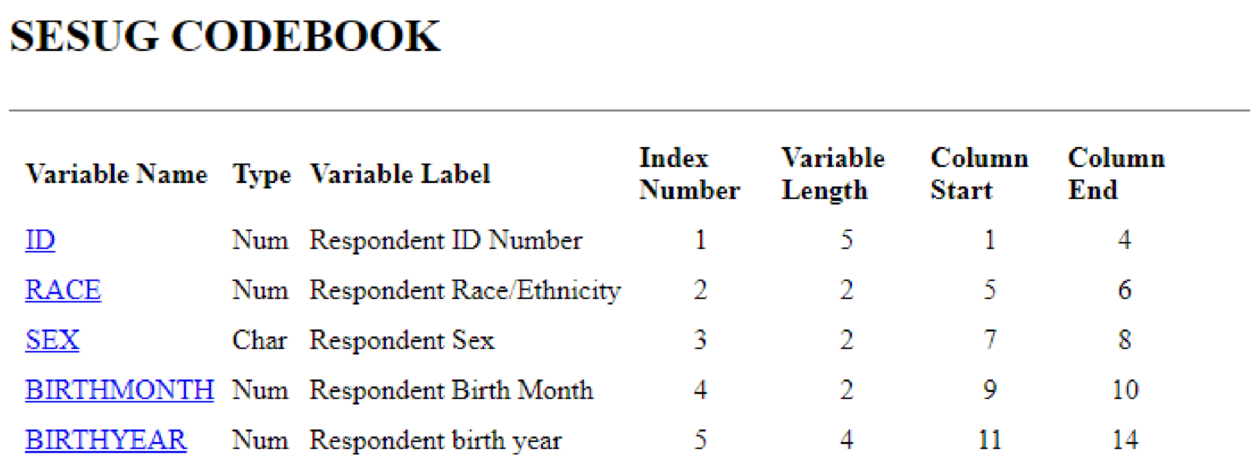
```
          </tr>';
    end;
    do until (last.index);
    put '<tr>
         <td align=left><a href="#' Index ' ">' Varname '</a></td>
          <td align=left>' VarType '</td>
          <td align=left nowrap>' VarLabel'</td>
          <td align=center>' Index '</td>
          <td align=center>' Varlength '</td>
          <td align=center>' Colstart '</td>
          <td align=center>' Colend '</td>
         </tr>';
    end;
    run;
```

As before, this is accomplished within a DATA step. The FILE statement points SAS to the HTML file. It's important to note that a MOD option is added to the end of the FILE statement here. This tells SAS to add the following HTML output to the file we first created when outputting the codebook title, rather than overwriting the HTML file with new output.

From there, a SET statement is used to point SAS to the "Variables" sheet that was first read in from our Excel file via a PROC IMPORT (before merging the file onto the variable values sheet). The BY statement then tells SAS to output the following HTML code for each entry in the "Variables" sheet since each entry in the "Variables" sheet has a unique index number.

The "if index=1 then do;" statement tells SAS to process the HTML code within the first PUT statement only once (rather than 5 times it would normally process this part of the code since there are 5 entries in the variable metadata sheet). The HTML code within this first PUT statement creates the headings for our variable layout table.

Afterwards, the "do until (last.index)" tells SAS to process and output the following HTML code for each index number that appears in our "Variables" sheet. You'll note that within this part of the HTML code, the fields from our "Variables" sheet are listed outside of the quotations that contain the actual HTML code. This is telling SAS to output the information listed in these fields from our "Variables" sheet. Running this code in conjunction with the code that created our codebook title produces the following output:



SESUG CODEBOOK

Variable Name	Type	Variable Label	Index Number	Variable Length	Column Start	Column End
ID	Num	Respondent ID Number	1	5	1	4
RACE	Num	Respondent Race/Ethnicity	2	2	5	6
SEX	Char	Respondent Sex	3	2	7	8
BIRTHMONTH	Num	Respondent Birth Month	4	2	9	10
BIRTHYEAR	Num	Respondent birth year	5	4	11	14

Figure 5. Codebook Title and Variable Layout Listing

VARIABLE ENTRIES

Once the codebook title and variable layout listing has been created, the individual variable entries (the main part of the codebook) is created. This is achieved with the same methodology that was shown above, but now uses our combined file that merged the "Variables" sheet and the "Variable Codes and Labels" sheet. The following code creates each variable entry from our combined file:

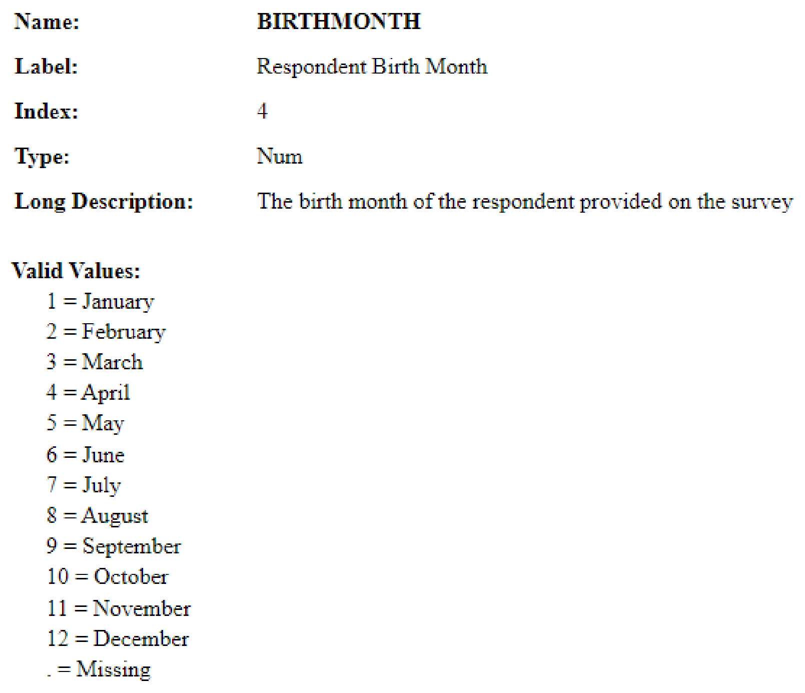
```
proc sort data=combined; by varname sequence; run;

data combined;
set combined;
Valid_values=strip(Code)||" = "||Label;
run;

data _NULL_;
set combined;
by varname sequence;
file ' C:/Users/wzsmith/SESUG 2023/SESUG_CODEBOOK.html' mod;
if first.varname then do;
      put '<hr size="1" width="100%" noshade
style="color:#000000"/>';
      put '<br><table id=' Index 'align=left cellspacing=2
style="width:35%; line-height:1.7; line font_size:1.0em;
margin:10px">
               <tr>
                <th scope="row" nowrap align=left> Name:</th>
                <td align=left><b>' Varname '</b></td>
                </tr>
                <tr>
                <th scope="row" nowrap align=left> Label:</th>
                <td align=left>' VarLabel '</td>
                </tr>
                <tr>
               <th scope="row" nowrap valign=top align=left>
Index:</th>
                <td align=left>' Index '</td>
                </tr>
                <tr>
                <th scope="row" nowrap valign=top align=left>
Type:</th>
                <td align=left>' VarType '</td>
                </tr>
                <tr>
                <th scope="row" nowrap valign=top align=left>Long
Description:</th>
                <td align=left>' Description '</td>
                </tr>
               </table>';
     put '<br><br><table align=left cellspacing=0
style="width:80%; line-height:1.2; line font_size:1.0em;
margin:10px">
```

```
                <tr><td style="text-align:left"><strong> Valid
Values: </strong></td>';
end;
put '<tr><td style="text-align:left">
&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;' Valid_values '</td>';
run;
```

First, the SAS program sorts our combined file by "Varname" and "Sequence". Then, a DATA step concatenates the codes and codes labels into a "Valid_Values" variable. The "Valid_Values" variable is what will appear in our codebook, along with the other variable metadata information.

From there, the same methodology that was presented when creating the variable layout table is utilized again. Because the combined file has each variable entry repeated based on the number of value labels the variable has (as shown in Figure 3), we want the variable metadata to be processed and output only once. This is achieved from the "if first.varname then do" statement within our DATA step. Within this DO loop, a solid line is created in the first PUT statement to visually separate each variable entry, and our second PUT statement then outputs the variable metadata information for each variable in our combined file.

Once our variable metadata information has been processed and output, we will end this DO loop and output every "Valid_Value" that exists for each variable we want included in the codebook that exists in our combined file. This is achieved from the initial BY statement that was used in the beginning of the DATA step, which will process and output each "Valid_Value" in the combined file by each "Varname" and "Sequence" combination within the file. The output for a single variable entry using this code is as follows (and will appear sequentially for each variable in the file):



Name:	**BIRTHMONTH**
Label:	Respondent Birth Month
Index:	4
Type:	Num
Long Description:	The birth month of the respondent provided on the survey

Valid Values:
1 = January
2 = February
3 = March
4 = April
5 = May
6 = June
7 = July
8 = August
9 = September
10 = October
11 = November
12 = December
. = Missing

Figure 6. Variable Entry Example

CONCLUSION

As shown above, SAS users can fully integrate HTML code into their programs to output seamless HTML codebooks that provide data users with an easier way to navigate variable metadata. This paper is limited in its scope so that its contents are accessible and easily digestible for novice SAS users, though does require a basic understanding of how to read and write HTML code. Advanced HTML methods can be used to further modify the code provided in this paper for more aesthetically pleasing output. Please refer to Appendix A for the full code shown piecemeal throughout this paper.

APPENDIX A

```
Proc import datafile C:/Users/wzsmith/SESUG 
2023/SESUG_Codebook_Spreadsheet.xlsx"
      out=variables
      dbms=xlsx
      replace;
     sheet='Variables';
      run;

Proc import datafile C:/Users/wzsmith/SESUG 
2023/SESUG_Codebook_Spreadsheet.xlsx"
      out=variable_values
      dbms=xlsx
      replace;
      sheet='Variable Codes and Labels';
      run;

proc sort data=variables; by varname; run;
proc sort data=variable_values; by varname; run;

data combined;
merge variables variable_values;
by varname;
drop i;
run;

data _NULL_;
file ' C:/Users/wzsmith/SESUG 2023/SESUG_CODEBOOK.html';
put '<p align="left" style="font-size:1.6em">
      <br>
      <strong> SESUG CODEBOOK </strong>
        <br>
      </p>'
;
put '<hr size="1" width="100%" noshade style="color:#000000"/>';
run;

proc sort data=variables; by index; run;

data _NULL_;
file 'C:/Users/wzsmith/SESUG 2023/SESUG_CODEBOOK.html' mod;
set variables;
by index;
```

```
if index=1 then do;
put '<table align=left cellspacing=8 cellpadding=1 style="width:35%;
font_size:1.0em">
     <tr>
      <td><b>Variable Name</b></td>
      <td><b>Type</b></td>
      <td><b>Variable Label</b></td>
      <td><b>Index Number</b></td>
      <td><b>Variable Length</b></td>
      <td><b>Column Start</b></td>
      <td><b>Column End</b></td>
      </tr>';
end;
do until (last.index);
put '<tr>
     <td align=left><a href="#' Index ' ">' Varname '</a></td>
      <td align=left>' VarType '</td>
      <td align=left nowrap>' VarLabel'</td>
      <td align=center>' Index '</td>
      <td align=center>' Varlength '</td>
      <td align=center>' Colstart '</td>
      <td align=center>' Colend '</td>
     </tr>';
end;
run;

proc sort data=combined; by varname sequence; run;

data combined;
set combined;
Valid_values=strip(Code)||" = "||Label;
run;

data _NULL_;
set combined;
by varname sequence;
file 'C:/wzsmith/SESUG 2023/SESUG_CODEBOOK.html' mod;
if first.varname then do;
     put '<hr size="1" width="100%" noshade style="color:#000000"/>';
     put '<br><table id=' Index 'align=left cellspacing=2
style="width:35%; line-height:1.7; line font_size:1.0em; margin:10px">
              <tr>
               <th scope="row" nowrap align=left> Name:</th>
               <td align=left><b>' Varname '</b></td>
               </tr>
               <tr>
               <th scope="row" nowrap align=left> Label:</th>
               <td align=left>' VarLabel '</td>
               </tr>
               <tr>
          <th scope="row" nowrap valign=top align=left> Index:</th>
               <td align=left>' Index '</td>
               </tr>
```

```
                <tr>
           <th scope="row" nowrap valign=top align=left> Type:</th>
                <td align=left>' VarType '</td>
                </tr>
                <tr>
           <th scope="row" nowrap valign=top align=left>Long 
Description:</th>
                <td align=left>' Description '</td>
                </tr>
               </table>';
     put '<br><br><table align=left cellspacing=0 style="width:80%;
line-height:1.2; line font_size:1.0em; margin:10px">
                <tr><td style="text-align:left"><strong> Valid Values:
</strong></td>';
end;
put '<tr><td style="text-align:left">
&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;' Valid_values '</td>';
run;
```

CONTACT INFORMATION

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

William Zachary Smith
RTI International
919-541-6987
wzsmith@rti.org