ABSTRACT

RTI has developed a suite of data tools to help keep data preparation on schedule and within budget. This toolkit provides a remarkably easy, low-cost way to create codebooks, master lists of SAS data sets for a project, reports of variables needing special investigation, and data crosswalks showing the relationship of variables across datasets. Traditionally, these documents are produced at the end of a project with a great deal of programming or are manually produced, but our tools allow the programmer to seamlessly create these documents at any time during the data preparation task. We have found that producing these documents early in data collection improves data quality and communication between the data collection team and client. This toolkit provides a comprehensive way to document and review not only datasets you create, but also datasets that you receive, especially if they do not have good documentation. Our paper illustrates how to run these macros and provides tips to add embellishments or tailor the documents to your project needs.

INTRODUCTION

Survey research organizations are often required to produce documentation to accompany their datasets. The documentation provides a roadmap for using the data and is as important as the data itself.

The Toolkit that follows describes a set of SAS macros that a programmer can use to easily generate codebooks, master lists of SAS data sets and crosswalks. These resulting documents display the relationship of variables across datasets and aid in discovering data relationships within datasets, problematic or otherwise. The documents can not only be used to review generated datasets but also review received datasets. They are especially helpful if the received datasets are not already well documented.

The core macro (Terry and Chantala, 2010) contained within the Toolkit was introduced in 2010 and has been widely accepted by various research centers and departments at the University of North Carolina, as well as by several international organizations. The current version has added new macros for speeding data preparation by identifying unexpected duplicate observations, creating crosswalks and harmonizing combined data sets.

The following sections include how-to descriptions on creating the codebooks, crosswalks and master lists of SAS data sets; on finding duplicate observations; and on harmonizing datasets with just a few key strokes.

Using the toolkit will increase efficiency in cleaning data, enhance communicating with clients about the status of data during data collection, and reduce the time needed to acquire solid data management skills for new programmers. Information on obtaining the toolkit is provided at the end of this document.

GETTING STARTED: TIPS FOR PREPARING THE SAS DATA SET

The macros in the toolkit use information stored with the variables and SAS data sets to create codebooks, crosswalks, and master data set lists. You can create your SAS data set with this information by adding:

- Labels to all variables
- User defined formats assigned to all categorical variables.
- User defined formats defining the valid range assigned to numeric variables.
- SAS date format assigned to all SAS date variables.
- A label assigned to the SAS data set.
This section provides instruction on adding this information to a SAS data set.

**PREPARING YOUR SAS DATA SET**

SAS provides statements to easily add informative labels to variables, their values and data sets. A brief description of how to create these labels is provided below, followed by a SAS program that creates a data set ready for use with the toolkit. Additional information is available in the on-line SAS documentation (http://support.sas.com/).

**VARIABLE LABELS**

The SAS label statement is used to assign a text description to each variable in the data set:

```sas
label variable = "Description";
```

The description can be up to 256 characters in length and should be enclosed in apostrophes (') or quotes ("").

**FORMATS**

Formats are created with the value statement in PROC FORMAT and permanently associated with one of more variables using the format statement in a DATA step.

**User defined formats**

Proc format can be used to create informative labels assigned to a value or range of values for a variable. You can assign text description to each value or range of values for a variable by using the value statement. The value statement has the form:

```sas
value <$fmt_name format-option(s)> value-range-set(s);
```

The name of the format (`fmt_name`) can be 32 characters for numeric variables. For character variables, the first character for name of the format must be a $ (`$fmt_name`) and up to 31 characters in length.

The `format-option(s)` include arguments to specify default, minimum or maximum length of a format and a fuzz factor for matching values to a range. See the SAS documentation for more information.

The `value-range-set(s)` allow you to assign a text description (enclosed in " or ") up to 32,767 characters to a value or set of values.

Below are a few example formats that can be useful for creating a codebook.

```sas
proc format;
value $anytext ' '='Missing (blank)
  ' '.'='Period (.)'
  other='Data present';
value $showall default = 40 ' '='Missing (blank)
  99='Not answered'
value month 1-12 = "Valid range"
  -99='Not answered'
run;
```

The `$anytext` format can be assigned to any character variable that does not have meaningful categories. This will allow the data will be summarized in only three categories (Missing (blank), Period (.), and Data Present). This is a useful format for open-ended text or names that you prefer to have appear summarized as one category in the codebook.

The `$showall` format is useful to assign to character variables if you prefer that every unique value of the variable appears in the codebook. The default = 40 in the value specification causes a maximum of 40
characters to be displayed. If any values have more than 40 characters they will be truncated, so be sure
the number specified as the default length is large enough to accommodate values for this variable.

The month format is an example of creating a format defining the valid range for a variable. This type of
format is useful for numeric variables that do not have meaningful categories.

The toolkit macros require that the descriptions assigned in the value-range-sets be unique. For
example, the following format will cause problems in the output of the codebook macro:

```sas
proc format; *Do not use this format;
value status 2720 = "Web Survey started"
               2799 = "Complete"
               2820 = "Phone survey contact"
               2899 = "Complete";
```

This format has the same description assigned to values 2799 and 2899. Below are alternate methods of
writing the format that will work well for the toolkit macros. Use the following format to have one category
for "complete":

```sas
proc format; *Use this format;
value status 2720 = "Web Survey started"
               2820 = "Phone survey contact"
               2799, 2899='Complete';
```

To keep the 2799 and 2899 as separate categories, add information to the value labels so they are
unique:

```sas
proc format;
value status 2720 = "Web Survey started"
               2799 = "Complete Web Survey"
               2820 = "Phone survey contact"
               2899 = 'Complete Phone Survey';
```

**SAS date formats**

The TK_codebook macro program will recognize certain date-time formats and display the range
(minimum to maximum) of dates in the codebook. The following date time formats are recognized:

**DATETIME:** Any format beginning with DATETIME will display the minimum and maximum "date part" of
the variable using an mmdy10.format.

**MMDDYY:** Any format beginning with MMDDYY will display the minimum and maximum dates with the
assigned date format.

**TIME:** Any format beginning with TIME will display the minimum and maximum time values with the
assigned time format.

**SAS DATA SET LABEL.**

SAS also allows you to assign and store a text string of up to 256 characters to a SAS data set. This
label is used by the TK_codebook macro to provide additional information about the data set. Below is an
example assigning a label to a data set.

```sas
data folder.HWS (label="Final data for Healthy Worker Study");
```

**EXAMPLE: CREATING A DATASET TO USE WITH THE TOOLKIT MACROS**
The SAS program shown below illustrates how to add this type of internal documentation to the variables and data set.

* CREATE FORMATS;
  proc format;
  value value $anytext ' '=''Missing (blank)' other='Data present';
  value $showall default = 40 ' '=''Missing (blank)' ;
  value race 1 = 'White'
    2 = 'Hispanic'
    3 = 'Black'
    4 = 'Asian';
  value sex 1='Male' 2='Female';
  value days 1-10='Valid Range' -99='Presented, not answered (web only)';
run;

* CREATE A DATA SET AND ASSIGN A LABEL TO BE SAVED WITH THE DATA SET;
  data folder.HWS (label="Final data for Healthy Worker Study");
  merge demography health;
  by caseid;

* ASSIGN LABELS TO VARIABLES IN DATA SET;
  label race = 'Race/Ethnicity';
  label int_date = 'Interview date';
  label sex = "Sex of participant";
  label sick_days = "Days absent due to illness";
  label comment = "Text description of illness";
  label clinic = "Name of Clinic";

* ASSIGN FORMAT TO VARIABLE IN DATA SET;
  format race race.;
  format int_date mmddyy10.;
  format sex sex.;
  format sick_days days.;
  format comment $anytext.;
  format clinic $showall.;
run;

You now have a SAS data set ready for use with the toolkit macro programs.

CREATING CODEBOOKS: MACRO TK_CODEBOOK

Codebooks are essential to understanding and using a dataset. They provide information on the structure of the data set, when it was created, and the meaning of every variable and their values. You can easily create a codebook with the TK_codebook.sas macro. It is simple to use, requiring only the following information:

- Titles for the codebook
- A SAS data set that has labels and formats assigned to the variables
- A Format library
- Full path name and type (PDF, XLS, XLSX, RTF, or HTML) of the codebook being created.

Optionally, you can also request the following reports listing potential problems identified in the data:
• Incomplete formats
• Out of Range values
• No variation in response values
• Variables not assigned a user-defined format
• Variables missing labels

The codebook macro uses information from the variable and value labels assigned to each variable and the label assigned to the data set.

SYNTAX

The TK_codebook.sas macro can be run with the following statement:

```sas
%macro TK_codebook(lib=libname_data, file1=data_set_name , fmtlib=libname_formats, var_order=order_keyword, cb_type=type_keyword, cb_file=codebook_name, cb_output=output_dataset, organization=data_set_organization, include_warn=warn_keyword, cb_size=size_keyword);
```

Required Variables:

LIB: Name of library for SAS data set (for FILE1 variable)
FILE1: Name of SAS data set used to create the codebook
FMTLIB: Name of format library
CB_TYPE: Type of codebook (XML, XLSX, XLS, PDF, RTF)
CB_FILE: Name of file for the codebook being created

Optional Variables:

CB_OUTPUT: Name of data set created by TK_codebook macro to create the FULL or BRIEF codebook. See Section on output data set for complete list of variables.
VAR_ORDER: Controls order variables are listed in codebook:
  • CUSTOM_ORDER (order from file named work.custom_order, see ORDERING VARIABLES IN CODEBOOK section of this paper),
  • INTERNAL (order of variables as stored in data set),
  • Omitted defaults to alphabetical as determined by PROC REPORT.
ORGANIZATION: Text indicating the organization of observations in the data set. For example, there might be one record per CASEID or one record per CASEID*WAVE.
INCLUDE_WARN: Flag to control printing of WARNING messages to reports in Codebook (in addition to LOG file)
  • YES prints warnings in file specified by CB_FILE (default)
  • OTHER warnings printed only in LOG file.
CB_SIZE: Flag to control size of codebook by limiting the number of columns of information on each variable in codebook.
  • BRIEF requests reduced size
  • FULL requests complete listing
See section on output dataset for information on variables in FULL and BRIEF codebooks.

OUTPUT DATA SET

All of the following variables listed in Table 1 are used in the CB_SIZE=FULL option and are included in the output data set even if CB_SIZE=BRIEF is requested.

Table1. Variables in output data set from macro TK_codebook.

<table>
<thead>
<tr>
<th>Included in CB_SIZE = BRIEF</th>
<th>VARIABLE</th>
<th>Type</th>
<th>Length</th>
<th>Label</th>
</tr>
</thead>
<tbody>
<tr>
<td>YES</td>
<td>NAME</td>
<td>Char</td>
<td>32</td>
<td>Variable name</td>
</tr>
<tr>
<td>YES (noprint)</td>
<td>ORDER</td>
<td>Num</td>
<td>8</td>
<td>Variable number order</td>
</tr>
<tr>
<td>YES</td>
<td>DESC</td>
<td>Char</td>
<td>37</td>
<td>Value label (Frequency Category text description)</td>
</tr>
<tr>
<td>No</td>
<td>FORMAT</td>
<td>Char</td>
<td>32</td>
<td>Variable format</td>
</tr>
<tr>
<td>YES</td>
<td>LABEL</td>
<td>Char</td>
<td>256</td>
<td>Variable label</td>
</tr>
<tr>
<td>YES</td>
<td>RANGE</td>
<td>Char</td>
<td>40</td>
<td>Values assigned to this category</td>
</tr>
<tr>
<td>YES</td>
<td>FREQUENCY</td>
<td>Num</td>
<td>8</td>
<td>Frequency</td>
</tr>
<tr>
<td>YES</td>
<td>PERCENT</td>
<td>Num</td>
<td>8</td>
<td>Percent</td>
</tr>
<tr>
<td>No</td>
<td>CUMFREQUENCY</td>
<td>Num</td>
<td>8</td>
<td>Cumulative frequency</td>
</tr>
<tr>
<td>No</td>
<td>CUMPERCENT</td>
<td>Num</td>
<td>8</td>
<td>Cumulative percent</td>
</tr>
<tr>
<td>YES (noprint)</td>
<td>CNT</td>
<td>Num</td>
<td>8</td>
<td>Order of value as it would appear in PROC FREQ output</td>
</tr>
<tr>
<td>No</td>
<td>MEAN_CHAR</td>
<td>Char</td>
<td>15</td>
<td>Mean of values for numeric variable</td>
</tr>
<tr>
<td>YES</td>
<td>TYPE_LENGTH</td>
<td>Char</td>
<td>9</td>
<td>Variable type and length</td>
</tr>
<tr>
<td>YES (noprint)</td>
<td>ORDER_FLAG</td>
<td>Num</td>
<td>8</td>
<td>Requested order of variables and values (RANGE/DESC) categories</td>
</tr>
<tr>
<td>Yes (Header)</td>
<td>SET_NAME</td>
<td>Char</td>
<td>22</td>
<td>Data set name</td>
</tr>
<tr>
<td>Yes (Header)</td>
<td>SET_LABEL</td>
<td>Char</td>
<td>34</td>
<td>Data set label</td>
</tr>
<tr>
<td>Yes (Header)</td>
<td>DATE_CREATED</td>
<td>Char</td>
<td>16</td>
<td>Date data set created</td>
</tr>
<tr>
<td>Yes (Header)</td>
<td>NUM_OBS</td>
<td>Char</td>
<td>3</td>
<td>Number of observations in data set</td>
</tr>
<tr>
<td>Yes (Header)</td>
<td>NUM_VAR</td>
<td>Char</td>
<td>2</td>
<td>Number of variables in data set</td>
</tr>
</tbody>
</table>

ORDERING VARIABLES IN CODEBOOK

The VAR_ORDER option allows you to have the variables in the codebook listed alphabetically (default) or as stored in the data set (VAR_ORDER=INTERNAL). If you prefer a different ordering scheme, create a simple two variable file called work.custom_order before you call the macro. The first variable is NAME, a 32 character field with your variable name in UPPER CASE. The second variable is ORDER, a numeric field with the order in which you want the variables to be printed. An example data step creating a work.custom_order data set is shown below.

data custom_order;
  length name $ 32;
  name = "T1";  ORDER = 1; OUTPUT;
name = "HHID09"; ORDER = 2; OUTPUT;
name = "LINE09"; ORDER = 3; OUTPUT;
name = "H1D"; ORDER = 4; OUTPUT;
run;

The TK_codebook macro will look for this data set if you specify VAR_ORDER=CUSTOM.

EXAMPLE: CREATE A BRIEF CODEBOOK WITH POTENTIAL PROBLEM REPORTS

This example illustrates how easy it is to create a codebook. We are requesting TK_codebook.sas macro create our codebook by doing the following:

- Use the data in /SAS_Analytics_2017/SAS_Data/StudyA_prelim.sas7bdat (file1=StudyA_prelim, lib=SAS_data).
- Use the formats stored in the file /SAS_Analytics_2017/SAS_Data/formats.sas7bdat to find the meaning of the values of the variables in the StudyA_prelim.sas7bdat data set (fmtlib=library).
- Save the codebook in a file named/Codebook/test_cb_ProbRpt.rtf so that we can open it with WORD (cb_type=RTF, cb_file=/Codebook/test_cb_ProbRpt.rtf).
- Order the variables listed in the codebook with the same order as they are stored in StudyA_prelim.sas7bdat (var_order=internal).
- Limit the columns in the codebook to Variable Name, Label, Type, Values, Value labels (Frequency Category), Frequency, Percent (cb_size=BRIEF).
- Include information in the header on the organization of the data set (organization = One record per CASEID).
- Print the Potential Problem reports (include_warn=YES).

Below is the SAS code for this codebook.

```sas
title j=l height=.20in 'Master Codebook for Study A Preliminary Data';
libname library '/SAS_Analytics_2017/SAS_Data';
libname SAS_data '/SAS_Analytics_2017/SAS_Data';
%include 'TK_codebook.sas';
%TK_codebook(lib=SAS_data,
  file1=StudyA_prelim,
  fmtlib=library,
  cb_type=RTF,
  cb_file=/Codebook/test_cb_ProbRpt.rtf,
  var_order=internal,
  cb_size=BRIEF,
  organization = One record per CASEID,
  include_warn=YES);
run;
```

The codebook is shown below. The names of the format assigned to each variable is not included in this report but would be included in the FULL report. The variables INT_DATE and BIRTHDATE had the SAS format mmddyy10 assigned and the TK_codebook.sas macro used this information to display the range of values as actual dates rather than numeric values. No format was assigned to CASEID and RACE, so the TK_codebook.sas macro reported the numeric range and information on the amount of missing values. No format was given for the character variable CITY so all observations were lumped into one category called "Blank, Text, or Value supplied". None of the user defined formats had a category defined for missing, so the TK_codebook.sas added a category "SAS missing(.) to the description listed in the "Frequency Category."
### Master Codebook for Study A Preliminary Data

**Data Set:** StudyA_prelim.sas7bdat  
**Label:** Final data for fictional Study A  
**Date Created:** 28JUL17:11:55:25  
**Number of Observations:** 501  
**Number of Variables:** 16  
**Organization of Data Set:** One record per CASEID

<table>
<thead>
<tr>
<th>Variable Name</th>
<th>Label</th>
<th>Type</th>
<th>Values</th>
<th>Frequency Category</th>
<th>Frequency</th>
<th>Percent</th>
</tr>
</thead>
<tbody>
<tr>
<td>CITY</td>
<td>Fictional city where participant lives</td>
<td>Char 10</td>
<td><strong>OTHER</strong></td>
<td>Blank, Text, or Value supplied</td>
<td>501</td>
<td>100.00</td>
</tr>
<tr>
<td>MODE</td>
<td>Mode of data collection</td>
<td>Char 4</td>
<td>CAI</td>
<td>Computer Assisted Interview</td>
<td>139</td>
<td>27.74</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>CAPI</td>
<td>Computer Assisted Personal Interview</td>
<td>32</td>
<td>6.39</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>PAPI</td>
<td>Paper and Pencil Personal Interview</td>
<td>33</td>
<td>6.59</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>TDE</td>
<td>Touchtone Data Entry</td>
<td>126</td>
<td>25.15</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>VOIP</td>
<td>Web Cam Interview</td>
<td>33</td>
<td>6.59</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>WEB</td>
<td>Web Interview</td>
<td>138</td>
<td>27.54</td>
</tr>
<tr>
<td>CASEID</td>
<td>Unique identifier for participant</td>
<td>Num 8</td>
<td>10000 to 10500</td>
<td>Range</td>
<td>501</td>
<td>100.00</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>SAS missing (.)</td>
<td>0</td>
<td>0.00</td>
</tr>
<tr>
<td>INT_DATE</td>
<td>Interview date</td>
<td>Num 8</td>
<td>03/21/2017 to 04/20/2017</td>
<td>Range</td>
<td>501</td>
<td>100.00</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>SAS missing (.)</td>
<td>0</td>
<td>0.00</td>
</tr>
<tr>
<td>BIRTHDATE</td>
<td>Date of birth</td>
<td>Num 8</td>
<td>05/28/1967 to 03/30/1992</td>
<td>Range</td>
<td>456</td>
<td>91.02</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>SAS missing (.)</td>
<td>45</td>
<td>8.98</td>
</tr>
<tr>
<td>AGE</td>
<td>Age at interview date</td>
<td>Num 8</td>
<td>25 to 49</td>
<td>Range</td>
<td>456</td>
<td>91.02</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>SAS missing (.)</td>
<td>45</td>
<td>8.98</td>
</tr>
<tr>
<td>SEX</td>
<td>Gender</td>
<td>Num 8</td>
<td>1</td>
<td>Male</td>
<td>249</td>
<td>49.70</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>2</td>
<td>Female</td>
<td>252</td>
<td>50.30</td>
</tr>
<tr>
<td>RACE</td>
<td>Race/Ethnicity</td>
<td>Num 8</td>
<td>SAS missing (.)</td>
<td></td>
<td>30</td>
<td>5.99</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>1</td>
<td>White</td>
<td>270</td>
<td>53.89</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>2</td>
<td>Hispanic</td>
<td>47</td>
<td>9.38</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>3</td>
<td>Black</td>
<td>130</td>
<td>25.95</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>4</td>
<td>Asian</td>
<td>24</td>
<td>4.79</td>
</tr>
<tr>
<td>HEALTH</td>
<td>How is your health</td>
<td>Num 8</td>
<td>1</td>
<td>Excellent</td>
<td>127</td>
<td>25.35</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>2</td>
<td>Very Good</td>
<td>163</td>
<td>32.53</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>3</td>
<td>Good</td>
<td>129</td>
<td>25.75</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>4</td>
<td>Fair</td>
<td>48</td>
<td>9.58</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>5</td>
<td>Poor</td>
<td>12</td>
<td>2.40</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>6</td>
<td></td>
<td>22</td>
<td>4.39</td>
</tr>
<tr>
<td>WEIGHT</td>
<td>How would you describe your weight</td>
<td>Num 8</td>
<td>0</td>
<td>0</td>
<td>17</td>
<td>3.39</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>1</td>
<td>Very underweight</td>
<td>71</td>
<td>14.17</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>2</td>
<td>Somewhat underweight</td>
<td>47</td>
<td>9.38</td>
</tr>
</tbody>
</table>
### Master Codebook for Study A Preliminary Data

**Data Set:** StudyA_prelim.sas7bdat  
**Label:** Final data for fictional Study A  
**Date Created:** 28JUL17:11:55:25  
**Number of Observations:** 501  
**Number of Variables:** 16  
**Organization of Data Set:** One record per CASEID

<table>
<thead>
<tr>
<th>Variable Name</th>
<th>Label</th>
<th>Type</th>
<th>Values</th>
<th>Frequency Category</th>
<th>Frequency</th>
<th>Percent</th>
</tr>
</thead>
<tbody>
<tr>
<td>HEALTH</td>
<td></td>
<td></td>
<td>3 Healthy weight</td>
<td></td>
<td>239</td>
<td>47.70</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>4 Somewhat overweight</td>
<td></td>
<td>97</td>
<td>19.36</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>5 Very Overweight</td>
<td></td>
<td>30</td>
<td>5.99</td>
</tr>
<tr>
<td>CHG_WEIGHT</td>
<td>What are you trying to do about your weight?</td>
<td>Num 8</td>
<td>SAS missing (.)</td>
<td>58</td>
<td>11.58</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>1 Trying to lose weight</td>
<td></td>
<td>251</td>
<td>50.10</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>2 Trying to stay same weight</td>
<td></td>
<td>87</td>
<td>17.37</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>3 Trying to gain weight</td>
<td></td>
<td>52</td>
<td>10.38</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>4 Nothing</td>
<td></td>
<td>53</td>
<td>10.58</td>
</tr>
<tr>
<td>EXER_DAYS</td>
<td>In the Past 30 days, how many days did you exercise at least 30 minutes?</td>
<td>Num 8</td>
<td>0 to 34 Range</td>
<td>501</td>
<td>100.00</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>SAS missing (.)</td>
<td>0</td>
<td>0.00</td>
<td></td>
</tr>
<tr>
<td>PERIOD</td>
<td>period</td>
<td>Num 8</td>
<td>Range</td>
<td>0</td>
<td>0.00</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>SAS missing (.)</td>
<td>501</td>
<td>100.00</td>
<td></td>
</tr>
<tr>
<td>SMOKE</td>
<td>Do you currently smoke?</td>
<td>Num 8</td>
<td>1 to 1 Range</td>
<td>400</td>
<td>79.84</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>SAS missing (.)</td>
<td>101</td>
<td>20.16</td>
<td></td>
</tr>
<tr>
<td>IOSTART</td>
<td></td>
<td>Num 8</td>
<td>Range</td>
<td>0</td>
<td>0.00</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>SAS missing (.)</td>
<td>501</td>
<td>100.00</td>
<td></td>
</tr>
</tbody>
</table>

The potential problem reports follow the codebook.

**Incomplete Format Report**

This report lists formats that were assigned to variables having values that were not defined by the FORMAT. The name of the FORMAT is listed in the first column of the report. Check to see if this value was omitted from the FORMAT and make corrections if needed.

<table>
<thead>
<tr>
<th>FORMAT</th>
<th>Value not in Format</th>
<th>Number of Variables with Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>HEALTH</td>
<td>6</td>
<td>1</td>
</tr>
<tr>
<td>WEIGHT</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

**Out of Range Value Report**

This reports lists the names of variables having values not included in the assigned user-defined FORMAT. Investigate, correct, recode or document the out of range value.
No Variation in Response Report

Variables with no variation in response values need to be investigated in sample surveys. These variables might be system variables that are expected to have only one value, but research variables with no variation in response are uncommon. This report lists all numeric variables that have only one value (other than missing). Below we see that the only response for the SMOKE variable is a value of 1 and 79.84% of the participants gave this answer.

Variables with no assigned user format.

The next two tables list the Character and Numeric variables that do not have a user-defined format assigned. The report for character variables appears below.

For character variables that do not have a format assigned the TK_codebook.sas macro will lump all observations into one category called "Blank, Text, or Value supplied". Consider assigning a format that will separate out the "Blank" responses from the responses that contain actual information by using a format such as:

```sas
proc format;
  value $anytext '=''Missing (blank)' other='Text or value supplied';
run;
```

The report for numeric variables without an assigned user-defined format appears below. The TK_codebook.sas macro will report frequency and percent for the range of non-missing values for each of these variables. The variables BIRTHDATE and INT_DATE are date variables and have been assigned the SAS format mmddyy10 rather than a user-defined format. Since TK_codebook.sas will use this format to display the range of dates in the codebook you do not need to assign a different format.
POTENTIAL PROBLEM: NUMERIC VARIABLES NOT ASSIGNED USER FORMAT

Codebook uses categories 'Range', 'SAS Missing (.)'
Try format with categories 'Valid Range', 'Missing'
Values outside of Valid Range will be identified in Out of Range Report.

<table>
<thead>
<tr>
<th>Variable Name</th>
<th>type</th>
<th>Variable Length</th>
<th>Variable Label</th>
</tr>
</thead>
<tbody>
<tr>
<td>AGE</td>
<td>Num</td>
<td>8</td>
<td>Age at interview date</td>
</tr>
<tr>
<td>BIRTHDATE</td>
<td>Num</td>
<td>8</td>
<td>Date of birth</td>
</tr>
<tr>
<td>CASEID</td>
<td>Num</td>
<td>8</td>
<td>Unique identifier for participant</td>
</tr>
<tr>
<td>EXER_DAYS</td>
<td>Num</td>
<td>8</td>
<td>In the Past 30 days, how many days did you exercise at least 30 minutes?</td>
</tr>
<tr>
<td>INT_DATE</td>
<td>Num</td>
<td>8</td>
<td>Interview date</td>
</tr>
<tr>
<td>IOSTART</td>
<td>Num</td>
<td>8</td>
<td>period</td>
</tr>
<tr>
<td>PERIOD</td>
<td>Num</td>
<td>8</td>
<td>period</td>
</tr>
<tr>
<td>SMOKE</td>
<td>Num</td>
<td>8</td>
<td>Do you currently smoke?</td>
</tr>
</tbody>
</table>

For numeric variables that represent amounts rather than categories it is useful to assign a format to define the valid range. For example the EXER_DAYS variable should always be 0 to 30 days. If special codes are assigned to document the reason missing, then the following format would be useful to assign so that the codebook reports the actual number of valid responses, any out of range values, and the reason for missing. Out of range values to be identified and reported in the “Out of Range Value” report.

```
proc format;
  value DAYS -9 = "Don't Know" -8="Not Asked" -7="Refused" 0:30="Valid Response";
run;
```

This table includes some variables such as SMOKE that have values representing categories and need a format assigned:

```
proc format;
  value smoke 0='No' 1='Yes';
run;
```

Undefined Variable Label Report

The final report printed lists the names of the variables that are missing a label or have a label that matches the name of the variable.

```
<table>
<thead>
<tr>
<th>Variable Name</th>
<th>Label</th>
<th>Potential Problem</th>
</tr>
</thead>
<tbody>
<tr>
<td>IOSTART</td>
<td></td>
<td>Missing variable label</td>
</tr>
<tr>
<td>PERIOD</td>
<td>period</td>
<td>Label identical to variable name</td>
</tr>
</tbody>
</table>
```

CREATING A MASTER LIST OF DATASETS: MACRO TK_DATASET_LIST.SAS

This macro provides an easy way to create a description of all SAS data sets in a folder. It has been particularly useful for the following tasks:

- Document a group of datasets that are ready for delivery
- Report on data sets that were created by the an overnight SAS job
• Explore a group of new data sets to make sure they have the expected structure
• Provide documentation on project datasets for a new programmer.

For each dataset in the folder, TK_dataset_list.sas will capture information on the name, label, creation date, number of observations and number of variables.

**SYNTAX**

The TK_dataset_list macro can be run with the following statement:

```sas
%dataset_list(libref=libref_folder);
```

**Required Argument**

`libref_folder`: The libref (the name associated with the folder pathname by the LIBNAME)

**EXAMPLE: CREATE A MASTER LIST OF DATASETS**

The following SAS code uses the LIBNAME statement to associate a folder named 
/SAS_Analytics_2017/SAS_Data to the libref SAS_data, then runs the TK_dataset_list macro to create a table showing attributes of all the SAS datasets in that folder.

```sas
%LET DataFolder= /SAS_Analytics_2017/SAS_Data;
LIBNAME SAS_data "&DataFolder.";
%TK_dataset_list(libref=SAS_data);
```

Below is the report created by the TK_dataset_list macro. All six data sets in folder /SAS_Analytics_2017/SAS_Data have been analyzed to extract information on size and creation date and tabulated in the table.

<table>
<thead>
<tr>
<th>SAS Data set name (*.sas7bdat)</th>
<th>Data Set Label</th>
<th>Create Date</th>
<th>Observations in Data Set</th>
<th>Number of Variables</th>
</tr>
</thead>
<tbody>
<tr>
<td>DEMOGRAPHY</td>
<td>Final Demography Data for Study A</td>
<td>26JUL17:10:40:55</td>
<td>501</td>
<td>8</td>
</tr>
<tr>
<td>DEMOGRAPHY_A1</td>
<td>Fictional Data from Study A, Collection 1</td>
<td>25JUL17:20:35:45</td>
<td>403</td>
<td>8</td>
</tr>
<tr>
<td>DEMOGRAPHY_A2</td>
<td>Fictional Data from Study A, Collection 2</td>
<td>25JUL17:20:35:46</td>
<td>98</td>
<td>8</td>
</tr>
<tr>
<td>HEALTH</td>
<td>Health data for Study A</td>
<td>27JUL17:18:05:22</td>
<td>501</td>
<td>5</td>
</tr>
<tr>
<td>STUDYA</td>
<td>Final data for fictional Study A</td>
<td>27JUL17:19:11:30</td>
<td>501</td>
<td>12</td>
</tr>
<tr>
<td>STUDYA_PRELIM</td>
<td>Final data for fictional Study A</td>
<td>28JUL17:11:55:25</td>
<td>501</td>
<td>16</td>
</tr>
</tbody>
</table>

**IDENTIFYING DUPLICATE OBSERVATIONS: MACRO TK_FIND_DUPS.SAS**

An important step in data preparation is to ensure variables that uniquely identify an observation occur only on one observation. The tk_find_dups macro will examine a data set and report on any observations that have the same value of one variable or a list of key variables separated by asterisks. An optional output data set is available containing the values of the key variables that occur on duplicate observations. Any formats assigned to the variables are ignored in the examination.

**SYNTAX**

The tk_find_dups macro can be run with the following statement:

```sas
%tk_find_dups(dataset=data_set_name, one_rec_per=variable_list,
```
dup_output=output_set);

**Required Arguments**

- **data_set**: Name of SAS data set to be examined.
- **one_rec_per**: Variable or list of variables separated by asterisks defining unique identifiers of each observation.
- **dup_output**: Name of output data set with duplicate values of variables (if found) listed in one_rec_per keyword

**Optional Arguments**

- **dup_output**: Output data set for the variables specified in one_rec_per have that values occurring on more than one observation in the data set.

**CONCEPTS**

The `tk_find_dups` macro creates a macro variable named `&numobs` and data sets named `_dup_check_` and `_the_dups_`. These will overwrite existing macro variable or data sets having these names. Any formats assigned to the variables are omitted from the examination.

**EXAMPLE: FIND DUPLICATE OBSERVATIONS**

Below is a data set that has a few duplicated observations on the variables:

```sas
data STUDY;
input CASEID WAVE LOCATION;
label CASEID = 'Unique identifier for participant';
label WAVE = 'Wave of data collection';
label LOCATION = 'Location of interview';
datalines;
100 1 4
100 2 4
100 3 4
200 1 6
200 1 6
200 1 6
200 2 3
300 1 5
300 2 5
300 3 5
400 1 6
400 2 6
400 2 6
;
```

The following statement runs macro `tk_find_dups` to examine dataset `STUDY` for any occurrences of multiple observations with identical values of CASEID*WAVE. An output data set named `STUDY_DUPS` is requested.

```sas
%tk_find_dups(dataset=STUDY, one_rec_per=CASEID*WAVE, dup_output=STUDY_DUPS);
```
Below are the results of the tk_find_dups macro:

Data set being examined: STUDY (N=13)
Identification variables: CASEID*WAVE
There should be only one record for every unique value of CASEID*WAVE

<table>
<thead>
<tr>
<th>COPIES</th>
<th>Frequency</th>
<th>Percent</th>
<th>Cumulative Frequency</th>
<th>Cumulative Percent</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>8</td>
<td>80.00</td>
<td>8</td>
<td>80.00</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>10.00</td>
<td>9</td>
<td>90.00</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>10.00</td>
<td>10</td>
<td>100.00</td>
</tr>
</tbody>
</table>

There are 10 unique values of CASEID*WAVE in the 13 observations in dataset STUDY. The table shows there are 8 observations (1*8) that have unique values of CASEID*WAVE, two observations (2*1) that have the same value of CASEID*WAVE, and three observations (3*1) that have the same value of CASEID*WAVE.

The next table printed by tk_find_dups.sas show the values of CASEID*WAVE that occur multiple times in the data set:

Values of CASEID*WAVE occurring on more than one record in data set STUDY
COPIES = Number of observations with identical values of CASEID*WAVE

<table>
<thead>
<tr>
<th>CASEID</th>
<th>WAVE</th>
<th>COPIES</th>
<th>Frequency</th>
</tr>
</thead>
<tbody>
<tr>
<td>200</td>
<td>1</td>
<td>3</td>
<td>1</td>
</tr>
<tr>
<td>400</td>
<td>2</td>
<td>2</td>
<td>1</td>
</tr>
</tbody>
</table>

CASEID = 200 has three observations with values of WAVE=1 while CASEID=400 has two observations with values of WAVE=2. You can request this table as an output data set to make it easy to merge with the study data and print the records for the duplicate cases.

HARMONIZING DATASETS: MACRO TK_HARMONY.SAS

Often data for a study comes from many different sources and files. One example is a survey that is collected with different modes of collection such as web versus teleform (paper and pencil) interview. Although the data sets contain the same questions, the data sets created may not always be created with the same structure. The task to combine data from all sources into one set by concatenating the data sets must have identical data attributes for variables with the same name. The TK_harmony macro will compare the structure of two data sets and report on differences found in data type and label of the variables having the same variable name, and identify variables that are unique to each set. This information is needed to harmonize the data when the data sets are combined into one set.

SYNTAX

The macro TK_harmony can be run with the following statement:

```sas
%TK_harmony(set1 = data_set_name1, set1_id = set_abbreviation1, set2 = data_set_name2, set2_id = set_abbreviation2, out = output_set);
```
Required Arguments

set1: Name of first data set
set1_id: Abbreviation (maximum of 20 characters) of first data set name used in output report
set2: Name of second data set
set2_id: Abbreviation (maximum of 20 characters) of second data set name used in output report

Optional Arguments

out: Name of output data set

OUTPUT DATA SET

The contents of the optional output data set is listed below. Attributes of all variables from both data sets are included.

<table>
<thead>
<tr>
<th>Variable</th>
<th>Type</th>
<th>Length</th>
<th>Label</th>
</tr>
</thead>
<tbody>
<tr>
<td>harmony</td>
<td>Char</td>
<td>10</td>
<td>Harmony measure comparing type, length, and label of variable (Values: DIFF = different, SAME = All Match, SOLO = variable in only one file)</td>
</tr>
<tr>
<td>label1</td>
<td>Char</td>
<td>256</td>
<td>set1_id variable label</td>
</tr>
<tr>
<td>label2</td>
<td>Char</td>
<td>256</td>
<td>set2_id variable label</td>
</tr>
<tr>
<td>label_match</td>
<td>Char</td>
<td>3</td>
<td>Both have same variable label? (Values: Yes, No)</td>
</tr>
<tr>
<td>location</td>
<td>Char</td>
<td>20</td>
<td>File location of variable (Values: Both, set1_id, set2_id)</td>
</tr>
<tr>
<td>name</td>
<td>Char</td>
<td>32</td>
<td>Variable name</td>
</tr>
<tr>
<td>type_length1</td>
<td>Char</td>
<td>9</td>
<td>set1_id variable data type (Num, Char) and length</td>
</tr>
<tr>
<td>type_length2</td>
<td>Char</td>
<td>9</td>
<td>set2_id variable data type (Num, Char) and length</td>
</tr>
<tr>
<td>type_match</td>
<td>Char</td>
<td>3</td>
<td>Both have same data type? (Values: Yes, No)</td>
</tr>
</tbody>
</table>

EXAMPLE: HARMONIZING DATA SETS

The demographic data in this example were collected by both web and teleform. Before concatenating these files, we can use the TK_harmony macro to identify differences that might compromise data when combining the data sets. The following statements runs a comparison of two data sets (SAS_data.demography_a1 and SAS_data.demography_a2), assign nicknames of Web and Paper to these data sets, and obtains an output data set named test_harmony.

```
libname SAS_data "SAS_Analytics_2017/SAS_Data/";

%TK_harmony(set1= SAS_data.demography_a1, 
            set1_id=Web, 
            set2= SAS_data.demography_a2, 
            set2_id=Paper, 
            out=test_harmony);
```

The first table printed by TK_harmony shows a summary of the Harmony measures of the variables that have the same name in both data sets, and identifies variables that appear in only one data set.
The two data sets have 9 variables with unique names. Seven of these variables (Harmony measure equals DIFF or SAME) have identical names in both data sets while two of the variables (Harmony measure equals SOLO) are found in only one of the data sets.

The harmony measures of the variables show:

- 4 have the same data type and label (Harmony=SAME)
- 1 has the same label, but different data type (Harmony=DIFF)
- 2 have the same data type, but different label (Harmony=DIFF)
- 2 variables occur in only one of the data sets (Harmony=SOLO).

The next table printed by TK_harmony shows details of the variables with harmony measure equal DIFF or SOLO.
From this report we can tell the following differences need to be corrected when the two data sets are combined:

- AGE and SEX have different labels in the two data sets
- MODE has different data types (CHAR 3 vs CHAR 4) in the two data sets
- The Web data set has a variable CTIY which is likely to be a misspelling of the variable CITY.
- The data types for CITY and CTIY have different lengths in the two sets with a length of 10 characters in the “Web” data set and 5 characters in the “Paper” data set.

One you have this information, you are ready to make corrections to labels and data types before combining the data sets.

**CREATING CROSSWALKS: MACRO TK_XWALK.SAS**

A crosswalk is a table that maps the variables in one data set into the equivalent variables in one or more other data sets. Thus, a crosswalk shows the “union” (all variables in all sets) as well as the “intersection” (variables in common) of multiple data sets.

The TK_xwalk.sas macro considers variables in multiple data sets to be the same if they have identical names and provides additional information on differences in variable label, format and data type. It can be used to uncover the relationship between variables in multiple files.

**SYNTAX**

The macro TK_xwalk.sas can be run with the following statement:

```sas
%TK_xwalk( SetList = libref_dataset_list);
```

**Required Arguments**

SetList: List of any number of data sets (i.e. libref_a.dataset_a  libref_b.dataset_b libref_c.dataset_c etc.)

**EXAMPLE: CREATING A CROSSWALK**

This example will examine three sets of data and determine which variables exist in all sets, in one set, or a group of sets by creating a crosswalk. In the code below we are specifying three data sets to use to create the crosswalk. To run the TK_xwalk macro, provide a list of the data sets to be included in the crosswalk separated by blanks. Note that each data set is specified by the libref and data set name.

```sas
%TK_xwalk( SetList = SAS_data.studya SAS_data.demography SAS_data.health);
```

The output created by TK_xwalk.sas appears in the next table. The first column on the left contains the names of variables in all three data sets. The first row labelled “Total” indicates there is a total of 25 variables in the 3 sets, with 8 in the demography data set, 5 in the health data set, and 12 in the third data set. Some of these variables appear in multiple sets, so there is actually only 12 (number of rows) unique variable names in the three sets. An entry of 1 in any of the last three columns indicates that the variable is present in the data set. Note that the only variable present in all three data sets is CASEID.

<table>
<thead>
<tr>
<th>VARIABLE CROSSWALK</th>
<th>Total</th>
<th>Data Set</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Frequency</td>
<td>SAS_data.demography</td>
</tr>
<tr>
<td>Total</td>
<td>25</td>
<td>8</td>
</tr>
<tr>
<td>Variable Name</td>
<td>Variable Label</td>
<td>Type-Length</td>
</tr>
<tr>
<td>---------------</td>
<td>----------------</td>
<td>-------------</td>
</tr>
<tr>
<td>AGE</td>
<td>Age at interview date</td>
<td>NUM-8</td>
</tr>
<tr>
<td>BIRTHDATE</td>
<td>Date of birth</td>
<td>NUM-8 MMDDYY</td>
</tr>
<tr>
<td>CASEID</td>
<td>Unique identifier for participant</td>
<td>NUM-8</td>
</tr>
<tr>
<td>CHG_WEIGHT</td>
<td>What are you trying to do about your weight?</td>
<td>NUM-8 CHG_WT</td>
</tr>
<tr>
<td>CITY</td>
<td>Fictional city where participant lives</td>
<td>CHAR-10</td>
</tr>
<tr>
<td>EXER_DAYS</td>
<td>In the Past 30 days, how many days did you exercise at least 30 minutes?</td>
<td>NUM-8</td>
</tr>
<tr>
<td>HEALTH</td>
<td>How is your health</td>
<td>NUM-8 HEALTH</td>
</tr>
<tr>
<td>INT_DATE</td>
<td>Interview date</td>
<td>NUM-8 MMDDYY</td>
</tr>
<tr>
<td>MODE</td>
<td>Mode of data collection</td>
<td>CHAR-4 SMODE</td>
</tr>
<tr>
<td>RACE</td>
<td>Race/Ethnicity</td>
<td>NUM-8 RACE</td>
</tr>
<tr>
<td>SEX</td>
<td>Gender</td>
<td>NUM-8 SEX</td>
</tr>
<tr>
<td>WEIGHT</td>
<td>How would you describe your weight</td>
<td>NUM-8 WEIGHT</td>
</tr>
</tbody>
</table>

If the label, type or format differ for a variable that is in multiple sets differs, then all values will be displayed. For example the WEIGHT variable has a FORMAT assigned in the studya.sas7bdat data set, but not in the health.sas7bdat dataset. The same is also true for the CHG_WEIGHT and HEALTH variables. Different labels, data types, or formats assigned to variables can indicate that the variables have different meanings in the data sets. Further investigation will determine if these variables do represent the same concept.

**OBTAINING THE TOOLKIT**

The toolkit and documentation can be downloaded from github.com at:

https://github.com/dchantala/Data_Toolkit.git

**CONCLUSION**

Using the macros in the toolkit will help you create a roadmap to your data. You will be able to easily generate documents telling you:

- Where data files are located
- How the data files are related
- Structure of data files, meaning of study variables and their values
These macros improve communication between the researchers and programming staff, automates part of the data validation process, highlights potential problems and unexpected results, and are easily incorporated into routine data cleaning and preparation.

REFERENCES


ACKNOWLEDGMENTS

Jim Terry, applications programmer at the University of North Carolina at Chapel Hill, collaborated in developing the first version of the TK_codebook.sas macro which was released under the name of proc_codebook.sas.

CONTACT INFORMATION

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

    Kim Chantala
    RTI International
    dchantala@rti.org