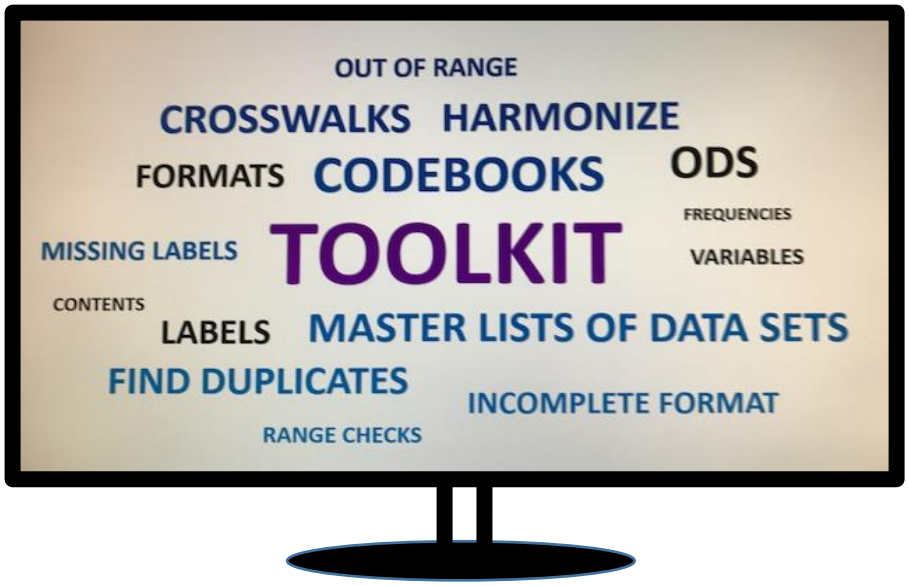
The Data Detective's ToolKit

Software Tools
for Data Management



RTI INTERNATIONAL

Author & Software Developer

Kim Chantala

Technical Advisor

Jean Robinson

Testers

Jean Robinson

Helen Smith

Documentation Specialist

Erin Friday

Acknowledgements: Jim Terry, applications programmer at UNC Chapel Hill, collaborated in developing the first version of the TK_codebook.sas macro. This was released under the name of proc_codebook.sas.

Contact Information: dchantala@rti.org

The Data Detective's Toolkit is available through Github.com:
https://github.com/dchantala/Data_Toolkit.git

Table of Contents

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 Data Detective's Toolkit that follows describes a software toolkit 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 the discoverability of 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 Data Detective's 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 Data Detective's 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.

An early version of this manual is provided in the SESUG 2017 Conference proceedings (SESUG Paper DM-77-2017, "A Software Toolkit for Data Management"). This manual provides updates and additional information to using the Toolkit.

The Data Detective's Toolkit is available through Github.com:
https://github.com/dchantala/Data_Toolkit.git

Getting Started: Tips for preparing the SAS data set

Overview

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 chapter 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:

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

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

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:

```
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.

```
proc format;
value $anytext ' '='Missing (blank)
       '.'='Period (.)'
       other='Data present';
value $showall default = 40 ' '='Missing (blank)' ;
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 indicates to display a maximum of 40 characters. 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 data tool macros require that the descriptions assigned in the *value-range-sets* be unique. For example, the format:

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

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

The categories of 2799 and 2899 would need to be recoded to the same value to use the description of ‘Complete’.

Below is an alternate methods of writing the format that will work well for the data tool macros. Use the following format to have one category for “complete”:

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

```
proc format;
value status 2720 = "Web Survey started"
    2799 = "Complete Web Survey"
    2820 = "Phone survey contact"
    2799, 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 mmddyy10. 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.

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

Example

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;
```

Creating codebooks: macro TK_codebook

Overview

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 TK_codebook macro uses information from the variable and value labels assigned to each variable and the label assigned to the data set. The chapter "Getting Started: Tips for preparing the SAS data set" will show you how to do this.

Syntax

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

%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+keywork*, 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 listed in codebook:

- CUSTOM_ORDER (order from file named work.custom_order, see *ORDERING VARIABLES IN CODEBOOK* section of this document),

- 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 are used in the CB_SIZE=FULL option and are included in the output data set even if CB_SIZE=BRIEF is requested.

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

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 1. 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).

```
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

Variable Name	Label	Type	Values	Frequency Category	Frequency	Percent
CITY	Fictional city where participant lives	Char 10	**OTHER**	Blank, Text, or Value supplied	501	100.00
MODE	Mode of data collection	Char 4	CAI	Computer Assisted Interview	139	27.74
			CAPI	Computer Assisted Personal Interview	32	6.39
			PAPI	Paper and Pencil Personal Interview	33	6.59
			TDE	Touchtone Data Entry	126	25.15
			VOIP	Web Cam Interview	33	6.59
			WEB	Web Interview	138	27.54
CASEID	Unique identifier for participant	Num 8	10000 to 10500	Range	501	100.00
				SAS missing (.)	0	0.00
INT_DATE	Interview date	Num 8	03/21/2017 to 04/20/2017	Range	501	100.00
				SAS missing (.)	0	0.00

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

Variable Name	Label	Type	Values	Frequency Category	Frequency	Percent
BIRTHDATE	Date of birth	Num 8	05/28/1967 to 03/30/1992	Range	456	91.02
				SAS missing (.)	45	8.98
AGE	Age at interview date	Num 8	25 to 49	Range	456	91.02
				SAS missing (.)	45	8.98
SEX	Gender	Num 8	1	Male	249	49.70
			2	Female	252	50.30
RACE	Race/Ethnicity	Num 8		SAS missing (.)	30	5.99
			1	White	270	53.89
			2	Hispanic	47	9.38
			3	Black	130	25.95
			4	Asian	24	4.79
HEALTH	How is your health	Num 8	1	Excellent	127	25.35
			2	Very Good	163	32.53
			3	Good	129	25.75
			4	Fair	48	9.58
			5	Poor	12	2.40
			6	6	22	4.39
WEIGHT	How would you describe your weight	Num 8	0	0	17	3.39
			1	Very underweight	71	14.17
			2	Somewhat underweight	47	9.38
			3	Healthy weight	239	47.70
			4	Somewhat overweight	97	19.36
			5	Very Overweight	30	5.99
CHG_WEIGHT	What are you trying to do about your weight?	Num 8		SAS missing (.)	58	11.58
			1	Trying to lose weight	251	50.10
			2	Trying to stay same weight	87	17.37
			3	Trying to gain weight	52	10.38
			4	Nothing	53	10.58
EXER_DAYS	In the Past 30 days, how many days did you exercise at least 30 minutes?	Num 8	0 to 34	Range	501	100.00
				SAS missing (.)	0	0.00
PERIOD	period	Num 8		Range	0	0.00

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

Variable Name	Label	Type	Values	Frequency Category	Frequency	Percent
				SAS missing (.)	501	100.00
SMOKE	Do you currently smoke?	Num 8	1 to 1	Range	400	79.84
				SAS missing (.)	101	20.16
IOSTART		Num 8		Range	0	0.00
				SAS missing (.)	501	100.00

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 was omitted from the FORMAT and make corrections.

POTENTIAL PROBLEM: INCOMPLETE FORMAT
Variable has value not in assigned FORMAT definition.
Check FORMAT definitions. If correct, check OUT OF RANGE VALUE report.

Format	Value not in Format	Number of Variables with Value
HEALTH	6	1
WEIGHT	0	1

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.

POTENTIAL PROBLEM: OUT OF RANGE VALUE
Variable has values not in FORMAT definition.
Investigate out of range value if FORMAT is correct.

Variable Name	Label	Out of Range Value	Format
HEALTH	How is your health	6	HEALTH
WEIGHT	How would you describe your weight	0	WEIGHT

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.

POTENTIAL PROBLEM: NUMERIC VARIABLES WITH NO VARIATION IN RESPONSE
All Non-missing values are the same for these variables

Variable Name	Label	Frequency Category	Minimum	Maximum	Frequency	Percent
SMOKE	Do you currently smoke?	Range	1	1	400	79.84

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 table for character variables appears below.

POTENTIAL PROBLEM: CHARACTER VARIABLES NOT ASSIGNED USER FORMAT
Codebook combines all values in the category 'BLANK, TEXT, OR VALUE SUPPLIED'
Try format with categories 'BLANK', 'TEXT OR VALUE SUPPLIED'

Variable Name	type	Variable Length	Variable Label
CITY	Char	10	Fictional city where participant lives

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:

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

The table 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.

Variable Name	type	Variable Length	Variable Label
AGE	Num	8	Age at interview date
BIRTHDATE	Num	8	Date of birth

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.			
Variable Name	**type**	**Variable Length**	**Variable Label**
CASEID	Num	8	Unique identifier for participant
EXER_DAYS	Num	8	In the Past 30 days, how many days did you exercise at least 30 minutes?
INT_DATE	Num	8	Interview date
IOSTART	Num	8	
PERIOD	Num	8	period
SMOKE	Num	8	Do you currently smoke?

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.

POTENTIAL PROBLEM: UNDEFINED VARIABLE LABEL Variable is missing label or has label matching variable name.		
Variable Name	**Label**	**Potential Problem**
IOSTART		Missing variable label
PERIOD	period	Label identical to variable name

Creating a Master List of Datasets: macro TK_dataset_list.sas

Overview

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:

%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 table show attributes of the files.

```
%LET DataFolder= /SAS_Analytics_2017/SAS_Data;

LIBNAME SAS_data "&DataFolder.";

%TK_dataset_list(libref=SAS_data);
```

Below are the results of the TK_dataset_list macro. All six data sets in folder /SAS_Analytics_2017/SAS_Data have been analyzed with information on size and creation date tabulated in the table.

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

Identifying Duplicate Observations: macro tk_find_dups.sas

Overview

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:

%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.

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

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

```
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.

```
%tk_find_dups(dataset=STUDY, one_rec_per=CASEID*WAVE,
dup_output=STUDY_DUPS);
run;
```

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

COPIES = Number of observations with identical values of CASEID*WAVE				
COPIES	**Frequency**	**Percent**	**Cumulative Frequency**	**Cumulative Percent**
1	8	80.00	8	80.00
2	1	10.00	9	90.00
3	1	10.00	10	100.00

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 , 2 observations (2*1) that have the same value of CASEID*WAVE, and 3 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

CASEID	WAVE	COPIES	Frequency
200	**1**	**3**	1
400	**2**	**2**	1

CASEID = 200 has three observations with values of WAVE=1 while CASEID=400 has 2 observations with values of WAVE=2.

```
proc contents data=STUDY_DUPS;
run;
proc print data=STUDY_DUPS;
run;
```

A partial listing of the output data set STUDY_DUPS appears below.

Data Set Name	WORK.STUDY_DUPS	Observations	2
Member Type	DATA	Variables	3

Alphabetic List of Variables and Attributes				
#	Variable	Type	Len	Label
1	CASEID	Num	8	Unique identifier for participant
3	COPIES	Num	8	Number of observations with identical values of CASEID*WAVE
2	WAVE	Num	8	Wave of data collection

The print procedure shows that the output data set STUDY_DUPS has the same data as the second table printed by tk_find_dups.

Obs	CASEID	WAVE	COPIES
1	200	1	3
2	400	2	2

Harmonizing Datasets: macro TK_harmony.sas

Overview

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:

%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.

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

Variable	Type	Length	Label
type_length2	Char	9	*set2_id* variable data type (Num, Char) and length
type_match	Char	3	Both have same data type? (Values: Yes, No)

Example

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.

Harmony of Variables: Web=SAS_data.demography_a1 and Paper=SAS_data.demography_a2				Total	
				N	Percent
Harmony measure	Variable location	Both have same data type?	Both have same variable label?		
DIFF	Both	No	Yes	1	11.11
		Yes	No	2	22.22
SAME	Both	Yes	Yes	4	44.44
SOLO	Paper	N/A	N/A	1	11.11
	Web	N/A	N/A	1	11.11
Total				9	100.00

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.

For the variables that have the Harmony measure of DIFF or SAME:

- 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.

Web=SAS_data.demography_a1 Paper=SAS_data.demography_a2 Inharmonic Variables			Data Type			Label		
Harmony Measure	Variable name	Variable Location	Same?	Web	Paper	Same?	Web	Paper
DIFF	AGE	Both	Yes	NUM 8	NUM 8	No	Age at interview date	Age of participant
SOLO	CITY	Paper	N/A	.	CHAR 5	N/A		Fictional city where participant lives
SOLO	CTIY	Web	N/A	CHAR 10	.	N/A	Fictional city where participant lives	
DIFF	MODE	Both	No	CHAR 3	CHAR 4	Yes	Mode of data collection	Mode of data collection
DIFF	SEX	Both	Yes	NUM 8	NUM 8	No	Gender	Sex of participant

From this report we can tell the following differences need to be corrected when the two data sets are concatenated:

- 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.

Creating Crosswalks: macro TK_xwalk.sas

Overview

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:

```
%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

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.

```
%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.

If the label, type or format differ for a variable that is in multiple sets, then all values will be displayed. For example the WEIGHT variable has a FORMAT assigned in the studys.sas7bdat data set, but not in the health.sas7bdat dataset. The same is also true for the CHG_WEIGHT and HEALTH variables.

VARIABLE CROSSWALK				Total	Data Set		
					SAS_data. demography	SAS_data. health	SAS_data. studya
				Frequency	Frequency	Frequency	Frequency
Total				25	8	5	12
Variable Name	Variable Label	Type-Length	Variable Format				
AGE	Age at interview date	NUM-8		2	1	.	1
BIRTHDATE	Date of birth	NUM-8	MMDDYY	2	1	.	1
CASEID	Unique identifier for participant	NUM-8		3	1	1	1
CHG_WEIGHT	What are you trying to do about your weight?	NUM-8		1	.	1	.
			CHG_WT	1	.	.	1
CITY	Fictional city where participant lives	CHAR-10		2	1	.	1
EXER_DAYS	In the Past 30 days, how many days did you exercise at least 30 minutes?	NUM-8		2	.	1	1
HEALTH	How is your health	NUM-8		1	.	1	.
			HEALTH	1	.	.	1
INT_DATE	Interview date	NUM-8	MMDDYY	2	1	.	1
MODE	Mode of data collection	CHAR-4	$MODE	2	1	.	1
RACE	Race/Ethnicity	NUM-8	RACE	2	1	.	1
SEX	Gender	NUM-8	SEX	2	1	.	1
WEIGHT	How would you describe your weight	NUM-8		1	.	1	.
			WEIGHT	1	.	.	1

References

Terry, James and Chantala, Kim "PROC_CODEBOOK, Automating the Review and documentation of SAS files", SESUG 2010, analytics.ncsu.edu/sesug/2010/BB14.Terry.pdf

Chantala, Kim, Robinson, Jean, and Smith, Helen "A Software Toolkit for Data management", SESUG 2017, available November, 2017

Office of Management and Budget, "Standards and Guidelines for Statistical Surveys", September 2006, https://www.nass.usda.gov/Publications/Methodology_and_Data_Quality/Advanced_Topics/standards_stat_surveys_OMB.pdf

"codebook – Stata", StataCorp. 2017. *Stata Statistical Software: Release 15*. College Station, TX: StataCorp LLC, www.stata.com/manuals13/codebook.pdf

"Overview (CODEBOOK command)", SPSS Inc. Released 2008. SPSS Statistics for Windows, Version 17.0. Chicago: SPSS Inc. https://www.ibm.com/support/.../en/SSLVMB...spss.../syn_codebook_overview.htm,

SAS Institute Inc. SAS Help and Documentation, Cary, NC: SAS Institute.