# Finding Needles in the Documentation Haystack

Susan Myers and David Smarrella
RTI International, RTP, NC

## ABSTRACT

Large complex data structures regularly require substantial documentation. Data from surveys released for public use typically promise confidentiality. Often this leads to certain confidential data being dropped or recoded. Commonly, documentation for the public use data will be based on the documentation for the original file. If the original documentation uses consistent terminology and notation with reference to the confidential data items, then an automatic search to locate areas of the documentation where changes are needed would be useful. We illustrate this by using Base SAS RX pattern matching functions to detect keywords and assure correctly documented data files.

## BACKGROUND - THE HAYSTACK

Our study is a large national household survey conducted yearly. For analytical purposes, a data file and codebook are generated for our client. We average close to 3,000 variables in each codebook. The resulting codebook for these variables runs approximately 800 pages. Each variable is well documented with the following parts:

- Comment which includes question text
- Label which is the SAS label included with the data file
- Levels that explain what each distinct value of the variable represents
- Frequencies of each level

### Figure 1. Excerpt from Codebook

```
                        NON-CORE DEMOGRAPHICS                          PAGE:    603
LABEL   LEN   DESCRIPTION                                                FREQ     %
-----   ---   -----------                                                ----     -
  (QD17)      NOTE:  The school enrollment variable SCHENRL is
              analogous to the variable ENROLED from prior years.
              SCHENRL takes into account information from
              follow-up probes if respondents originally
              reported that they were not enrolled in school.
              Based on these probes, respondents were coded as
              yes (i.e., enrolled) if they were on break from
              school and intended to return to school once their
              break was over.  SCHENRL was assigned a code of
              11 if the respondent reported currently being
              enrolled but there was uncertainty about the
              respondent's school enrollment status.

              The next questions are about school.  Are you now
              attending or are you currently enrolled in school?
              By "school," we mean an elementary school, a junior
              high or middle school, a high school, or a college
              or university.  Please include home schooling as well.

 SCHENRL  2   NOW ENROLLED IN ANY SCHOOL
               1 = Yes.....................................   26263  48.56
               2 = No......................................   27765  51.34
               5 = Yes LOGICALLY ASSIGNED (LFSCHWH2=601)...      12   0.02
              11 = Yes (SCHDSKIP = 30)....................       8   0.01
              94 = DON'T KNOW.............................       4   0.01
              97 = REFUSED................................       4   0.01
              98 = BLANK (NO ANSWER)......................      23   0.04
```

As illustrated in Figure 1, even level definitions can get extremely detailed. This particular variable has a value 5 that references another variable in the file, LFSCHWH2. While this can be a great asset to analysts, it can create difficulties when converting an Analytic File to a Public Use File.

## PROBLEM - THE NEEDLES

When a Public Use File is created, much care is taken to identify which variables might lead to a breach of confidentiality. Each variable in the Analytic File is then treated in one of the following ways:

- Left alone: does not present a problem
- Excluded from the Public Use File
- Recoded
- Substituted

To keep track of these decisions, an Excel Spreadsheet is created for these metadata. Figure 2 contains a small sample of this level of documentation. We have a column to indicate whether the variable will be dropped (Delete column), or Recoded, which will then have a new variable name as indicated in the New Name column. If either the Recode or Delete column have any value, we run the risk of having a documentation problem in our Public Use Codebook. As seen in Figure 1, with the variable SCHENRL referencing the variables LFSCHWH2 and SCHDSKIP, our documentation could include variable names that no longer exist, as a result of deletion or renaming.

**Figure 2. Metadata of Public Use Datafile Variable Treatment**

| | | | Public Use File | | | |
|---|---|---|---|---|---|---|
| **Analytic Order** | VARIABLE | **Analytic 2002** | **Recode** | **Substitute** | **Delete** | **New Name** |
| | | | | | | |
| 1 | **IDENTIFICATION** | | | | | |
| 2 | ID | X | | | | |
| 3 | AGE | X | 1 | Y | | AGE2 |
| 4 | BRTHDATE | X | | | A | |

With this in mind, how do we go about finding these possible inconsistencies in our 800 page document? We know we can generate a list of variable names to flag from the metadata. However, how can we effectively search for each flagged occurrence so that data managers can be alerted to each problem?

## SOLUTION - THE RX FUNCTION

SAS has always had an extensive set of very useful string functions. However, they did not seem to satisfy our needs in this particular situation. For example, if we used the INDEXW function to look for the variable AGE, would we find <AGE? Since INDEXW looks for white space as a delimiter, it would not detect this problem. Had we used INDEX, we would find our occurrence of AGE within <AGE. However, we would also find every instance of AGE within the word PAGE 795 times!

Fortunately, SAS introduced the flexible RX pattern matching functions. We can now add rules for our search. Character Classes are provided to the programmer to specify special searches. The Default Character Classes offered by SAS are listed in Figure 3. User defined Character Classes may be built by the programmer by concatenating the Character Class Complements (characters '^' or '~') to a string within quotes. Either character, '^' or '~', are interpreted as NOT symbols. The Complements used in conjunction with Character Classes add a powerful tool to a programmer's toolbox.

**Figure 3: Default Character Classes**

| | |
|---|---|
| `$a or $A` | matches any alphabetic upper- or lowercase letter in a substring ($'a-zA-Z'). |
| `$c or $C` | matches any character allowed in a version 6 SAS name found in a substring ($'0-9a-zA-Z_'). |
| `$d or $D` | matches any digit in a substring ($'0-9'). |
| `$i or $I` | matches any initial character in a version 6 SAS name found in a substring ($'a-zA-Z_'). |
| `$l or $L` | matches any lowercase letter in a substring ($'a-z'). |
| `$u or $U` | matches any uppercase letter in a substring ($'A-Z'). |
| `$w or $W` | matches any white space character, such as blank, tab, backspace, carriage return, etc., in a substring. |

Examples of building a User Defined Character Class are located in Program Step 3 below. By using our default $w in rx=rxparse("$w"||"&var"||"$w"), we can search for variable names surrounded by white space. This will eliminate the occurrences of finding the variable name AGE on every page (within the word PAGE). Our second Defined Character Class, rx2=rxparse("^'A-z'"||"&var"||"^'A-z'") allows us to search for variable names that may have non alpha characters immediately before and after, i.e. <AGE. By combining the two searches, we have captured our desired flagged occurrences while eliminating results that are not our concern.

**Program Step 1: Initialize**

```
%let year=2002;
%let yr=02;
%let path2=w:\archive\data&yr.\puf\;

/* text file to search */
filename cbk "z:\irb\&year.codebooks\puf\cbk.txt";

/* output file of found var names */
filename found "z:\smyers\codebook\&year.\puf\FoundText.txt";

options symbolgen nocenter;

filename xls&yr. DDE "Excel|&path2.[PUFVariableTrack.xls]sheet1!R5C1:R3035C28" NOTAB;
libname local "Z:\smyers\codebook\&year.\puf\";
```

### Program Step 2: Read Our Metadata

```
/* Read in excel file with variable names that have been dropped or renamed */
data drop_rec (keep=varname);
  infile XLS&yr. DLM='09'X DSD missover lrecl=500;
     input order : 3. varname : $8. fin01 : $1. recnote01 : $1. subnote01 : $1. delnote01 : $1.
recvar01 : $8.;
if recvar01 ^= ' ' or delnote01 ^= ' ' then output;
run;

data _null_;
 file found;
run;

/* Put the variable list into a table that can be looped through */
data _null_;
  set drop_rec end=last;
  call symput('v'||left(_n_),trim(left(varname)));
  if last then call symput('index',trim(left(_n_)));
run;
```

### Program Step 3: Search our Document

```
/* Loop through the list of variable names and search codebook text for them */

%macro SearchIt(var);

data cbk;
length line $84. search $10.;
 infile cbk truncover;
  input line $1-84;
/* Put a leading space for writing to output file */
  search=' '||"&var";
  file found mod;
  linenum=_N_;
/* On first observation, set search values */
  if _N_=1 then
    do;
/* $W puts a blank at the beginning and end */
     rx=rxparse("$w"||"&var"||"$w");
/* 2nd restriction on search values to check for nonalpha leading and trailing chars to catch
occurrences such as '<age' */
     rx2=rxparse("^'A-z'"||"&var"||"^'A-z'");
    end;
  retain rx rx2;
  match=rxmatch(rx,line);
  match2=rxmatch(rx2,line);
   if match > 0 or match2 >0 then
     put @1 linenum @9 search @20 line;
run;

%mend SearchIt;

/* Loop through the process with each variable name in the lbl file */
%macro readlbl1;
         %do i=1 %to &index;
             %SearchIt(&&v&i);
         %end;

%mend readlbl1;

/* Start the process */
%readlbl1;
```

## CONCLUSION

The RX pattern matching functions are excellent tools for finding relevant keywords in large documents. Though our example here covers only one situation, there are many instances where these functions can help validate data when integrity is in question. Commonly given examples include Social Security Numbers or Phone numbers. There are many options available to the programmer within the RX family. We encourage the user to explore these and make use of these powerful yet practical tools.

## CONTACT INFORMATION

Susan Myers, smyers@rti.org, (919) 541-7441
David Smarrella, dsmarrella@rti.org, (919) 541-7072

Each author can be reached at the mailing address:

RTI International
P.O. Box 12194
Research Triangle Park, NC 27709-2194.

## TRADEMARK NOTICE

SAS and all other SAS Institute Inc. product or service names are registered trademarks or trademarks of SAS Institute Inc. in the USA and other countries. ® indicates USA registration.

Other brand and product names are registered trademarks or trademarks of their respective companies.