ABSTRACT
At the conclusion of many survey-based data collecting projects, recoding the hundreds and thousands of character vari-
ables to ‘reserved scale’ specified numeric variables is a uncomplicated but cumbersome task for SAS® programmers. If
you are a person who likes to avoid a large amount of typing as much as I do, this paper will give you an idea of how to
maintain high quality for this recoding task with minimal typing. This paper also answers the following questions: How can
you create a powerful SAS® MACRO that will write the IF-ELSE-THEN Statement for you? How can you avoid any human
errors such as typos? And how do you use MS Excel to speed up your work?

INTRODUCTION
As a programmer, I am often in situations in which very vast quantities of survey data need to be recoded into correspond-
ing numeric values in a very short time. The recoding process is not necessarily a difficult one, but it has the potential to
be very time-consuming and frustrating.

If we ignore the matter of program efficiency for a moment, there are so many words to type, between using IF-ELSE-
THEN statements, SELECT-WHEN statements or PROC FORMAT tricks; the mere fact that there is so much to type
means that there is a great risk of making small typos, such as omitting a semicolon or misspelling a variable name, both
of which are errors that would require time and energy to de-bug. Even if you consider yourself to be the most detail-
orientated person in the world, human error almost seems inevitable. Do you ever dream of using powerful SAS® MAC-
RO to do all of the tedious typing for you? This paper will use a real example drawn from a specific project to demonstrate
the quick tricks to do just that. With a basic knowledge of SAS® MACRO, you can make this dream a reality.

TAKE FULL ADVANTAGE OF WHAT WE HAVE
For all of our examples we are using data that is in two different files. First, we have a raw data file (see Figure 1a), which
contains several records with answers to questions starting with variable Q1a (4001) in column H. Second, we have a
Names and Labels spreadsheet (see Figure 1b), which contains labels for each question in the raw data.

While the files we have may not seem to be hard to handle, they are actually quite messy; there is some character raw
data in Excel format, a separate Excel spreadsheet that contains the names and labels information for the new numeric
variables and a detailed Questionnaire description, and finally instructions for us to recode the data. So how can we inte-
grate these three files? Normally, we could write IF-ELSE-THEN codes that follow the logic reference provided by the
Questionnaire description and then apply these codes to the raw data. The variable names and labels spreadsheet will be
used to generate the SAS® labels. Unfortunately, this method still requires a great deal of effort. Instead, let us take an-
ochre look at the raw data and Labels files.

<table>
<thead>
<tr>
<th>Access Code</th>
<th>Level Reference Id</th>
<th>Module Id</th>
<th>Module Reviewed</th>
<th>Module Submitted</th>
<th>Q1a (4001)</th>
<th>Q1b (4002)</th>
<th>Q1c (4003)</th>
</tr>
</thead>
<tbody>
<tr>
<td>2 G6FA0</td>
<td>0315</td>
<td>07</td>
<td>No</td>
<td>Yes</td>
<td>YES</td>
<td>YES</td>
<td>NO</td>
</tr>
<tr>
<td>3 P6600</td>
<td>1005</td>
<td>07</td>
<td>Yes</td>
<td>Yes</td>
<td>NO</td>
<td>NO</td>
<td>NO</td>
</tr>
<tr>
<td>4 T6000</td>
<td>0232</td>
<td>07</td>
<td>No</td>
<td>Yes</td>
<td>YES</td>
<td>YES</td>
<td>YES</td>
</tr>
<tr>
<td>5 B631</td>
<td>0596</td>
<td>07</td>
<td>Yes</td>
<td>No</td>
<td>YES</td>
<td>YES</td>
<td>NO</td>
</tr>
<tr>
<td>6 E760M</td>
<td>1087</td>
<td>07</td>
<td>No</td>
<td>Yes</td>
<td>YES</td>
<td>YES</td>
<td>YES</td>
</tr>
<tr>
<td>7 J66E</td>
<td>1077</td>
<td>07</td>
<td>No</td>
<td>No</td>
<td>NOT DISPLAYED</td>
<td>NOT DISPLAYED</td>
<td>NOT DISPLAYED</td>
</tr>
<tr>
<td>8 R6E3</td>
<td>1036</td>
<td>07</td>
<td>No</td>
<td>Yes</td>
<td>YES</td>
<td>YES</td>
<td>NO</td>
</tr>
<tr>
<td>9 L6403</td>
<td>0568</td>
<td>07</td>
<td>No</td>
<td>Yes</td>
<td>YES</td>
<td>NO</td>
<td>NO</td>
</tr>
<tr>
<td>10 V66DA</td>
<td>0706</td>
<td>07</td>
<td>No</td>
<td>Yes</td>
<td>YES</td>
<td>YES</td>
<td>YES</td>
</tr>
<tr>
<td>11 G660K</td>
<td>0926</td>
<td>07</td>
<td>No</td>
<td>Yes</td>
<td>YES</td>
<td>YES</td>
<td>YES</td>
</tr>
<tr>
<td>12 P60C</td>
<td>0149</td>
<td>07</td>
<td>No</td>
<td>Yes</td>
<td>YES</td>
<td>YES</td>
<td>YES</td>
</tr>
<tr>
<td>13 V690F</td>
<td>0290</td>
<td>07</td>
<td>No</td>
<td>Yes</td>
<td>YES</td>
<td>NO</td>
<td>NO</td>
</tr>
<tr>
<td>14 D669D</td>
<td>0210</td>
<td>07</td>
<td>No</td>
<td>Yes</td>
<td>YES</td>
<td>NO</td>
<td>NO</td>
</tr>
<tr>
<td>15 F660C</td>
<td>0446</td>
<td>07</td>
<td>Yes</td>
<td>Yes</td>
<td>YES</td>
<td>YES</td>
<td>NO</td>
</tr>
<tr>
<td>16 B664</td>
<td>0351</td>
<td>07</td>
<td>No</td>
<td>Yes</td>
<td>YES</td>
<td>NO</td>
<td>NO</td>
</tr>
<tr>
<td>17 F6A02</td>
<td>0917</td>
<td>07</td>
<td>No</td>
<td>Yes</td>
<td>YES</td>
<td>YES</td>
<td>YES</td>
</tr>
<tr>
<td>18 X66E</td>
<td>0495</td>
<td>07</td>
<td>No</td>
<td>Yes</td>
<td>YES</td>
<td>YES</td>
<td>YES</td>
</tr>
<tr>
<td>19 O1777</td>
<td>0122</td>
<td>07</td>
<td>No</td>
<td>Yes</td>
<td>YES</td>
<td>NO</td>
<td>NO</td>
</tr>
<tr>
<td>20 L700C</td>
<td>1125</td>
<td>07</td>
<td>No</td>
<td>Yes</td>
<td>YES</td>
<td>NO</td>
<td>NO</td>
</tr>
<tr>
<td>21 R650E</td>
<td>0765</td>
<td>07</td>
<td>No</td>
<td>Yes</td>
<td>YES</td>
<td>YES</td>
<td>NO</td>
</tr>
<tr>
<td>22 K66ECE</td>
<td>0954</td>
<td>07</td>
<td>No</td>
<td>Yes</td>
<td>YES</td>
<td>NO</td>
<td>NO</td>
</tr>
</tbody>
</table>
Figure 1a. Snapshot of the raw data

![Table of Question Variables and Labels](image)

Figure 1b. Snapshot of the Names and Labels Spreadsheet

The column heading “Q1a(4001)” in Figure 1a represents the first question of the survey, and it corresponds to the first variable name in Figure 1b, END142, which also represents the first question in the survey. Traditionally, we could write codes such as: 'IF Q1a(4001)="YES" THEN END142=1;'. However, we should take advantage of the fact that we already know how the contents of these two files correspond to each other.

**PROC FREQ—IT'S NOT JUST ABOUT THE FREQUENCIES**

It might seem like a very strange idea to use the PROC FREQ procedure to begin recoding data. But in fact, this procedure works perfectly.

![Output of Proc Freq](image)

Figure 2. Output of Proc Freq.

Take a close look at Figure 2 and you will notice that the values in the column for the variable labeled “F8” are all of the possible values this variable can have in the raw data. The output shown in Figure 2 almost follows the structure of a complete IF-THEN-ELSE statement. Although it is missing a few components of the IF-THEN-ELSE format, we can use this information to write the entire statement quite easily. The following diagram illustrates how to do this.
We must start with either the phrase “IF” or “ELSE IF”, then reference the name of the variable in question (which we will retrieve by using a SAS® MACRO), an equal sign, and the value of the F8 variable (which was already generated, see Fig. 2). Next, we add the phrase “THEN” and then follow a similar format: the new variable names that we can retrieve from the Labels spreadsheet (Figure 1b), an equal sign, and the appropriate numeric value. Finally, we add a semicolon to indicate the end of the statement.

You may be wondering what the F8 variable stands for. Since the raw data is in Excel format, we can use the GETNAMES function in our PROC IMPORT code. In this example, GETNAMES=NO is recommended. We can use this to easily assign sequential orders to the character variable names, which will be beneficial for the following steps.

**DYNAMIC SAS® MACROS—MACROS THAT WILL DO ALL OF THE WORK FOR YOU**

In the last step, we created the general frame of our codes. What we need to do now is to find an effective way to create a functional S SAS® MACRO that contains one MACRO variable to retrieve the value of F8, one to retrieve the variable names from the raw data and an other to retrieve the new variable names from the Labels spreadsheet (Figure 1b) and combine those MACRO variables together.

```sas
%macro freq;
%do i=8 %to 68;
proc freq data=hea noprint;
table F%i/out=q%eval(&i-7) (rename=(F%i=fvalue));
run;
data q%eval(&i-7);
set q%eval(&i-7);
length Raw $15;
Raw="F%i";
%end;
%mend;
%freq;
```

The code above will generate PROC FREQ output for every character variable which needs to be recoded. We developed a new variable, “Raw”, to store the variable names in the raw data. You may notice we use %eval(&i-7) here, since the variables need to be recoded starting from F8, and doing this will give us a better one to one correspondent relationship between variables in the raw data and the new variable names. That is, it passes variables F1 through F7 and begins with variable F8 as the first variable to recode.

There are a few tricks that are worth noticing.

First, we use a CALL SYMPUT DATA step to define MACRO variables. The advantages of this method include not only assigning a sequential number to each variable or each data set, but also providing a convenient way to store the compound character values. In the following code, the order of observation _N_ is used to assign a sequential number to each variable name. The total number of variables is also retained in &nvar.

**Figure 2 Partial output of MACRO %freq**

<table>
<thead>
<tr>
<th>F8</th>
<th>Frequency</th>
<th>Percent of Total Frequency</th>
<th>Raw</th>
</tr>
</thead>
<tbody>
<tr>
<td>DISPLAYED</td>
<td>1</td>
<td>0.1149275362 F8</td>
<td></td>
</tr>
<tr>
<td>NO</td>
<td>110</td>
<td>15.942028986 F8</td>
<td></td>
</tr>
<tr>
<td>NOT ANSWERED</td>
<td>14</td>
<td>2.028985072 F8</td>
<td></td>
</tr>
<tr>
<td>NOT DISPLAYED</td>
<td>48</td>
<td>5.985217391 F8</td>
<td></td>
</tr>
<tr>
<td>YES</td>
<td>517</td>
<td>74.927536232 F8</td>
<td></td>
</tr>
</tbody>
</table>
For each data set, we set up a variable order equal to _N_, which determines values of V1; in other words, it determines if we should use "IF" or "ELSE IF" at the beginning of the statement.

We can apply this basic logic in the codes. According to the questionnaire description, we know that 90% percent of the variables use the same logic to define how to convert "YES", "NO", "SKIPPED" and "NOT ANSWERED". I was unwilling to incorporate every possible situation in the codes, such as "NO DISTRICT-LEVEL STAFF IN THIS AREA", which are all values that are long and can easily prompt human errors. Keep in mind that we are trying to avoid a large amount of typing.

```sas
%macro combine;
data _null_;set name;
call symput(compress("varname"||_N_),trim(F1));
call symput(compress("nvar"),compress(_N_));
run;
%do i=1 %to &nvar;
data a&i;
   length name $50 ffvalue $150;
   set q&i;
   name="&&varname&i";
   ffvalue=fvalue;
   order=_N_;run;
   data b&i(keep=v1 v2 v3 v4 v5 v6 v7 v8 v9);
   set a&i;
   length v1 v2 v3 v4 v5 v6 v7 v8 v9 $150;
   if order=1 then v1="if";
   else v1="else if";
   v2=Raw;
   v3="=";
   v4="'|'\trim(ffvalue)||'\'';
   v5="then";
   v6=name;
   v7="'|'=";
   if upcase(ffvalue)="YES" THEN v8="1";
   else if upcase(ffvalue)="NO" THEN v8="2";
   else if ffvalue in ("SKIPPED","SK") THEN v8=".A";
   else if ffvalue in ("NOT ANSWERED","\",""NA",""DISPLAYED",""NOT DISPLAYED") THEN v8=".C";
   else if ffvalue in ("NOT APPLICABLE") THEN v8=".N";
   ELSE v8=" ";
   V9=";
run;%end;%mend;%combine;
```

Figure 3. Partial output of MACRO %combine

As Figure 4 shows, we now have every element we need for the IF-ELSE-THEN statement.

OUTPUT THE RESULT IN AUTO FILTER EXCEL FORMAT

Although the output in Figure 4 contains all of the information we need, we need to realize that we cannot do anything to the data sets unless we transfer the output into a particular format. First I tried to use a text file since we can simply CALL the recodes file using an %INCLUDE statement. However I soon discovered the drawbacks of this method. Remember, in the previous step we only coded the common values, and ignored the longer values to avoid human error. So now, for those long values we will have blank values in column V8. If we output the data in a text file format, it won't be easy to find those blanks. That is a risk we do not want to take.
The benefit of outputting data into auto filter excel format is that it makes finding values that are not recoded an easy job; also, you can easily fill in the blanks. Furthermore, we can use this output as double check. For example, if you find any blank values which are not associated with some long words, you’d better check your program again. After all of the blanks in V8 column are filled in, the recodes are done. Although this paper refers to this as a one hour fix, it’s actually even shorter than one hour. If we compare this to manually coding everything, the time you save with automation and the human error you avoid are valuable not only to you but to your clients also.

CONCLUSION
This paper serves as an introduction to creating a lot of recodes with minimal typing. The programs and method presented in this paper can be easily modified to fit the needs of different users. I hope to have provided sufficient explanation for even a beginner level SAS® programmer to learn how to handle such tasks. Although my methods are basic, they are extremely useful even in an advanced user’s day-to-day programming.

ACKNOWLEDGMENTS
I would like to thank my boss Tonja Kyle and our client (CDC/OID/NCHHSTP) for providing me with great challenges over the years. I also want to thank my colleague Kamya Khanna, who is the co-author for this paper, for her helpful discussions.

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

**Wen Song**
Company: ICF International
Address: 11785 Beltsville Dr, suite 300
City, State ZIP: Beltsville, MD, 20705
Work Phone: 301-572-0963
E-mail: wen.song@icfi.com

**Kamya Khanna**
Company: ICF International
Address: 11785 Beltsville Dr, suite 300
City, State ZIP: Beltsville, MD, 20705
Work Phone: 301-572-0963
E-mail: kamya.khanna@icfi.com