Week 5 Walkthrough
Renaming and Labeling Variables

James Robertson

January 31, 2025

Changing Variable Names

Today's topic is a big one! Or at least a big deal. I've mentioned time and again how coders are lazy creatures and SAS is here to enable us in all the best ways. Today we're going over the RENAME and LABEL functionality in SAS DATA **steps**. Let's start with some motivation.

Motivation

I have a dataset in Excel that looks like the image below.

Duration (in seconds)	Finished	Q1.2 - Did you attend the	Q1.3 - In what country did you r	Q1.4 - 50 States, D.C. and Pu	Q1.5 - In wh
24	TRUE	No			
24	TRUE	No			
570	TRUE	Yes	United States of America	Texas	Dallas
616	TRUE	Yes	United States of America	Minnesota	Saint Paul
0	TRUE	Yes	United States of America	District of Columbia	Washington,
566	TRUE	Yes	United States of America	District of Columbia	Washington
1230	TRUE	Yes	United States of America	District of Columbia	Washington,
1375	TRUE	Yes	United States of America	Florida	St. Petersbu
941	TRUE	Yes	United States of America	California	Riverside
516	TRUE	Yes	United States of America	Arizona	Phoenix

Looking at these column names, an overwhelming sense of dread grows in the pit of my stomach. These column names are *hideously long*, so long that they have all been truncated on-screen. And that's before even touching on the column names having spaces, parentheses, hyphens, commas, and goodness knows what else. *Horrible!*

Solution Using RENAME

When I'm staring down data like this with the understanding that I *must* read it in, the first thing I do is toss out all of those hideous names and use GETNAMES=NO in my PROC IMPORT step.

```
PROC IMPORT DATAFILE=REFFILE
    DBMS=XLSX
    OUT=lib.dataset;
    REPLACE;
    GETNAMES=NO;
RUN;
```

When I use GETNAMES=NO, every column gets assigned a **default** name, specifically the columns will be named VAR1, VAR2, VAR3, and so on. Not very descriptive! So we've solved the problem of *hideous, evil* names and replace it with the problem of *boring, useless* names. A better problem to have in my book, but we're not quitting until the problem is *actually* solved! This is where we would like to use RENAME to set things right.

You may have noticed that I have not referred to RENAME as a **statement**, this isn't because it's *not* a **statement**, but because it's not *only* a statement. Let me explain.

When it comes to RENAME, there are two ways to go about it. You can have a RENAME **statement** standing freely in a DATA **step** or you can have RENAME as an **option** in a SET statement. Either way, the way we write it is much the same, so let's start by looking at how to write out our RENAME.

The **syntax** (grammar) of RENAME has only a few rules. The first rule is you have to start with RENAME! Without that, how will SAS ever know what to do? The second rule is that after the RENAME statement you write the *current* name of your variable, an equals sign =, and then the *new* name of your variable. The third rule is that you can then repeat this process as much as you like, with spaces or new lines separating the variables under consideration, and (if using the RENAME **statement**) **end with a semicolon** ;. This works out to look like the following.

```
RENAME old_name_1 = new_name_1
       old_name_2 = new_name_2
       old_name_3 = new_name_3;
```

Always remember `old = new`, the order is important! Since we're often renaming variables from `VAR1` or similar, remembering that these default names go on the left is one way to think about it.

In the example shown earlier, we have six (6) columns that are currently named `VARn` (where `n` is some number) we would like to rename. Whenever I rename variables, I want the name to be *as short as possible* while still being descriptive. Why? Because I don't want to type a long name a bunch of times! For this example, let's start by using a `RENAME` **statement** to give these some more helpful names.

```
DATA lib.survey2;
    SET lib.survey;
    RENAME VAR1 = duration
           VAR2 = finish
           VAR3 = q1_2
           VAR4 = q1_3
           VAR5 = q1_4
           VAR6 = q1_5;
RUN;
```

We can achieve the same thing via the `RENAME` **option** and it looks pretty similar.

```
DATA lib.survey3;
    SET lib.survey (RENAME = (VAR1 = duration
                              VAR2 = finish
                              VAR3 = q1_2
                              VAR4 = q1_3
                              VAR5 = q1_4
                              VAR6 = q1_5));
RUN;
```

Notice that when `RENAME` is an **option**, we have to have an equals sign and parentheses. Notice also that in both of these examples, I have saved the output into a new dataset (`lib.dataset2` and `lib.dataset2`)! This is often a good ideas as if we overwrote `lib.dataset` and then re-ran this code, we would start getting errors saying `VAR1` doesn't exist *because we already renamed it*! If you get an error saying a variable doesn't exist, try peeking at the data to be sure you haven't renamed the variable.

Going Further with `LABEL`

Now that our variables have all been renamed, it's time to begin coding! Except I already forgot what some of the variables represent. Oops...

I can open Excel back up and stare at columns there, but I'd really rather have all the information I'm looking for in `SAS` so I can send someone just my `.sas` and `.sas7bdat` files and ignore their emails from there[1]. This is where `LABEL` comes in! Just like `RENAME`, `LABEL` can be an **option** or a **statement**. The **syntax** (grammar) is pretty much identical, shown below. The big change is instead of a new variable name, we give some amount of decription.

```
DATA lib.survey2;
    SET lib.survey;
    LABEL VAR1 = "Duration (in seconds)"
          VAR2 = "Finished"
          VAR3 = "Did you attend..."
          VAR4 = "Country of attendance"
          VAR5 = "State of attendance"
          VAR6 = "City of attendance";
RUN;

DATA lib.survey3;
    SET lib.survey (LABEL = (VAR1 = "Duration (in seconds)"
                             VAR2 = "Finished"
                             VAR3 = "Did you attend..."
                             VAR4 = "Country of attendance"
                             VAR5 = "State of attendance"
```

[1]For legal reasons, this is a joke. I would remind you that responding to emails in a timely manner is important as it preserves your rapport with collaborators and coworkers.

```
                                VAR6 = "City of attendance"));
RUN;
```

It is important to note that I am using the *old names* as I am pulling from the original dataset with my SET **statement**. Ordering matters for this as if we rename the variables before labelling, then we have to use the new names. **Remember that options in the SET statement will be executed before any other statements, this comes up often when labelling and renaming!**