PharmaSUG 2015 - Paper QT08

Hands Free: Automating Variable Name Re-Naming Prior to Export

John Cohen, Advanced Data Concepts LLC, Newark, DE

ABSTRACT

Often production datasets come to us with data in the form of rolling 52 weeks, 12 or 24 months, or the like. For ease of use, the variable names may be generic (something like VAR01, VAR02, etc., through VAR52 or VAR01 through VAR12), with the actual dates corresponding to each column being maintained in some other fashion – often in the variable labels, a dataset label, or some other construct. Not having to re-write your program each week or month to properly use these data is a huge benefit.

Until, however, you may need to capture the date information to properly document – in the variable names (so far VAR01, VAR02, etc.) – prior to, say, exporting to MS/Excel® (where the new column names may instead need to be JAN2011, FEB2011, etc.). If the task of creating the correct corresponding variable names/column names each week or month were a manual one, the toll on efficiency and accuracy could be substantial.

As an alternative we will use an approach using a "program-to-write-a-program" to capture date information in the incoming SAS® dataset (from two likely alternate sources) and have our program complete the rest of the task seamlessly, week-after-week (or month-after-month). By employing this approach we can continue to use incoming data with generic variable names and output our results with specific (and correct!) variable names, all hands free.

METADATA

Metadata are data about data, a description of the contents of a data set including variable names, variable lengths, data types, value labels, and the like.[1] These are stored, if we know how to access them, in SAS data sets. As experienced SAS programmers, we are practiced at manipulating SAS data sets. Today we may be reporting on quarterly sales results, calculating the P-value of the statistical difference between test and control data sets, comparing defect rates between production lines in two different locations, average test scores by school district, manipulating character values to identify most-mentioned products, or the like.

We can just as easily report on data set contents, calculating differences in data set update dates, or manipulating character variables to modify variable labels or renaming variables in bulk. We will describe here just such a process, taking metadata for a SAS data set and using these data to create a process for deriving a modified version of the original data set, updating according to our requirements. The values of the incoming data will NOT be changed (at least in this instance). What WILL be modified are some of the metadata data – such as variable names, value labels, etc.

PROGRAM TO WRITE A PROGRAM – THE APPROACH

Any program is basically reasonable to write the first time. After all, that is (at least part of) what we do. And the very process of discovery, of testing and building, creates much of the value in our resulting code. However, updating the program because a few things have changed the next week, and again the following week, and so on, likely does not add the same value. If substantial logic changes are required, again that is part and parcel of the life of a SAS programmer. But if the changes consist of incremental updating of, say, date-related variable names and labels, listings based on the latest customer lists, rankings of sales results by territory, or the like, then the value of meticulously "hand-coding" the program modifications is negligible. And the QC requirements and probability of introducing new errors each time is certainly a costly time sink.

If you are thinking that there must be a better way, well, sometimes there is. A collection of techniques have been developed by some of the clever folks in our SAS user community. These are referred to collectively as table-driven, dynamic data-driven, and the like.[2] Some of these are quite powerful and also quite involved. But the common theme is that we build programs which take advantage of the underlying data to help write the program,

[1] For a more extensive discussion of the concept and usage of metadata set DiIorio, "Metadata 101: A Beginner's Guide to Table-Driven Applications Programming".
[2] See DiIorio and Abolafia, Fehd and Carpenter, or Koslova and Berestizhevsky for three different examples.

thus obviating some of the more mundane, lower-value programming tasks, reducing error rates, and help to make the program more self-documenting. This means that when the data change, our programs will also change – all without programmer intervention. (This is the dynamic part.) It also means that somehow, some of the programming components will be created for us as part of the complete execution of the program. (This is the program-to-write-a-program component.)

ACCESSING THE METADATA

Our first requirement will be to access the metadata – data about our data. (A representation of our data is visible in Figure 1.) We will use what for most should be the most accessible technique – PROC CONTENTS with the OUT= option.[3] This option will write out the contents – with which we are likely quite familiar from the report this procedure generates – to a SAS data set. And we all know what to do with a SAS data set! The default output data set contains many more variables than we will need. Instead we will only keep those required for the next step in the process.

Figure 1 – Our Data: Rolling 52-Week Sales Figures by Territory

Territory_Name	Week_01	Week_02	Week_03	"	"	"	Week_52
Boston, MA	127	143	113	"	"	"	157
Portland, ME	115	98	103	"	"	"	112
Wilmington, DE	93	47	86	"	"	"	101
"	"	"	"	"	"	"	"

Our **PROC CONTENTS** statement might look like this:

Figure 2 – Accessing the Metadata

```
Proc Contents data=Weekly_Sales
              OUT=Weekly_Sales_Contents(keep=name label);
Run;
```

In Figure 2 we create the SAS metadata SAS data set (did you get that) with the OUT= option. We named this new data set Weekly_Sales_Contents and kept only the variables NAME (the name of the variables in the Weekly_Sales SAS data set) and LABEL (the corresponding variable labels). Again, the metadata data (at least those components which we want to access for the next step) are stored as the SAS data set Weekly_Sales_Contents and look something like Figure 3. FYI, each observation in the metadata reflects one variable in our original data set.

Figure 3 – Our Metadata

Name	Label
Week_01	week_ending_2010_01_08
Week_02	week_ending_2010_01_15
Week_03	week_ending_2010_01_22
"	"
Week_52	week_ending_2010_12_31

[3] Some of the more elegant solutions alluded to previously use combinations of PROC SQL and the SAS macro language, both powerful but also more advanced components of Base SAS than is appropriate here.

DYNAMIC DATA-DRIVEN COMPONENT

We next bring this metadata SAS data set into a DATA step and manipulate it as in the example in Figure 4. Our goal is to take the variable labels – each one capturing the actual (week ending) date of that variable/data column – and rename the variables – all generic names in the incoming data set – so that the resulting output data set variable names become self-documenting.

Figure 4 – Manipulating the SAS Metadata & Creating the Program Component

```
                                                    /** create temporary storage space **/
Filename RENAME ‘C:NESUG 2011\CC04\update_rename.sas’;

Data _null_;
   Set Weekly_Sales_Contents end=eof;
   File RENAME;                                      /** access temporary storage space **/

   If _n_ = 1 then put @1 'rename ';                 /** beginning of program component **/

   New_name = substr(label,13,10);                   /** “calculate” values **/

   put @8 name $8. ' = '  new_name $10.;             /** write rename statements **/

   if eof then put @3 ';' ;                          /** complete the SAS statement **/
run;
```

Simply put, we have taken the metadata, captured the specific (week ending) date for each variable/observation in turn, and created a statement which will rename the original generic variable names of Week_1, Week_2, etc. to _2010_01_08, _2010_01_15, etc. These rename statements are created one observation at a time, and the final entire programming component, from the initial “RENAME”, through each individual variable/observations respective rename step, to the final statement-ending semi-colon (“**;**”) will be captured in a text file named “update_rename.sas” – in fact a little SAS program fragment. It may look a bit like Figure 5.

Figure 5 – The Program Component in the Flesh

```
Rename
        Week_01 = _2010_01_08
        Week_02 = _2010_01_15
        Week_03 = _2010_01_22
           "    "      "
           "    "      "
           "    "      "
        Week_52 = _2010_12_31
   ;
```

To access this program component/fragment, a simple **%INCLUDE** statement referring to the original **FILENAME** will do the trick as in Figure 6, which includes what might be a representative program. We create a (rolling) annual total – made much simpler with the generic variable names – then rename the original weekly variables and output to a new SAS data set. We may then pass this data set along to a colleague, or vendor in a different company, or export to, say, MS/Access® or MS/Excel. The (week ending) dates of the data columns are now captured in the outgoing data set variable names, and at least this component of documenting our data will require no additional effort on our part.

Figure 6 – Accessing the Program Component

```
Data report;
   Set Weekly_Sales;
   Rolling_Total = sum(of Week_01 - Week_52);
   %INCLUDE RENAME;
Run;

   /** forward to colleague, e-mail to outside vendor, or export to Excel **/
   /** Dates of columns are now documented each time the program executes **/
   /** with no need for any additional program modifications              **/
```

Through testing, we know that these are valid variable names and specific to the contents of each variable/column. This means that when we want to run the program again next week (or next year), we can re-run the same program **AS IS** – no changes!!!!! We do not need to hand-code **`Week_01 = _2010_01_15`** when we run again next week, **`Week_01 = _2010_01_22`** the following week, and so forth. The program will do this for us with much greater accuracy and requiring little time.

CONCLUSIONS

Some programming tasks, once a program is well-past initial development and testing, become both tedious and fraught with potential for errors quite out-of-keeping with the value of the programming time we need to invest in making incremental changes and the testing thereof. Where these changes are reflected in the underlying data in ways which we can manipulate – once captured in the metadata – we may be in the position to create dynamic, data-driven programs which can simplify some of the regular updating tasks. These will then both save us programming time – much of it mundane – and improve our accuracy – a win/win.

But few things in life are free. This comes at a cost of an increase in upfront programming complexity and in development and testing time. Further, as the program can almost become "turnkey," there may be a need to build in automatic validation processes. The payoff comes in the ease of running our process in the future. The approach described here is hopefully relatively easy to understand and replicate.

REFERENCES:

Frank DiIorio, "Metadata 101: A Beginner's Guide to Table-Driven Applications Programming", in **Proceedings of the 20th Annual Northeast SAS Users Group Conference**, Nov. 11 – 14 2007, Baltimore, MD.

Frank DiIorio and Jeff Abolafia. "Dictionary tables and views: Essential tools for serious applications", in **Proceedings of the 29th SAS User Group International Conference**,May 9-12, 2004, Montréal, Canada.

Fehd , Ronald J. and Art Carpenter, "List Processing Basics: Creating and Using Lists of Macro Variables", in **SGF 2007 Conference Proceedings**, March 22-25, 2009, National Harbor, Maryland.

Kolosova , Tanya and Samuel Berestizhevsky, **Table-Driven Strategies for Rapid SAS Applications Development**, Cary, NC: SAS Institute Inc., 1995.

ACKNOWLEDGMENTS

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.

CONTACT INFORMATION

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

John Cohen
Advanced Data Concepts LLC
Newark, DE
(302) 559-2060
Jcohen1265@aol.com