Paper SAS637-2017

A Long-Time SAS® Programmer Learns New Tricks

Lisa Horwitz, SAS Institute Inc.

ABSTRACT

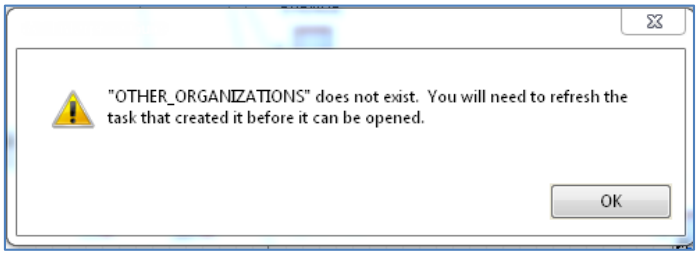
When a large and important project with a strict deadline hits your desk, it's easy to revert to those tried-and-true SAS programming techniques that have been successful for you in the past. In fact, trying to learn new techniques at such a time can prove to be distracting and a waste of time. However, the lull after a project's completion is the perfect time to reassess your approach. Once you've met your deadline, find out whether any new features—or features that are new to you—have been added to the SAS arsenal that could be of great use the next time around. Such a post-project reassessment has provided me with the opportunity to learn about several features that will prove to be extremely valuable as I rework a key project for Round 2. These features include:

- The PRESENV option and procedure
- Fuzzy matching with the COMPGED function
- The ODS POWERPOINT statement
- SAS® Enterprise Guide® enhancements, including copying and pasting process flows and the macro variable viewer

The application in question involves a great deal of validation and reconciliation to match up data from numerous regularly refreshed sources; data manipulation and transposition; and generation of standard and ad hoc reports at the detail and summary levels. I had no inkling of how dramatically this project would grow and expand in scope as I worked on it, but I did have the foresight to use SAS Enterprise Guide to keep the various and ever-increasing number of steps and processes organized. I use of the Enterprise Guide INPUT task extensively, which makes quick work of converting Excel spreadsheets, the format in which most of my data arrives, into SAS data sets. Besides the INPUT task, I mostly use code nodes, which create multiple intermediate temporary SAS data sets and permanent output data sets.

THE PRESENV OPTION AND PROCEDURE

The ad hoc report requests that I receive occasionally require data in a format that exists in one of the intermediate temporary SAS data sets that my application generates. Because temporary SAS data sets do not persist from one session of SAS Enterprise Guide to the next, absent-mindedly clicking on the icon for a temporary data set in the process flow produces this message:



In the first iteration of this project, my solution was to re-run the branch of the process flow that produced the temporary data set that I needed. But of course, in doing so, I was re-creating all of the data sets, both temporary and permanent, that are produced by the code. As my data volume has grown larger and larger over the last year, this approach has taken more and more time and was proving to be quite inefficient. Fortunately, my investigation of previously unfamiliar SAS features and options revealed that there's an easier way.

The PRESENV option records your SAS sessions so that you can restore your environment as you left it. This feature includes returning temporary SAS data sets to the WORK library, recompiling macro code, reassigning values to macro variables, reissuing global statements and global options, and more.

I include the PRESENV option at the beginning of the first code node that I run in my SAS Enterprise Guide process, which contains LIBNAME statements, macro library references, and other global statements. For example:

```
options presenv;
```

I then run the various branches of the process as needed to produce reports and perform analyses, supplement the process flows with new code, or perform maintenance. When I am ready to end my SAS Enterprise Guide session, I run another code node that contains the PRESENV procedure:

```
libname savedata 'c:\presenv\sasdata';
filename savecode 'c:\presenv\code\store.sas';
proc presenv permdir=savedata sascode=savecode
   show_comments;
```

The LIBNAME statement identifies the library in which copies of the temporary data sets, compiled macro code, macro variable values, and other such SAS content are stored. The FILENAME statement indicates where to save the code to restore the environment. The procedure uses PERMDIR= and SASCODE= options to reference the LIBREF and FILEREF previously assigned. An additional option, SHOW_COMMENTS, which can be omitted, produces some descriptive information, most notably a list of the temporary SAS data sets and compiled macro code that have been stored:

#	Name	Member Type	File Size	Last Modified
1	E_E_SORTED	DATA	851968	12/20/2016 15:51:46
2	IDENTIFY_TIERS	DATA	1703936	12/20/2016 15:51:46
3	OTHER_ORGANIZATIONS	DATA	786432	12/20/2016 15:51:46

(More rows)

9	SASGOPT	CATALOG	13312	12/20/2016 15:51:46
10	SASMACR	CATALOG	156672	12/20/2016 15:51:46

The store.sas program that I referenced in this example provides some interesting reading. For example, if you run code more than once, it shows the iterations of the code with the previous versions commented out.

```
*LIBNAME SAVEDATA 'c:\presenv\sasdata';
*FILENAME SAVECODE 'c:\presenv\code\store.sas';
LIBNAME SAVEDATA 'c:\presenv\sasdata';
FILENAME SAVECODE 'c:\presenv\code\store.sas';
```

This suggests an intriguing use of the procedure as a way to trace and debug your code!

It also shows the values assigned to macro variables in the program:

```
data _null_;
      call symput('ADDRESS','OAK AVENUE' );
      call symput('NAME','CAMDEN' );
      (more statements)
```

A look at the log after running the procedure is also instructive. It indicates that if the storage location already contains copies of the temporary data sets, the contents are first deleted before the data sets are recopied. This feature ensures that only the most current versions of the data sets are saved.

```
NOTE: Deleting SAVEDATA.E_E_SORTED (memtype=DATA).
NOTE: Deleting SAVEDATA.IDENTIFY_TIERS (memtype=DATA).
NOTE: Deleting SAVEDATA.OTHER_ORGANIZATIONS (memtype=DATA).

NOTE: Copying WORK.E_E_SORTED to SAVEDATA.E_E_SORTED (memtype=DATA).
NOTE: There were 1458 observations read from the data set WORK.E_E_SORTED.
NOTE: The data set SAVEDATA.E_E_SORTED has 1458 observations and 20 variables.
NOTE: Copying WORK.IDENTIFY_TIERS to SAVEDATA.IDENTIFY_TIERS (memtype=DATA).
NOTE: There were 1148 observations read from the data set WORK.IDENTIFY_TIERS.
NOTE: The data set SAVEDATA.IDENTIFY_TIERS has 1148 observations and 16 variables.
NOTE: Copying WORK.OTHER_ORGANIZATIONS to SAVEDATA.OTHER_ORGANIZATIONS (memtype=DATA).
NOTE: There were 1458 observations read from the data set WORK.OTHER_ORGANIZATIONS.
NOTE: The data set SAVEDATA.OTHER_ORGANIZATIONS has 1458 observations and 20 variables.
```

But the real value of the procedure isn't apparent until the session has ended and a new session is started at some future time. To restore the environment, I add a %include statement to a code node to run the store.sas program:

```
%include 'c:\presenv\code\store.sas';
```

The log shows the copy operation from the PERMDIR= location identified in the procedure back into the WORK library:

```
NOTE: Copying SAVEDATA.E_E_SORTED to WORK.E_E_SORTED (memtype=DATA).
NOTE: There were 1458 observations read from the data set SAVEDATA.E_E_SORTED.
NOTE: The data set WORK.E_E_SORTED has 1458 observations and 20 variables.
NOTE: Copying SAVEDATA.IDENTIFY_TIERS to WORK.IDENTIFY_TIERS (memtype=DATA).
NOTE: There were 1148 observations read from the data set SAVEDATA.IDENTIFY_TIERS.
NOTE: The data set WORK.IDENTIFY_TIERS has 1148 observations and 16 variables.
NOTE: Copying SAVEDATA.OTHER_ORGANIZATIONS to WORK.OTHER_ORGANIZATIONS (memtype=DATA).
NOTE: There were 1458 observations read from the data set SAVEDATA.OTHER_ORGANIZATIONS.
NOTE: The data set WORK.OTHER_ORGANIZATIONS has 1458 observations and 20 variables.
```

This means that instead of seeing the "does not exist" error that I referenced previously, I can access my temporary data sets without re-running the code to create them.

In addition, the compiled macro code is restored:

```
NOTE: Copying entry CHECKFMT.MACRO from catalog SAVEDATA.SASMACR to catalog WORK.SASMACR.
NOTE: Copying entry ECLIBASSIGN.MACRO from catalog SAVEDATA.SASMACR to catalog WORK.SASMACR.
```

And the previously defined macro variable values are retrieved. This snippet from the log shows that the values of the macro variables are the same as they were when I exited my last session:

```
24          %put macro variable name=&name;
macro variable name=CAMDEN
25          %put macro variable address=&address;
macro variable address=OAK AVENUE
```

The session has been restored to the state in which I left it, and I can resume my work where I left off.

FUZZY MATCHING WITH THE COMPGED FUNCTION

The data that I used in this project comes from different systems and sources. The main roll-up field, ORGANIZATION, is not standardized across the tables. Furthermore, in another data source provided by a third party, the organization name is entered in a freeform text field, introducing even more variation.

Some ways that ORGANIZATION varies from data source to data source are listed here:

Explanation of Variation	Examples
Name has different corporate designation	A Big Company Ltd.; A Big Company Inc.
Name has different spacing	ABigCompany; A BigCompany
Name has different punctuation	A Big Company, Ltd.; A Big Company Ltd.
Name has different capitalizations	A Big COMPany; A BIG COMPA NY
Name is misspelled	A Bigg Company
Name is abbreviated	ABC Ltd.
Name is similar to another organization's name	Big Large Company

It's important to address these variations, because SAS expects an exact match in a comparison of values. Any attempt to filter, query, summarize, or otherwise process the data will, by default, handle every variation as a separate, unrelated entry.

The SAS Data Management solution is your best option to address these types of variations and all of the other data quality issues that come up in production data preparation at the enterprise level, and in fact the next iteration of this project will likely include a rewrite using this technology. Currently, for my non-enterprise-level project, I first used a rather heavy-handed approach to reconcile these variations, IF-THEN logic. Here's an example:

```
if trim(left(organization)) in ( 'A Big Company Ltd.', 'A Big Company Inc.'
'ABIGCOMPANY','A Bigg Company') then new_organization='A Big Company Inc.';
```

It isn't hard to see that this technique will very quickly become tedious, cumbersome, and difficult to keep current. Every time a new variation of the name occurs in the data, the logic has to change to accommodate it in the code. Functions like UPCASE, INDEX, SUBSTR, SCAN, and others can help, but the task to match up the data is still daunting.

Discussions with fellow SAS programmers and some research revealed a number of approaches using functions that do fuzzy matching. (See the Appendix for sources.) After some testing, I determined that the one that worked best for me was COMPGED. This function compares two strings—in my case, the organization as it appears in the official list (the lookup table) of organizations, and the organization name as it appears in my different data sources.

According to *SAS 9.4 Language Reference by Name, Product, and Category*, COMPGED measures the generalized edit distance, which is "a measure of dissimilarity between two strings … the number of deletions, insertions, or replacements of single characters that are required to transform string-1 into string-2."

For example, let's assume that the standardized name of an organization (LOOKUP) is A Big Company Inc. Here, I've used the syntax for COMPGED without any optional arguments:

```
difference=compged(lookup,organization);
```

The following table shows the values that are returned by COMPGED when the variations of ORGANIZATION are compared to it:

organization	difference
A Big Company Ltd.	300
A Big Company Inc.	0
ABigCompany	210
A BigCompany	200
A Big Company, Ltd.	330
A Big Company Ltd.	300
A Big COMPany	490
A BIG COMPANY	990
A Bigg Company	210
ABC Ltd.	900
A Big Company	190
Big Large Company	1140

Because case, spacing, and punctuation play a significant role in increasing the number of steps to get a match, let's try the function with the option to ignore case ('i'), and the COMPRESS function to remove embedded blank spaces and punctuation:

```
difference2=compged(compress(lookup,' ,.'),compress(organization,'
,.'),'i');
```

In most cases, these minor adjustments decreased the number of steps, and in some cases considerably.

organization	difference	difference2
A Big Company Ltd.	300	300
A Big Company Inc.	0	0
ABigCompany	210	150
A BigCompany	200	150
A Big Company, Ltd.	330	300
A Big Company Ltd.	300	300
A Big COMPany	490	150
A BIG COMPANY	990	150
A Bigg Company	210	220
ABC Ltd.	900	800
A Big Company	190	150
Big Large Company	1140	1000

In my application, the lookup table has nearly 800 entries and the number of rows of data to match varies from several to many thousands, depending on the source. In addition to knowing that I can be reasonably sure which names are actually just variants of the correct name using COMPGED, I can also see values that appear to be anomalies, such as ABC Ltd. and Big Large Company. Accordingly, setting a cutoff for a matching value, such as 300 in this example, helps to identify any organization names that are more than 300 steps from the lookup value and that require a separate review.

Using fuzzy matching requires a fair amount of experimentation; you have to determine how to get the closest match without inadvertently grouping values that seem similar but that are truly different values. But the results are very gratifying, and the technique is simpler than some others that you might be using.

THE ODS POWERPOINT STATEMENT

I am often asked to report on my findings, which I usually do in Microsoft® PowerPoint® format. For example, I use the FREQUENCY task in SAS Enterprise Guide to get current counts of the number of organizations in each program tier, Silver, Gold, and Platinum. It isn't that onerous to cut and paste the FREQUENCY procedure output into a PowerPoint slide, but of course doing it this way means that the slide must be manually updated whenever the counts change. On the other hand, using the ODS POWERPOINT statement in a program means that this reporting step can be incorporated into the overall process flow, and the slide is updated when the data is updated, automatically.

The ODS POWERPOINT statement joins a lengthy list of ODS (Output Delivery System) features that have been available for quite a while. The general syntax for the statement, as documented in the ODS User's Guide, is as follows:

```
ODS POWERPOINT <option(s)>;
```

Options include information about where to save the PowerPoint slide, the style template to use, the layout format, and more. With the third maintenance release of SAS 9.4, extensive additional flexibility was added to incorporate the advance, repeat, sound, and transition features that we're familiar with when we build slides in the PowerPoint environment.

My ODS statement looks like this:

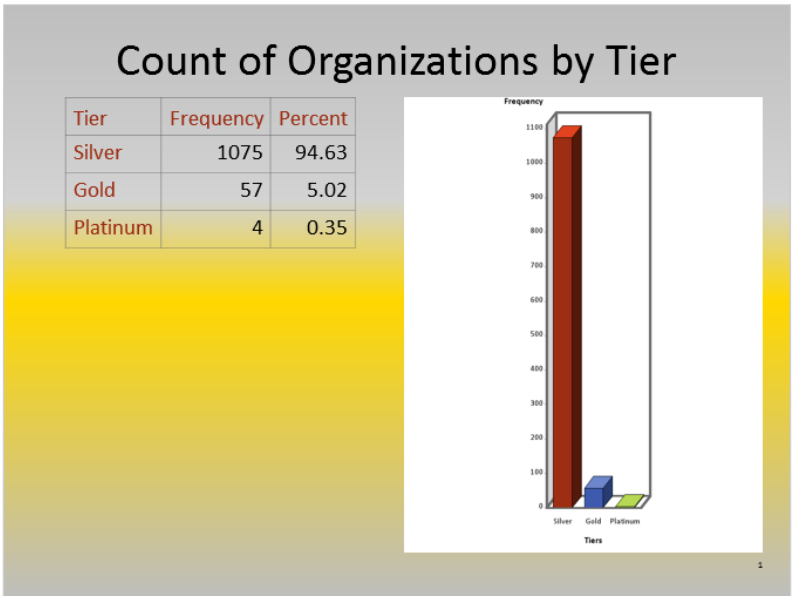
```
ods powerpoint file="c:\SGF 2017\Sample1.pptx"  style=styles.example1
layout=twocontent nogtitle nogfootnote;
```

The FILE= option indicates where I'd like to store the PowerPoint slide. The STYLE= option refers to a custom stylesheet that I created. You can learn how to create a stylesheet using PROC TEMPLATE from the ODS documentation. Because I'd like to display a frequency count and a bar chart side-by-side, I use the LAYOUT=TWOCONTENT option. And to suppress default titles and footnotes, I use the NOGTITLE and NOGFOOTNOTE options.

Most of the rest of the program consists of the SAS code to produce the frequency and bar chart reports.

```
ods html close;
ods escapechar = "^";
title1 'Count of Organizations by Tier';
options nodate;

ods powerpoint file="c:\SGF 2017\Sample1.pptx"  style=styles.example1
layout=twocontent nogtitle nogfootnote;


proc freq data=sgf.organizations (where=(tier in ('Silver' 'Gold'
'Platinum'))) notitle order=freq;
   tables tier / nocum;
run;

axis1 style=1 width=4  minor=none label=("Frequency");
axis2 style=1 width=4 label=("Tiers");

proc gchart data=sgf.count 31dec;
vbar3d  tier / sumvar=count shape=block frame discrete type=sum nolegend
descending coutline=black raxis=axis1 maxis=axis2 patternid=midpoint;
run; quit;

ods _all_ close;
```

The resulting PowerPoint slide looks like this:

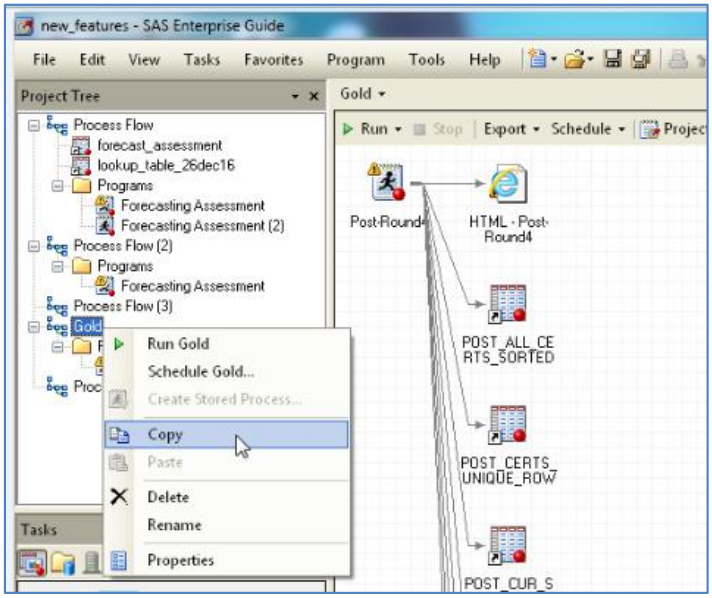


SAS ENTERPRISE GUIDE ENHANCEMENTS

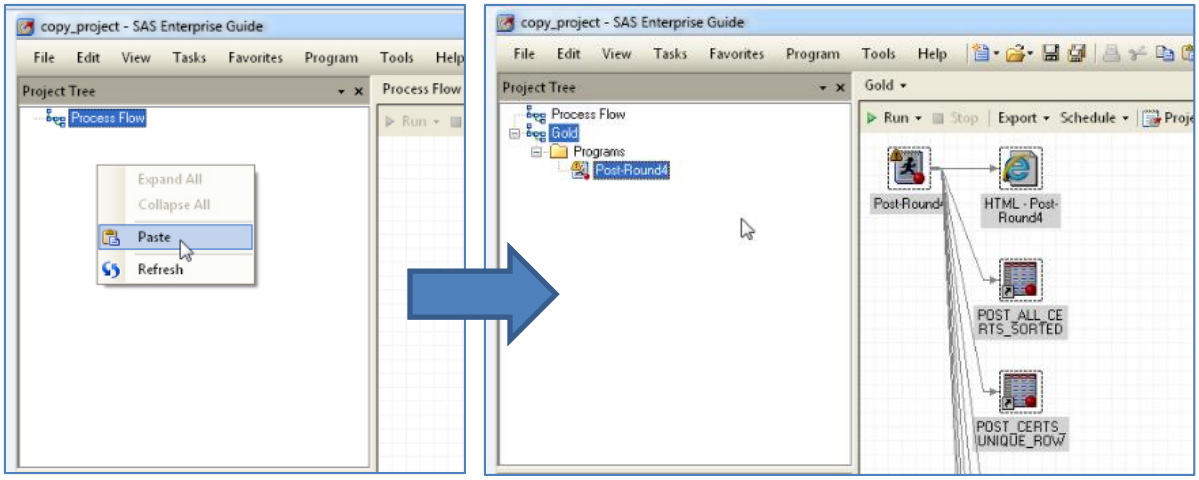
Every new release of SAS Enterprise Guide comes with a list of enhancements to try out and incorporate into your routine. There are many enhancements that I've started to make use of, but I'll focus on just a few of these: copying and pasting process flows and the macro variable viewer (both introduced in version 7.1), and—hot off the presses–the DATA step debugger, an enhancement in version 7.13.

The ability to copy and paste objects, branches, and now entire process flows from one project to another is extremely handy and very easy to accomplish.

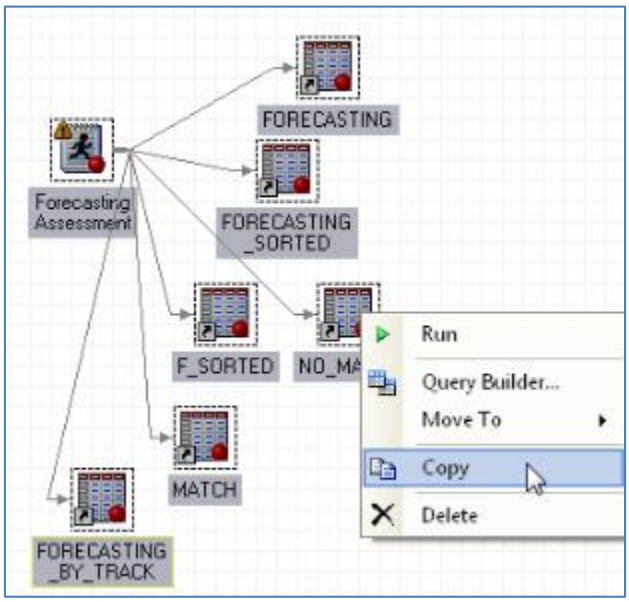
To copy an entire process flow, find it listed in the source project tree, right-click it, and select **Copy**. In my example, I'm making a copy of the Gold process flow:



Then, in the destination project tree, right-click and select **Paste**. The Gold process flow is copied to the new project.

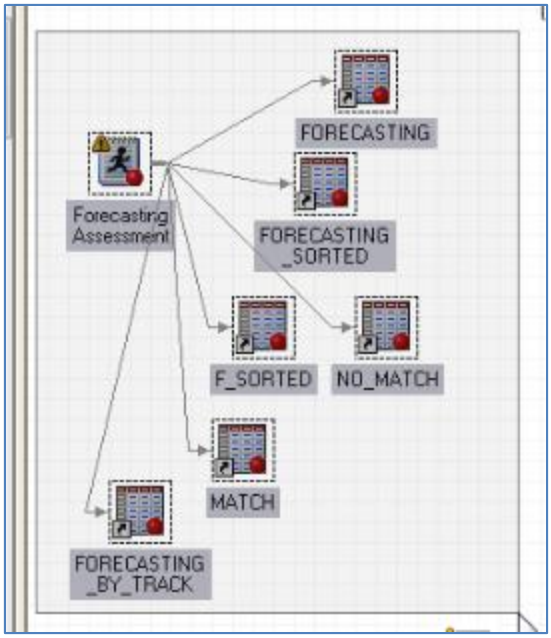


You have a couple of options to copy a single branch from one project to another project or from one process flow to another process flow. Holding down the **Ctrl** key, you can click to activate each of the nodes that you want to copy (causing dotted lines to appear around each node), right-click, and select **Copy**.



Then right-click at the destination location and select **Paste**.

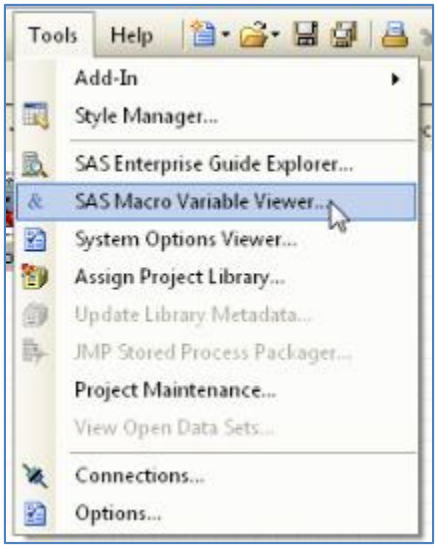
Alternatively, you can draw a selection box around the branch that you want to copy using your left mouse button, which has the effect of activating each node that you want to copy.



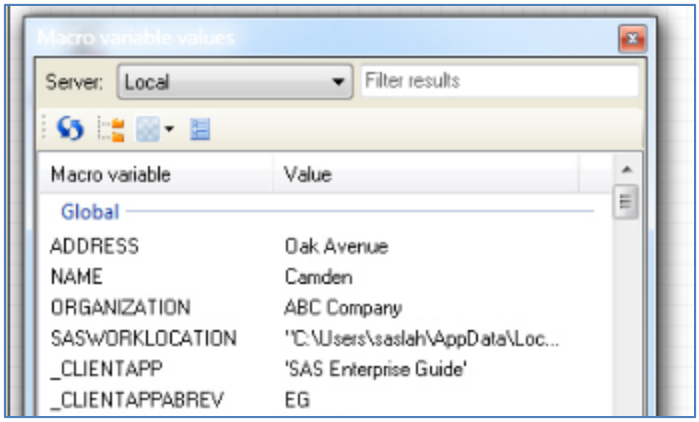
Right-click and select **Copy**, and then right-click again to paste the branch into its new process flow or project.

These techniques make it very easy to reuse portions of your project in new applications, test new features, create a backup, and so on.

Another useful feature of SAS Enterprise Guide that I've started to use to my advantage is the macro variable viewer. Access it by clicking the Tools menu and selecting **SAS Macro Variable Viewer**.



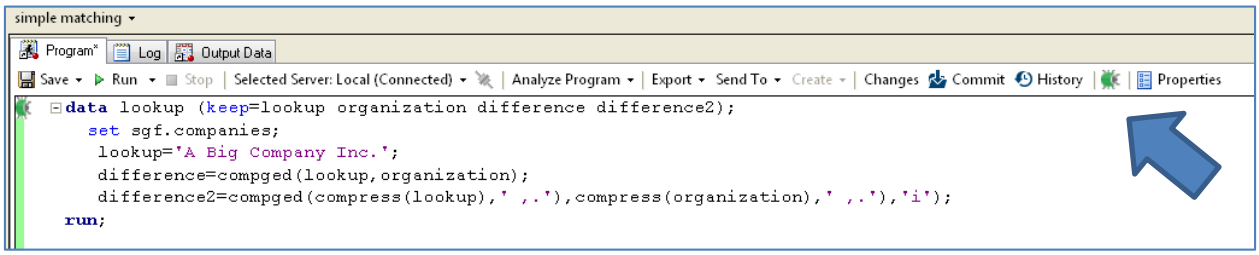
This viewer shows you the current value of all user-defined and automatic macro variables. Also, because macro variables are used as prompts in Enterprise Guide, the viewer is invaluable for debugging your process flows and stored processes, confirming that macro variables are global in scope, and validating that values for the macro variables are being assigned as expected.



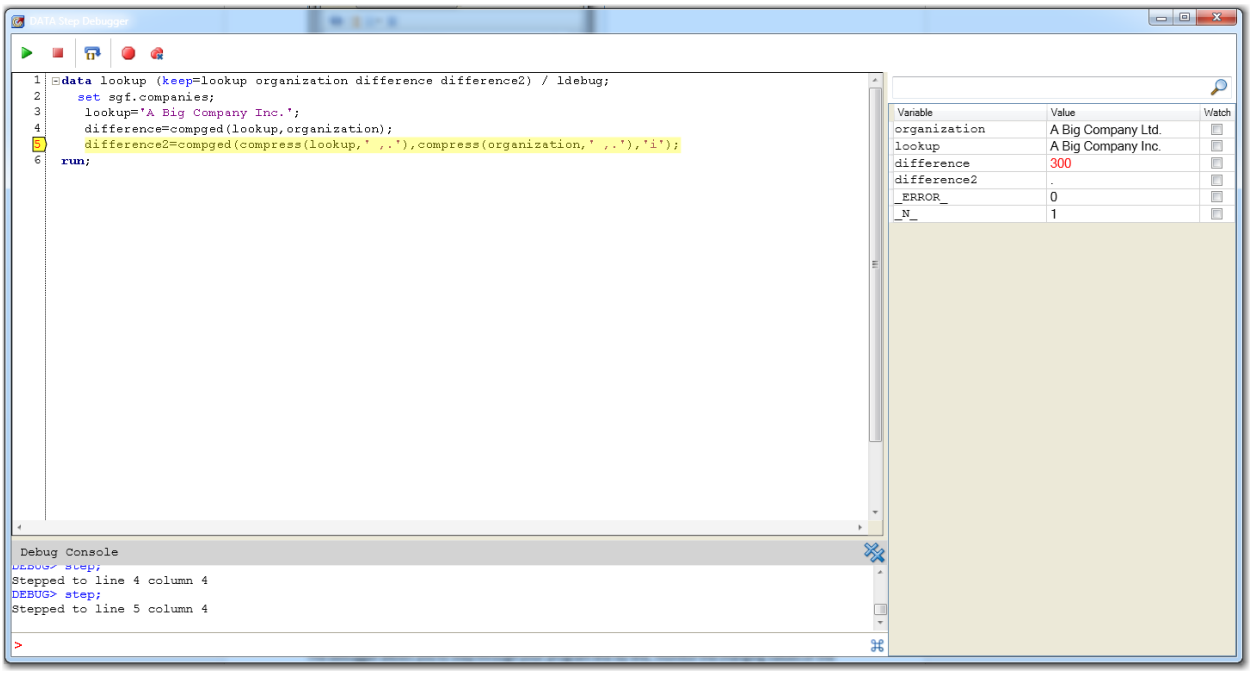
As discussed earlier in this paper, if it is advantageous to preserve macro variable values from one session of SAS Enterprise Guide to another, the PRESENV option and procedure can perform this work for you.

Finally, a brand-new enhancement to SAS Enterprise Guide is the introduction of the DATA step debugger. The debugger allows you to step through your DATA step code line-by-line to monitor the changing values of the variables, see the logic of the statements play out, and watch the branching and looping caused by IF-THEN-ELSE and DO loops. It does not debug procedure step code.

You can toggle the debugger on and off by clicking the "bug" toolbar button while viewing your program in the Program Editor window. The program in the window shown here is the one that I used to produce the COMPGED function results that I discussed earlier in this paper.



Once the debugger has been enabled, clicking on the bug on the left opens the DATA step debugger window with its various components. If you've used the DATA step debugger in SAS Display Manager, you'll notice some additional graphical features like the variable window on the right side, and that instead of typing instructions, you can click **F10** to step forward and can click other buttons to perform additional tasks. Or, if you prefer, you can enter the commands at the red "greater than" prompt at the bottom of the window. If you've never used the DATA step debugger before, you are likely to be delighted by this enhancement to SAS Enterprise Guide. In the image below, the line that is about to be executed is highlighted in yellow. As variables are assigned new values, they appear in red, such as the variable DIFFERENCE, newly assigned the value 300.



More advanced features include skipping ahead past sections of code that are working properly to get to the troublesome lines using breakpoints; calculating expressions and seeing the result; jumping to a certain line in the program; and much more. See Chris Hemedinger's blog post and the SAS Enterprise Guide documentation for more information.

CONCLUSION

With every new version, SAS adds many new features to its products and solutions that can provide significant advantages. And there are many ways to learn about these new features: they are often spotlighted in conference papers like this one; they are addressed in blogs and in the SAS Communities; they are the subject of SAS Education classes; and they are documented in the What's New editions that accompany each SAS release. It is well worth the time to investigate these new features, because you might find, as I did, some features that will prove to be very useful and big time-savers for you.

ACKNOWLEDGMENTS

Many thanks go to Maura Stokes (Senior Director, Advanced Analytics R&D, SAS), Amy Peters (Principal Product Manager, SAS), Kathy Wisniewski (Applications Analyst, the University of North Carolina at Chapel Hill), and Merry Rabb (Manager, Public Health, RTI International) for their great ideas and excellent counsel. Conversations with them are always highly inspirational!

REFERENCES

PRESENV OPTION AND PROCEDURE

Fredlund, Keith, Grand Valley State University, and Wai, Thinzar, Grand Valley State University, "*Painless Extraction: Options and Macros with PROC PRESENV*," Midwest SAS User Group *Proceedings* 2016, http://www.mwsug.org/proceedings/2016/SA/MWSUG-2016-SA06.pdf

Squillace, Jan, SAS, "Flexibility by Design: A Look at New and Updated System Options in SAS® 9.4," http://support.sas.com/resources/papers/Flexibility_by_Design.pdf

Base SAS® Procedures Guide, Sixth Edition

FUZZY MATCHING

Dunn, Toby, Dunn Consulting, "Getting the Warm and Fuzzy Feeling with Inexact Matching," SAS Global Forum Proceedings 2014, http://support.sas.com/resources/papers/proceedings14/1316-2014.pdf

Foley, Malachy, University of North Carolina at Chapel Hill, "Fuzzy Merges: Examples and Techniques," SAS Users Group International 24 Proceedings, http://www2.sas.com/proceedings/sugi24/Advtutor/p46-24.pdf

Queen, Mary Kathryn, SAS Institute Inc., "How to Find Your Perfect Match Using SAS® Data Management," SAS Global Forum Proceedings 2016, http://support.sas.com/resources/papers/proceedings16/SAS3560-2016.pdf

Kevin Russell, SAS, "how to perform a fuzzy match using SAS functions," SAS Users blog, January 27, 2015, http://blogs.sas.com/content/sgf/2015/01/27/how-to-perform-a-fuzzy-match-using-sas-functions/

Schreier, Howard, U.S. Department of Commerce, "*Using Edit-Distance Functions to Identify 'Similar' E-mail Addresses*," SAS Users Group International 29 Proceedings, http://www2.sas.com/proceedings/sugi29/073-29.pdf

Staum, Paulette, Paul Waldron Consulting, "Fuzzy Matching Using the COMPGED Function," NESUG 2007 *Proceedings*, http://www.lexjansen.com/nesug/nesug07/ap/ap23.pdf

Syphus, Stacey, SAS, "What's your favorite function in SAS?" SAS Learning Post blog, September 22, 2016, http://blogs.sas.com/content/sastraining/2016/09/22/whats-favorite-function-sas/

Additional references for fuzzy matching are listed on the SAS Community wiki, at http://www.sascommunity.org/wiki/Fuzzy_Matching

SAS® 9.4 Language Elements by Name, Product, and Category

ODS POWERPOINT STATEMENT

Hunter, Tim, SAS, "A First Look at the ODS Destination for PowerPoint," SAS Global Forum 2013 *Proceedings*, https://support.sas.com/resources/papers/proceedings13/041-2013.pdf

SAS 9.4 Output Delivery System: User's Guide, Fifth Edition

SAS ENTERPRISE GUIDE ENHANCEMENTS, INCLUDING DATA STEP DEBUGGER

Hemedinger, Chris, SAS, "*Using the DATA step debugger in SAS Enterprise Guide*," The SAS Dummy blog, http://blogs.sas.com/content/sasdummy/2016/11/30/data-step-debugger-sas-eg/?utm_source=feedburner&utm_medium=feed&utm_campaign=Feed%3A+ASasBlogForTheRestOfUs+%28The+SAS+Dummy%29

About SAS Enterprise Guide, accessed via Help drop-down menus within the product user interface

CONTACT INFORMATION

Lisa Horwitz
SAS Institute Inc.
SAS Campus Drive
Cary, NC 27513
Email: Lisa.Horwitz@sas.com