

## Enterprise Guide for New and Experienced SAS Users

SAS Enterprise Guide (EG) has revolutionized the way SAS is used to process, model, and visualize data. This presentation will look at the advantages and disadvantages of EG from the perspective of both a user new to the SAS environment and an experienced SAS programmer. We will cover tips for using EG more efficiently, how EG leads to better project documentation, and when to not use EG. Data used to illustrate the application of EG will be taken from actual educational research projects. We used version 4.3 to develop this paper.

We were coworkers at an educational assessment company. One of us was new to SAS and found the EG interface to be more intuitive for starting with SAS. The other had been using SAS extensively for eight years but was a novice to EG. Join us as we provide tips, praise, and complaints from our experience with this tool.

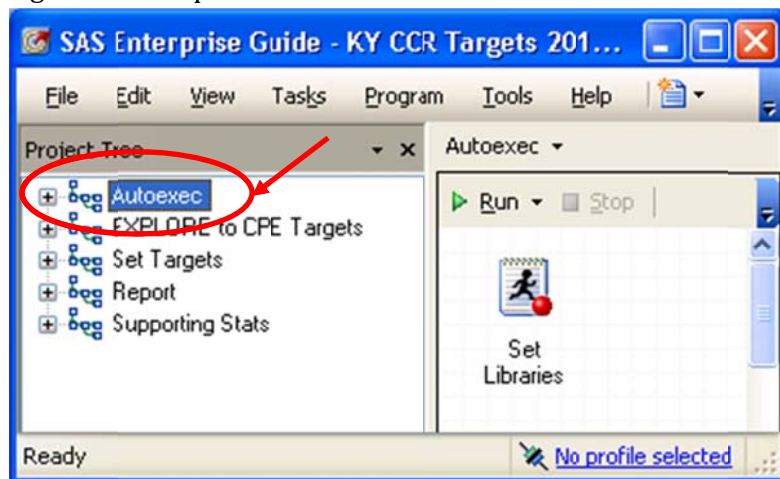
### Tips for using EG

#### Autoexec process flow

With the traditional SAS interface, a SAS program called autoexec.sas is run each time SAS is opened. The autoexec file might be used to automatically set up libraries, create formats, and define macros.

Enterprise Guide offers similar functionality through the use of an autoexec process flow. Simply create a process flow in your project and rename it “autoexec” (Figure 1). Fill the autoexec process flow with any Task, SAS program, etc. that you would any other process flow. We like to put it at the top of the process flows to remind us that these run first. By default Enterprise Guide will prompt you to run the autoexec process flow each time it is opened.

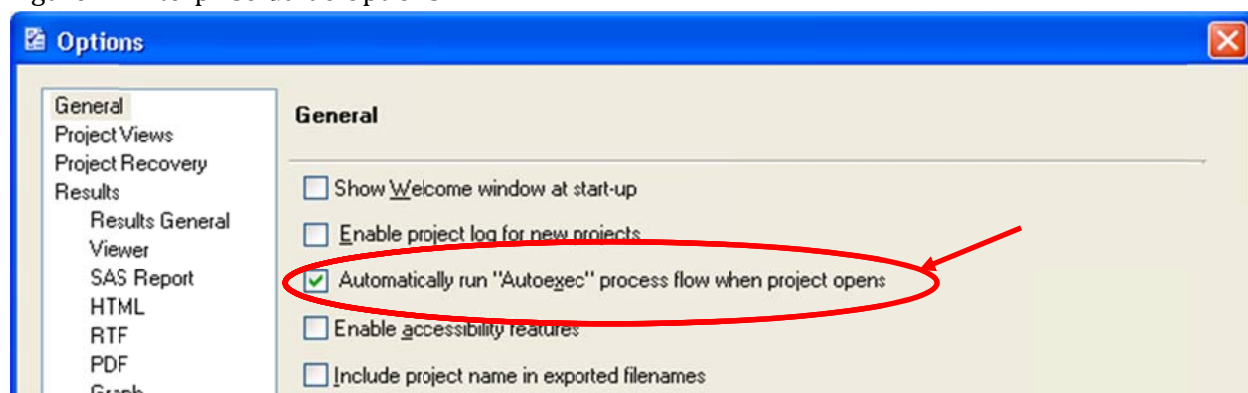
Figure 1: Example of Autoexec Process Flow



Once you get annoyed with having to confirm to run the autoexec process flow each time, you will want to turn on the option that automatically runs the autoexec. Simply go to the Tools menu and

select Options. From the General option on the left-hand menu select the 3<sup>rd</sup> checkbox “Automatically run “Autoexec” process flow when project opens” (Figure 2).

Figure 2: Enterprise Guide Options



### Borrow code from EG

One development paradigm that we have found useful in EG is to create basic tasks in EG, borrow the SAS code, and make expansions to extend the task to additional options not available in the point-and-click interface. For example, the Summary Tables Wizard Task is the EG interface to PROC TABULATE. We find the wizard approach to be a more intuitive than coding PROC TABULATE directly, but there are some nice formats that are not available in the wizard. We applied the Summary Tables Wizard to the SASHELP.CARS data to produce Table 1.

Table 1: Default Table produced by Summary Tables Task

	MSRP	Horsepower
	Average	Average
<b>Origin</b>		
Asia	24741.32	190.70
Europe	48349.80	251.89
USA	28377.44	212.82
<b>Total</b>	32774.86	215.89

We would like to format each average in a different way, but EG does not have an option for this. Rather than write the code entirely from hand, we borrow from what has already been written in EG. View the Submitted Code (Figure 3).

Next, copy the code to the clipboard. Create a New Program, and paste the code into the new program adding the formatting statements into PROC TABULATE (Figure 4). The reformatted table is shown as Table 2.

### Clean up unneeded datasets at end of program

Because EG visually represents all of the datasets that a task creates, it is good practice to clean up unneeded datasets at the end of SAS Programs. This issue existed in the traditional SAS interface, but somehow it didn’t affect our mental state when it was behind the scenes in the WORK library. We have taken to adding a simple PROC DATASETS step at the end of program code so that the

visual representation of the project only shows those datasets needed for subsequent steps (Figure 5).

Figure 3: Submitted Code for Summary Tables Task

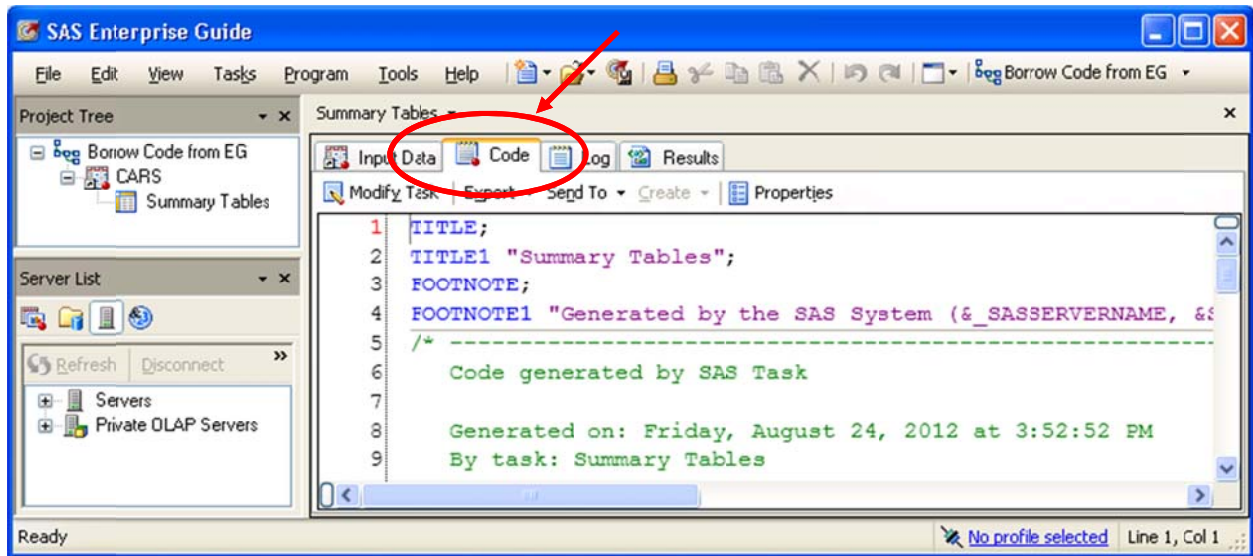


Figure 4: PROC TABULATE Code with Additional Formatting Statements

```

19 PROC TABULATE
20 DATA=SASHELP.CARS
21
22 ;
23
24 VAR MSRP Horsepower;
25 CLASS Origin / ORDER=UNFORMATTED MISSING;
26 TABLE
27     /* ROW Statement */
28     Origin
29     all = 'Total' ,
30     /* COLUMN Statement */
31     (MSRP * Mean={LABEL="Average" * f=dollar6.0
32     Horsepower * Mean={LABEL="Average" * f=3.0 )
33 ;
34
35 RUN;

```

A red circle highlights the formatting statements in the PROC TABULATE code, and a red arrow points to it from the right side of the window.

Table 2: Reformatted PROC TABULATE Table

	MSRP	Horsepower
	Average	Average
Origin		
Asia	\$24741	191
Europe	\$48350	252
USA	\$28377	213
Total	\$32775	216

Figure 5: PROC DATASETS Code to Clean Up Unneeded Datasets

```
39 proc datasets nolist lib=work ;  
40     delete est1 targets ;  
41 run; quit;
```

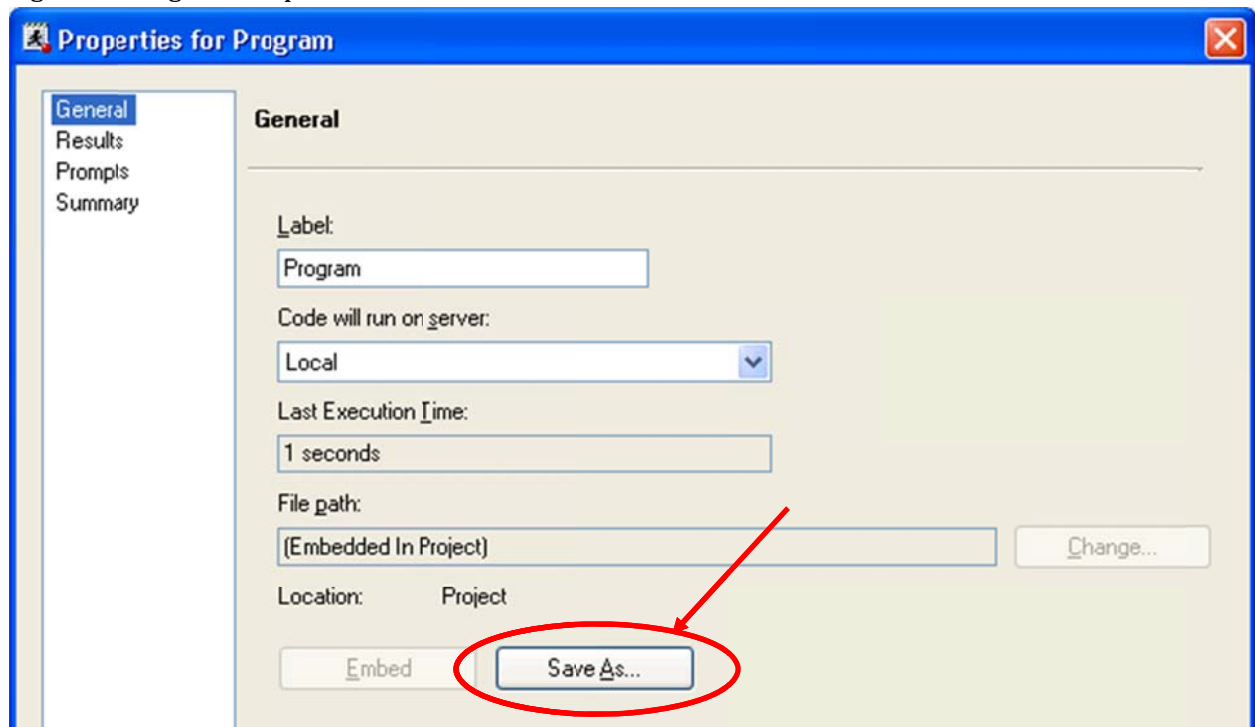
The NOLIST option prevents the entire list of objects in the library from being printed to the log. The LIB= option points to the library where the datasets reside. The DELETE statement allows a listing of the datasets that should be deleted to be made.

### For tasks that take a long time, Save Program as... and run in batch

One of the nice features about EG is the ability to set a program running while still working on another one in the same session. There seems to be a limit, though, to how much activity EG can handle. Our experience has been that processing intensive tasks can bog down EG so that no other work can get done. In this situation we have found it better to save the program externally and run it in batch mode so that other work within the project can continue.

First open the Program Properties, click Save As..., and provide a program name and path (Figure 6). This saves the program externally as the traditional .sas file type. The .sas file can then be run in batch like any other.

Figure 6: Program Properties



Some tips to remember:

- The external program will not have access to libraries and temporary datasets that have been created within the EG session. You might have to redefine libraries in the program and save any required input datasets to a permanent library.

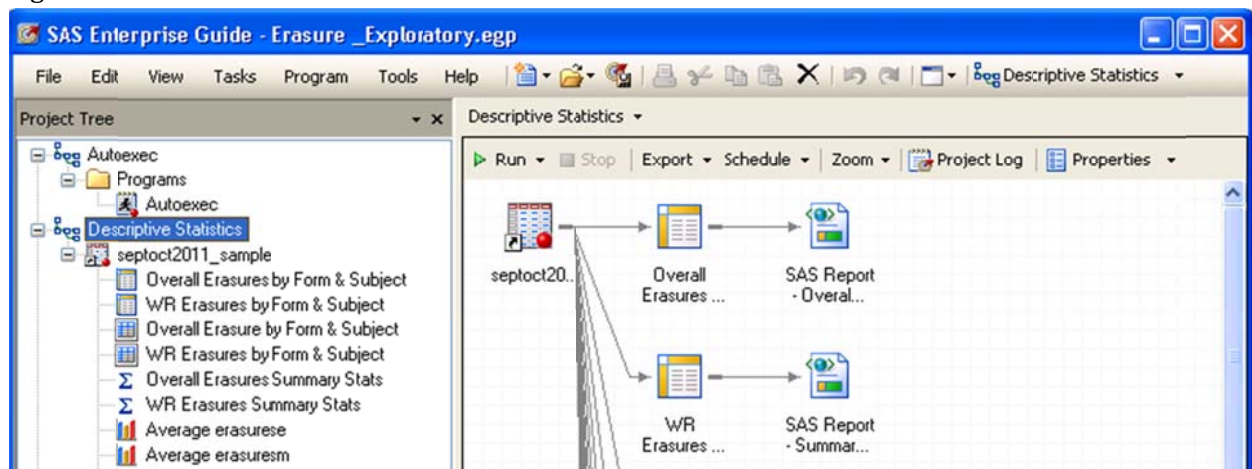
- EG won't automatically display the permanent output datasets from the program. You might have to manually open them and link the program to them to have the flow documented.

## Better Documentation

### Visual representation of process flow.

One useful aspect of EG is that it allows users to effectively track their work. With this idea, EG utilizes the process flow to provide an overall picture of how everything fits together (Figure 7). Generally, a process flow displays data sets, tasks, reports, and results, as well as the relationships between these objects. Many process flows can exist within a project, but it is recommended to separate them by activity for easier access and organization (e.g., autoexec, descriptive statistics, and models). EG users often review process flows to document their progress throughout the project.

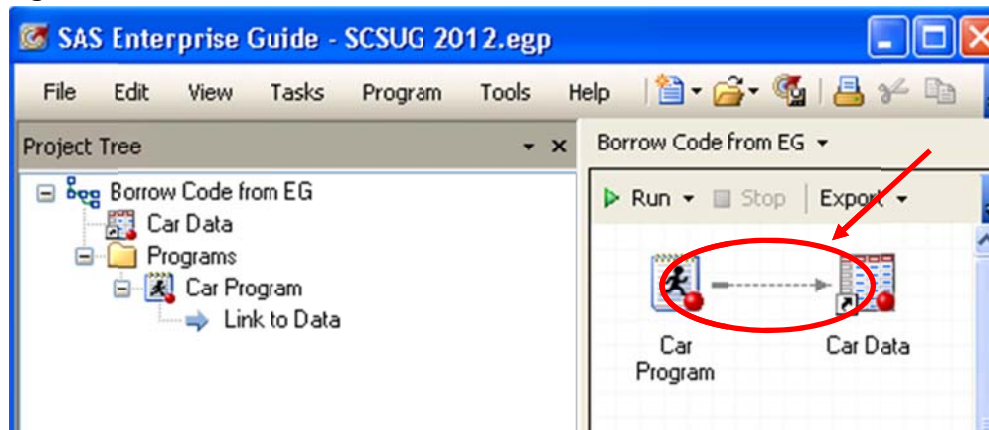
Figure 7: Process Flow



### Creating manual links

As mentioned in the previous segment, a process flow presents relationships between objects inside a project. These relationships are shown by arrows connecting one object to another (e.g., program to dataset, or graph to report). Sometimes, these relationships are implicit as EG automatically identifies these relationships and display them through links in the process flow, without control from the user. However, EG also allows users to create their own links among objects (Figure 8). When EG does the linking on its own, the link appears as a solid line. When it is user-defined, the link appears as a dashed line. Whether it is automatic or manual, links help to control program execution. To create the user-defined link below, we right clicked Car Program, selected Link Program to, and selected Car Data.

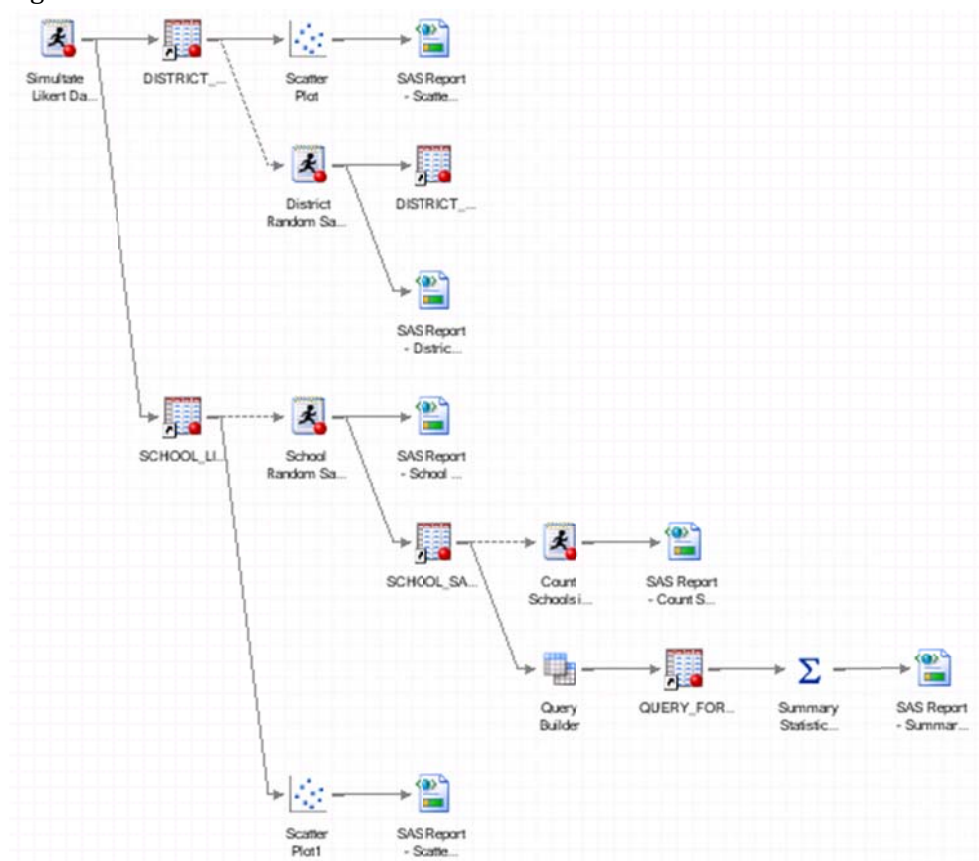
Figure 8: Manual Link



## Many smaller programs

Another strength of EG is its ability to divide large programming tasks into smaller, more manageable programs. In the traditional SAS interface, we tended to write long programs consisting of hundreds of lines of code. In EG, our development process tends to break up the code into smaller programs with more visual representations of the links between code pieces (Figure 9). We believe this provides better documentation and allows new analysts to more quickly pick up maintenance on projects.

Figure 9: Smaller Code Chunks in EG

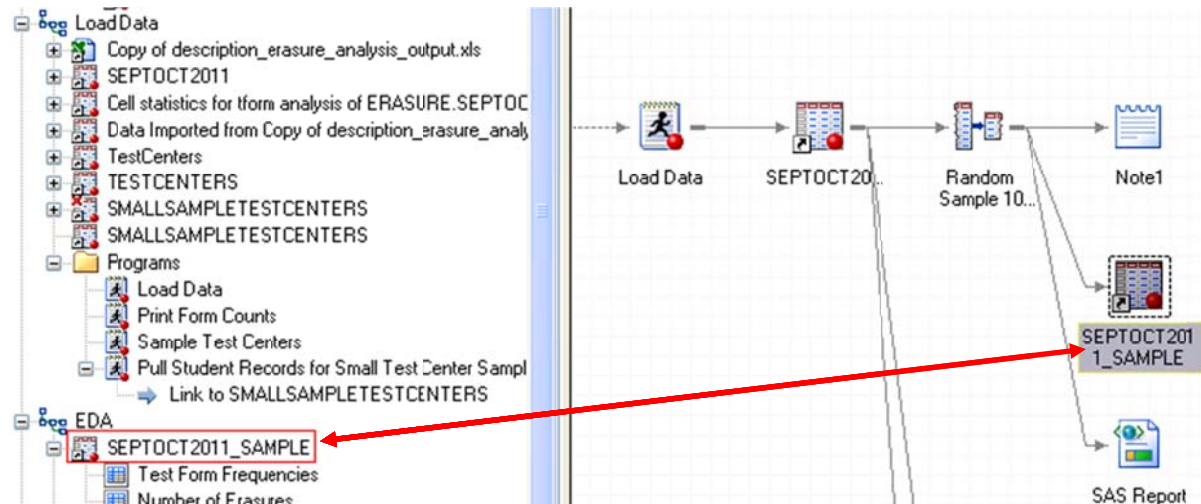




### Many process flows saving permanent version between.

Often times, users may want to save permanent data between multiple process flows (Figure 10). This minimizes the effort of having to reload temporary data each time EG is run. If not specified, EG automatically saves datasets in the WORK library where they will be held until the project is closed. To save a permanent dataset, simply create a user-defined library and assign your dataset to that specific library. It's handy to save a dataset as permanent if you plan to use it throughout your project. The permanent dataset can be transferred from one process flow to another.

Figure 10: Permanent SAS Dataset Saved as Output from One Process Flow and Used as Input in Another



### **Determine which tasks are easier to build interactively and which tasks are best coded manually.**

As we have worked with Enterprise Guide, we have developed certain processes that we prefer to perform through the point-and-click interface and others that we prefer to code directly.

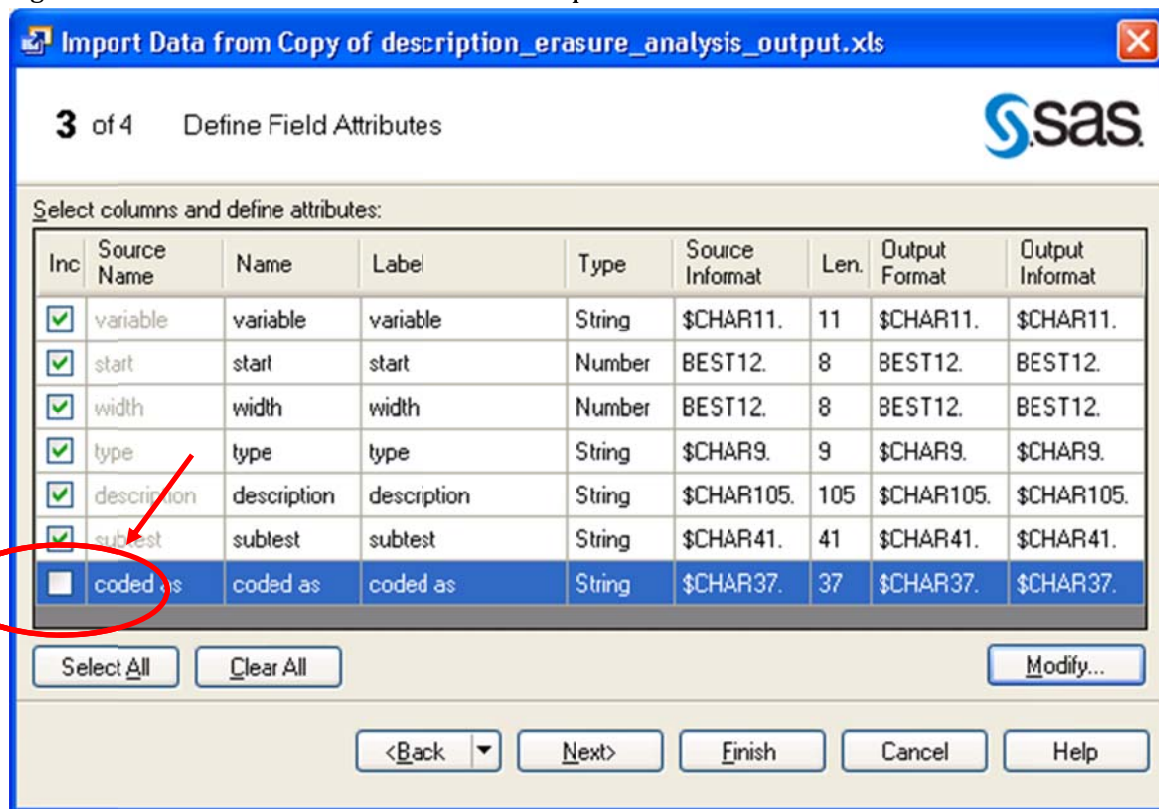
#### Better built interactively in EG

Some of the activities we prefer to use the point-and-click interface for include data importing, dataset merging, and table creation.

#### *Data Import*

For most standard Excel and delimited files, the data import in EG works well and saves us from having to remember all of the options for the INFILE statement. From the File menu choose Import Data and browse to the file you wish to import. A favorite screen in the import process is the Define Field Attributes where we can define which variables to import, what to name them, informats and formats on one screen (Figure 11). In this example, we have chosen not to import the last variable by unchecking the box in the Inc column.

Figure 11: Define Field Attributes for Data Import



### Query Builder

For merging data by PROC SQL, we have found the Query Builder Task in EG to be quite useful. Multiple tables can be imported and only chosen fields kept in the merged dataset (Figure 12).

Figure 12: Select Data Tab in Query Builder

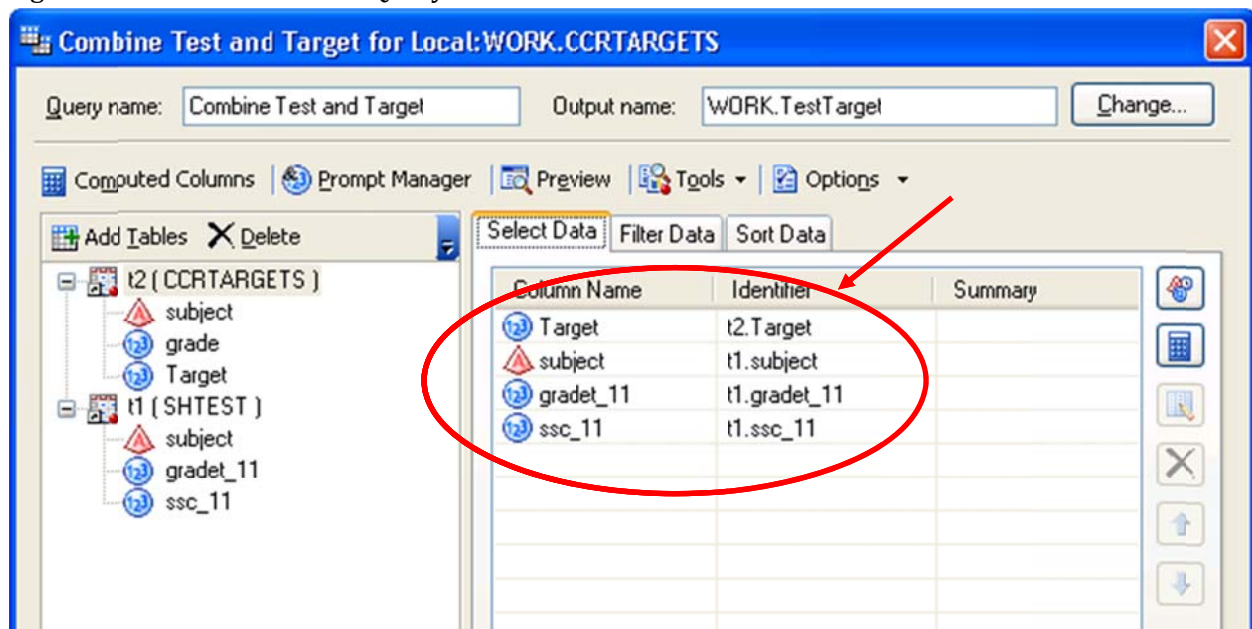
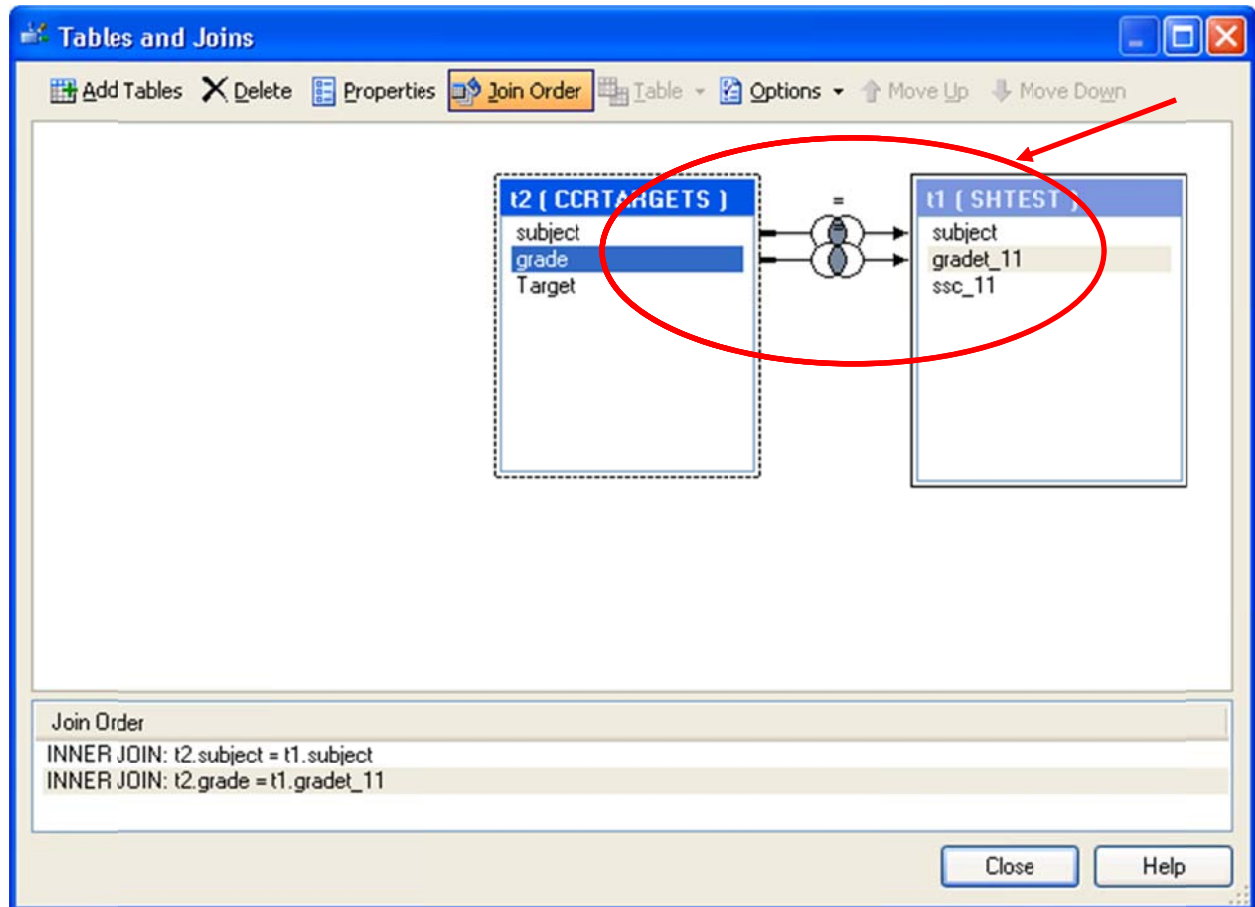




Table joins can be managed visually in the Query Builder Task (Figure 13).

Figure 13: Table Joins in Query Builder Task



### ***Proc Tabulate***

Since the goal of PROC TABULATE is to produce a visually informative table, building the table graphically is more convenient. The EG task for this is Summary Tables. Some of the options available include the Select analysis variables and statistics screen where you can choose the analysis variables, what statistics to build off of them and where to place them in the table (Figure 14). These options add the analysis variables to the VAR statement and place the requested statistics in the appropriate places in the TABLE statement. The Preview portion of the dialog allows you to visualize what the completed table will look like.

You may also choose and arrange classification variables (Figure 15). These options build the CLASS statement and place the classes in the appropriate places in the TABLE statement. Once again the Preview is invaluable in setting up the table as you wish.

### **Better coded manually**

We have found some tasks that are still easier to code manually.

Figure 14: Select Analysis Variables and Statistics in Summary Tables Task

**Summary Tables for Local:SASHELP.CARS**

2 of 6 Select analysis variables and statistics

Analysis variables:

Variable	Statistic
MSRP	Average
Horsepower	Average

Preview:

		MSRP	Horsepower
		Average	Average
Origin		999	999
		999	999
Total		999	999

Analysis variable labels: in columns

Statistics labels: in columns

Select table format

Browse...

Figure 15: Select Classification Variables in Summary Tables Task

**Summary Tables for Local:SASHELP.CARS**

3 of 6 Select classification variables

Columns:

(Optional)  
Click Add to insert a column variable.

Rows:

Origin

Preview:

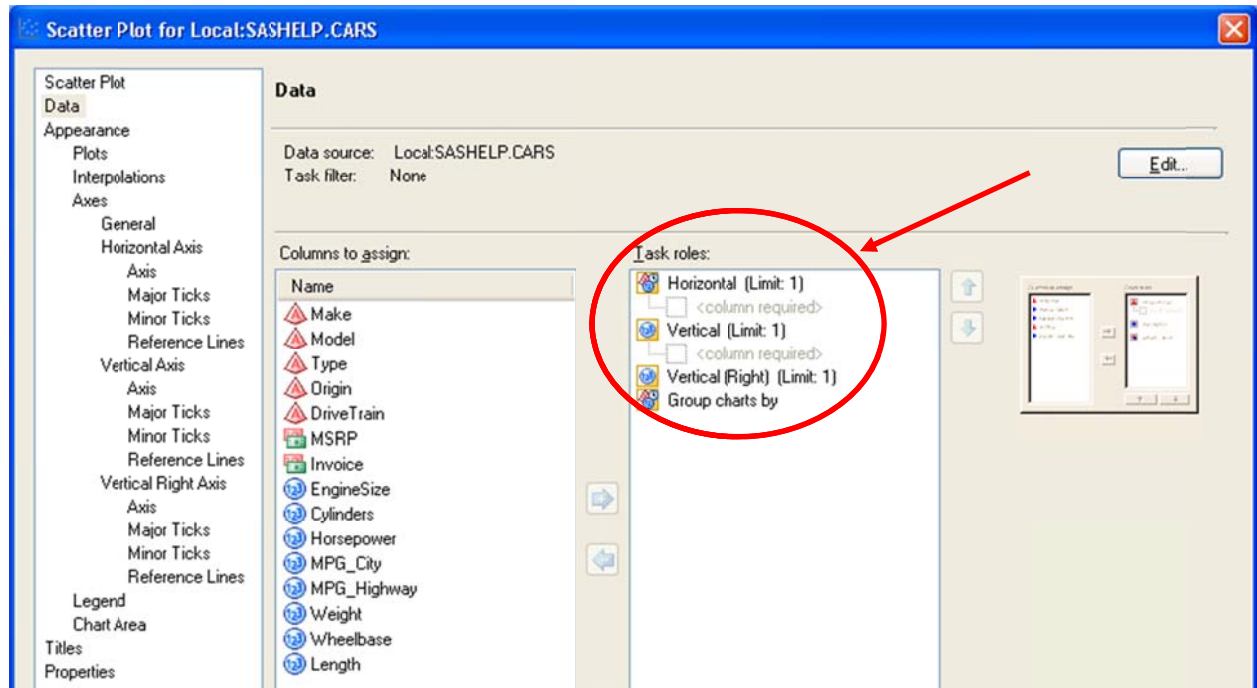
		MSRP	Horsepower
		Average	Average
Origin		999	999
		999	999
Total		999	999

## Graphics

Although EG documentation touts the inclusion of an interface to graphics, we have found those graphics tasks provided to be too simple for most of our work. As an example take the Scatter Plot task (Figure 16). You are limited to a single Horizontal variable and a single Vertical variable. We

happen to know that the PLOT statement in PROC GPLOT can handle multiple variables and produce multiple scatter plots so this limitation in the EG task seems puzzling. Another limitation of the EG interface to graphics is the lack of support for the new SG Procedures for making statistical graphics.

Figure 16: Example of Limitation in EG Interface to SAS/Graph



### ***Complex data step manipulations***

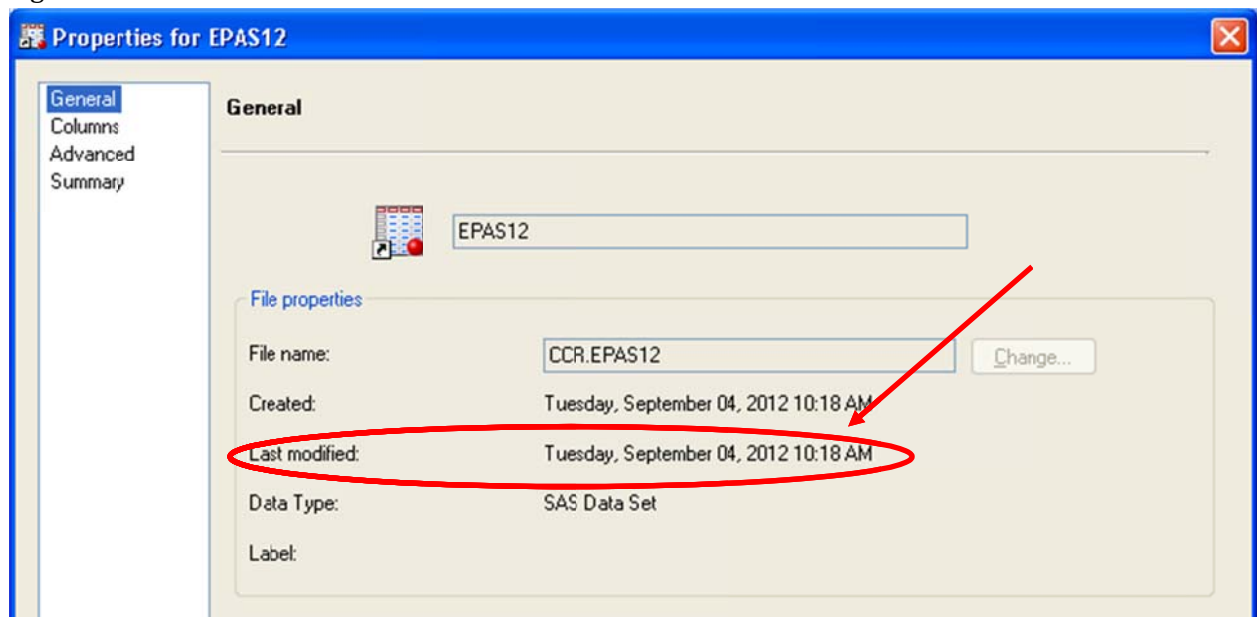
For all but the simplest data manipulations, we still prefer to code directly rather than using the EG interface. EG provides Tasks for various processes like sorting and transposing data which call PROC SORT and TRANSPOSE, respectively. However, the rich and efficient data manipulations available in the DATA Step are still the domain of the experienced SAS coder.

## **EG Issues**

### **Dataset info not updated in interface**

You can use and display permanent SAS datasets as part of your project. We often need to check that datasets have been updated in the correct sequence. We expected that you could request the last modified time for the permanent data set by viewing the dataset properties in EG (Figure 17). However, the time displayed there is the last time that the dataset was modified within the EG session. If you have not modified the dataset within the current EG session, the time the EG session started is displayed. You still need to use Windows Explorer to view the actual modification time for the dataset.

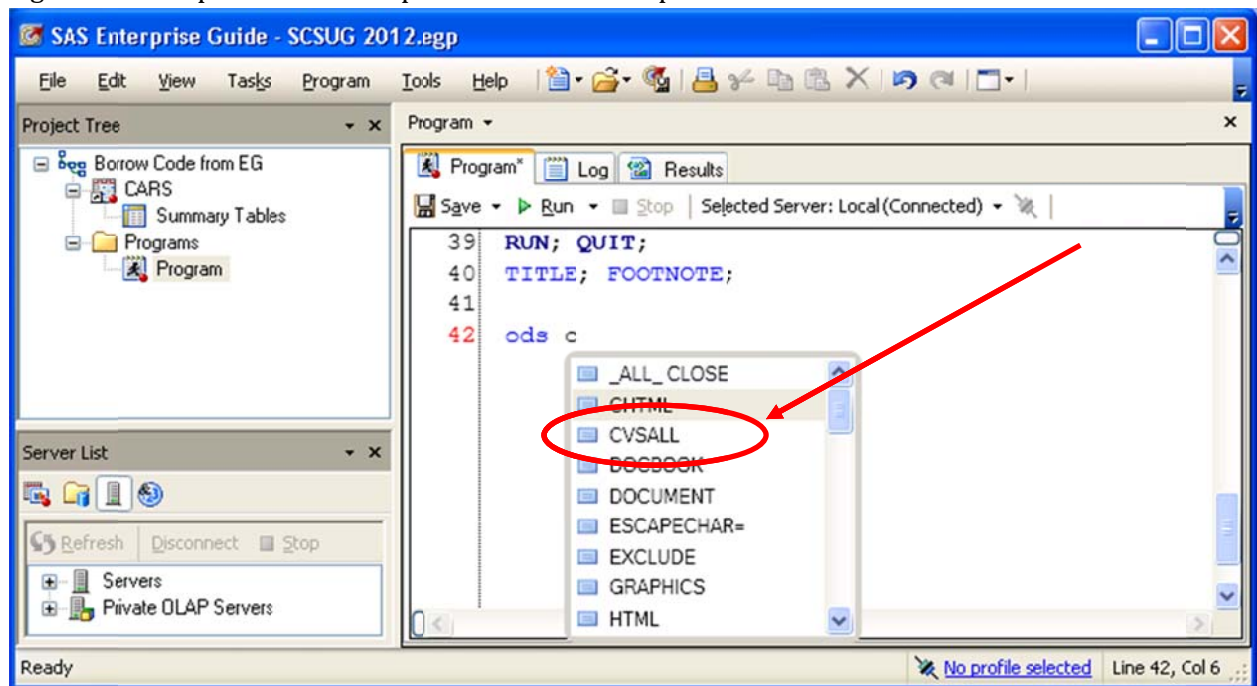
Figure 17: Incorrect Last Modified Time for Permanent SAS Dataset



### ODS CVSALL

Autocomplete for the EG SAS code editor is a wonderful feature, but there may still be some missing or misspecified options. One that we have noticed is the ODS CSVALL statement to create a .csv file. The EG editor has this misspelled as CVSALL (Figure 18).

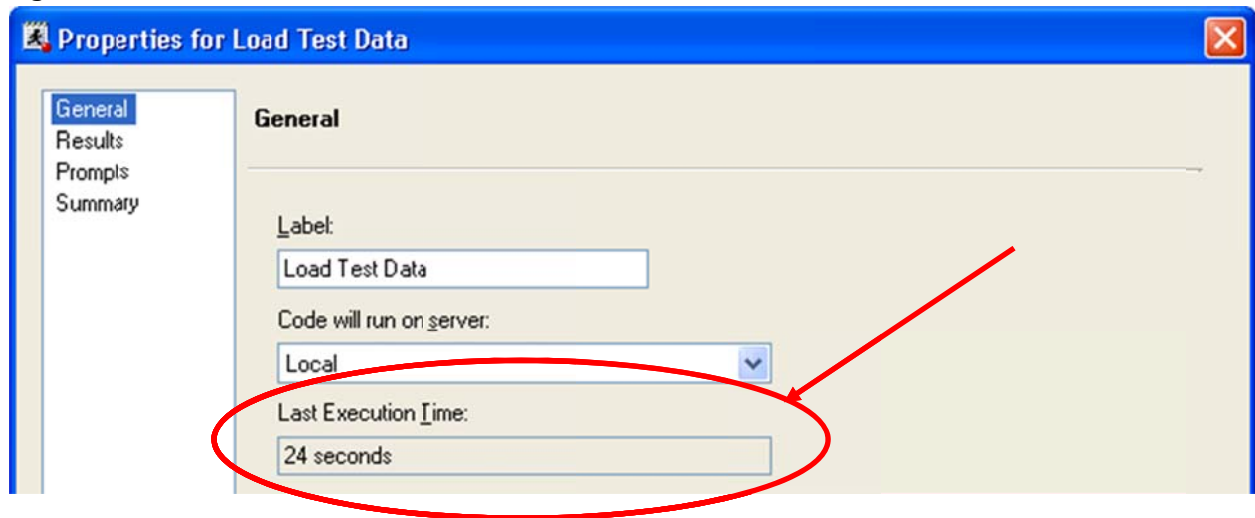
Figure 18: Misspelled CSVALL option in EG Autocomplete Feature



### Lack of Total run time in log

Within EG, we miss the reporting of total run time in the Log files that was automatic in the traditional SAS interface. This is often useful information for planning when and how to run a process. This information can be viewed in the Task properties as Last Execution Time (Figure 19), but it seems like a simple feature to add back to the SAS Log at least for the old-time SAS programmers.

Figure 19: Last Execution Time for EG Task



### Conclusion

We hope you have found useful tips on approaching SAS Enterprise Guide. For the most part, we have found this tool to be a great addition for productivity and documentation. We look forward to discovering even more powerful features of EG.

### Suggested Reading

First, Jennifer and Steven First. 2012. Productivity Tips for SAS Enterprise Guide Users. SAS Global Forum paper 301-2012.

Ravenna, Andy. 2011. Becoming a Better Programmer with SAS Enterprise Guide 4.3. SAS Global Forum paper 307-2011.

Schacherer, Chris. 2012. Take a Fresh Look at SAS Enterprise Guide: From point-and-click ad hocs to robust enterprise solutions. SAS Global Forum paper 294-2012.

Slaughter, Susan J. and Lora D. Delwiche. 2010. The Little SAS Book for Enterprise Guide 4.2. Cary, NC: SAS Institute Inc.

### Author Contact Information

Phuong Pham  
Clarity Services, Inc.  
ppham@clarityservices.com

Steve Fleming  
Clarity Services, Inc.  
sfleming@clarityservices.com