USE OF SAS PROC COMPARE AND AUTOCALL TO REDUCE THE CODE NEEDED FOR A SAS SYSTEM BY 75%

Arthur H. Hensler
The Firestone Tire & Rubber Company, Akron, Ohio

BACKGROUND

Last year I received a 16,000 line SAS system from an end user. The system came with no documentation, but a list of proposed modifications. A 27 member partitioned dataset contained the code. The code did not use macros, and the code in each partition was run from a separate job.

Upon examination, I found many large sections of code that appeared similar. Before making any changes, I decided to determine how the system worked, and to provide complete documentation. The best way to do this was to break the code down into smaller, more manageable chunks and document each of these.

I planned to convert the similar pieces of code into SAS AUTOCALL macros, and to use the partitioned dataset as the AUTOCALL library. In order to do this, I needed to find the differences (if any) in the similar pieces of code. Working this manually proved to be too time consuming and error prone, so I generated the following automated method of comparison using SAS PROC COMPARE.

PROC COMPARE METHOD

Copy the sections of code to be compared into new datasets for editing. The SAS procedure below generates a SAS dataset for each group of SASCODE being compared and runs PROC COMPARE to find differences between the groups of code.

ED!T ---- TSQTDJ5.SASCODE.PATA(SCAS1) - 01.01 ---

If needed, FIRSTOBS can be added to the INFILE statement to designate the beginning of a section being compared and OBS can be used to limit the comparison to a given number of lines.

PROC COMPARE only prints the first 20 characters of any variable which differs from 1 dataset to another. Using 4 variables displays differences more than 20 characters into a line.

After reviewing the output from PROC COMPARE, change spacing and capitalization and insert blank lines to make the 2 versions of code as much alike as possible. When all changes have been made, the differences left are the differences between the two versions of the code, which can be included in the macro formed from the code.

EXAMPLE USING PROC COMPARE

The two datasets below containing SAS code will be compared and a macro written which will do the work of both.

EDIT ---- TSQTDJ5.SASCODE.PATA(SCAS1) - 01.02 ---

Activate SAS and allocate the datasets and the SAS code containing PROC COMPARE.

Command "'" """" """" """" """" """" """" """" """" """" """" """" """" """" """" """" """" """" """" """" """" """" """" """" """" """" """" """" """" """" """" """" """" """" """" """" """" """" """" """" """" """" """" """" """" """" """" 

548
Run the procedure.

The output from PROC COMPARE is shown below.

The comparison shows 3 lines different between the 2 datasets. The word "ONE" is capitalized in one dataset, but not the other. Since this will not affect how the code runs, make it caps in both datasets. "RUN" appears in line 4 of one dataset, but line 5 of the other. The PROC PRINT showing in the dataset does not compare with the PROC PRINT in the compare dataset. This is because PROC COMPARE only compares variables for observations in the base dataset.

The added line with the UPCASE function shifted all following lines down a line in its dataset. To account for this, insert a blank line before RUN in SAS1.

This is how SAS1 appears after the editing.

Running PROC COMPARE code again produces the following result.

This shows that only 1 line of code is different between the two datasets. SAS3 is the macro generated to run both cases of SAS1 and SAS2.

SAS4 shows the finished macro with documentation in the macro header.

RESULTS OF CHANGES

While this is a very simple case, the techniques used here are extremely helpful in generating macros from large SAS programs. In the system I received, I reduced 16,000 lines of SAS code to under 4,000 lines, while documenting and adding major enhancements to the system. Making the system macro driven has increased the number of members in the partitioned dataset from 27 to 35.

Going to a macro based system and reducing the volume of code have made the system much easier to understand and document, and have greatly reduced the effort required for maintenance changes.