CC-10

Automation of Data Updates: A Case Study

Carry W. Croghan

ABSTRACT

In December of 2009, the United States Environmental Protection Agency initiated a year long study to measure a suite of traffic-related air pollutants adjacent to I-15 in Las Vegas, Nevada. Measurements were collected simultaneously at four sites and stored at five minute intervals. The data were transmitted daily to a central computer using WinCollect Data Evaluation and Reporting Software (Version: 3.3). Once a week, the data were exported into an Excel (2003) spreadsheet which was then translated to a SAS® dataset. To efficiently update the SAS database on a weekly basis several different SAS tools were utilized including PROC IMPORT, PROC UPDATE, %INCLUDE, and batch calls with parameter pass throughs. Each of these tools will be described with details including why they were utilized. By automating the process of updating the SAS database, the time to complete the weekly data extraction task and the probability of human error were greatly reduced.

INTRODUCTION

The United States Environmental Protection Agency (EPA) performed a year long study in Las Vegas, NV to measure air pollutants near a major roadway in partnership with the Federal Highway Administration. The air pollutants were measured simultaneously at four monitoring sites located at different distances from I-15. At each of the monitoring sites, a suite of 15-20 instruments were connected to a computer running WinCollect Data Evaluation and Reporting Software (Version: 3.3) that recorded the over ninety measurements at five minute intervals twenty-four hours a day, seven days a week for over fifteen months. Once a day, the data from the individual sites were transmitted to a computer at our main campus in North Carolina using WinCollect. The data were examined in graphical format to ensure that the instruments were operating within normal parameters. Once a week, the data were then loaded into a SAS database for more extensive statistical analyses.

METHOD

WinCollect was extremely useful in that it controlled multiple instruments in the field and permitted remote access to the data for viewing and downloading. There were, however, problems with generating output from data. Although there is a feature that automatically generates data reports, the interface to design the reports was touchy and would crash after only a few edits were performed. Because of this, the data dump feature was utilized instead of generating reports. The data dump feature is menu driven and could not be automated. Once a week, the data from the previous week were downloaded into separate Excel Spreadsheets for each of the sites.

As a first step to translating the data from the Excel Spreadsheets to SAS, the IMPORT Wizard in interactive SAS was utilized. The SAS IMPORT Wizard tool is equivalent to the PROC IMPORT procedure with the additional feature of guiding you through the import process. Therefore, it is not necessary to memorize or look up the syntax for the PROC IMPORT. The main advantage to performing this task in this manner is an opportunity to verify quickly if the import was successful by consulting the log and viewing the resulting SAS dataset. Once it was determined that the import was successful, the code from the IMPORT Wizard was saved to a file. The main disadvantage of IMPORT Wizard is that it is interactive and requires input from the user. For these reasons, it is considered a useful tool only for one time imports. The PROC IMPORT code used within a program is a more efficient method of importing multiple datasets.

The SAS IMPORT Wizard tool is accessed in interactive SAS from a command in the File menu labeled 'Import Data'. The IMPORT Wizard will take you through a series of frames extracting the necessary information from you. The first is the type of data to be imported. The default is Microsoft Excel. The second frame requests the location of the file to be imported and the sheet name. The destination and name of the output dataset is requested in the third frame. Once providing this information you can either "Finish" which will import the data or "Next" which will take you to a frame that will permit you to save the code generated by the Wizard. My goal when running the wizard is obtaining the code so it can be run in the future with little modification.

Separate code was generated for each of the four sites. Each of the sites had over ninety variables. Although effort was made to maintain consistency between each of the sites, there were differences. Initially the differences between sites were relatively minor. For example there were small differences in the variable names between sites that required some variables to be renamed in SAS. As the study progressed, the differences between sites increased. In particular, there were modifications to one site as several additional instruments were added. These differences necessitated separate code for each of the four sites.

Once the new data were translated into SAS, the new records were combined with the previously downloaded data. In SAS there are three ways to combine datasets together in a data step: set, merge, and update.

- Set - appends records. If a by statement is used, then the values are combined in such a way as to preserve the sort order.
- Merge - combined the records from each dataset into one record consisting of all variables. It is recommended to always use a by statement to ensure the files are matched correctly. If there are variables that have the same name in both files, the value in the later files over writes the value in the previous files.
- Update - appends records so that no duplicate records are generated. A by statement that uniquely identifies each record is required. Only two datasets can be combined with update. The first is the master dataset and the second is the dataset that is used to update the master dataset. For each matching record, update replaces values in the master with new non-missing values from the update. If an observation has missing value in the update file and has a non missing value in the master file, the observation will not be updated.

Although the most commonly used statements are the set and merge, the update statement is very useful because it modifies an existing dataset with new information. Update is usually used to make corrections to a dataset. Only those records and variables, which have corrections, would be contained in the updating dataset.

In our situation, the data were being downloaded on a weekly basis. The last day from the previous weeks download was always downloaded again. This was done to insure that complete days were captured. The time in which the data were uploaded from the remote sites was not constant. Since the data were being downloaded for entire days, there were times in which portions of a day would have been downloaded twice. If the datasets were combined with a simple SET statement there would have been duplicate records created. To avoid this problem, the UPDATE statement was used.

One problem recognized with the UPDATE statement in this case was that it only replaces values if the new value is non-missing. There were times where previously downloaded data that were non-missing were tagged as invalid making it missing on the second download. Making these types of changes to the master dataset requires more extensive programming. One method for dealing with this problem is to removing those problem records from the master dataset prior to combining with the updating dataset. Another option is to use the SET statement with an IN option to identify records from each of the two datasets and delete the duplicate records from the master dataset.

At the beginning, there were multiple problems with connecting to the remote computers. At times, it was not possible to access the data. Maintaining the code to load and update the datasets in separate programs permitted selective running. However eventually all those difficulties were resolved. In our computing environment, only one SAS program can be run at a time. It was necessary to wait for one program to be completed prior to initiating another program. This became burdensome. The solution was to combine the programs together into a single program. Instead of copying the code, the new program consists of a series of %INCLUDE calls to the existing programs.

The %INCLUDE statement brings in code from another file into the current program and runs that code as if it is contained in the current program. The syntax is simple:

```
%include filename;
```

The filename can be the physical location and name of the external program or it can be a pointer that was previously defined. It is possible to reference multiple files with a single include statement. The code will be run in the order in which it appears on the statement. However, it is preferable for sake of readability and ease of debugging to use a separate %INCLUDE statement for each file. You cannot limit the lines to be included from an external file. The entire file is always included and run.

Using %INCLUDE statements maintains the individuality of the code but allows the code from different programs to function as if they were a single program. While there continued to be modifications to the various sites and resulting data, the individual program dedicated to that site could be modified and corrected to meet the new requirements without affecting the calling program or the other sites' programs. In addition if there were any communication problems, single site programs could be run separately without fear of not being the most up to date code. This modular approach is a time and sanity saver.

Over time, most of the problems were resolved. The process of downloading the data and updating the SAS data base became routine. The names of the output files were standardized to be the site name and the date of the download. The only thing necessary to be changed within the program was the date. Each week, the program was opened and the new date placed in a macro variable. Then the program was run in batch mode. The process of opening the file, while not long, was not trivial. Since there only needed to be a single edit made, it seems to be even more arduous to locate the variable and make the change. There should be an easier way to get the information to SAS for the single value. The solution was to pass that value to SAS during the batch call of the program using the 'sysparm' system option.

A system option modifies the manner in which SAS performs. Commonly used system options include: nodate, obs=, and pagesize=. Some system options can be restricted by a site administrator and cannot be modified. PROC OPTIONS will give you a list of options, their values, and whether they are restricted. For those options that can be modified there are three ways of doing so:

1. on the command line or in a configuration file
2. in an OPTIONS statement
3. in the SAS System Options window in interactive SAS.

How and where you modify the system option will determine how pervasive the change is. For example, options modified in an OPTIONS statement will override those set when the program was initialized.

SYSPARM is a system option that allows you to pass a character string to a SAS program. The character string can be referenced within the program using either the SYSPARM() function within a DATA step or the macro variable &SYSPARM. The syntax is

```
-sysparm <"> character string <">
```

when including on a command line call and

```
sysparm = <"> character string <">
```

when set on an OPTION statement.

If you do not enclose the character string in quotes, the resulting value passed to the program may be converted to all caps.

To facilitate the use of the SYSPARM option, a batch program was written. The program consisted of a few lines of directory changes and the important line:

```
C:\Program Files\SAS\SAS 9.1\sas.exe" -batch -sysin UpdateStationall.sas -sysparm
"20100216"
```

This code consists of six elements. The first is the call to the SAS executable program. The second –batch is a program switch that makes the SAS run in batch mode instead of interactive mode. The third is another switch -sysin which indicates the next piece of information name of the SAS program that is to be run. The final two elements are the –sysparm option and the value that is being passed to the SAS program. In our case, it was the date of the download that is used to identify the Excel files that are to be imported into the database.

Since the batch code was so short, the time to open the file was greatly reduced and it was simple to locate the necessary field to be modified.

At first the process of updating the SAS database took a couple of hours each week. Most of the time was consumed by editing the individual programs and running them separately. Then output from each site was verified before proceeding to the next site. Once all of the datasets were stabilized and the batch call process was used the time to edit and initiate the programs was reduced to mere minutes.

CONCLUSIONS

Occasionally, there are tasks that need to be done repeatedly. In our situation, the task was the weekly updating of 4 site specific datasets. The datasets consisted of over 90 measurements taken at 5 minute intervals, 24 hours a day and 7 days a week (over 181,000 data points). Each week an Excel spreadsheet was created from the WinCollect software with data from the previous week for each of the four sites. These data were imported into SAS and appended to the existing SAS datasets. The IMPORT Wizard was used to generate the PROC IMPORT code to translate the Excel spreadsheets into SAS format. There were separate programs written for each of the sites to address idiosyncrasies of the specific site. At the beginning, this task was far from routine. There were multiple changes to the file structures and the problems with connectivity and data availability. Once the task became routine, various SAS tools were be used to automate the task. The %INCLUDE statement allowed for a modular approach to combining the different programs into a single program. Using the SYSPARM option to pass information to the SAS program was the final element to automating the process. Opening and editing a short DOS batch program is much quicker than opening a SAS program into a program editor.

SAS is a very flexible programming language that permits automation of tasks at several different levels. Establishing automatic processes assures consistency and efficiency. However before you can automate the task, you must first have a working solution. The final solution should be built upon previous work.

ACKNOWLEDGMENTS

The United States Environmental Protection Agency through its Office of Research and Development funded and managed the research described here. It has been subjected to Agency administrative review and approved for publication. Mention of trade names or commercial products does not constitute endorsement or recommendation for use.

REFERENCES

SAS Institute, Inc. SAS On-line Documentation 9.1.2 (2004), http://support.sas.com/onlinedoc/912/docMainpage.jsp.

CONTACT INFORMATION

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

Carry Croghan
EPA E205-04
109 T.W. Alexander
RTP, NC 27711
Work Phone: 919 541 3184
Email: Croghan.Carry@epa.gov