Paper PO-08

Using Windows Batch Files to Sequentially Execute Sets of SAS® Programs Efficiently

Matthew Psioda, Department of Biostatistics,
The University of North Carolina at Chapel Hill, Chapel Hill, NC

ABSTRACT

SAS users commonly have the need to repeatedly execute large sets of SAS programs and efficiently perform electronic log scans. We discuss one simple method that uses a utility SAS program to create windows batch (*.bat) files that can be used to sequentially submit a set of programs and scan the resulting SAS logs. Furthermore, SAS programs are often stored in a standard folder structure which compartmentalizes the programs into subfolders according to purpose. In this setting, users need to execute the SAS programs in one or more of the subfolders in a consistent and pre-specified sequence. We describe a method that is based on an easy-to-construct windows batch file, called a global batch file, which gives the user simple prompts to determine what sets of programs need to be executed. Based on user input, the SAS programs are executed in sequence, logs are scanned for each set of programs, and a batch report is generated. When an organization uses consistent folder structure from project to project, these tools are completely portable allowing efficient batch processing with virtually no modification.

INTRODUCTION

Any efficient process for executing a series of SAS programs requires a well-organized directory structure in which to store the SAS programs. Figure 1 presents a simple, easily generalizable directory structure which will serve as the foundation for the discussions included in this paper.

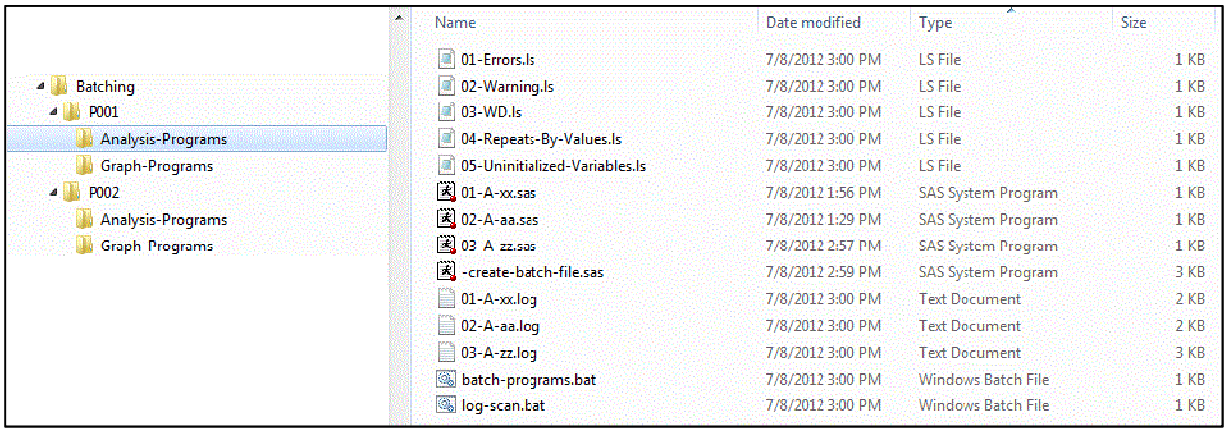

Figure 1: Sample Directory Structure

We consider an example directory with 2 projects, designated P001 and P002, which contain subfolders for analysis SAS programs and graphics SAS programs. While each project has the same underlying directory structure, the SAS programs contained therein may be similar or completely different. In this paper, we present a process that allows sequential execution of the SAS programs stored in a single folder (i.e. SAS programs in P001\Analysis-Programs) and sequential execution of the SAS programs stored in multiple standard folders (i.e. SAS programs in P001\Analysis-Programs followed by SAS programs in P001\Graph-Programs).

At the heart of this process is a utility SAS program, called -CREATE-BATCH-FILE.SAS, which is executed to create a Windows batch file which we call BATCH-PROGRAMS.BAT. When the Windows batch file is executed, it sequentially batch submits each of the SAS programs stored in the folder in alpha-numeric order based on the program names. After completion of the batch submission of the set of programs, it then executes a second Windows batch file to perform a log scan of the newly created SAS logs. Log scans can be customized to search for standard terms (ERROR, WARNING, etc.), or user-defined terms (UNEXPECTED VALUE, NOTE TO SELF, etc.). Using the proposed process, any set of SAS programs stored within a single folder can be easily executed and the corresponding logs scanned with the click of a single file.

The second tool executes sets of SAS programs which are stored in different folders. This is done with another application of Windows batch files using what we refer to as a global batch file. When executed, the global batch file prompts the user for information regarding what project and folders are to be included in the global batch. Based on the user input, the programs in each of the selected folders are batch submitted in the standard order.

CREATING AND USING THE BATCH-PROGRAMS.BAT FILE

We utilize SAS to write out the contents of the BATCH-PROGRAMS.BAT file which uses the Windows command line to perform several tasks, including:

1. Deleting files associated with previous SAS program runs (e.g. logs)
2. Batching each SAS program in the directory
3. Calling the LOG-SCAN.BAT file after completing batch submission of the SAS programs

The most fundamental portion of the utility SAS program uses the SASHELP.VEXTFL view to find the location of the utility SAS program and store the location in a macro variable named BATCHPATH. The following SAS code provides one of many methods for how this can be done:

```
proc SQL noprint;
 select substr(xpath,1,index(xpath,'-create-batch-file')-2)
  into :batchpath
   from sashelp.vextfl
    where find(xpath,'-create-batch-file','i');
quit;

%let batchpath = %trim(%left(&batchpath));
```

Once the path to the utility SAS program is identified, we navigate the working SAS directory to the location of the utility SAS program and create a list of SAS programs found in the directory.

```
x "cd &batchpath.";

filename insas pipe "dir *.sas  /on/b";
data SASPrograms;
 infile insas;
  length SASName $100.;
  input SASName $;
   if substr(SASName,1,1) ^= '-';
run;
filename insas clear;
```

In the above code we use FILENAME statement with an unnamed PIPE to determine the names of all SAS programs in the working SAS directory. The quoted commands following the PIPE keyword are DOS commands. The asterisk wildcard selects all files with a SAS extension. We use both the /on and /b options (or switches) in the command line code. The /on option returns files in alphabetical order based on the file names and the /b command suppresses summary and header information which results from the default application of the DIR command. We use an INFILE statement in the subsequent data step to read the SAS program names into a SAS data set for further processing. Note the use of the subsetting IF statement in the data step. This allows the user to exclude some SAS programs from being included in the batch file. Any program with a name beginning with a dash is excluded. Hence, starting utility programs names with a dash (such as -CREATE-BATCH-FILE.SAS), is an easy way to avoid having them run as a part of the automated batch process.

Now that we have the SAS program names stored in a SAS data set, we can create a data step variable that contains the appropriate commands to batch the SAS program using the Windows command line. The following SAS code could be used:

```
data RunPrograms;
 set SASPrograms end = Last;
  length Line $300.;
  if _n_ = 1 then do;
   Line  ="rem [BEGIN: Run SAS Programs]";  output;
  end;
   Line  = 'start "Batch Submission" /abovenormal /wait';
   Line  = catx(" ",Line,'"C:\Program Files\SASHome\SASFoundation\9.3\SAS.EXE"');
   Line  = catx(" ",Line,strip(sasname), "-nologo -icon -rsasuser") ; output;
  if Last then do;
   Line  ="rem [END: Run SAS Programs]"; output; Line  =" "; output; output; output;
   Line  ="rem [BEGIN: Call Log Scan Batch File]";  output;
   Line  ="call log-scan.bat";  output;
   Line  ="rem [END: Call Log Scan Batch File]";  output;
  end;
  keep Line;
run;
```

An example of the resulting SAS data set is shown in Figure 2.

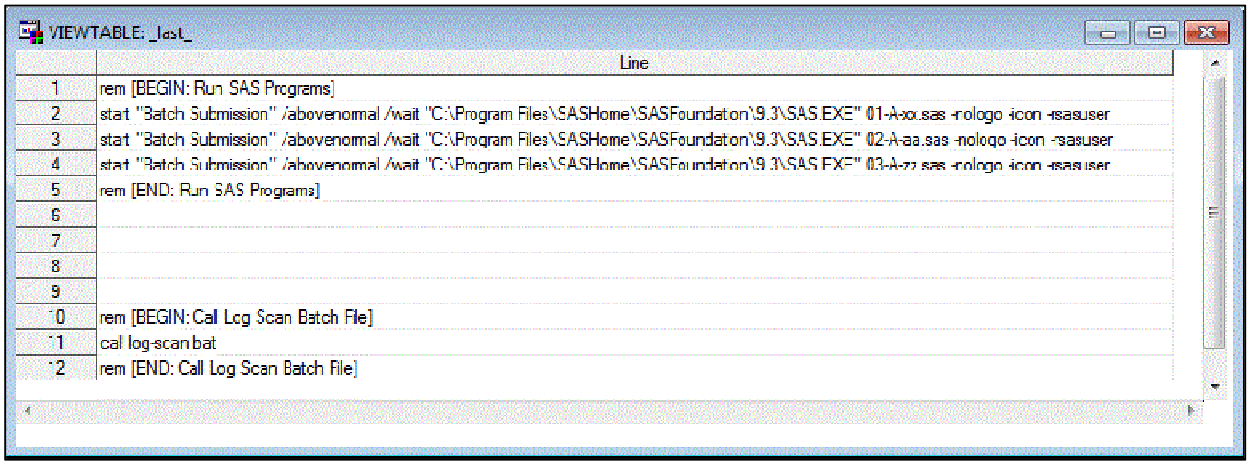

Figure 2: Contents of the RUNPROGRAMS Data Set

The /abovenormal option affects the priority of the SAS program executing in the operating system environment. The /wait option requires that one SAS program complete before starting execution of another. If sufficient system resources exist such that execution of a large number of programs at once is feasible and if there are not dependencies among the SAS programs being executed, one might consider removing the /wait option to accelerate completion of the entire process. The -rsasuser option instructs SAS to access the users profile in read only-mode. This can be helpful to avoid the common SAS warning that results when multiple SAS sessions have been opened at the same time by the same user.

Using an identical process to that described above, we can examine the working SAS directory for files that we wish to scour at the start of a new batch execution. The following SAS code could be used:

```
filename inall pipe "dir *.*  /on/b";
data AllFiles;
 infile inall;
  length FileName $100.;
  input FileName $;
   if find(FileName,'.sas','i')=0 and
      find(FileName,'.bat','i')=0 and
      find(FileName,'.ls','i')=0;
  length DelLine $300.;
  DelLine = "del *."||strip(scan(FileName,2,'.'));
run;
proc sort data = AllFiles nodupkey; by DelLine; run;
filename inall clear;

data Scour;
 set AllFiles end=Last;
  length Line $300.;
  if _n_ = 1 then do;
   Line  =" ";  output;
   Line  ="rem [BEGIN: Delete Existing Files in Directory]";  output;
  end;
  Line  = DelLine;  output;
  if Last then do;
   Line  ="rem [END: Delete Existing Files in Directory]";  output;
   Line  =" ";  output; output; output; output;
  end;
  keep Line;
run;
```

We use the *.* wildcard to get a list of all files in the working directory and a subsetting IF statement to remove any program that we do not wish to delete. For example, we would not want to remove programs with a SAS extension. As an alternative to using the contents of the directory to determine what files to remove, one could simply hard code deletion statements for all the file types that are expected and desired to be automatically removed.

Finally, once the two data sets containing batch file commands are created, we write out the contents of the BATCH-PROGRAMS.BAT file with a simple data step. The following code could be used:

```
filename Batch "batch-programs.bat";
data _null_;
 file Batch;
  set Scour RunPrograms;
   put Line;
run;
filename Batch clear;
```

The contents of the resulting Windows batch file are shown below in figure 3.

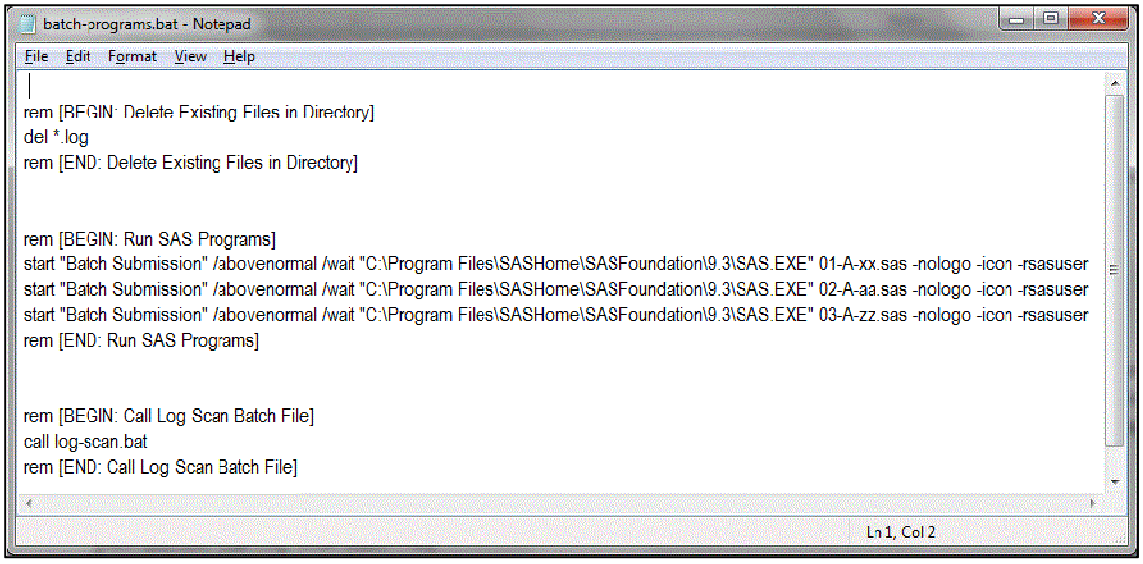

Figure 3: Contents of Batch-Programs.bat File

The BATCH-PROGRAM.BAT file can be executed with a simple double-click. Note that lines starting with “rem” do not execute. Starting a line with “rem” allows the user to write comments in the batch file. Additionally, Windows batch files can be edited with any text editor. This allows users to customize the batch file to run a smaller collection of programs. The modified batch file can be saved under a different name to allow for long-term use. If ones wishes to store modified batch files for long term use, the batch file creation program should be written to not automatically scour files with the BAT extension during the batch file creation process.

CREATING AND USING THE LOG-SCAN.BAT FILE

The LOG-SCAN.BAT file is designed to scan the SAS logs for a set of pre-defined key words that could potentially indicate problems with the SAS code execution. Common examples of keywords are ERROR and WARNING. Depending on the application, a variety of other key words might be included. We use SAS to write out the contents of the LOG-SCAN.BAT file in the same way as was shown above. The LOG-SCAN.BAT is a static file in the sense that its contents do not depend on the associated directories contents. We write out the contents of this file using the same utility SAS program that writes out the BATCH-PROGRAMS.BAT file just for simplicity. The following example code could be used:

```
filename LogScan "log-scan.bat";
  data _null_;
   file LogScan;
    length Line $300.;
    Line  ="rem [BEGIN: Log Scan]";  put Line;
    Line  ="del *.ls";  put Line;
    Line  ='for %%i in (*.log) do find /N "ERROR" %%i >>  01-Errors.ls'; put Line;
    Line  ='for %%i in (*.log) do find /N "WARNING" %%i >>  02-Warning.ls'; put Line;
    Line  ="rem [END: Log Scan]";  put Line;
  run;
filename LogScan clear;
```

The above code includes commands to scan for ERRORs and WARNINGs. Other key words can be added using the same syntax. The search is case sensitive but this can be controlled with the command line option /I. However, given that the scan searches for standard SAS log messages, a case sensitive scan is suggested to avoid false findings. The /N option requests the display of the line number of the log file where the keyword is found. In the process we described above, the LOG-SCAN.BAT file is automatically called by the BATCH-PROGRAMS.BAT file when it is executed. However, the LOG-SCAN.BAT file can be manually executed manually as well. This is valuable when one SAS program in a set must be updated based on the findings of a batch log scan. In this scenario, the single SAS program can be updated and individually batch submitted and then the log scan can be manually run. The log scan will result in one text file per key word defined in the search and each file will include a section for each SAS log in the directory. An example is provided in Figure 4.

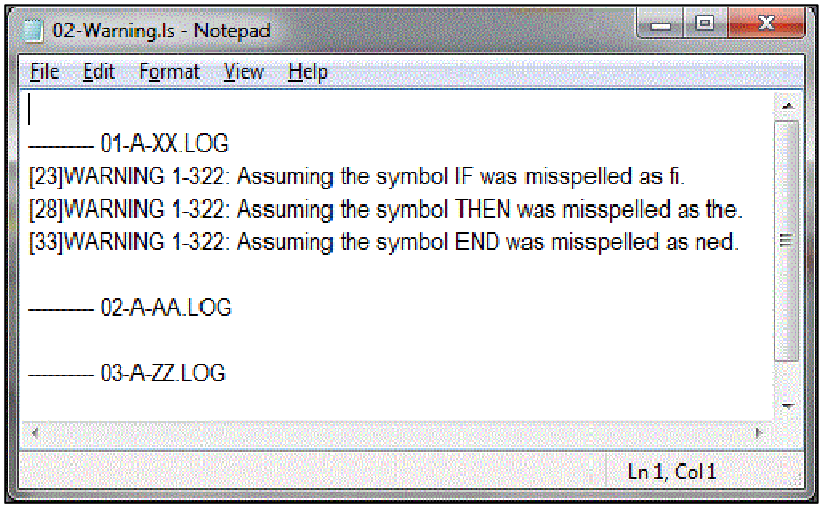

Figure 4: Example Log Scan Report

GLOBAL BATCH EXECUTION FOR MULTIPLE FOLDERS

We have outlined a process for how one can batch execute a set of SAS programs in a single folder using a Windows batch file. The next step is to develop a simple process for executing sets of programs from multiple folders in a

desired order. When a consistent directory structure is used from project to project, this is a simple task. We will use a static global batch file to perform this task.

The first step of the global batch file is to determine what project the user is going to submit SAS programs for and then what folders within that project. There are a variety of ways this can be done using the Windows command line and we present one very simple method here based on user prompts. Figure 5 presents a sample set of DOS window prompts.

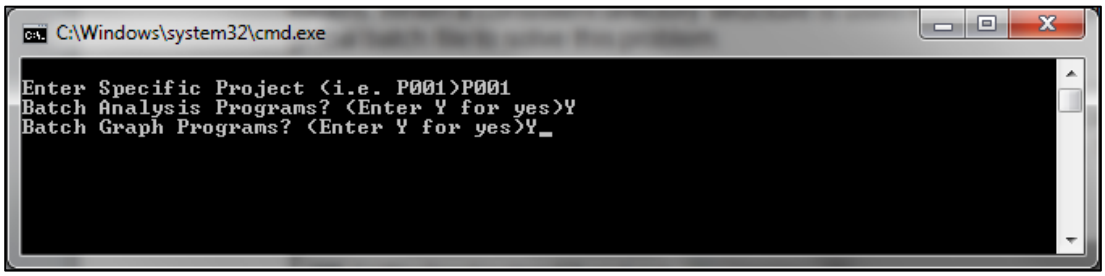

Figure 5: DOS Window Prompts

Each of these prompts request information from the user that is stored in a DOS variable that can be used conditionally within the batch file. The following command line code can be used to issue the prompts above to create the DOS variables FOLD, ANALYSIS, and GRAPHS, respectively.

(i) set /p FOLD=Enter Specific Project (i.e. P001)

(ii) set /p ANALYSIS=Batch Analysis Programs? (Enter Y for yes)

(iii) set /p GRAPHS=Batch Graph Programs? (Enter Y for yes)

The /p option allows the DOS variable to be set to a user entered value. The string to the right of the equal sign is the user prompt that will be displayed in the DOS window. With the above variables defined, we can complete the global batch process by calling the -CREATE-BATCH-FILE.SAS program that creates the BATCH-PROGRAMS.BAT file in each selected folder and then subsequently the BATCH-PROGRAMS.BAT file that the program creates. The following figure presents command line code that could be used for conditional execution of the analysis SAS programs if the directory is selected by the user.

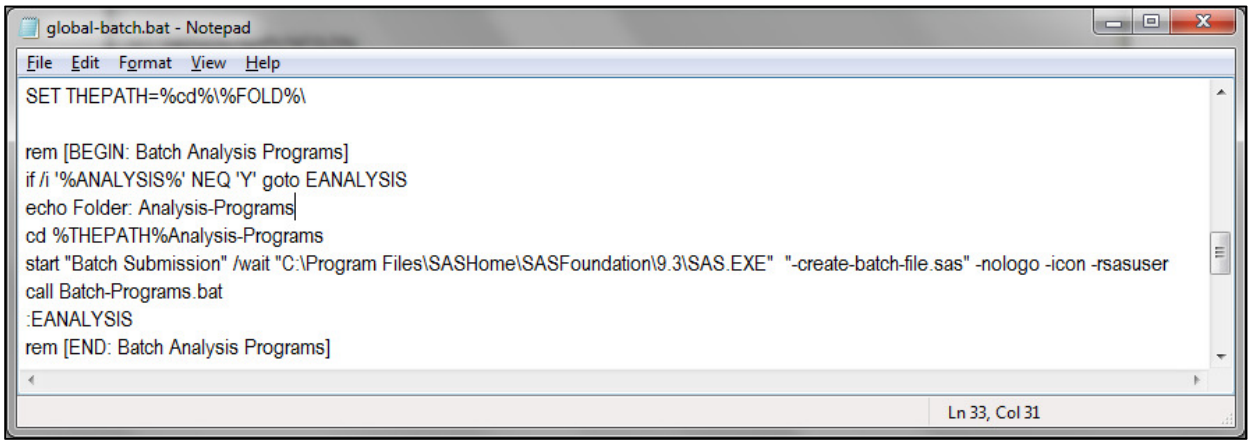

Figure 6: Global Batch File (Excerpt)

The code above uses the SET command to define the THEPATH variable based on the values of the CD and FOLD variables. Note how percent signs are used surrounding a DOS variable when it is referenced in a batch file. The CD variable is an automatic variable which stores the directory where the GLOBAL-BATCH.BAT file is located. When the GLOBAL-BATCH.BAT file is stored in a folder containing the project folders, as is done in this application, the THEPATH variable stores the precise path to the selected project directory.

The next section of code in the batch file uses an IF statement comparing the ANALYSIS variable to a value of Y to determine if the analysis programs should be executed. The use of the /I option make the comparison case insensitive. If the user inputs the letter y in any case, the subsequent code is executed. If the user inputs anything other than y, the batch file skips over the analysis programs directory.

The ECHO command prints the current folder being processed to the DOS window as a means of tracking the process of the global batch. The change directory command (cd) changes the working DOS directory to the Analysis-Programs folder. The subsequent code is the familiar batch submission code for a SAS program. After the utility SAS program is executed, the newly created BATCH-PROGRAMS.BAT file is immediately executed using the CALL command. Similar code can be added to the global batch file for each standard folder in a directory structure and the order in which such directories should be processed. For example, if analysis programs make data sets that are used by the graph programs, analysis program execution should come before graph program execution in the global batch file.

By default, each line of the batch file will print to the DOS window as it is executed. This can cause a great deal of clutter in the window and can obscure informative user-written messages. It is recommended that each batch file starts with the command @ECHO OFF. This command will suppress the default behavior so that only user printed messages (with the ECHO command) are displayed in the DOS window.

The ECHO command can be used to produce a permanent report of what directories have been batch submitted in addition to providing real time information in the DOS window. This can be a useful way to track how long a global batch of an entire project directory takes. The syntax for writing to an ASCII file is very simple and is presented below in Figure 7.

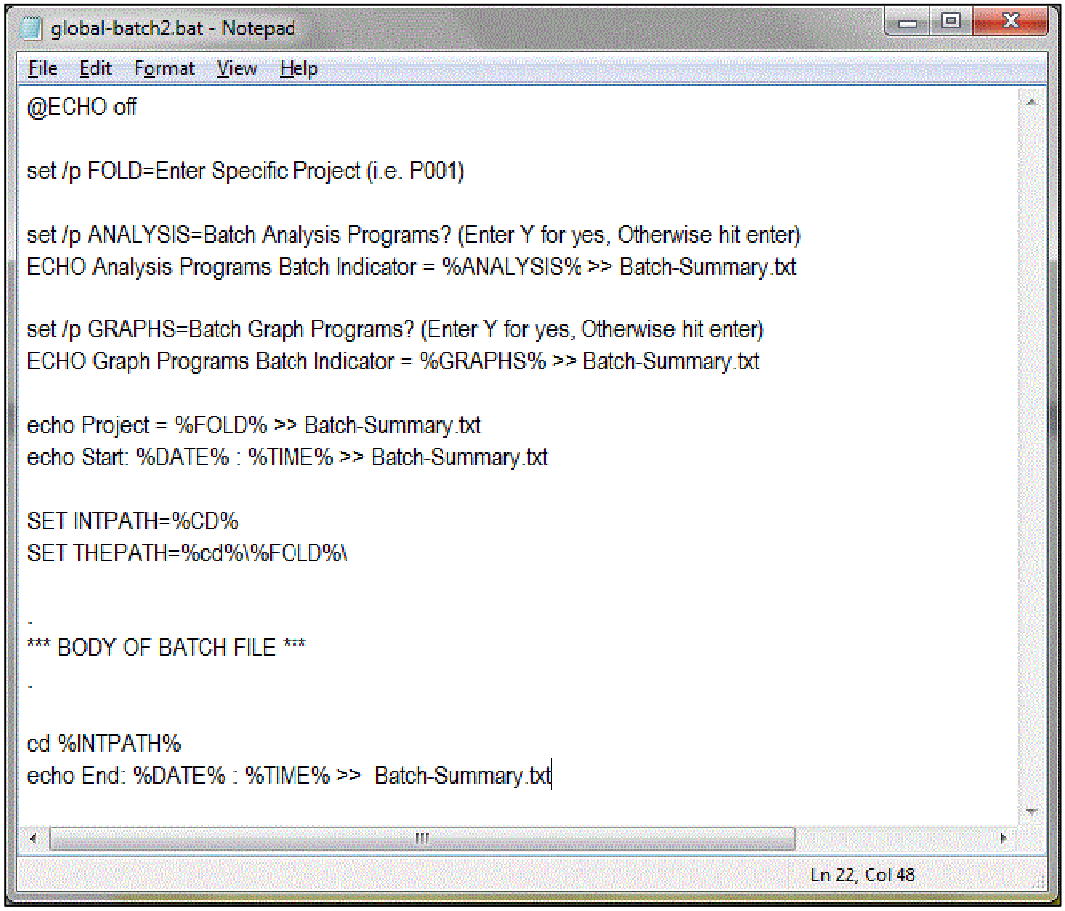

Figure 7: Sample Code to Generate a Batch Report

Note the format of the ECHO commands in the global batch code above. The use of ">>" instructs DOS to write the message to the BATCH-SUMMARY.TXT file instead of the DOS window. Hence, the report will contain a printed message of the folder input by the user, indicators for what project subdirectories are batch submitted, the start time of the global batch, and the end time of the global batch. Also note the use to the INTPATH DOS variable. This variable is created based on the location of the global batch file and is used in a change directory command just before the final message is written. This is necessary to ensure all messages are written to the same text file. If the change directory command were not included, a second BATCH-SUMMARY.TXT file would be created in the location of the last program directory that was included in the global batch. Figure 8 presents a sample batch report.

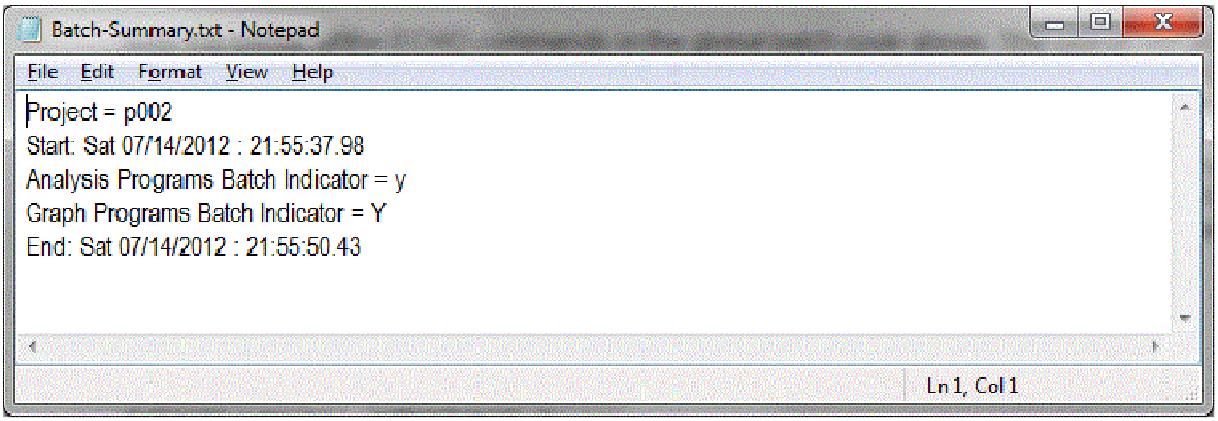

Figure 8: Sample Batch Report

CONCLUSION

In this paper we have presented one of the many methods that users can implement to batch submit a large number of SAS programs. The advantages of this approach are that it is highly effective, simple to implement, and it is easily adaptable to complex directory structures with many folders. For users who require a project-specific directory structure, we suggest that a utility program be created to auto-generate a project-specific global batch file. The fundamental component of the process, the utility program that creates the BATCH-PROGRAMS.BAT file, need not change at all across projects.

REFERENCES

Windows XP Professional Product Documentation: Command-line reference A-Z. 2012. Microsoft Corporation. <http://www.microsoft.com/resources/documentation/windows/xp/all/proddocs/en-us/ntcmds.mspx?mfr=true>

ACKNOWLEDGMENTS

I would like to thank my friend and colleague Sandy Ferber for her review and feedback on draft versions of this paper.

CONTACT INFORMATION

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

Name: Matthew Psioda
Enterprise: UNC Department of Biostatistics, Student
E-mail: psioda@live.unc.edu