Paper 2884-2015

Managing Files by Click and Drag?

Jason Wachsmuth, The University of Iowa, Public Policy Center, Iowa City, IA

ABSTRACT

File management is a tedious process, but it can be automated by using SAS® to create and execute a Windows Command Script. The macro in this paper copies files from one location to another, identifies obsolete files by the version number, and then moves them to an archive folder. Assuming some basic conditions are met, this macro is intended to be easy to use and robust. Windows users who run routine programs for projects with rework may want to consider this solution.

INTRODUCTION

Managing files by copying them to the appropriate storage location is a tedious process, but it can be automated by using SAS® and a Windows Command Script (CMD) (Watson, 2013). This paper will present an enhancement to this routine task. Besides copying files from one location to another, this macro will also segregate obsolete files as indicated by the version number and move them to an archive folder. Assuming some basic conditions are met, the FILEMGMT macro is intended to be file type-agnostic, and easy to use by consisting of just two parameters. Windows users who run routine programs for projects with rework may want to consider this solution depending on their version control methods.

ASSUMPTIONS

In order to directly use the FILEMGMT macro there are certain conditions that must be met:

- a version number is specified by a leading character followed by two digits, e.g., v01, w10
- the version number is located at the end of the filename, e.g., elig_apr15_cty_v01
- filenames are homogeneous across version numbers, e.g., elig_apr15_cty_v02
- the length of filename is ≤ 100 characters
- the length of filename plus the length of directory (i.e., var1-var4) is ≤ 200 characters
- directories do not contain special characters, e.g., &, (

The first three assumptions do not have a workaround mentioned within the scope of this paper, however the last three do and will be explained briefly now. For instance, the variable filename is assigned a length of $100, but this can be updated in the LENGTH statement. The length of variables var1-var4 are limited by the default value of $200, but these can be reassigned, too. Last, if the directories include any special characters then they must be masked with the %STR macro quoting function (“Special characters in directory names,” 2014; see the macro invocations in the Appendix for examples).

USING THE MACRO

The FILEMGMT macro consists of two positional parameters composed of directory locations. The first defines the origin of the files to be copied, and the second defines where the files are to be stored. In addition, two Windows system options are specified before the %MACRO statement: XMIN prevents the command processor from starting in an active state, and NOXWAIT eliminates the need to type EXIT in order to reactivate the SAS session (“Running Windows or MS-DOS commands from within SAS,” 2014).

```
 1 options xmin noxwait;
 2 %macro filemgmt(source, destination);
 …
46 %mend filemgmt;
47 %filemgmt(c:\health\apr15\output, p:\ppc\health\apr15\output)
```

BACKING UP THE FILES

Five X statements and one DATA step is all it takes to programmatically back up the files. The first part of this section consists of two X statements where the cd command changes the default directory to the source directory, and the dir command with /b/l options generates a brief (i.e., no heading or summary information) listing of files and folders in lowercase letters ("Command-line reference A-Z," 2014). Keep a clean source folder, though, because everything in the listing is about to be copied. Furthermore, if there are exact files between folders, the source files will overwrite the destination files without warning.

```
 3 x cd "&source.";
 4 x dir /b/l > filenames.txt;
```

The beginning of the DATA step specifies the data with an INFILE statement, and the LENGTH and INPUT statements capture the variable filename. Remember, one of the limitations of using the list input style to read and store raw character data, without the LENGTH statement, is that the values will be cut off after 8 bytes. Next, an evaluative statement deletes the unnecessary observations to copy over which includes a folder named archive and the text listing file. Two assignment statements follow after the data is clean. Var1 is composed of the constant &source. and the variable filename; var2 is composed of the constant &destination. and the variable filename. Last, the FILE statement specifies an external file, and the PUT statement writes out syntax for the copy command. Table 1 shows the product of the DATA step.

```
 5 data _null_;
 6   infile "&source.\filenames.txt" dlm='09'x;
 7   length filename $100;
 8   input filename $;
 9   if filename in("archive", "filenames.txt") then delete;
10   var1 = quote(catt("&source.\", filename));
11   var2 = quote(catt("&destination.\", filename));
12   file "&source.\copy_files.cmd";
13   put "copy " var1 var2;
14 run;
```

copy "c:\health\apr2015\output\elig_apr15_cty_v02.sas7bdat" "p:\ppc\health\apr2015\output\elig_apr15_cty_v02.sas7bdat"
copy "c:\health\apr2015\output\elig_apr15_cty_v02.txt" "p:\ppc\health\apr2015\output\elig_apr15_cty_v02.txt"
copy "c:\health\apr2015\output\elig_apr15_cty_v02.xlsx" "p:\ppc\health\apr2015\output\elig_apr15_cty_v02.xlsx"

Table 1. Content of copy_files.cmd

The last part of this section executes the CMD, and the delete command discards extra files in the &source. folder.

```
15 x cd "&source.";
16 x copy_files.cmd;
17 x del copy_files.cmd filenames.txt;
```

ARCHIVING THE FILES

Archiving files incorporates similar code as backing up files with the addition of a few steps. Like before, the first part of this section changes the directory and creates a list of its files and folders. However, a third X statement is added to ensure the archive folder exists. Although if previously created, nothing happens besides an implicit message that notes the subdirectory already exists.

```
18 x cd "&destination.";
19 x dir /b/l > filenames.txt;
20 x md archive;
```

The first DATA step does several things. After the data is read using modified list input and then cleaned, the number of periods that appear in filename are assigned to cou. The cou variable will compose part of the count argument in the CALL SCAN routine:

Syntax
CALL SCAN(<string>, count, position, length <, <charlist> <, <modifier(s)>>>);

Count specifies the number of the word in the character string that the CALL SCAN routine will select. The intention is to capture the word following the last delimiter (i.e., the file extension) so the cou+1 expression is used. This strategy accounts for filenames that could contain more than one period. Once count identifies the word, the position argument returns the starting point to the variable pos. In other words, now the starting position of the file extension is known. The last argument, length, returns the length of the word to the variable len. Note that the trim function was used in the string argument so that length would not include trailing blanks.

The two assignment statements that conclude this DATA step create the final variable where the values consist of everything except the version number. The first statement uses the variables pos and len in the SUBSTR function to create the variable ext which is the file type. The second statement uses the CAT and SUBSTR functions to achieve our objective as illustrated in Table 2.

```
21 data fn_01;
22   infile "&destination.\filenames.txt" dlm='09'x;
23   input filename :$100.;
24   if filename in("archive", "filenames.txt") then delete;
25   cou = count(filename, '.');
26   call scan(trim(filename), cou+1, pos, len, '.');
27   ext = substrn(filename, pos, len);
28   base = cat(substrn(filename, 1, pos-5), ext);
29 run;
```

STATEMENT	NUM VALUE	CHAR EQUIVALENT
call scan(trim(filename),cou+1,pos,len,'.');	2	elig_apr15_cty_v01.sas7bdat
call scan(trim(filename),cou+1,pos,len,'.');	20	elig_apr15_cty_v01.sas7bdat
call scan(trim(filename),cou+1,pos,len,'.');	8	elig_apr15_cty_v01.sas7bdat
ext = substrn(filename,pos,len);		sas7bdat
base = cat(substrn(filename,1,pos-5),ext);	15	elig_apr15_cty_v01.sas7bdat
base = cat(substrn(filename,1,pos-5),ext);		elig_apr15_cty_sas7bdat

Table 2. Three statements create a grouping variable needed hereafter

Now that the base variable exists, it is time to sort the data. Base is the first BY variable because these values must be grouped. Filename is the second BY variable to ensure that the latest version is the last observation within each base group.

```
30 proc sort data = fn_01 out = fn_02; by base filename; run;
```

base	filename
elig_apr15_cty_sas7bdat	elig_apr15_cty_v01.sas7bdat
elig_apr15_cty_txt	elig_apr15_cty_v01.txt
elig_apr15_cty_xlsx	elig_apr15_cty_v01.xlsx
elig_apr15_cty_sas7bdat	elig_apr15_cty_v02.sas7bdat
elig_apr15_cty_txt	elig_apr15_cty_v02.txt
elig_apr15_cty_xlsx	elig_apr15_cty_v02.xlsx

Table 3. Fn_01 data before sorting

base	filename
elig_apr15_cty_sas7bdat	elig_apr15_cty_v01.sas7bdat
elig_apr15_cty_sas7bdat	elig_apr15_cty_v02.sas7bdat
elig_apr15_cty_txt	elig_apr15_cty_v01.txt
elig_apr15_cty_txt	elig_apr15_cty_v02.txt
elig_apr15_cty_xlsx	elig_apr15_cty_v01.xlsx
elig_apr15_cty_xlsx	elig_apr15_cty_v02.xlsx

Table 4. Fn_02 data after sorting

The next step is to identify the last value of each BY group for the base variable. This is accomplished with the BY statement where SAS creates two temporary variables: FIRST.base and LAST.base. The evaluative statement that follows deletes the last observation per base group and the remaining observations are the filenames that are going to be moved to the archive folder.

```
31 data fn_03;
32   set fn_02;
33   by base;
34   if last.base then delete;
35 run;
```

base	filename
elig_apr15_cty_sas7bdat	elig_apr15_cty_v01.sas7bdat
elig_apr15_cty_txt	elig_apr15_cty_v01.txt
elig_apr15_cty_xlsx	elig_apr15_cty_v01.xlsx

Table 5. Fn_03 results include only files to be archived

The macro concludes with a DATA step writing out the move command and two variables to a CMD file (results are shown in Table 6), and three X statements to execute the script and delete extraneous files.

```
36 data _null_;
37   set fn_03;
38   var3 = quote(catt("&destination.\", filename));
39   var4 = quote("&destination.\archive");
40   file "&destination.\archive_files.cmd";
41   put "move " var3 var4;
42 run;

43 x cd "&destination.";
44 x archive_files.cmd;
45 x del archive_files.cmd filenames.txt;
```

move "p:\ppc\health\apr2015\output\elig_apr15_cty_v01.sas7bdat" "p:\ppc\health\apr2015\output\archive"
move "p:\ppc\health\apr2015\output\elig_apr15_cty_v01.txt" "p:\ppc\health\apr2015\output\archive"
move "p:\ppc\health\apr2015\output\elig_apr15_cty_v01.xlsx" "p:\ppc\health\apr2015\output\archive"

Table 6. Content of archive_files.cmd

CONCLUSION

File management can be time consuming, particularly if you back up files by copying them from your local drive to a shared drive, and then move obsolete files into the destination's archive folder. If this sounds like your process, using the FILEMGMT macro has two major benefits. First, it saves time; especially when managing multiple folders (e.g., log, data, deliverables) because they can be processed all at once in seconds. Second, accuracy is increased because files are copied, sorted, and moved programmatically instead of manually. Now, the question is, why would you continue to manage files by click and drag?

REFERENCES

Running Windows or MS-DOS commands from within SAS. (n.d.). Retrieved October 2, 2014, from SAS 9.2 Documentation website, http://support.sas.com/documentation/cdl/en/hostwin/63285/HTML/default/viewer.htm#exittemp.htm.

Special characters in directory names. (n.d.). Retrieved October 2, 2014, from SAS 9.2 Documentation website, http://support.sas.com/documentation/cdl/en/scracclgpug/62997/HTML/default/viewer.htm#p1fzy89uyrmwuzn1q8qdeoia19xi.htm.

Command-line reference A-Z. (n.d.). Retrieved December 8, 2014, from Windows XP Professional Product Documentation website, http://technet.microsoft.com/en-us/library/bb490890.aspx.

Watson, R. (2013, May). *Let SAS do your DIRty work*. Paper presented at PharmaSUG '13, Chicago, IL.

CONTACT INFORMATION

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

Jason Wachsmuth
The University of Iowa, Public Policy Center, Health Policy Research Program
310 S Grand Avenue
Iowa City, IA 52242
jason-wachsmuth@uiowa.edu

APPENDIX

```
 1 options xmin noxwait;
 2 %macro filemgmt(source, destination);
 3 x cd "&source.";
 4 x dir /b/l > filenames.txt;
 5 data _null_;
 6     infile "&source.\filenames.txt" dlm='09'x;
 7     length filename $100;
 8     input filename $;
 9     if filename in("archive", "filenames.txt") then delete;
10     var1 = quote(catt("&source.\", filename));
11     var2 = quote(catt("&destination.\", filename));
12     file "&source.\copy_files.cmd";
13     put "copy " var1 var2;
14 run;
15 x cd "&source.";
16 x copy_files.cmd;
17 x del copy_files.cmd filenames.txt;
18 x cd "&destination.";
19 x dir /b/l > filenames.txt;
20 x md archive;
21 data fn_01;
22     infile "&destination.\filenames.txt" dlm='09'x;
23     input filename :$100.;
24     if filename in("archive", "filenames.txt") then delete;
25     cou = count(filename, '.');
26     call scan(trim(filename), cou+1, pos, len, '.');
27     ext = substrn(filename, pos, len);
28     base = cat(substrn(filename, 1, pos-5), ext);
29 run;
30 proc sort data = fn_01 out = fn_02; by base filename; run;
31 data fn_03;
32     set fn_02;
33     by base;
34     if last.base then delete;
35 run;
36 data _null_;
37     set fn_03;
38     var3 = quote(catt("&destination.\", filename));
39     var4 = quote("&destination.\archive");
40     file "&destination.\archive_files.cmd";
41     put "move " var3 var4;
42 run;
43 x cd "&destination.";
44 x archive_files.cmd;
45 x del archive_files.cmd filenames.txt;
46 %mend filemgmt;
47 %filemgmt(c:\health\apr15\output, p:\ppc\health\apr15\output)*apr15;
48 %filemgmt(c:\health\apr%str(&)15\output, p:\ppc\health\apr%str(&)15\output)*apr&15;
49 %filemgmt(c:\health\apr%str(%')15\output, p:\ppc\health\apr%str(%')15\output)*apr'15;
50 %filemgmt(c:\health\apr%str(%(15%))\output,p:\ppc\health\apr%str(%(15%))\output)*apr(15);
```