PharmaSUG 2013 - CC02

Beep, Beep, Beep, Back It Up!

A Fool Proof Approach to Archiving with no Copying

Kristen Reece Harrington, BS, Rho® Inc., Chapel Hill, NC, USA

ABSTRACT

Have you been asked to revise a derivation and then several months later asked to revert back to the previous derivation code? Or when a project ends, were you asked to remove it from the network to free up space? What if Archival Software is not an option? This paper presents an "archiving" macro (%Archive()) which takes the contents of a directory, including subdirectories, and creates a zip file which contains all files and mirrors the parent's directory structure. Like magic, your code is saved and easily retrieved! Using an elaborate array of SAS® functions, an X command, and WinZip®, the archiving macro determines the execution location and the contents of the subdirectories associated with that location. With virtually no specified parameters, anyone can archive and look like a pro.

INTRODUCTION

In any statistical programming industry, it is imperative to SAVE YOUR WORK. You never know when the sponsor of a project which ended three years ago will ask for a program delivery, or when you may be asked to revise an imputation multiple times only to revert back to the original definition six months later. While saving your work seems obvious, the key question is "Was all the necessary code saved"? Imagine someone needs to recover a specific program from a directory which was completed months earlier. The person realizes that the included supporting macro was located in a separate directory that was never included in the archive and now has a timestamp of yesterday? The %Archive() macro was designed while keeping in mind that anyone, even a non-programmer or someone unaffiliated with the project, could be executing the macro. The end goal of this macro is to create a snapshot in time by producing one .zip file which contains all subdirectories and files EXCEPT the Archive folder where the .zip file(s) will be stored.

THE %ARCHIVE() MACRO

```
%Archive(Sponsor=,Protocol=);
```

The %Archive() macro is designed to determine the location from which the macro is being called using %SYSGET(SAS_EXECFILEPATH). This approach helps eliminate potential user error involving hidden spaces and typos. It also ensures that the common practice of copying and pasting programs does not cause the accidental zipping of incorrect directories. %SYSGET(SAS_EXECFILEPATH) kicks off this process by providing the full pathname including the program name. The following code determines the full pathname, create a macro variable containing the full pathname, strips off the program name and then finally creates a macro variable containing the just path:

```
%LET CurFile = %SYSGET(SAS_EXECFILEPATH);
%LET CurName = %SCAN(&CurFile, -1, \/ );
%LET CurPath = %SUBSTR(&CurFile, 1, %EVAL(%LENGTH(&CurFile)-
%SYSFUNC(INDEXC(%SYSFUNC(REVERSE(&CurFile)),\/))) );
```

Once &CurPath is set to the appropriate pathname (the location where the macro is being executed), a supporting macro is called. The %DIR macro utilizes SAS system return codes and the SAS functions DOPEN, FOPEN, DREAD, FINFO, and FCLOSE to produce a WORK dataset containing full pathnames for all files and subdirectories within the &CurPath location.

The supporting %DIR macro creates the WORK.FILES dataset. The goal is to include all files and subdirectories EXCEPT the Archive folder and its contents. The next step is to edit WORK.FILES to remove data points associated with the Archive folder as well as ensure all contents of subdirectories are included in the .zip file.

```
DATA files;
 SET files;
  IF type = 'DIR' AND name = 'Archive' THEN delete;
/* this removes the Archive folder and associated files*/
  IF type = 'DIR' THEN fullpath = LEFT(TRIM(fullpath))||"\*.*";
/* Using wildcard characters, we ensure all folders contents are included */
RUN;
```

Now that the list of files to be zipped is complete, confirmation of a location to save the .zip file that will be created needs to occur. If a location does not exist a folder needs to be created. This is easily done with X commands:

```
OPTIONS NOXWAIT;
%IF %SYSFUNC(FILEEXIST(&CurPath.\Archive)) = 0 %THEN %DO;
  x "cd &CurPath. ";
  x "mkdir Archive";
%END;
```

The NOXWAIT system option “specifies that the command processor automatically returns to the SAS session after the specified command is executed. You do not have to type EXIT.” (SAS Institute, 2005). This is done so that the DOS Window will not stay open. The %SYSFUNC(FILEEXIST(xxx)) statement returns a 1 (yes) or 0 (no) as to whether the file, or directory in this case, exists. If the answer is 0, the %do loop is processed and the Archive folder is created via two X commands. X “cd &CurPath” which makes sure the starting point is the same directory as our program location and X “mkdir Archive” which makes a new subdirectory called Archive within the &CurPath directory.

A .zip file without a date in the filename can cause issues. Therefore, the next step is to create a macro variable containing the date of execution, i.e. today’s date.

```
%GLOBAL fdate;
DATA _NULL_;
 CALL SYMPUT("fdate",LEFT(PUT("&sysdate"d,date9.)));
RUN;
```

After creating the data set containing a list of files to .zip and checking to make sure an Archive folder in which to create the .zip file exists. The next step is to create a .txt file containing the contents of WORK.FILES. This has two purposes: 1. It provides an audit trail of what was zipped. 2. The .txt file can be used to create the .zip file.

```
DATA _NULL_;
 SET files;
 FILE "&CurPath.\Archive\Archive_&Sponsor._&Protocol._&fdate..txt" DLM=',';
 PUT fullpath;
RUN;
```

Note that &Sponsor and &Protocol were the only two parameters defined in our macro. These two parameters are only used in the .txt and .zip filenames.

The last step is creating the .zip file(s). First, a macro variable is created containing the full pathname of the .zip file (%let zipfile=). Next, the X command is used to tell WinZip to execute. The following options are used in the X command statement: 1. –a adds a file, folder, etc. 2. –r is used to include subfolders 3. –p is used to retain the original directory structure within the final .zip file.

```
%let zipfile = &CurPath.\Archive\Archive_&fdate..zip;
x " ""c:\program files\winzip\WINZIP32.EXE"" -a -r -p ""&zipfile.""
@""&CurPath.\Archive\Archive_&Sponsor._&Protocol._&fdate..txt"" ";
```

CONCLUSION

Everyone should always save their work for future use or reference. There isn't always time to go through each program and determine what supporting macros, files, documents, etc. are needed to recreate specific output. Saving the entire directory structure and all contents within that directory is a fool proof method of ensuring that a point in time has been captured and can be recreated with two clicks of a button.

REFERENCES

SAS Institute Inc. 2005. SAS OnlineDoc® 9.1.3. Cary, NC: SAS Institute Inc.

Fairfield-Carter, Brian and Hunt, Stephen (2007), 'Zipping Right Along: Push button SAS® Transfers via Command-line Invocation of the WinZip32 Executable', Proceedings of the 2007 SAS Global Forum Conference.

CONTACT INFORMATION

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

Name: Kristen Harrington
Enterprise: Rho, Inc.
Address: 6330 Quadrangle Drive Suite 500
City, State ZIP: Chapel Hill, NC 27517
Work Phone: 919-599-6377
Fax: 919-408-0999
E-mail: Kristen_Harrington@RhoWorld.com
Web: www.rhoworld.com