MWSUG 2017 - Paper TT10

Fully Automated Updating of Arbitrarily Complex Excel Workbooks

David Oesper, Lands' End, Dodgeville, Wisconsin

ABSTRACT

You can generate some very sophisticated Excel workbooks using ODS EXCEL and ODS TAGSETS.EXCELXP, but sometimes you'll want to create your Excel workbook in Microsoft Excel or someone else will provide it to you. I'll show you how you can use SAS® to dynamically update (and distribute) any existing Excel workbook with no manual intervention required. You'll need only Base SAS 9.4 TS1M2 or later and SAS/ACCESS to PC Files to use this approach. Examples in both the Linux and Windows operating environments will be presented.

INTRODUCTION

As useful as ODS EXCEL® is (and it is getting better all the time), you may encounter a situation where you need to update an existing Excel workbook that has been created by you or someone else, and that you would rather not try to replicate with SAS® because of its complex formatting. For example, you may find yourself needing to refresh an Excel worksheet that is not a simple grid of rows and columns. We can automate this refresh process by first creating a static version of the Excel workbook we wish to update and adding to it one or more hidden data sheets that will be referenced. Each time we wish to apply an update, we'll clone the static workbook to create our working copy, and replace the data sheets in that working copy using the LIBNAME XLSX access method that became available in the SAS/ACCESS to PC Files product beginning with release SAS 9.4 TS1M2. Finally, we'll save the updated Excel workbook to the destination of our choice and/or email it to a list of recipients.

STEP 1: CREATE A MODEL EXCEL WORKBOOK WITH REFERENCED DATA SHEETS

Create a version of your Excel workbook that adds one or more data sheets of simple rows and columns. Then change each static data value on your main sheet to a reference to a cell on a data sheet. For each data sheet, the rows will correspond to observations and the columns will correspond to variables in an update SAS dataset.

It is beyond the scope of this paper to illustrate how to add data sheets and cell references to a Microsoft Excel workbook. Information about how to do this is readily available from various internet sources such as GCFLearnFree.org® (GCF LearnFree.org 2017).

STEP 2: CLONE THE MODEL EXCEL WORKBOOK TO CREATE A WORKING COPY

In this example, our model Excel workbook is called **U.S. Census Divisions.xlsx** and is located in the Linux directory /sasdevr/devr180/. We'll be making a copy and putting it over in /sasdevr/devr7 using Linux system commands. First, we define macro variables for the input and output filenames and pathnames at the top of our program:

```
/* Path and File Name of model Excel workbook */
%let xlinpath  = /sasdevr/devr180;
%let xlinfile  = U.S. Census Divisions.xlsx;

/* Path and File Name of output Excel workbook */
%let xloutpath = /sasdevr/devr7;
%let xloutfile = U.S. Census Divisions 20170807.xlsx;
```

Next, we use CALL SYSTEM commands to perform the copy:

```
/* Copy the Excel Workbook */
data _null_;
   call system("cd &xlinpath");
   call system("cp -f %unquote(%bquote(')&xlinfile%bquote(')) &xloutpath");
   call system("cd &xloutpath");
   call system("mv %unquote(%bquote(')&xlinfile%bquote('))
                   %unquote(%bquote(')&xloutfile%bquote('))");
run;
```

In SAS for Windows, the copy process looks like this:

```
/* Path and File Name of model Excel workbook */
%let xlinpath  = C:\Users\David\Documents\SASIN;
%let xlinfile  = U.S. Census Divisions.xlsx;

/* Path and File Name of output Excel workbook */
%let xloutpath = C:\Users\David\Documents\SASOUT;
%let xloutfile = U.S. Census Divisions 20170807.xlsx;

/* Copy the Excel Workbook */
options noxwait noxsync;
data _null_;
   call system(%unquote(%bquote(')
   copy "&xlinpath\&xlinfile" "&xloutpath\&xloutfile" %bquote(')));
run;
options xwait xsync;
```

STEP 3: UPDATE THE WORKING COPY OF YOUR EXCEL FILE WITH NEW DATA SHEETS

You'll need to create a SAS dataset that has the same number of rows (observations) and columns (variables) as each Excel data sheet you are going to replace. In this example, you are going to replace a data sheet named **Data** with the SAS dataset **sasuser.newpop**. While a BY variable isn't explicitly required if both your data sheet and SAS dataset are in the same sort order, it helps ensure that each row of your data sheet is replaced with the correct observation from your SAS dataset. Incidentally, since your data sheet is hidden in the model Excel workbook, it will remain hidden after you've made your updates.

The data sheet to be updated in this example is in the Excel workbook defined by LIBNAME xlout and the data sheet name is 'Data'.

```
/* Use LIBNAME XLSX access method */
libname xlout xlsx "&xloutpath/&xloutfile";
/* Update data sheet with a merge */
data xlout.'Data'n;
   merge xlout.'Data'n sasuser.newpop;
   by statename;
run;
```

STEP 4: SAVE YOUR UPDATED EXCEL WORKBOOK

To copy your updated Excel workbook from a Linux directory to a Windows directory, we can use the FILENAME FTP access method and a byte copy program (Hemedinger 2011).

```
/* FILENAME same as output LIBNAME */
filename xlout "&xloutpath/&xloutfile";

/* FILENAME for FTP to Windows drive */
filename ftpout ftp "DIRMARKET124/BA/Staff/Dave/SASOUT/&xloutfile" recfm=s
                    host='********' user='********' pass='********';

/* Copy the file byte-for-byte */
data _null_;
   length filein 8 fileid 8;
   filein = fopen('xlout','I',1,'B');
   fileid = fopen('ftpout','O',1,'B');
   rec = '20'x;
   do while(fread(filein)=0);
      rc = fget(filein,rec,1);
      rc = fput(fileid,rec);
      rc = fwrite(fileid);
   end;
   rc = fclose(filein);
   rc = fclose(fileid);
run;
```

STEP 5: EMAIL YOUR UPDATED EXCEL WORKBOOK

To email your updated Excel workbook, use the FILENAME EMAIL access method. Make sure you only send an email if your program runs without error!

```
/* Output FILENAME uses the EMAIL access method - notice we are */
/* emailing the Excel workbook we created on Linux              */
filename eml email to = ('Dave.Oesper@landsend.com')
                    cc = ('DaveOesper@mac.com')
                    type = 'text/plain'
                    emailid = 'Dave.Oesper@landsend.com'
                    attach = ("&xloutpath/&xloutfile"
                              ct="application/octet-stream" ext="xlsx")
                    subject = 'Updated U.S. Census Data';

/* Send email, but only if no errors occur in the program */
%macro sendemail;
%if %length(&syserrortext) ne 0 %then %return;
data _null_;
   file eml;
   put "Here are the updated population estimates
        for each state from the U.S. Census
        Bureau." /;
   put 'Best Regards,' /;
   put 'Dave';
 run;
%mend sendemail;

%sendemail;
```

CONCLUSION

We used the LIBNAME XLSX engine, a part of SAS/ACCESS to PC File Formats, to update data sheets in an Excel workbook using a simple DATA step MERGE. We then used FILENAME FTP to copy our updated Excel workbook from a Linux server to a Windows directory. Finally, we used FILENAME EMAIL to email our updated Excel workbook to a recipient list. This gives us a versatile and automated way to update and distribute Excel workbooks that were originally created using Microsoft Excel.

REFERENCES

GCF LearnFree.org®. 2017. "Using cell references with multiple worksheets." Accessed August 2, 2017. Available at https://www.gcflearnfree.org/excel2013/relative-and-absolute-cell-references/3/.

Hemedinger, Chris. "How to use SAS DATA step to copy a file from anywhere." SAS Blogs. June 17, 2011. Available at http://blogs.sas.com/content/sasdummy/2011/06/17/how-to-use-sas-data-step-to-copy-a-file-from-anywhere/.

CONTACT INFORMATION

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

David Oesper
Lands' End
Dave.Oesper@landsend.com