Paper RF-04-2013

Copy and Paste from Excel to SAS®

Arthur S. Tabachneck, Ph.D., myQNA, Inc., Thornhill, Ontario Canada
Matthew Kastin, I-Behavior, Inc., Penn Valley, PA

ABSTRACT

Most, if not all, of us are far too familiar with the problems that one can confront when trying to import an Excel workbook into a SAS® dataset. Numeric fields might be imported as character fields, dates might be imported as either character fields or dates that are sixty (60) years earlier than what they actually represent, and fields containing time values might be imported as representing one-half of a second when they actually represent 43,200 seconds. While such discrepancies can be corrected by following the use of PROC IMPORT with a carefully written datastep, the present paper presents an alternative that only requires one step, uses less code, only requires base SAS and, on our test data, ran almost fifty (50) times faster than using PROC IMPORT.

THE PROBLEM

One could easily write an entire book describing the various problems a SAS user might confront when attempting to import an Excel workbook. For example, relevant headers might span across multiple columns or simply not exist for some columns, date, time and datetime values might be imported as character variables or import as dates that are significantly different from what the values they actually represent, and values that were formatted in Excel might totally lose their assigned formats. And, while SAS users had more than enough problems to potentially confront with older versions of Excel and SAS, the additional complexity of having to deal with 32 and 64 bit mismatches between the two systems has made such efforts even more challenging.

For recurring tasks, of course, the best choice would be to discover, refine and ultimately implement a method that repeatedly obtains the desired result by submitting a set of proven code. However, not all tasks fit into that category. Sometimes one is confronted with the need to analyze a particular data table, and have to accomplish the task without the benefit of an easy to use proc. Such a situation can arise for numerous reasons, for example when a proc simply doesn't exist for accomplishing the particular task, a site doesn't license an add-on product that would be needed for a proc to work as needed, or when a table simply doesn't conform to a proc import's design specifications.

Some examples that immediately come to mind are if you have to import a table from: (1) an Excel 2010 workbook, but you are still using SAS 9.13; (2) an Excel workbook, but you don't have either SAS/ACCESS Interface to PC file formats or SAS/ACCESS Interface to ODBC; or (3) a table that is embedded in a Word document or web page.

THE TEST DATA

Two Excel workbooks were created by running the following two sets of code:

```
data short (keep=field204 quarter product_group quantity profit);
  length a1-a8 $40;
  set sashelp.orsales ;
  length field204 $204;
  a1="This represents the first 40 characters";
  a2="of a silly long sentence we created for";
  a3="testing whether the various methods for";
  a4="creating SAS datasets by using datastep";
  a5="methods to input tables from a system's";
  field204=catx(" ",of a1-a5);
run;

PROC EXPORT DATA= short
            OUTFILE= "c:\short.xlsx"
            DBMS=XLSX REPLACE;
run;
```

```
data long (keep=field327 quarter product_group quantity profit);
  length a1-a8 $40;
  set sashelp.orsales ;
  length field327 $327;
  a1="This represents the first 40 characters";
  a2="of a silly long sentence we created for";
  a3="testing whether the various methods for";
  a4="creating SAS datasets by using datastep";
  a5="methods to input tables from a system's";
  a6="clipboard. We broke the sentence into 8";
  a7="40 character strings so we could easily";
  a8="show what we were including in our test";
  field327=catx(" ",of a1-a8);
run;

PROC EXPORT DATA= long
            OUTFILE= "c:\long.xlsx"
            DBMS=XLSX REPLACE;
run;
```

The actual starting point for the import methods described in the present paper is using Excel to open one of the workbooks created by the above code and then entering the following the control key combinations: CTRL-A, followed by entering CTRL-C. The first action selects all of the workbook's columns and rows, and the second copies the selected cells to the computer's clipboard.

Then, while the table is in the clipboard, the code for the particular method is run.

Also, while a table was still in the clipboard, we opened a copy of Microsoft Word, pasted the table into the Word document (i.e., entered the following control key combination: CTRL-V), and then used the following instructions from the Microsoft Word documentation to select and copy the table to our system's clipboard:

> In Print Layout view, rest the pointer on the table until the table move handle ⊞ appears
>
> Click the table move handle to select the table.
>
> Press CTRL+C.

Then, while the table was in the clipboard, the code for the particular method was run.

THE SOLUTIONS

The Clipboard Access Method. The Clipboard Access Method was introduced in Version 9, is mentioned in the SAS documentation and, at first glance, looks like the easiest method to use. Unfortunately, while not mentioned in the documentation, the *lrecl* and *notab* options can't be used, thus record length is limited to 256 characters and tab delimiters will automatically be converted to six spaces, thus a normal input statement can't be used if there are any missing cells. However, if all of a table's rows were less than 257 characters, the following code could import from the clipboard regardless of whether the data had been copied from Excel or Word:

```
filename clippy clipbrd;
data orsales_short;
  attrib
   Quarter label='Quarter'  length=8 informat = yyq6. format=yyq6.
   Product_Group label='Group' length=$ 25 informat= $25. format=$25.
   Quantity label='Number of Items' length=8 informat=6. format= 6.
   Profit label='Profit in USD' length=8 informat=12. format=12.2
   Field163 label='Short' length=$ 163 informat=$163. format= $163.;
  infile clippy missover firstobs=2;
  input;
  _infile_ = transtrn( trim( _infile_ ) , '      ' , '09'x );
  quarter       = input( scan( _infile_ , 1 , '09'x ,'m') , yyq6. ) ;
  product_group =        scan( _infile_ , 2 , '09'x ,'m')           ;
  quantity      = input( scan( _infile_ , 3 , '09'x ,'m') , 6.    ) ;
  profit        = input( scan( _infile_ , 4 , '09'x ,'m') , 12.   ) ;
  field163      =        scan( _infile_ , 5 , '09'x ,'m')           ;
run;
filename clippy clear;
```

The DDE/Clipboard Method. The DDE/Clipboard Method requires less code than the Clipboard Access Method, but requires specifying one's input statement slightly differently. This method will work for longer records, but can only be used on systems that support Dynamic Data Exchange, and we were only able to run the method successfully using a clipboard that was copied from an Excel workbook.

```
filename clippy dde 'clipboard';
data want;
  infile clippy lrecl=400 dsd notab missover dlm='09'x firstobs=2;
  informat long_field $char327.;
  informat quarter yyq6.;
  format quarter YYQ6.;
  informat product_group $char24.;
  input quarter product_group quantity profit long_field;
run;
filename clippy clear;
```

Using functions in a datastep. This method was the most versatile of the three methods. The method should work on any system that allows one to copy data to its clipboard and worked for longer records, records with missing cells, and with clipboards created by copying from either Excel workbooks and Word documents.

```
data want;
  attrib _inline_ length = $ 32767
   Quarter label='Quarter'  length=8 informat = yyq6. format=yyq6.
   Product_Group label='Group' length=$ 25 informat= $25. format=$25.
   Quantity label='Number of Items' length=8 informat=6. format= 6.
   Profit label='Profit in USD' length=8 informat=12. format=12.2
   field327 label='Long' length=$ 327 informat=$327. format= $327.;
  keep quarter product_group quantity profit field327;
  rc = filename( 'clippy' , ' ' , 'clipbrd' );
  if ( rc  ne 0 ) then link err;
  fid = fopen( 'clippy' , 's' , 32767 , 'V' );
  if ( fid eq 0 ) then link err;
  do _n_=1 by 1 while( fread( fid ) = 0 );
    rc = fget( fid , _inline_ , 32767 );
    _inline_ = transtrn( trim( _inline_ ) , '      ' , '09'x );
    if _n_=1 then continue;
    quarter       = input( scan( _inline_ , 1 , '09'x,'m' ), yyq6. ) ;
    product_group =        scan( _inline_ , 2 , '09'x,'m')          ;
    quantity      = input( scan( _inline_ , 3 , '09'x,'m'), 6.     ) ;
    profit        = input( scan( _inline_ , 4 , '09'x,'m'), 12.    ) ;
    field327      =        scan( _inline_ , 5 , '09'x,'m')          ;
    output;
  end;
  rc = fclose( fid );
  rc = filename( 'clippy' );
  stop;

  err:
    do;
      m = sysmsg();
```

```
          put m;
          stop;
        end;
    run;
```

SUMMARY AND CONCLUSION

Three methods were presented for accomplishing one-time imports of data from Excel workbooks and tables embedded in Word documents. The Clipboard Access Method can be used to import both Excel Workbooks and Word tables, but only where no individual record is longer than 256 characters. Additionally, since one can't include a *notab* option in the filename statement, input statements can't be defined according to normal standards in order to account for potential missing cells.

The DDE/Clipboard method was able to overcome the 256 character limitation, but can only be used to import data that had been copied from Excel workbooks, can only be used on systems that support Dynamic Data Exchange, and is a quite finicky regarding allowable variable informats

Using functions in a datastep was the most versatile of the three methods. The method worked for longer records, worked with clipboards created by copying data from either Excel workbooks or Word documents, and should work on any system that allows one to copy data to its clipboard.

DISCLAIMER

The content of this paper is the work of the authors and does not necessarily represent the opinions, practices or recommendations of the authors' organizations.

CONTACT INFORMATION

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

Arthur Tabachneck, Ph.D., President
myQNA, Inc. Thornhill, ON Canada
E-mail: atabachneck@gmail.com

Matthew Kastin
I-Behavior, Inc.
Penn Valley, PA
E-mail: matthew.kastin@gmail.com

REFERENCES

COPY AND PASTE ALMOST ANYTHING, Tabachneck, A., Herbison, R., King, J., DeVenezia, R., Derby, N., and Powell, B., SGF 2012 Proceedings, http://support.sas.com/resources/papers/proceedings12/238-2012.pdf

FILENAME, CLIPBOARD Access Method, SAS 9.2 Documentation, SAS Institute 2012, http://support.sas.com/documentation/cdl/en/lrdict/64316/HTML/default/viewer.htm#a002571877.htm