

Paper 040-30

A Macro for Importing Multiple Excel Worksheets into SAS® Data Sets

Helen Sun, Robarts Clinical Trials, London, ON, CANADA

Cindy Wong, Robarts Clinical Trials, London, ON, CANADA

ABSTRACT

Microsoft Excel is a widely available application and is commonly used to capture and transmit data. These data are often imported to SAS for manipulation and analysis. Many methods exist to import data from Excel to SAS, such as PROC IMPORT, IMPORT WIZARD and ODBC. In this paper, we introduce a macro, %xl2sas, which uses the powerful Dynamic Data Exchange (DDE) technique¹ to import all the worksheets within an Excel workbook by one invocation. %xl2sas can automatically handle a variable number of worksheets and generate separate SAS datasets without specifying the name of each worksheet as required in the method presented by Qi². This SAS macro was initially developed as a solution to facilitate data transfers in a clinical trials setting. As Excel files often contain multiple worksheets, we believe %xl2sas is a flexible and user-friendly tool that has wide applicability for the importing of Excel to SAS.

INTRODUCTION

PROBE (Program to Reduce Orthopedic Blood Exposure), sponsored by the Ontario Ministry of Health and coordinated by Robarts Clinical Trials, is a study to evaluate the effectiveness and cost-effectiveness of a comprehensive blood conservation program in patients undergoing primary hip surgery. Twenty-nine Ontario hospitals were randomly allocated to implement the program (intervention sites) or to continue usual care (control sites). To monitor the performance of the intervention sites, the hospitals were asked to submit their data weekly in a log form (LOG data). The LOG data were maintained in an Excel file with each site's data kept in a separate worksheet.

Although the outcome of the clinical trial is to be based on data extracted from patients' medical charts (CHART data) in both intervention and control sites, some of the data captured on the log form are specific to the implementation of the blood conservation program and are needed for the economic analysis. The health economist requested that the data be placed in a single Excel worksheet containing both CHART and LOG data with specific data requirements. Several interim data transfers were made to the health economist prior to the completion of the study.

%xl2sas was developed to automate the import of the LOG data from Excel to SAS. This macro builds on Vyerman's powerful DDE technique¹. As the number of worksheets could vary for each interim transfer, %xl2sas handles the importing of multiple worksheets programmatically in one invocation. SAS was then used to combine the LOG data with CHART data, manipulate the data to meet the health economist's data requirements, and export the data back to Excel as a single worksheet.

This paper presents the SAS codes of %xl2sas, the macro used to automate the data transfer process for the PROBE study. The %xl2sas macro was tested on SAS Version 8.2 and Windows 2000 Professional.

THE %XL2SAS MACRO

The syntax of macro %xl2sas is as follows:

```
%macro xl2sas(Path=,
              File=,
              StartRow=1,
              StartCol=1,
              EndRow=,
              EndCol=,
              GetVarName=1);
```

Parameters Path and File are mandatory. Other parameters are optional. If EndRow and EndCol are not specified, the numbers of rows and columns of each worksheet will be obtained. GetVarName is an indicator variable to denote if variable names are to be retrieved from a row in the worksheet. The default value of GetVarName is 1 (i.e. true), which means variable names will be taken from StartRow number with data values to follow in subsequent rows. If the value of GetVarName is 0 or missing (i.e. false), StartRow number contains the first row of data and default variable names COL1, COL2, etc. will be used.

For the PROBE study, the %xl2sas macro was invoked as follows:

```
%xl2sas(path=S:\Probe3\SAS\EXCEL\,file=log.xls);
```

Now let us walk through the SAS codes in detail.

1. LAUNCH THE EXCEL APPLICATION

To launch the Excel application, we used Roper's method³:

```

%*launch Excel;
options noxsync noxwait noxmin;
%local excelstarted;
filename excelchk dde 'excel|system' command;
%let excelstarted = %sysfunc(fopen(excelchk,S));
data _null_;
  length fid rc start stop time 8;
  fid=&excelstarted;
  if fid le 0 then do;
    rc=system('start excel');
    start=datetime();
    stop=start+10;
    do while (fid le 0);
      fid=fopen('excelchk','s');
      time=datetime();
      if (time ge stop) then fid=1;
    end;
  end;
end;
run;
%let rc = %sysfunc(fclose(&excelstarted));
filename excelchk clear;

```

2. OBTAIN THE NUMBER OF WORKSHEETS AND WORKSHEET NAMES

We first opened the Excel workbook. We then modified Vyverman's macro⁴ %loadname to obtain the number of worksheets (stored in macro variable &nsheets) and their names (stored in macro variable &SheetNames). Please note that worksheet names may contain special characters such as a space or a period. However the codes presented here have not allowed for special characters that could conflict with SAS codes such as a comma or a quotation mark.

```

%*open the excel file and insert macro sheet;
filename xl2sas dde 'excel|system' lrecl=200;
data _null_;
  length string $200;
  file xl2sas;
  string='[open("||"&path.&file."||"',0,false)]';
  put string;
  put '[workbook.next()]';
  put '[workbook.insert(3)]';
  string='[workbook.move("Macro1","||"&file"||"',1)]';
  put string;
run;
filename xlmacro dde "excel|macro1!r1c1:r1000c1" notab lrecl=200;

%*initialize local macro variables;
%local nsheets sh wn i sheet xlrows xlcols EndNewRow EndNewCol
      MaxCol VarWds ind rowi row1 row2 StartRowNew VarNames VarNum;

%*get number of sheets and put in &nsheets;
%let nsheets=;
data _null_;
  file xlmacro;
  put '=select("r1c2:r1c2)';
  put '=set.name("nsheet",selection())';
  put '=set.value(nsheet,get.workbook(4))';
  put '=halt(true)';
  put '!dde_flush';
  file xl2sas;
  put '[run("macro1!r1c1)]';
  put '[error(false)]';
run;
filename nsheets dde "excel|macro1!r1c2:r1c2" notab lrecl=200;
data _null_;
  length nsheets 8;
  infile nsheets;

```

```

        input nsheets;
        call symput('nsheets',trim(left(put(nsheets,2.))));
run;
%let nsheets=%eval(&nsheets-1);
filename nsheets clear;

%*clear macro sheet;
data _null_;
    file xl2sas;
    put '[workbook.activate("macro1")]';
    put '[select("r1c1:r1000c2")]';
    put '[clear(1)]';
    put '[select("r1c1")]';
run;

%*get names of sheets and put in &SheetNames;
data _null_;
    length MacCmd $200;
    file xlmacro;
    %do sh=1 %to &nsheets; /*choose one sheet at a time*/
        MacCmd="=select(!$b$&sh,!$b$&sh)";
        put MacCmd;
        put '=set.name("cell",selection())';
        %do wn=1 %to &sh;
            put '=workbook.next()'; /*move to current sheet*/
        %end;
        put '=set.value(cell,get.workbook(3))'; /*get name of current sheet */
        put '=workbook.activate("Macro1)';
    %end;
    put '=halt(true)';
    put '!dde_flush';
    file xl2sas;
    put '[run("macro1!r1c1")]';
    put '[error(false)]';
run;

filename sheets dde "excel|macro1!r1c2:r&nsheets.c2" lrecl=200;
data _sheet_names;
    length junk SheetName $256;
    infile sheets delimiter=' ';
    input junk SheetName;
    drop junk;
run;
filename sheets clear;
data _null_;
    file xl2sas;
    put '[workbook.activate("macro1")]';
    put '[select("r1c1:r1000c2")]';
    put '[clear(1)]';
    put '[select("r1c1")]';
run;

proc sql noprint;
    select SheetName into: SheetNames separated by '@'
    from _sheet_names;
quit;
proc datasets nolist;
    delete _sheet_names;
run;

```

3. IMPORT ALL WORKSHEETS INTO SAS DATASETS

We first collected the numbers of rows and columns for each worksheet. Next we checked the width of each column using the method posted by Vyverman⁵ and stored it in a macro variable &VarWds. We then imported all the data in each worksheet into a separate SAS dataset. The SAS datasets were named SHEET1, SHEET2, etc., one dataset for each worksheet in the Excel workbook. The worksheet name was stored in the first variable named SHEETNAME

in each dataset. All columns were imported as character variables. Please note that since special characters are not allowed in SAS variable names, they were first removed from the column headings. After the special characters are removed, column headings should contain unique names or names that begin with an alphabet (or an underscore), otherwise default names COL1, COL2, etc. are used for SAS variable names.

```

%do i=1 %to &n sheets; /*for every sheet*/
%let sheet=%scan(&SheetNames,&i,@);
%*get # rows and # cols of sheet;
data _null_;
  length ddestring $ 200;
  file xlmacro;
  put '=select("r1c2:r1c2")';
  put '=set.name("rows",selection())';
  put '=select("r2c2:r2c2")';
  put '=set.name("cols",selection())';
  ddestring='=workbook.activate("'||"&sheet"||'")';
  put ddestring;
  put '=set.value(rows,get.document(10))'; /*last row number of sheet*/
  put '=set.value(cols,get.document(12))'; /*last column number of sheet*/
  put '=workbook.activate("macro1")';
  put '=halt(true)';
  put '!dde_flush';
  file xl2sas;
  put '[run("macro1!r1c1")]';
  put '[error(false)]';
run;

filename xlrows dde "excel|macro1!r1c2:r1c2" lrecl=200;
filename xlcols dde "excel|macro1!r2c2:r2c2" lrecl=200;
%let xlrows=0;
data _null_;
  length xlrows 8;
  infile xlrows;
  input xlrows;
  call symput('xlrows',trim(left(put(xlrows,5))));
run;
%if &xlrows=0 %then %goto exit; /*if this sheet is empty, no more processing*/
%let xlcols=1;
data _null_;
  length xlcols 8;
  infile xlcols;
  input xlcols;
  call symput('xlcols',trim(left(put(xlcols,5))));
run;
filename xlrows clear;
filename xlcols clear;

%if &EndRow= %then %let EndNewRow=&xlrows; %else %let EndNewRow=&EndRow;
%if &EndCol= %then %let EndNewCol=&xlcols; %else %let EndNewCol=&EndCol;
  %put EndNewRow=&EndNewRow EndNewCol=&EndNewCol;
%*clear macro sheet;
data _null_;
  file xl2sas;
  put '[workbook.activate("macro1")]';
  put '[select("r1c1:r1000c2")]';
  put '[clear(1)]';
  put '[select("r1c1")]';
run;

%*get width of each column and put in &xlcols+1 column in &sheet;
%*since dates in Excel can take on length from 5 to 10, extend length to 11
  for dd-mmm-yyyy format;
%let MaxCol=%eval(&xlcols+1);
%let VarWds=; /*width of columns*/

```

```

%do ind=&StartCol %to &EndNewCol;
  %let rowi=%eval(&StartRow+&ind);
  data _null_;
    file xlmacro;
    length ddestring $200 c1 c2 $1;
    /*switch to Excel Application as front end*/
    put '=app.maximize()';
    ddestring ='=app.activate("Microsoft Excel - '||"&file"||'")';
    put ddestring;
    ddestring ='=workbook.activate("'||"&sheet"||'")';
    put ddestring;

    chars='ABCDEFGHJKLMNOPQRSTUVWXYZ';
    if floor(&ind/27) then c1=substr(chars,floor(&ind/27),1);
    else c1=' ';
    if mod(&ind,26) then c2=substr(chars,mod(&ind,26),1);
    else c2='Z';

    ddestring ='=select("r' ||trim(left("&rowi"))||'c' ||"&MaxCol"||'")';
    put ddestring;
    ddestring='=send.keys("=max({}len({}' ||trim(left(c1))||trim(left(c2))
    ||"&StartRow.:" ||trim(left(c1))||trim(left(c2))||"&xlrws"||'{}{}))")';
    put ddestring;
    put '=send.keys("^+{return}")';
    put '=halt(true)';
    put '!dde_flush';
    file xl2sas;
    put '[run("macro1!r1c1")]';
  run;
%end;

%*clear macro1 sheet;
data _null_;
  file xl2sas;
  put '[workbook.activate("macro1")]';
  put '[select("r1c1:r1000c2")]';
  put '[clear(1)]';
  put '[select("r1c1")]';
run;

%let row1=%eval(&StartRow+&StartCol);
%let row2=%eval(&StartRow+&EndNewCol);
filename xlmax dde "excel|&sheet!r&row1.c&MaxCol:r&row2.c&MaxCol" lrecl=200;
data _xlmax_;
  length xlmax 8 ;
  infile xlmax;
  input xlmax;
  if xlmax=0 then xlmax=1;
  else if xlmax<=10 then xlmax=11;
run;
filename xlmax clear;

%*put widths of columns to &VarWds;
proc sql noprint;
  select xlmax into: VarWds separated by ' '
  from _xlmax_;
quit;
proc datasets nolist;
  delete _xlmax_;
run;
%if &VarWds= %then %goto exit;

%*put all variable names to &VarNames;
%if &GetVarName %then %do;
  %* get names and numbers from start row;
  data _null_;
    length ddestring $100;

```

```

        ddestring="'excel|&sheet!r&StartRow.c&StartCol.:r&StartRow.c&EndNewCol'";
        call symput('ddestring',ddestring);
    run;
    filename GetName dde &ddestring notab lrecl=2000;
    data _VarName_;
        length TempVar $200;
        infile GetName dlm='09'x ;
        input TempVar @@;

        /*remove special characters, only keep A-Z, a-z and 0-9;
        len=length(trim(left(TempVar)));
        length var $32; /*truncate to 32 if longer*/
        do i=1 to len;
            if not verify(substr(trim(left(TempVar)),i,1),
                '_0123456789abcdedfghijklmnopqrstuvwxyzABCDEFGHIJKLMNPOQRSTUVWXYZ')
                then var=trim(var)||substr(trim(left(TempVar)),i,1);
        end;
        /*if the compressed variable name is empty, then name it as Col#;
        if var=' ' then var='Col' || compress(put(_n_,best3.));
        /*if the compressed variable name begins with a number,
        then add underscore before it;
        if indexc('0123456789',substr(trim(left(var)),1,1))
        then var='_' || trim(left(var));
        drop len i TempVar;
    run;
    /*check for duplicate names, if duplicate name, rename as Col#;
    proc transpose data=_VarName_ out=_VarNamet;
        var var;
    run;
    proc sql noprint;
        select count(*)
        into: VarNum
        from _VarName_;
    quit;
    data _VarNamet;
        set _VarNamet;
        %do ind=1 %to &VarNum-1;
            %do indj=&ind+1 %to &VarNum;
                if col&ind=col&indj then col&indj="Col&indj";
            %end;
        %end;
    run;
    proc transpose data=_VarNamet out=_VarName;
        var %do ind=1 %to &VarNum; Col&ind %end;;
    run;
    /*put all variable names to &VarNames;
    proc sql noprint;
        select var
        into: VarNames separated by ' '
        from _VarName;
    quit;
    filename GetName clear;
    proc datasets nolist;
        delete _VarName_ _VarNamet _VarName;
    run;
    %let StartRowNew=%eval(&StartRow+1);
%end;
%else %do; /*simply name variables as Col1, Col2, etc*/
    %let VarNames=;
    %let VarNum=%eval(&EndNewCol-&StartCol+1);
    %do ind=1 %to &VarNum;
        %let VarNames=&VarNames Col&ind;
    %end;
    %let StartRowNew=&StartRow;
%end;
%if &VarNum=0 %then %goto exit;

```

```

%*if length of variable is 1 (the column is empty) and variable name is Col#,
  then put it to &DelCols to be deleted;
%let DelCols=;
%do ind=1 %to &VarNum;
  %if %scan(&VarWds,&ind)=1 and %index(%scan(&VarNames,&ind),Col)=1
    %then %do;
      %let DelCols=&DelCols %scan(&VarNames,&ind);
    %end;
%end;

%*input data to SAS;
data _null_;
  length ddestring $100;
  ddestring="'excel|&sheet!r&StartRowNew.c&StartCol.:r&EndNewRow.c&EndNewCol'";
  call symput('ddestring',ddestring);
run;
filename indata dde &ddestring notab lrecl=2000;
data sheet&i;
  length SheetName $30;
  SheetName="&sheet";
  infile indata notab dsd dlm='09'x missover;
  length %do ind=1 %to &VarNum;
    %scan(&VarNames,&ind) $%scan(&VarWds,&ind)
  %end;;
  input &VarNames;
  %if &DelCols ne %then drop &DelCols;;

  %*delete observation if all variables are empty;
  if %do ind=1 %to &VarNum-1;
    %scan(&VarNames,&ind)=' ' and
    %end;
    %scan(&VarNames,&VarNum)=' '
  then delete;
run;
filename indata clear;

%exit;;
%end; /*end of %do i=1 %to &n sheets "for every sheet"*/

```

4. CLOSE THE EXCEL APPLICATION

We then close the Excel application:

```

%*close the Excel file without saving it;
filename xlmacro clear;
data _null_;
  file xl2sas;
  put '[error(false)]';
  put '[close]';
run;
%*if Excel Application is opened by this program, close it;
%if &excelstarted=0 %then %do;
  data _null_;
    file xl2sas;
    put '[quit]';
  run;
%end;
filename xl2sas clear;
%mend;

```

APPLICATION OF %XL2SAS TO PROBE STUDY

LOG data from the PROBE study sites were kept in separate worksheets in an Excel workbook. The site identifier was used to name each worksheet. %xl2sas was run to import the worksheets into SAS datasets. The SAS datasets were named SHEET1, SHEET2, etc., one dataset for each study site as ordered in the Excel workbook. In each SAS dataset, the worksheet name (i.e. the site identifier) was stored in the first variable named SHEETNAME.

The SAS datasets, SHEET1, SHEET2, etc. were then combined into one dataset and merged with the CHART data for manipulation. %xl2sas imported all columns as character variables to ensure that all values in columns containing mixed data types were moved to SAS as they appear in Excel. In our situation, further data manipulations were required to be performed in SAS nevertheless to meet the health economist's data requirements. Date variables were separated into three (Year, Month, Day) fields, numeric variables were converted from character type back to numeric type, and variables were encoded as required, prior to exporting the data back to Excel (codes not shown).

LIMITATIONS

There are several restrictions with %xl2sas. First, certain special characters are not allowed in the worksheet name such as a comma or a quotation mark. Further work will be required to automate the removal of these characters using SAS. Therefore prior to running %xl2sas, the user will need to review the worksheet names and if necessary rename them in Excel. Second, all data in the Excel file are imported as character variables although some columns may contain all dates or all numeric data. Again, further work will be required to first check the data type of each cell within a column to determine the data type for the column. However the current approach is a simple way to ensure that all data in a column will be transferred to SAS without any values being defaulted to null. The variable types can be easily converted in SAS if needed. Third, the names of the SAS datasets are set to SHEET1, SHEET2, etc. with the name of the Excel worksheet stored in the first variable SHEETNAME in the dataset. This was adequate for our purpose as SHEET1, SHEET2, etc. were created only temporarily to be combined later into a single dataset. To give the SAS datasets more meaningful names, SHEET# can be easily renamed to the value of the first variable in the dataset. The first variable in the dataset can then be deleted to match the content of the Excel worksheet. Please note that worksheet names that are longer than 32 characters will be truncated due to the length limitation of SAS dataset names.

```
data _null_;
  set sheet1 (obs=1);
  call symput('dsn',substr(trim(translate(trim(left(
    compress(SheetName, ' )@!#$%^*:/-.'+))), '_',' '),1,32));
run;
data &dsn;
  set sheet1(drop=SheetName);
run;
```

Although the structure of the worksheets in our case was the same for all the study sites, this is not a requirement for %xl2sas. As Excel worksheets often contain different kinds of data and hence the SAS datasets are to be maintained separately, it would be useful to assign the same Excel worksheet names as the SAS dataset names.

CONCLUSION

%xl2sas provides a flexible way to import a variable number of worksheets in an Excel workbook into SAS datasets. The macro is built using the powerful DDE technique to import data from Excel to SAS. It illustrates an innovative application and adaptation of existing concepts and SAS codes in response to a specific data transfer need. Although %xl2sas was developed for a particular clinical trials situation, we believe it can be easily applied to other circumstances.

REFERENCES

1. Vyverman, K. "Using Dynamic Data Exchange to Export Your SAS® Data to MS Excel – Against All ODS, Part I". Proceedings of the 26th Annual SAS Users Group International Conference, Paper 011-26, 2001.
2. Qi, H. "A Practical Approach to Transferring Data from Microsoft Excel® to SAS® in Pharmaceutical Research". Proceedings of the 29th Annual SAS Users Group International Conference, Paper 082-29, 2003.
3. Roper, C. A. "Using SAS and DDE to execute VBA macros in Microsoft Excel". Proceedings of the 25th Annual SAS Users Group International Conference, Paper 098-25, 2000.
4. Vyverman, K. "Creating Custom Excel Workbooks from Base SAS® with Dynamic Data Exchange: A Complete Walkthrough". Proceedings of the 27th Annual SAS User Group International Conference, Paper 190-27, 2002.
5. Vyverman, K. "MS Excel column widths". Posted in Newsgroup: comp.soft-sys.sas, 2002-01-10.

CONTACT INFORMATION

Helen Sun
SAS Programmer
hsun@robarts.ca
Robarts Clinical Trials
Robarts Research Institute
P.O. Box 5015, 100 Perth Drive
London, Ontario, Canada
N6A 5K8
Tel: 519-663-3400
Fax: 519-663-3807

SAS and all other SAS Institute Inc. product or service names are registered trademarks or trademarks of SAS Institute Inc. in the USA and other countries. ® indicates USA registration.

Other brand and product names are trademarks of their respective companies.