

An Animated Guide©:
Sending SAS files to Excel

Concentrating on a D.D.E. Macro

Russ@Russ-Lavery.com

GOALS For This Presentation

- **A Quick comparison of various ways to send a SAS table to Excel**
- **Walk through a generic D.D.E. macro**

Sending Files to Excel

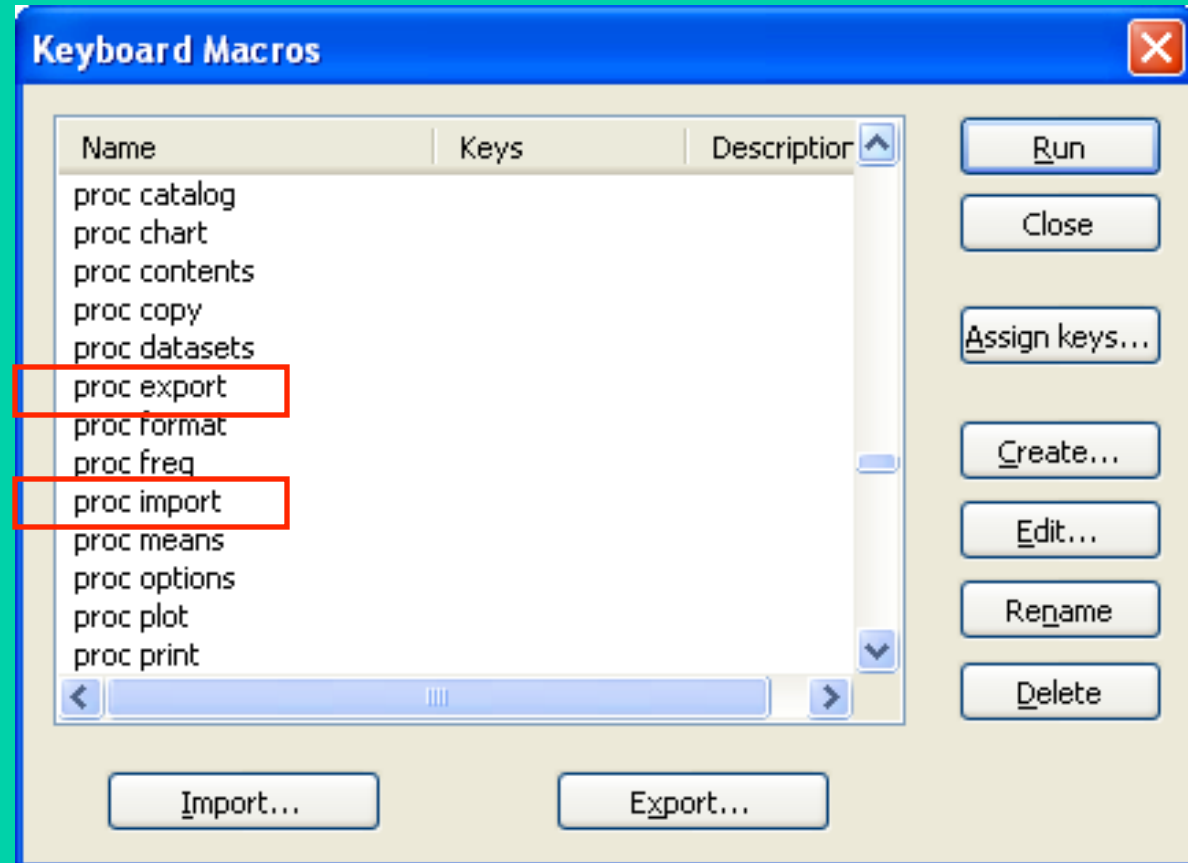
Many excellent papers address this topic

Each method has good/bad features

Method you use might be a company policy

Main deliverable
is a generic DDE
macro

Put this DDE
macro in your
abbreviation file



Sending Files to Excel

Proc Export

Fast - if in abbreviation file
Easily overwrites old workbooks
Can write several sheets
Writes other formats (csv, Access)
Not a lot of control - no formatting
Difficult to use with a template

Libname Engine

Fast - if in abbreviation file
Can write to pages or named ranges
Can write several sheets
Does not overwrite data
**Can have problems with data set
dimensions not agreeing with
dimensions of named ranges**
**Workbook is locked until the libref
is cleared**

Sending Files to Excel

DDE

Control

Easy to use with a template

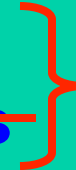
Can write several sheets

Fast - if in abbreviation file

Obsolete???

~~**Fussy syntax**~~

~~**Bad Error messages**~~



Sections of the generic D.D.E. macro

Declare parameters

Open Excel

Open Template - if one exists

Clean up working files

**Filter out ROWS of data on variables
that we may not want to send to Excel**

**Drop any extra vars. - Load # of variables we want to
export into NoOfVars - create a macro array**

Print the variables we want to export to the log

**Put the number of rows in a macro variable and give
a bit of notice if the file is getting large**

Issue the D.D.E. statement to dimension the “writing area”

**Use a data _null_ and a loop to write the D.D.E. put
statement**

**If &SaveNowYN =Y, perform any special writes, make
the summary sheet active, save and close**

Declare parameters

```
%MACRO Write2XlsV2( /*Write ONE SAS file to ONE Tab of an Excel Workbook*/
  FirstWrite=      /*A Y here will cause SAS to start Excel.
                    AND MUST be valued as Y for the Macro to open an Excel template
                    - used for first call fo XLS sheet*/

  ,SASFile=        /*File to send to the XLSDataTab Xls sheet*/
  ,whereClause=    /*a complete where statement eg:   %str(where 1=1;)   */
  ,DropVars=      /*a complete "drop data set option" with ()
                    and withOUT semicolon eg; (drop=SortOrder4Prod) */

  ,XLSDataTab=     /*Tab to which to write the table in SASFile*/
  ,XLSSmryTab=     /*Often,at the end of loading a template,
                    we write one or two pieces of info to a summary tab*/

  ,StartRowNo=5    /*You do not have to start writing in R1C1 -Look at the template-
                    See where the data section starts*/
  ,StartColNo=1    /*You do not have to start writing in R1C1 -Look at the template-
                    See where the data section starts- sometimes we start at column
2*/

  ,templateLoc=    /*path to the template. If you value this AND set firstWrite to Y
                    , the macro will open the template*/
  ,SaveNowYN=N     /*Saves & Closes Excel. After writing last tab in a multi-tab SS,
                    change to Y & enter path in NewName*/
  ,NewName=C:\Temp\DashBoard.xls /*save the XLS sheet under this name.
                    Saving only happens when SaveNowYN=Y */
);
```

Open Excel

```
%macro OpenExcel;  
  %if (&FirstWrite =Y) or (&FirstWrite =y) or (&FirstWrite =YES) %then  
    %do; /*This is the first time we write to the XLS sheet - we need to open it*/  
      options noxwait noxsync;  
      /* The ocmmand below will not work!  
      x "C:\Program Files\Microsoft Office97\Office\EXCEL.EXE";  
      Use the command dir /X to see the real path */  
      x "C:\Progra~1\MIFF2D~1\Office\EXCEL.EXE";  
      DATA _NULL_;  
      Rc = SLEEP(5);  
      RUN;  
    %end;  
  %mend OPenExcel;  
  %OpenExcel;
```

Open Template - if one exists

```
%macro OpenTemplate;
```

```
/*This macro can be used to write to several pages to a new SS, or template,  
and we ONLY open the Template: 1) on the first call and 2) if it exists
```

```
- We Must tell the macro if this is the first call and if the template exists*/
```

```
%if ( (%length(&templateLoc) GT 0 ) /*if GT 0, User has specified a template*/  
and ( %upcase(&FirstWrite =Y) or (%upcase(&FirstWrite) =YES ))  
    ) %then
```

```
%do;
```

```
/*if above is true, user has specified a template to which he wants to write*/
```

```
/*if no template specified, write to the sheet that opens when we open excel*/
```

```
Filename Excel dde 'EXCEL|SYSTEM';
```

```
data _null_;
```

```
file excel;
```

```
put "[open('"&TemplateLoc"')]"
```

```
run;
```

```
%end;
```

```
%else
```

```
%do;
```

```
%put No template specified;
```

```
%end;
```

```
%mend OpenTemplate;
```

```
%OpenTemplate;
```

Clean up working files

```
%macro CleanUp;  
  %if %sysfunc(exist(GoodObs))NE 0 %then  
    %do;  
      Proc SQL;  
        Drop table GoodObs;  
      quit;  
    %end;  
  %else  
    %do;  
      %put NO table cleanup required - no table to drop;  
    %end;  
  %mend CleanUp;  
  
%cleanUp;
```

Filter out ROWS of data on variables that we may not want to send to Excel

```
/*We might want to filter observations on a variable we do not want to send to excel*/
```

```
/*Filter observations ASAP and drop variables in the next step*/
```

```
Data GoodObs;
```

```
set &SASFile;
```

```
&whereClause; /*<----- Drop rows you do NOT want to print*/
```

```
/*This, and multiple calls of the macro,  
can be used to split files that are too large to fit on an XLS sheet*/
```

```
run;
```

Drop any extra vars. - Load # of variables we want to export into NoOfVars - create a macro array

/*for the DDE, We need to know how many obs and variables are in the file we are exporting. Put the number of rows and columns into macro variables so we can dimension the DDE writing area*/ **/*Note &DropVars. You do not have to export all the variables */**

```
Proc Contents data=GoodObs&DropVars      varnum noprint
          out=VarsToExport ;
run; /* create data set with variables info--*/
proc sort data=VarsToExport ;
by varnum;
run;
```

After dropping variables, we have a file that contains the variables to export

```
data _null_;
  /*How many variables - This number is used to dimension the DDE "writing" area*/
  if 0 then set VarsToExport Nobs=NOfObs;
  call symput("NoOfVars",Strip(left(NOfObs)));
run;
```

Number of variables in NoOfVars

```
%put We have &NoOfVars Variables to export from &&SASFile ;
```

```
data _null_;
  /*in the DDE, We need to be able to loop and "write" the names of the variables*/
  set VarsToExport(keep=name);
  /*create an array of macro variables*/
  call symput("vn"||left(Put(_n_,7.)),name); /*DANGER hardcode*/
run;
```

Macro array holding variable names (in order)

Print the variables we want to export to the log

```
%macro LeaveRecord;  
  /* As a debugging tool, write the variables we want to export in the log */  
  
  %do QC=1 %to &NoOfVars;  
    %put for loop number &qc the variable we write is &&vn&QC;  
  %end;  
  
%mend LeaveRecord;  
  
%LeaveRecord;
```

Print the macro array holding variable names (in order)

Put the number of rows in a macro variable and give a bit of notice if the file is getting large

```
data _null_;  
/*How many rows - This number is used to dimension the DDE "writing" area */  
if 0 then set &SASFile Nobs=NOfObs;  
  
call symput("NoOfRows",Strip(NOfObs));  
  
If NOfObs GT 65000 then  
do; /*DANGER HARDCODE*/  
  put "*****";  
  put "WA" "RNING **";  
  put "There are " NOfObs " in this file.";  
  put "    Check to see if the might fill up an XLS sheet";  
  put "*****";  
end;  
run;
```

We used to be able to overload Excel.

%put

After applying the where clause, we want to print &NoOfRows from the file &SASFile;

Issue the D.D.E. statement that dimensions the "writing area"

```
filename blah  
dde
```

You do not have to send data to Row1 col1

```
"excel|&XLSDataTab!
```

```
R&StartRowNo.C&StartColNo.:
```

```
R%eval(&StartRowNo+&NoOfRows) C%eval(&StartColNo+&NoOfVars)"
```

```
NOTAB;
```

Calculate the dimensions of the writing area

```
/* in statement above, we dimensioned the DDE "writing" area */
```

Use a data _null_ and a loop to write the D.D.E. put statement

```
Data _null_; /*this data null reads the GoodObs table and does the writing to excel*/  
set GoodObs&DropVars;
```

```
file blah LRECL=2050;  
/*The value of Lrecl can range from 1 to 1,048,576 (1 megabyte). According to  
the  
experience of SAS experts, an Lrecl of 8192 covers 99% of the cases that most  
SAS programmers encounter.  
*/
```

```
%macro LoopOverVars;  
/*use a loop to list all the variable names in the DDE statement*/  
put %do k=1 %to &NoOfVars;
```

```
&&vn&k "09"X
```

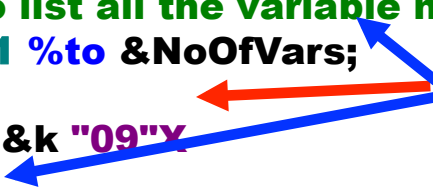
```
%end ;
```

```
%mend LoopOverVars;
```

```
%LoopOverVars;
```

```
run;
```

Use a loop to write the put statement



If &SaveNowYN) =Y, perform any special writes, make the sumamry sheet active, save and close

```
%if (%upcase(&SaveNowYN) =Y) or (%upcase(&SaveNowYN) =YES) %then
%do; /* If you tell the macro you have written the last data tab, it will
1)** Write any information to the "summary sheet"
2)** Put the date in Cell A2 of Sheet 1 3)** Save the file under a new name */
```

```
filename TwoCells dde "excel|&XLSSmryTab!R1C1:R3C2" NOTAB; /*HARDCODED range*/
/*This writing should not "Overlap" the range where we write data*/
Data _null_; /*HARDCODE - /*We write the 6 cells below into a 3 by 2*/
file TwoCells LRECL=2050;
Put "Put in Cell A1" "09"X "Put in Cell A2" "09"X ;
Put "Run On:" "09"X "%sysfunc(Date(),worddate18.)" "09"X ;
Put "Put in Cell A3" "09"X "Put in Cell B3" "09"X ;
run;
```

```
filename Final dde "excel|&XLSSmryTab!R1C1:R1C1" NOTAB; /*HARDCODE*/
/*(above) Make &XLSSmryTab active before saving then (below)Save and rename*/
filename outexcel dde "EXCEL|SYSTEM";
data _null_;
file outexcel;
put "[save.as('"'&NewName'"')]" ;
put "[close]";
put "[quit]";
run;
%end;
%Mend Write2XlsV2;
```

The End

Russ Lavery

Russ@Russ-Lavery.com

Thanks to Dr. Steve Bajgier

Presentation Structure

3

- Preliminary Information
- Equally likely outcomes of random variables
 - Coins: formulas for EV and variance
 - Dice: formulas for EV and variance
 - The Central limit theorem and dice
 - Linear Combinations of Random Variables
- The Hypergeometric - *We'll get there eventually*
 - Combinations vs Permutations
 - Combinations of equally likely outcomes
 - The product rule for equally likely outcomes
 - Finally - the hypergeometric itself
- Fisher's exact test
 - What if with Excel
- The CMH Test
 - What if with Excel