PharmaSUG2011 - Paper CC23

Choosing the Best Method to Create an Excel Report

Romain Miralles, Clinovo, Sunnyvale, CA

ABSTRACT

PROC EXPORT, LIBNAME, DDE or excelXP tagset ? Many techniques exist to create an Excel file using SAS®. Which one is the best one? Well, it depends; each of them offers some unique advantages.

LIBNAME statement is a new and useful option available with SAS 9 but old methods like DDE are still very powerful and offer some unique capabilities. Knowing and understanding the different techniques is essential for SAS programmers to quickly and effectively produce a report that will meet the requirement provided by the customer.

This paper will briefly describe the main techniques to generate an Excel file from SAS, and provide recommendations on the appropriate method to use when an Excel output must be created.

INTRODUCTION

SAS programmers often have to prepare reports and listings for customers. Multiple methods exist in SAS to program listings and reports. Choosing the best technique does not necessarily mean selecting the most powerful solution. The role of the SAS programmer is to determine the best technique to use for a project taking into consideration the customer's needs, constraints, and time frame.

Many clients are also required to have an Excel version of the output. They want to be able to view, sort and filter the data, and Excel is usually the tool they master best. In addition, Excel is an excellent means to share information.

We will see in this paper the techniques available to generate Excel reports using SAS, we will explain the different functionalities, and weigh the pros and cons of each solution depending on the purpose and requirements.

PROC EXPORT

PROC EXPORT is the most common way to export a SAS dataset to a Microsoft Excel document. Its simple syntax makes it easy to use. Usually, it is the first method from which a SAS programmer will learn to export data. It is a basic technique that proves useful in many situations.

SYNTAX

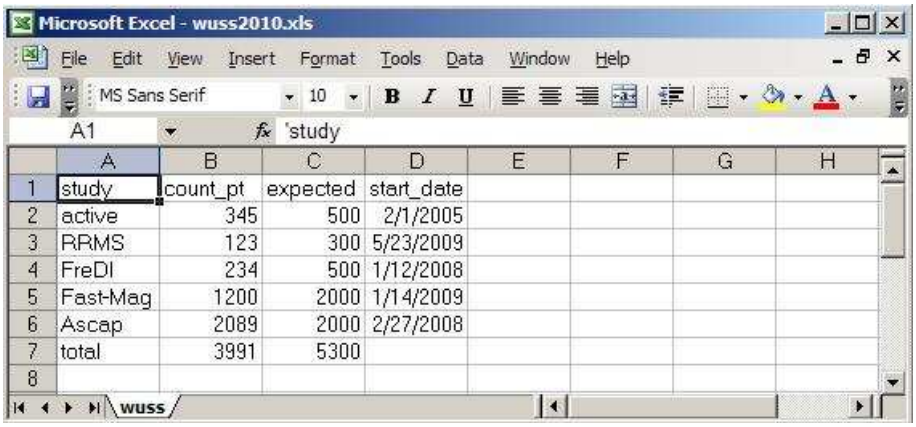
```
proc export data = phsug.phsug
      dbms=excel
      outfile = "c:\phsug2011.xls"
      replace;
      sheet='phsug';
run;
```

data: SAS dataset to export.
outfile: Path and filename for the output.
dbms: To create an Excel file, DBMS option needs to be set to Excel.
replace: Overwrites an existing file.
sheet: Sheet name in Excel. If this option is not included, then a sheet name based on DATA= will be created.

PROC EXPORT can be used in 2 different ways:

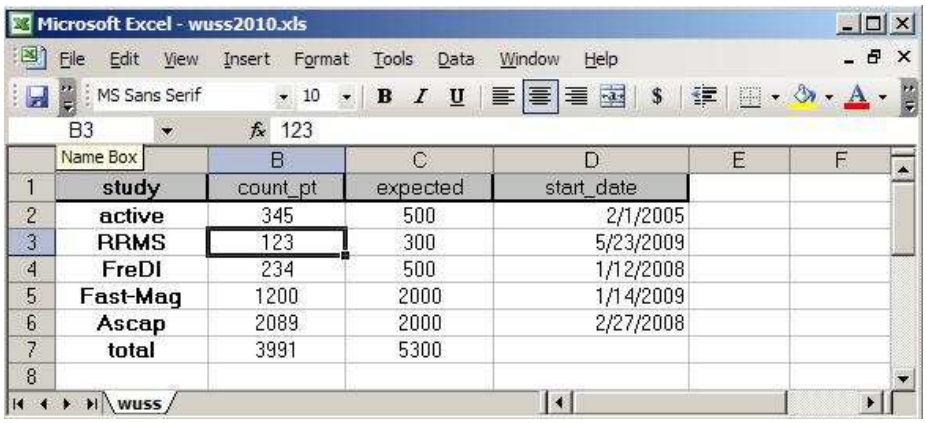
- First, we can use it to create an Excel file with the data from the dataset. With this method, it is not possible to format the output.

- The second solution is to use an Excel template and to populate the file with data from the dataset. An Excel document is saved on the computer and the columns are formatted as needed. Then the path and filename of the Excel template are entered in the PROC EXPORT OUTFILE parameter.



	A	B	C	D	E	F	G	H
1	study	count_pt	expected	start_date				
2	active	345	500	2/1/2005				
3	RRMS	123	300	5/23/2009				
4	FreDI	234	500	1/12/2008				
5	Fast-Mag	1200	2000	1/14/2009				
6	Ascap	2089	2000	2/27/2008				
7	total	3991	5300					
8								

Figure 1. PROC EXPORT output without using a template. Variable labels are not exported.



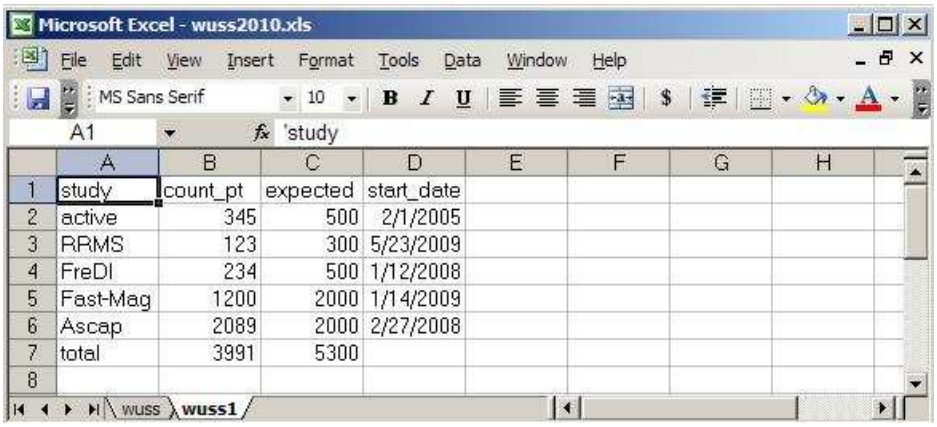
	B	C	D	E	F
1	**study**	count_pt	expected	start_date	
2	**active**	345	500	2/1/2005	
3	**RRMS**	123	300	5/23/2009	
4	**FreDI**	234	500	1/12/2008	
5	**Fast-Mag**	1200	2000	1/14/2009	
6	**Ascap**	2089	2000	2/27/2008	
7	**total**	3991	5300		
8					

Figure 2. PROC EXPORT using a pre-formatted template.

You might encounter the following error message when using PROC EXPORT with an Excel template:

```
-ERROR: Error attempting to CREATE a DBMS table. ERROR: Execute: Too many fields
defined.
WARNING: File deletion failed for _IMEX_.phsug.DATA.
ERROR: Export unsuccessful.  See SAS Log for details.
```

This message means that you are trying to export more columns than the number of columns defined in your named range. To resolve this issue, you need to change your named range in Excel (named ranges are further described in the LIBNAME engine section).
Named range issues can also generate incorrect reports. For Instance, in the Figure 3 below, an unexpected tab appears. This error implies you need to check the named range.



	A	B	C	D
1	study	count_pt	expected	start_date
2	active	345	500	2/1/2005
3	RRMS	123	300	5/23/2009
4	FreDI	234	500	1/12/2008
5	Fast-Mag	1200	2000	1/14/2009
6	Ascap	2089	2000	2/27/2008
7	total	3991	5300	
8				

Figure 3. Named range is not defined properly in the Excel template. PROC EXPORT does not populate the existing tab and create a new one.

WHEN TO USE IT

PROC EXPORT is a good method for easy data manipulation. The Excel file can have one or multiple spreadsheets. However, it is impossible to create elaborated reports that will impress customers.
PROC EXPORT is a good technique if the customer expects a simple listing developed quickly with limited formatting.
Many programmers only use PROC EXPORT and Export Wizard to create Excel reports. This is a good solution overall, but in many situations other techniques can produce a better result and broaden the horizons of SAS programmers.

LIMITATIONS

Customization of a report is limited even when using an Excel template. PROC EXPORT starts in the default, top, left cell (A1) and fills out the necessary rows and columns. Moreover, PROC EXPORT cannot use the SAS labels as column names in Excel unless you are using SAS 9.2. This limits the use of PROC EXPORT for report programming. Typically, users have no knowledge of the SAS dataset structure and SAS variable names are not meaningful to them. In addition, variable names in SAS have restrictions that column labels don't share.

The order of columns in the exported worksheet is the same as the order of variables in the original dataset. Variables need to be ordered before using the PROC EXPORT. All the variables existing in the dataset will be output using this technique.

EXCELXP TAGSET

Using ExcelXP tagset is a technique available in SAS version 9.1. It utilizes the Extended Markup Language (XML) but does not require SAS programmers to know XML. Using the ExcelXP Tagset is a powerful method for formatting a spreadsheet. It is used much like the ODS HTML destination. Common ODS options can be used as well as many other helpful options.

ExcelXP tagset is different from the other techniques described in this paper in that it doesn't need an Excel template to create a formatted report. It can be downloaded from the SAS website. Installation is easy: You open the SAS program and simply execute it. This will create or update the ExcelXP tagset on your computer.

SYNTAX

ExcelXP tagset can be used to export the results of PROC REPORT, PROC TABULATE, or PROC PRINT. It can display multiple tables per worksheet as well as multiple worksheets.
The ODS option 'style' can be used with the ExcelXP tagset. Many styles are available in SAS and styles can also be customized or created using PROC TEMPLATE.

```
ODS listing;
ODS results;

ODS listing close; /*Turn off the standard line printer destination*/
ODS noresults;      /*Prevents results from appearing within SAS viewer*/
```

```
ods tagset.ExcelXP
      file="c:\phsug2011.xls"
      style=phsug  /*Styles to control appearance of output*/


      options     (Embedded_titles = 'yes'
                   Embedded_Footnotes   = 'yes'
                   sheet_name= 'Phsug'
                   autofilter= 'yes'
                   frozen_headers= '3'
                   autofit_height= 'yes'
                   absolute_column_width= '15,10,10,13');

title1 "List of ongoing clinical trials";

footnote1 "May 8th 2011";

proc Report data=phsug.phsug NOWD;
     Column study count_pt expected start_date;
     define study /center  style(column)=[font_weight=bold] style(header) =
     [background = CX4D7EBF];   ;
     define count_pt /center    style(header) = [background = CX4D7EBF];
     define expected /center    style(header) = [background = CX4D7EBF];
     define start_date /center  style(header) = [background = CX4D7EBF];
run;



ods tagset.ExcelXP close; /* Close and release the xml file so it can be opened with
Excel*/

ODS listing;
ODS results;
```

ExcelXP includes a wide range of formatting options. In this example, we used the options described below but many more options are available and documented online:

Embedded_Footnotes: Add footnotes to the footer section of the Excel worksheet.
Embedded_Titles: Add Titles to the header section of the Excel worksheet.
sheet_name: Name the current sheet.
autofilter: This option will add an autofilter to your headers.
frozen_headers: If your listing has many rows, scrolling down will move the headers off the screen. This useful option will freeze the rows you want to use as headers.
autofit_height: Excel automatically sets the row heights so that wrapped text can be seen.
absolute_column_width: ExcelXP automatically estimates the width on columns based on several factors. This option allows you to correct the column widths. The column widths are entered as a comma separated list. Different values can be tried until you get the expected output.

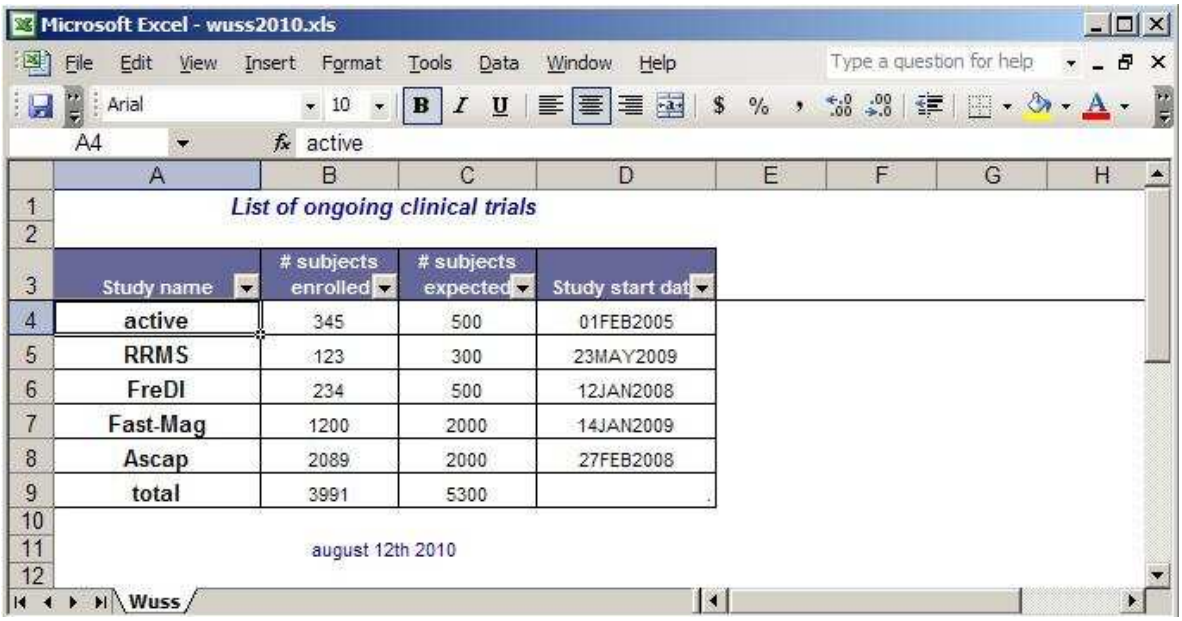


Figure 4. Output produced by ExcelXP tagset using a SAS style and some tagset options.

WHEN TO USE IT

ExcelXP tagset has an amazing number of options and high flexibility, allowing it to accomplish almost any report. It reduces or eliminates the need for manual formatting, as all formatting and layouts are performed by SAS: There is no need to create a template or edit the Excel workbook. ExcelXP includes a lot of options that control the appearance of the report and many papers are available online to get a better understanding of all them. However, if you need to use an Excel template for your report, the ExcelXP tagset is not the best solution.

ExcelXP tagset is one of the best ways to create a file with multiple sheets. Unlike other techniques, there is no need to prepare a template. All formatting is performed by SAS, so it becomes very easy to define a style and apply it to all the worksheets.

LIMITATIONS

In order to take advantage of the latest ExcelXP features, the tagset must be downloaded and installed on the machine. Computers without the latest version of ExcelXP might not be able to run SAS programs to create reports. Additionally, ExcelXP is still evolving, thus functionalities may change in the future.

Excel does not handle date variables like SAS. Date variables exported from SAS with ExcelXP tagset are interpreted like a text variable. Excel will understand SAS dates only if they are converted to a specific format before using ExcelXP (the paper 'The Devil Is in the Details: Styles, Tips, and Tricks That Make Your Microsoft Excel Output Look Great!" gives a great explanation about date format with ExcelXP).

ExcelXP tagset produces an XML file designed for Excel. XML is a great way to store data; everything is in a human-readable format. However, XML files are not readable on every computer. Some customers might have an old version of Excel that could not open the file created by ExcelXP tagset. One of the solutions to avoid this issue is to open the XML file and save it in XLS format. This can be automated using a SAS macro and a VBS script.

DDE

Dynamic Data Exchange (DDE) is an old protocol, but it is one of the most powerful methods to integrate SAS and Excel.
DDE is the direct communication between SAS and Excel using a server/client model. Excel acts as a server and SAS as a client. It is the only technique that can use visual basic language, the most powerful feature of Excel, and provides total control over the output.

However, this technique might seem obscure to people who have no experience with it. There are two ways to use DDE:

- The first approach, and probably the most difficult one, is to execute all the code directly in SAS using X4ML functions to provide instructions to the Excel application. DDE can enable much of the functionality of Excel within SAS.
- The second solution is to create a pre-formatted Excel template and populate it using DDE. With just two macros to open and close the Excel file, it becomes very easy to program a fancy report. This solution is easier and faster. Anyone can prepare a nice spreadsheet using the power of Excel, and populate it with the SAS data.

SYNTAX

```
%let stufile=C:\phsug;
/*************************************************************************************/
/*                                TO OPEN XLS FILE                                   */
/*************************************************************************************/

%MACRO OPENXLS(FOLDER=,IN=);
%LET FIL=;
       DATA _null_;
              LENGTH FILE $300.;
              FILE="'&STUFILE\&folder.&in.'";
              FILE="'"!!TRIM(LEFT(TRANWRD(FILE,"'",'"')))!!"'";
              CALL SYMPUT ("FIL",TRIM(LEFT(FILE)));
       RUN;
options noxwait noxsync;
x &fil.;
filename commands dde "Excel|system";
%MEND OPENXLS;
/*************************************************************************************/
/*     TO SAVE ACTIVE XLS FILE    UNDER SPECIFIC FOLDERS AND QUIT EXCEL           */
/*************************************************************************************/
%MACRO CLOSEXLS(out=,quit=0);
%LET FIL=;
       DATA _null_;
              LENGTH FILE $300.;
              FILE="[save.as('&STUFILE.\&out.')]";
              FILE="'"!!TRIM(LEFT(TRANWRD(FILE,"'",'"')))!!"'";
              CALL SYMPUT ("FIL",TRIM(LEFT(FILE)));
       RUN;
       %PUT &FIL=;

       data _null_;
              file commands;
              put &fil.;
              put '[CLOSE()]';
              %IF &quit=0 %then %do;
                     put '[QUIT()]';
                     stop;
                     run;
                     filename commands clear;
              %END;

              %IF &quit=1 %then %do;
                     run;
              %END;

%MEND CLOSEXLS;

%OPENXLS(FOLDER=template\,IN=phsug2011_DDE_temp.xls); /* Open the Excel template*/
option missing="" ;

/* First dataset is output in a range starting at row 5 column 3  and ending at row 30
and column 10*/
filename TAB DDE "EXCEL|[phsug2011_DDE_temp.xls]phsug!R5C3:R30C10";
filename xlcmds DDE "EXCEL|SYSTEM";
```

```
data _null_;  /* Data step to output the data in the Excel file */
   set phsug.phsug;
   file TAB notab lrecl=7000;

   PUT  study              '09'x
   count_pt                '09'x
   expected                '09'x
   start_date              '09'x;
run;

/* Second dataset is output in a range starting at row 16 column 1 and ending at row
18 and column 2*/
filename TAB DDE "EXCEL|[phsug2011_DDE_temp.xls]phsug!R16C1:R18C2";
filename xlcmds DDE "EXCEL|SYSTEM";

data _null_;
   set phsug.param;
   file TAB notab lrecl=7000;
   PUT  var1               '09'x
   var2                    '09'x;
run;

data _null_; /* Invoke a VB macro 'lastline' embedded in the Excel template */
       file xlcmds;
       PUT '[RUN("lastline")]';
run;

%CLOSEXLS(OUT=phsug_DDE_&today..xls);
```

filename TAB DDE "EXCEL|[phsug2011_DDE_temp.xls]phsug!R5C3:R30C10": Excel template name inside brackets followed by the Excel sheet name.
'R5C3:R30C10' are row and column numbers where the data will be loaded. Just make sure that all data can fit in the range defined.

In this example we used an Excel macro 'lastline' programmed in VBA and embedded in the template. This macro changes the background color and the font of the table's last line. We could have prepared a template with the last line pre-formatted. However, if the number of records changes frequently in the SAS dataset, we would need to manually update the template. Using a VBA macro, Excel can determine automatically the last line and format the data.

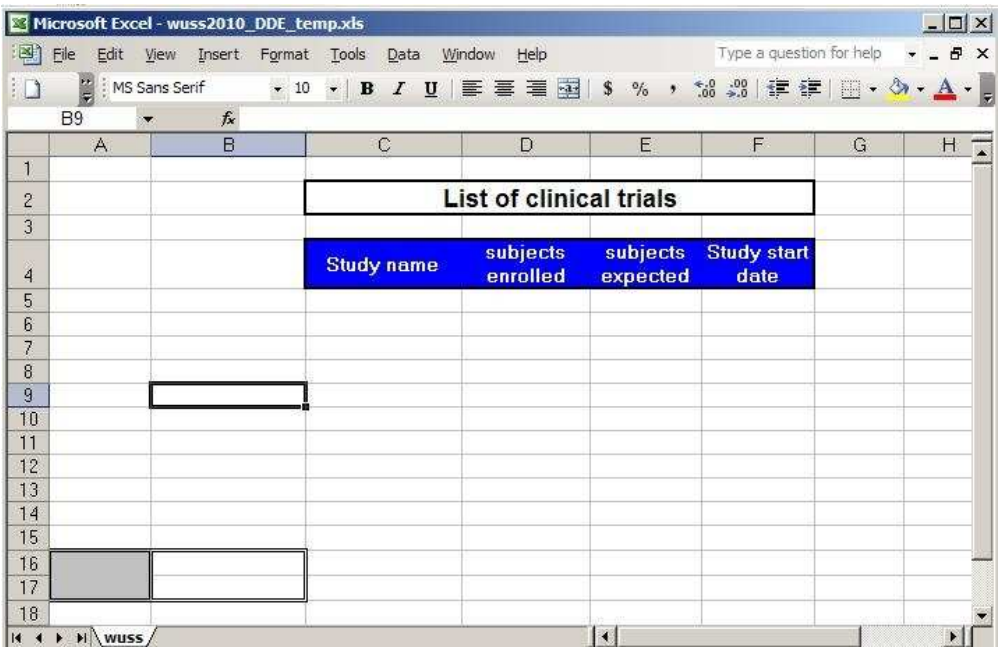


Figure 5. Pre-formatted Excel template used with DDE.

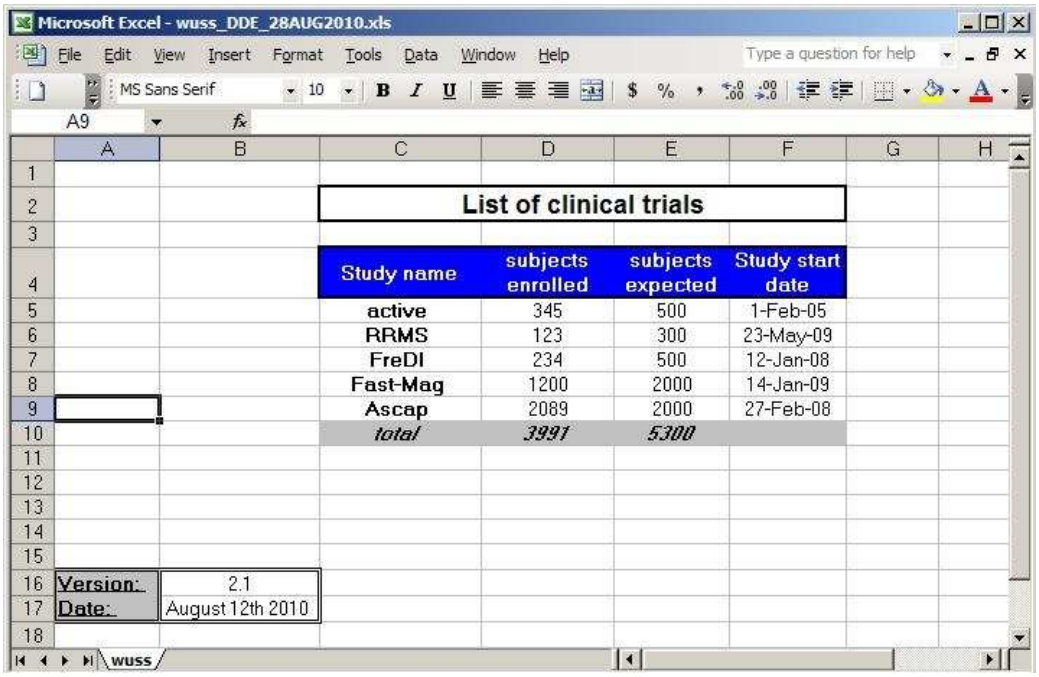


Figure 6. Excel file created using DDE. The last line 'total' is formatted using a VBA macro.

WHEN TO USE IT

DDE is a great solution to create sophisticated reports. It provides full control of Excel and allows you to leverage the powerful capabilities of this reporting tool. Anyone knowing Excel can prepare an Excel template with pre-formatted cells and columns.
VBA macros can be programmed and called directly from SAS. They can be very powerful and can help you build the perfect report. If you need to use a VBA macro for your report, DDE is the best solution.

LIMITATIONS

Both Excel and SAS need to be installed and active for DDE to work. This makes DDE impossible to use in an environment where SAS is installed on a server and Excel on a desktop.
In addition, knowledge of Excel programming is required. DDE uses Excel version 4 macro language, which has been superseded by Visual Basic for Applications (VBA). Finding documentation for this macro language is not always easy. However, even if SAS cannot issue visual basic command, it can still invoke VB macros that are embedded in a sheet.

One frequent issue with DDE is the carriage return. If a cell value contains a carriage return in SAS, it will be interpreted as a line return by Excel and generate an incorrect report. One solution to avoid this issue is to replace the carriage returns by another character with SAS before using DDE.

LIBNAME ENGINE

LIBNAME engine is the newest method to get information from SAS into Excel. It is available with SAS 9.0 and lets you use Excel as SAS library.
Like DDE, LIBNAME engine allows you to heavily customize your output. It does not give full control of Excel like DDE, but it has one big advantage: Excel doesn't need to be installed on the machine running SAS.

SYNTAX

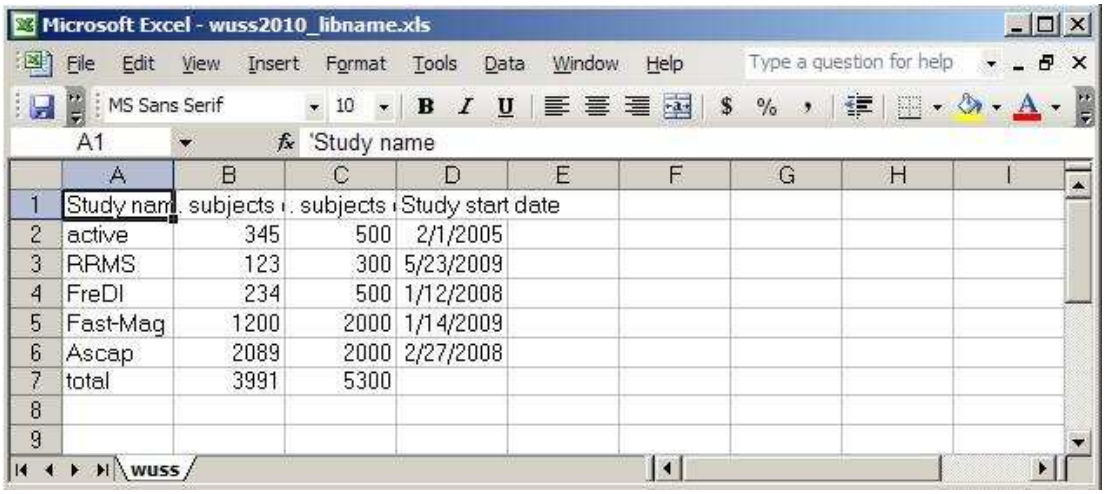
LIBNAME can be used to create a new workbook. It will create the Excel file on the machine and then populate it such as a PROC EXPORT. The file should not exist on the drive as LIBNAME engine cannot overwrite.

```
libname toexcel EXCEL 'c:\phsug2011_libname.xls' VER=2002 ;

DATA toexcel.phsug (DBLABEL=YES);
      SET phsug.phsug ;
RUN;
LIBNAME toexcel CLEAR;
```

Ver=2002: To specify the most current version. The default is Excel 97
DBLABEL=YES: Write the variable labels

A data step is used to export the data. We can choose the name of the tab in the Excel output and we can export the variable labels with the option DBLABEL=YES.



	A	B	C	D	E	F	G	H	I
1	Study nam	subjects	subjects	Study start date					
2	active	345	500	2/1/2005					
3	RRMS	123	300	5/23/2009					
4	FreDI	234	500	1/12/2008					
5	Fast-Mag	1200	2000	1/14/2009					
6	Ascap	2089	2000	2/27/2008					
7	total	3991	5300						
8									
9									

Figure 7. Excel file created with LIBNAME without using a template. SAS variable labels can be exported as column names.

LIBNAME can also be used with a template to create a customized output. The first step is to prepare a template and define the named ranges to receive the SAS data. Then, the file can be populated using the Excel LIBNAME. Without properly defined named ranges, LIBNAME engine cannot export the data to Excel. A range is a subset of cells defined using Excel option **insert->name->define.**

Named range defined in Excel. Named range is case-sensitive and must match exactly the name in the SAS code.

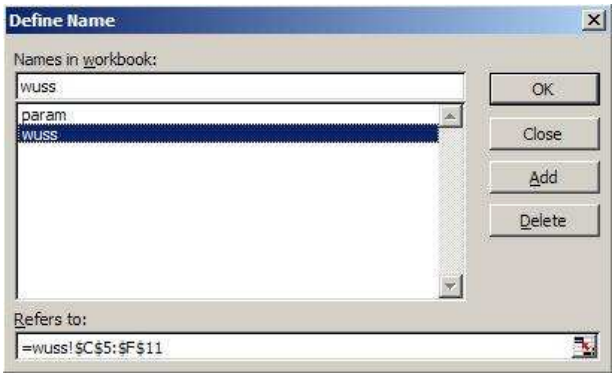


Figure 8. Define named ranges dialog

```
libname toexcel EXCEL 'c:\phsug2011_libname_2.xls' VER=2002 ;

/* Delete the data from the range to be populated. Without prior deletion, SAS will
not populate the range*/
PROC DATASETS LIB=toexcel;
      DELETE phsug;
      RUN;
QUIT;

PROC DATASETS LIB=toexcel;
      DELETE param;
      RUN;
```

```
QUIT;

DATA toexcel.phsug;
      SET phsug.phsug ;
RUN;

DATA toexcel.param ;
      SET phsug.param ;
RUN;
LIBNAME toexcel CLEAR; /* Disconnect from workbook*/
```

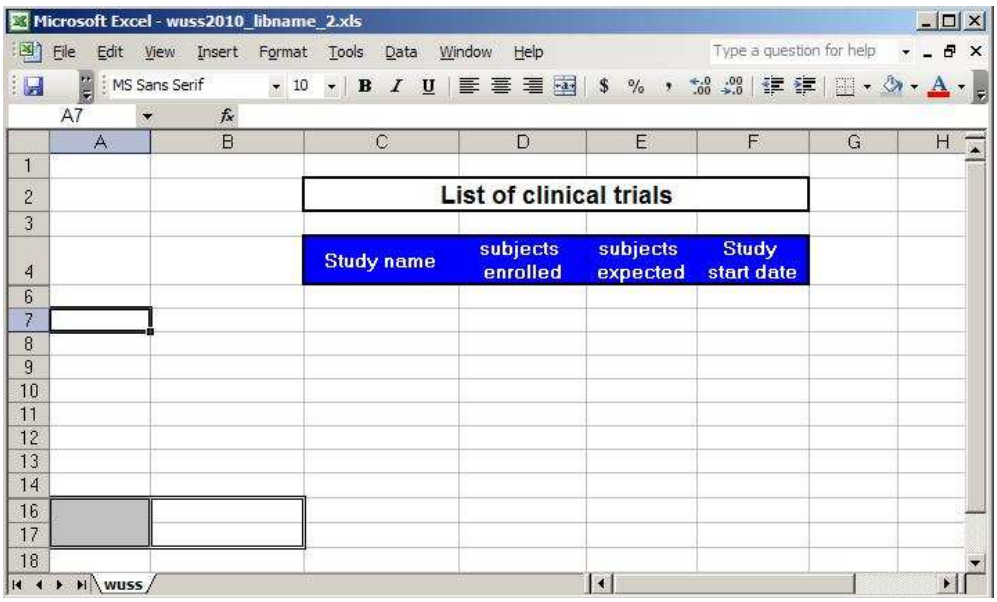


Figure 9. Template used with LIBNAME engine. Rows 5 and 15 are hidden in order not to display the variable names in the final report.

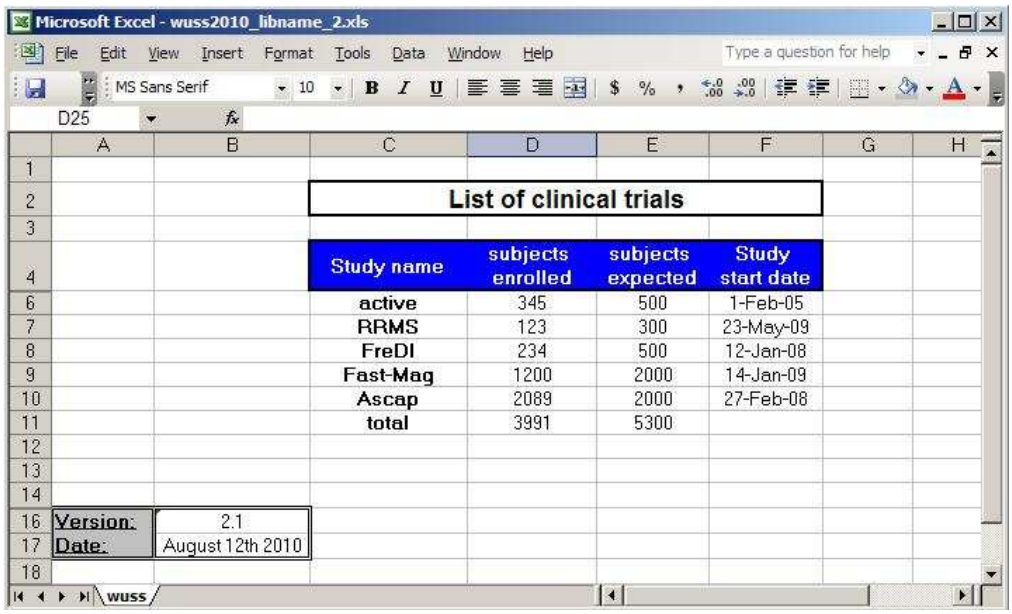


Figure 10. Output with LIBNAME engine. Variable names are output in the report but hidden to the user. Two named ranges have been defined in the template and used in the SAS code.

WHEN TO USE IT

LIBNAME engine is another great method to generate customized Excel reports. It does not need Excel present or activated, and a simple data step can be used to populate the output. The Excel spreadsheet can be manipulated much like a SAS dataset. Additionally, unlike PROC EXPORT (before SAS 9.2), LIBNAME engine can export the variable labels.

LIBNAME engine is also a possible solution to program a fancy report. It is not as powerful as DDE but it is sufficient for most of the programs. Pre-defined and customized Excel worksheets can be easily used after understanding named ranges in Excel.

LIMITATIONS

Using LIBNAME can be confusing at the beginning. Each dataset appears twice when you open the Excel file with LIBNAME:

- One with the expected name. This dataset is the named range.
- One with a trailing "$". This dataset is the spreadsheet.

It requires a little time to understand Excel structure and the use of named ranges. Without a correctly named range in Excel, LIBNAME cannot execute and displays an error message in the log.

By default, LIBNAME engine writes the variable names in the first row of a range. When defining the named ranges, users need to remember that the first row will be populated with the variable names. The only solution not to display the headers is to hide the first row of the range in the Excel template.

NO FORMATING AND QUICK REPORT

If the report does not require any formatting, the easiest methods to use are PROC EXPORT and LIBNAME without template. The Excel file is created directly from the SAS code. LIBNAME engine offers more advantages than PROC EXPORT, such as the ability to export the labels or to create new columns as you are writing to the spreadsheet.

Both techniques have a very simple syntax. PROC EXPORT is a SAS procedure with multiple options widely described online. LIBNAME engine uses a data step to export the data.
These techniques can be used to quickly create a new Excel file. You can export a dataset to Excel even faster without programming by right-clicking on the dataset.

USING AN EXCEL TEMPLATE

We sometimes need to use an Excel template designed by an Excel expert or by ourselves. In that case, the two best methods are LIBNAME engine and DDE. PROC EXPORT can be used with an Excel template but formatting options are too limited.
Syntaxes are quite different between LIBNAME and DDE. With the LIBNAME engine, we need to use named ranges in Excel and make sure the data will fit in the template. For DDE, only the column names need to be set up in the Excel template. It doesn't require named ranges, and data will be output in the rows and columns defined in the SAS code.
The biggest advantage of DDE over LIBNAME engine is the possibility to call and execute VBA macros.

CREATING A MULTI-SHEET REPORT

Choosing the best method for a multi-sheet report depends on the report you want to create. For multi-tabs without formatting, PROC EXPORT or LIBNAME engine are sufficient. If the output needs to be formatted, different techniques can be used. A template can be prepared in different tabs and populated with LIBNAME or DDE. However, using a template is not a good idea if the report has many tabs. It would take too much time to prepare each tab.

ExcelXP tagset is the ideal solution to export SAS data to Excel workbooks that contain multiple worksheets. By using styles and the numerous options available with the tagset, you can create a custom template that will be applied to all the tabs. The tagset will create the Excel file with the tabs and style defined in the SAS code.

CONCLUSION

The goal of the methods described above is the same: Creating an Excel file using SAS. However, syntax, capabilities and limitations are different. Some SAS programmers choose to learn one method to create an Excel report and use it for any program. This strategy narrows the ability to take full advantage of broad and powerful SAS capabilities.

Knowing several techniques is a better strategy that allows SAS programmers to effectively develop a report in a short period of time. By acquiring knowledge on different SAS techniques, SAS programmers have a broader range of options available to create the exact report expected by the customer and can decide on the method best suited taking into account time constraints and the ultimate project goal..

REFERENCES

Generating Custom Report Tables: Using SAS with DDE and VBA, Ying Feng, SUGI 2005
http://www2.sas.com/proceedings/sugi30/161-30.pdf

De-Mystifying the SAS® LIBNAME Engine in Microsoft Excel: A Practical Guide, Paul A. Choate, Carol A. Martell, SUGI 2006
http://www2.sas.com/proceedings/sugi31/024-31.pdf

Printable Spreadsheets Made Easy: Utilizing the SAS® Excel XP Tagset, Rick Andrews, NESUG2008
http://www.nesug.org/proceedings/nesug08/ap/ap06.pdf

Creating Multi-Sheet Excel Workbooks the Easy Way with SAS, Vincent DelGobbo, pharmasug 2007
Play Nice with Others: Waiting for a Lock on SAS Table, John Leveille, Jimmy Hunnings, pharmasug 2005.
http://www.lexjansen.com/pharmasug/2005/posters/po33.pdf

Excellent Ways of Exporting SAS Data to Excel, Ralph Winters, NESUG2004
http://www.nesug.org/proceedings/nesug04/io/io09.pdf

Choosing the Right Tool from Your SAS® and Microsoft Excel® Tool Belt,
Steven First, Jennifer First, SUGI2010
http://support.sas.com/resources/papers/proceedings10/144-2010.pdf

More Tips and Tricks for Creating Multi-Sheet Microsoft Excel Workbooks the Easy Way with SAS, Vincent DelGobbo, SGF 2009
http://support.sas.com/resources/papers/proceedings09/152-2009.pdf

The Devil Is in the Details: Styles, Tips, and Tricks That Make Your Microsoft
Excel Output Look Great!, Eric Gebhart, SUGI2008
http://www2.sas.com/proceedings/forum2008/036-2008.pdf

CONTACT INFORMATION

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

Romain Miralles
Clinovo.
1208 E. Arques Avenue, suite 114
Sunnyvale, CA 94085
E-mail : mrom34@gmail.com
Web : http://www.clinovo.com