Reading Data Tables: PROC IMPORT and DDE

Data sets are often created using spreadsheet packages, which are quite popular. How can you read the data from one of these spreadsheet packages? Here are three different ways that this can be done. Microsoft Excel is used to illustrate this, but SAS is not limited to Excel.

Below are the contents of the Microsoft Excel file: data1.xls

TRT	X	Y	Z
A	5	15	15
A	3	21	44
A	5	11	79
A	7	55	46
A	1	22	16
A	6	23	54
A	8	.	52
B	4	44	32
B	1	11	53
B	6	77	51
B	.	88	24
B	1	55	26
B	8		59
B	9	11	54
B	4	33	21

INFILE

Previously, INFILE was used to read in data from external text files. These files could be space or tab delimited or some other delimiter could be specified and recognized by INFILE. Suppose we begin with the Excel sheet above. In Excel, the worksheet can be saved as a text file by selecting **File – Save As** and specify the file type. You could choose either space delimited or tab delimited text files. If there are missing observations in the data columns that are blank rather than denoted by '.' you should save the file as a tab delimited file. Keep all column headings.

Objective 1: Using the data above, create an Excel file data1.xls and a tab delimited text file. For this objective the file is saved on the a:\ drive. Specify and use the drive:\folder you have available where a: is specified below.

Using the Data Step Information 3, the SAS code for reading the data set is:

```
FILENAME one  'a:\data1.txt';
DATA a ;
INFILE one MISSOVER FIRSTOBS=2 DSD DLM='09'X ;
INPUT trt $ x y z;
RUN;
QUIT;
```

Having done this, the temporary SAS data set WORK.A can be used in a SAS program during the current SAS session.

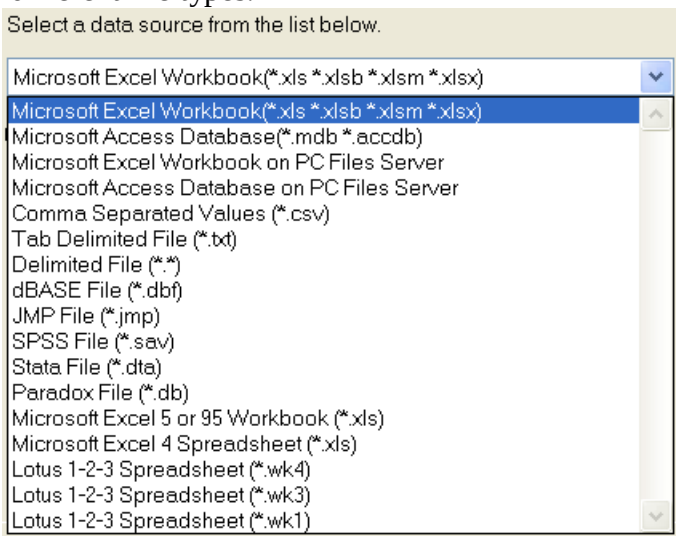
However, it is also possible to use an Excel file directly, that is, it does not have to be saved as a text file in an intermediate step. Whenever one takes a file developed in one software package for use in another software package, care should always be taken. Specifically, one should be mindful of how missing observations are handled and examine both character and numeric values.

PROC IMPORT The IMPORT procedure can read data files created using software other than SAS. While there is syntax that one can learn for this procedure, it is presented here using the **Import Wizard.** The **Import Wizard** is a menu-driven procedure for importing data files. Menu items and other selections you make by clicking appear in **boldface** in these objectives and instructions.

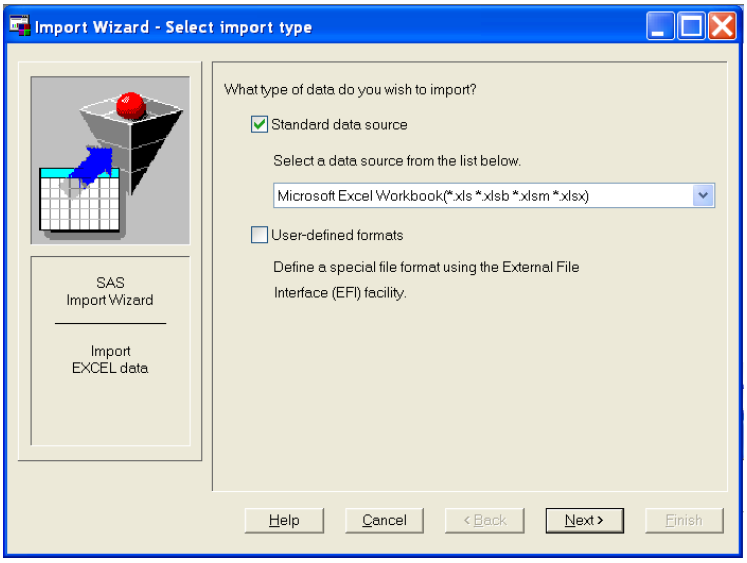
Objective 2:

1. Create an Excel data file given the information on the previous page. Call this file data1.xls. Save this file to a:\ if you have not already done so. Close data1.xls. (You do not need to close Excel.) You may have done this in the earlier objective.

2. Initiate a SAS session. From the pull-down menus select: **File – Import Data**. This will open an Import Wizard. **What type of data do you wish to import?** Observe that you can select from several different file types.

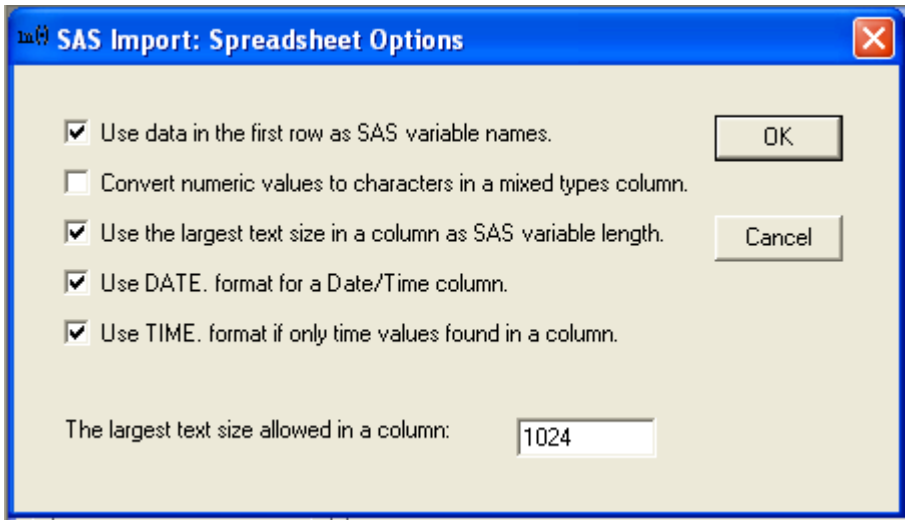


For this objective select from the list, Microsoft Excel Workbook (*.xls *.xlsb *.xlsm *.xlsx). Then select **Next**.



Note: SAS v9.1 does not import Office 2007 files, and SAS v9.2 does not import Office 2010 files. If you are using SAS v9.1, you will need to save an Excel 2007 file in 2003 format prior to importation of the data. Likewise, for any file formats, you'll need to save as an older format if your SAS release year is predates the file format.

3. **Connect to MS Excel** That is, where is the file located? Type in the blank: **a:\data1.xls** or click on **Browse** and select the directory:\folder and filename from the lists you have available. **OK**
   What table do you want to import? If you have a multi-sheet spreadsheet and the data are not on the first sheet, you can click on the down arrow of the dialog window to view the available worksheet names in the targeted Excel file. The default worksheet names are "Sheet1$" and so on. If you have renamed the worksheets in Excel, the Import dialog will identify your sheet names. Longer worksheet names will not "work" with the Import procedure. It may be necessary to shorten the sheet name(s) in Excel before the IMPORT procedure may work. Select **Options** to identify any special case situations described in this window.



Use data in the first row as SAS variable names - It is strongly recommended that the first

line of the Excel file contain column headings to be used as variable names. For now, keep these column headings to one word. Later, using **View Table,** you'll learn how to work with longer column headings in Excel. Check the box if you have allowed one line for column headings. Limit, one line for headings. No headings on the worksheet? Deselect the box by clicking on it.

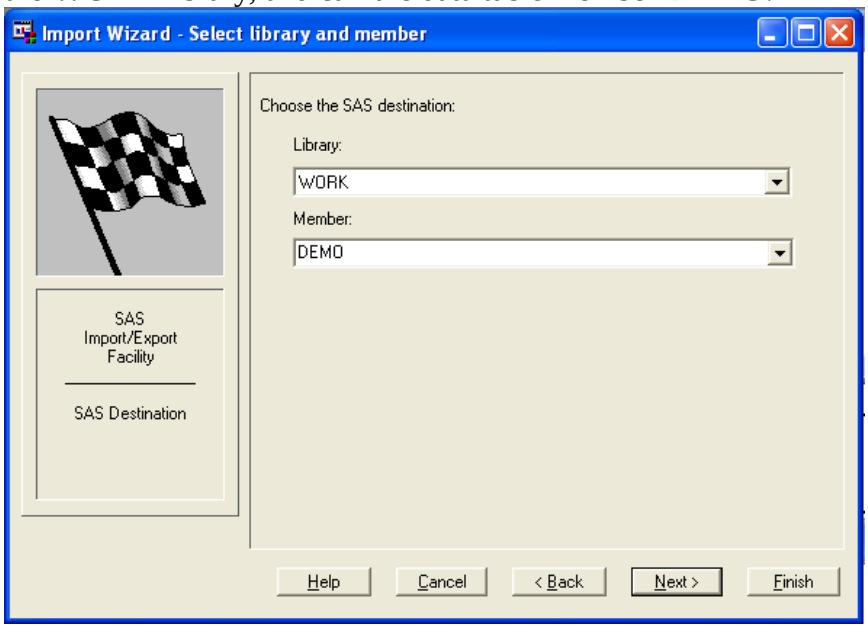
Convert numeric values to characters in a mixed types column - SAS will determine whether the data entries are character or numeric based on the values in the first line of data (below headings). You will want to check this item if the column entries could be either numeric or character. For example, suppose you have a column header "Location," and the values in the column are: 56, 84, 29N, 29S, . . . If 54 is the first entry in the "Location" column, then it is presumed that all of the data in the column are numeric. When the values "29N" and "29S" are encountered, these will be read in as missing values since they are character strings. Likewise, if a character string is the first entry, all value in the column are presumed to be character strings.

Use the largest text size in a column as SAS variable length – If your data set contains text expressions in a single column, you will want to select this option so that the text strings are not truncated.

DATE and TIME formats have not yet been covered in this material. This default setting where these last two items are selected is generally recommended for the beginning SAS programmer.

After making your **Spreadsheet Options** selections: **OK - Next**

4. **Select library and member:** Select the name of a SAS library and the name the data table will be referred to in SAS. Notice that there is a down arrow for the list of available libraries. If the library you wish to use does not appear in this list, you will need to create a new library first. See SAS Libraries and Permanent Data Tables for information. For this objective, use the **WORK** library, and call the data table member **DEMO**.

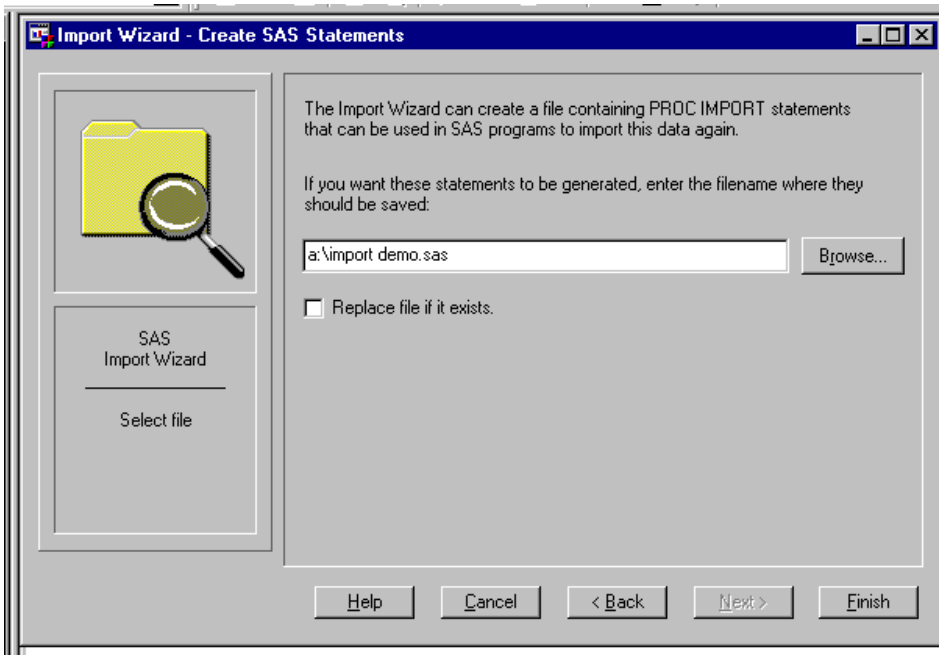


At this point you can click on the **Finish** button. You will get confirmation of the successful completion of the IMPORT procedure in the log window:

```
NOTE: WORK.DEMO was successfully created.
```

Or you will receive a message indicating cancellation or other errors with importation.

If you will be importing the Excel file again, you can save the SAS code for the IMPORT procedure that this menu driven procedure has "written" by selecting **Next** instead of **Finish**. (See next screen capture.) This would be advantageous is if the Excel file continues to change, and you wish to begin working on it prior to its completion.
Create SAS Statements - For this objective type "a:\import demo.sas" in the final dialog window. Check **Replace file if it exists** if that is appropriate for your situation. Then click on the **Finish** button. This will create a file in drive:\folder (a:\ ) called "import demo.sas." Observe the note in the log window. If the IMPORT wizard is successful, you will get a note in the log window that the data set has been created just as indicated above.



6. After the completion of the Import Wizard, examine the resulting data table in the WORK library. In the SAS Explorer window, select **Libraries – WORK – DEMO** to open the data table in VIEWTABLE. Note that the missing values denoted by the . or the space in initial Excel spreadsheet are both correctly recorded as missing in SAS. Always verify how missing observations are managed.

VIEWTABLE: Work.Demo

	TRT	X	Y	Z
1	A	5	15	15
2	A	3	21	44
3	A	5	11	79
4	A	7	55	46
5	A	1	22	16
6	A	6	23	54
7	A	8	.	52
8	B	4	44	32
9	B	1	11	53
10	B	6	77	51
11	B	.	88	24
12	B	1	55	26
13	B	8	.	59
14	B	9	11	54
15	B	4	33	21

7. You can view the IMPORT procedure code requested by moving to the **Enhanced Editor**. Then **File – Open a:\import demo.sas**. Notice the code that appears. (The default setting (GETNAMES) assumes that there are column headings in the Excel document. Notice also that the worksheet name within the Excel file is a part of the program also.)

```
PROC IMPORT OUT= WORK.demo
            DATAFILE= "a:\data1.xls"
            DBMS=EXCEL REPLACE;
     SHEET="Sheet1$";
     GETNAMES=YES;
     MIXED=YES;
     SCANTEXT=YES;
     USEDATE=YES;
     SCANTIME=YES;
RUN;
```

Keeping the SAS code for importing the data file is useful if the Excel file may be updated periodically and used in SAS programs. If a permanent SAS data set is created, IMPORT will overwrite the previous permanent SAS data set in the specified library, but the original Excel file will remain intact.

Programming Notes:

- While the IMPORT procedure above was demonstrated using all point-and-click methods, one can always write a simple program to submit from the Editor.
- Suppose you have several files and/or worksheets to import into SAS. One could perform all steps of the Import Wizard once and save the IMPORT code (as demonstrated above). Then the resulting code could be opened in the Editor. This code could be copied, pasted, and edited to target another `DATAFILE =` and/or `SHEET = .`
- The FILENAME statement can also be used in conjunction with the IMPORT procedure code. For example, multiple sheets from data1.xls could be imported. In the code below, notice that the Excel target file is identified in the FILENAME statement. Secondly, notice that each IMPORT procedure produces a different data table: WORK.DEMO1 and WORK.DEMO2. (Definitely assign different names here!) Finally, notice that

different worksheets in the data1.xls file are targeted with the SHEET option.

```
FILENAME xyz 'a:\data1.xls' ;
PROC IMPORT OUT= WORK.demo1
      DATAFILE= xyz
      DBMS=EXCEL REPLACE;
SHEET="Sheet1$";
GETNAMES=YES;
MIXED=YES;
SCANTEXT=YES;
USEDATE=YES;
SCANTIME=YES;
RUN;

PROC IMPORT OUT= WORK.demo2
      DATAFILE= xyz
      DBMS=EXCEL REPLACE;
SHEET="Sheet2$";
GETNAMES=YES;
MIXED=YES;
SCANTEXT=YES;
USEDATE=YES;
SCANTIME=YES;
RUN;
```

One could use multiple FILENAME statements to identify multiple target files and the worksheets within them. Exercise caution with your SAS data table naming conventions.

DDE (Dynamic Data Exchange) DDE allows you to read from and write to other software packages. This utility will also be demonstrated using Microsoft Excel.

To first learn DDE, the cooperating software program must have the data source file "open." DDE is not limited to this however.

To use DDE, one must use a FILENAME statement. For usage with Microsoft Excel:

FILENAME *fileref* DDE 'excel|sheet1!r2c1:r16c4' ;

The fileref is the temporary name by which the external file will be recognized during the current SAS session. The part in quotation marks is called the DDE triplet. It identifies the *application_name*|*topic*!*item*. In this instance, the *application_name* is Microsoft Excel, the *topic* is sheet1 (or the appropriate sheet number or sheet name), and the *item* is the block of data in Excel identified by the upper left hand corner (r2c1 – row 2 column 1) and the lower right hand corner (r16c4 – row 5 column 4). A colon (:) separates the two designations. You can change the sheet number if multiple sheets in an Excel document have been used. If you have changed the names of the sheets in Excel, you should use the modified sheet names as the topic in the triplet.

If you wish to access an Excel file that is not currently "open in Excel," you can do this by specifying the full path and name of the file enclosed in quotation marks for the topic. Here is an example:

```
filename test dde 'Excel|a:\[data1.xls]Sheet1!R2C1:R16C4'
```

The application-name is **`Excel`**, the topic is a:\[data1.xls]Sheet1!, and the range is **`R2C1:R15C4`**. This has accessed the Excel file data1.xls on the a:\ drive (no folder specified). Excel does not have to be "open" for this to execute.

Objective 3:
Open the data file data1.xls in Microsoft Excel. Enter the following program in the Editor. Note that you cannot have a blank space in Excel for a missing value. Use a period for all missing values. Make this correction in row 14 of data1.xls.

```
dm 'log; clear; output; clear; ';

***************** Reading Data from an Excel file ********************;
*** Data appears in the first four columns of sheet 1 ***;
*** The first row contains column headers              ***;
*** Missing Data must have a . in the Excel file       ***;

FILENAME a  DDE 'excel|sheet1!r2c1:r16c4' ;
```

```
DATA one;
INFILE a missover;
INPUT trt $ x y z ;

proc print data=one;
run;
quit;
```

Notice that the item specification in the triplet does not include column headers. If you do not insert a period for the missing value in row 14 of the Excel file, the value of y is read as 59. The MISSOVER option is necessary if there are missing values.

Objective 4:
Results computed in SAS can be printed in an Excel file also. To do this you must invoke DDE using the FILENAME statement. The DDE triplet must identify the destination.

Within a DATA step, the FILE statement can be used to identify the destination of the data set. The PUT statement within the DATA step, identifies which information, and in what order items can be "written." The PUT statement will produce a single line of information in the destination.

1. Run this objective after the previous DDE objective. Excel must be open for this objective to work.

```
***************** Printing to an Excel file ***************************;

FILENAME b DDE 'excel|sheet2!r2c2:r2c10';

PROC MEANS DATA=one;
VAR x y z ;
OUTPUT OUT=two MEAN = xbar ybar zbar STD= xstd ystd zstd N=xn yn zn ;

DATA three;
SET two ;
FILE b  ;
PUT xbar ybar zbar xstd ystd zstd xn yn zn ;

run;
quit;
```

Run the program above in SAS. Move to Excel and examine sheet 2 of the Excel Workbook. Note that the MEANS procedure results appear on sheet 2.

You can also use the PUT statement to write text in the file.

2. Modify the program above by PUT "X mean = " xbar "Y mean=" ybar "Z mean=" zbar ; Experiment with spacing and different text options.

How might one write multiple lines in a destination file? *Multiple PUT statements*

How could a single column of information be written to Excel? *Modify the item or range.*

3. Delete the FILE statement in the above objective. Where are the values PUT? *Look in the log window!*