How do I import Excel files into SAS using the SAS Tutorials?

Authored by

stats writer

June 24, 2024

The process of importing Excel files into SAS using the SAS Tutorials involves using the Import Wizard tool, which allows users to easily transfer data from an Excel spreadsheet into a SAS dataset. This tutorial provides step-by-step instructions and guidance on how to navigate the Import Wizard and properly format the data for successful importation. By following this tutorial, users can efficiently import Excel files into SAS and utilize the data for further analysis and manipulation.

Importing Data

Most of the time when you start a new project, your data will not be saved in a SAS dataset file format (*.sas7bdat). Your data might be in the form of a spreadsheet in Excel, an SPSS dataset, or a text file. The most common and new-user friendly method for reading a non-SAS dataset into SAS is by using the Import Wizard.

Update 2017 June: The first version of this tutorial was written for the 32-bit version of SAS 9.3. The steps used for that version of SAS do not work in 64-bit SAS. We have updated the tutorial to include directions for both versions of SAS.

If you are using SAS 9.3, 32-bit:

Import Wizard (PROC IMPORT)

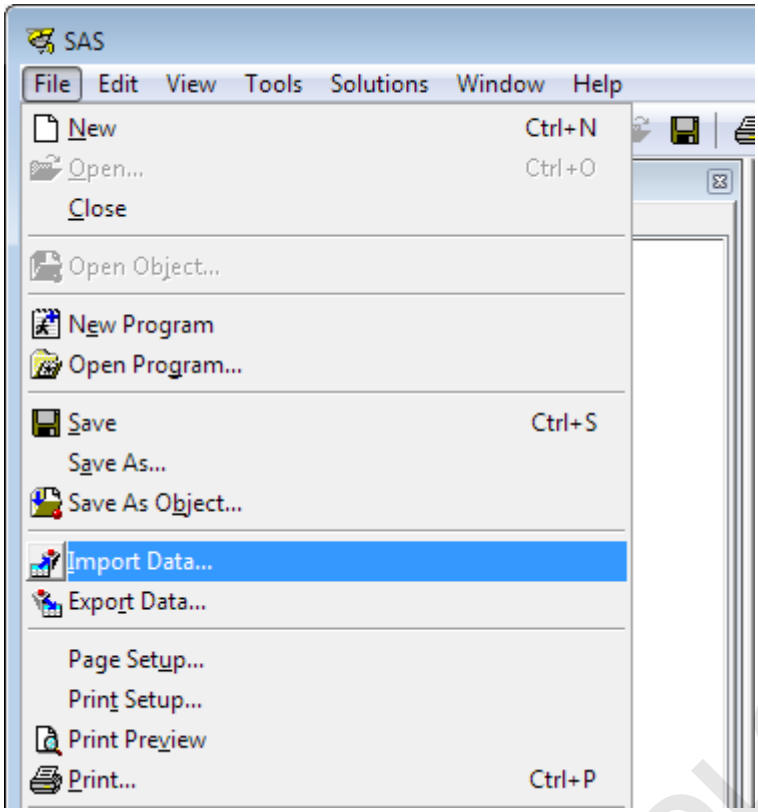
If you are using SAS 9.3 or 9.4 64-bit:

LIBNAME PCFILES

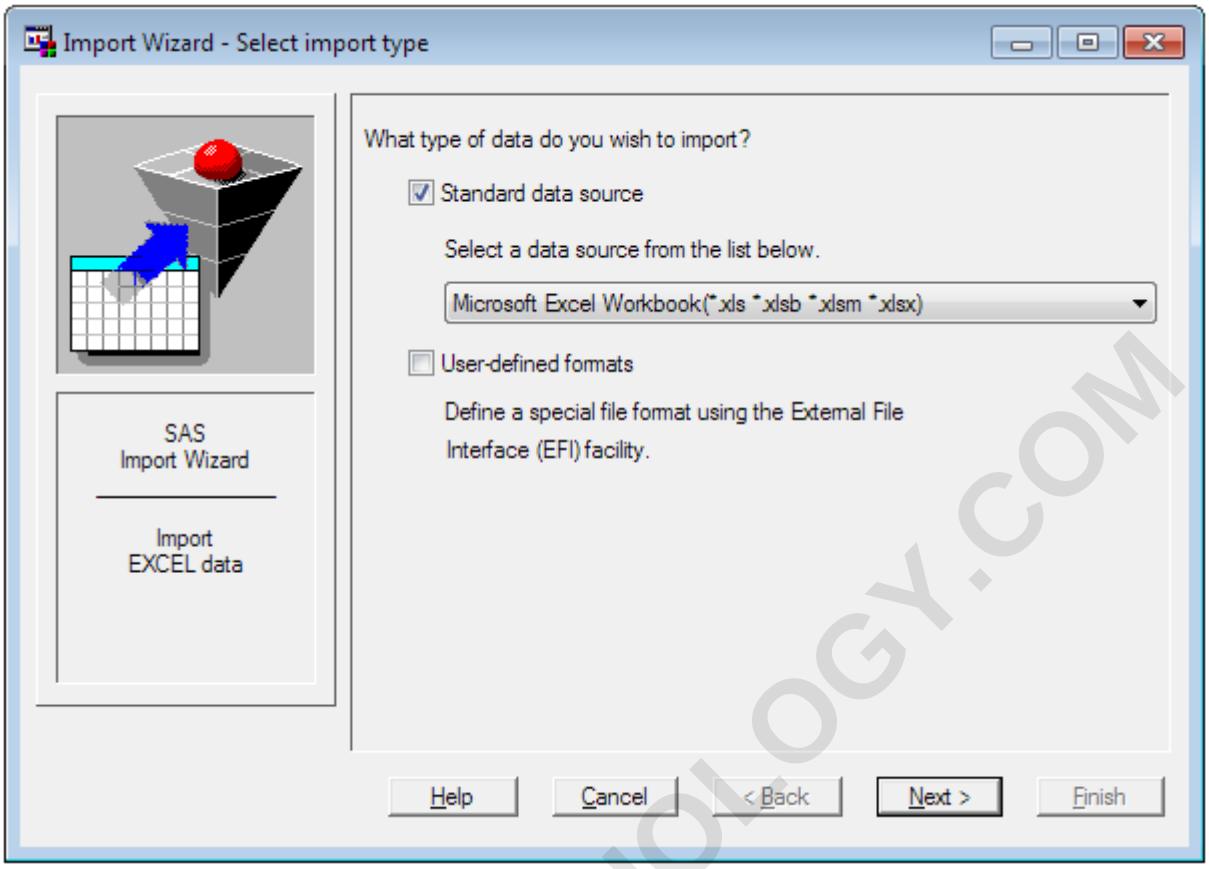
You can check what version of SAS you have by examining the Log window when you first launch SAS.

Importing Excel Files into SAS 9.3 (32-bit) Using the Import Wizard

To start the Import Wizard, click **File > Import Data**. Let's import our sample data, which is located in an Excel spreadsheet, as an illustration of how the Import Wizard works.



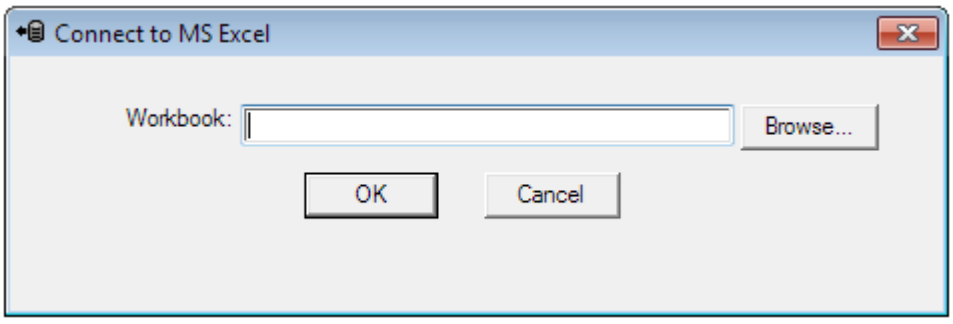
A new window will pop up, called "Import Wizard - Select import type".



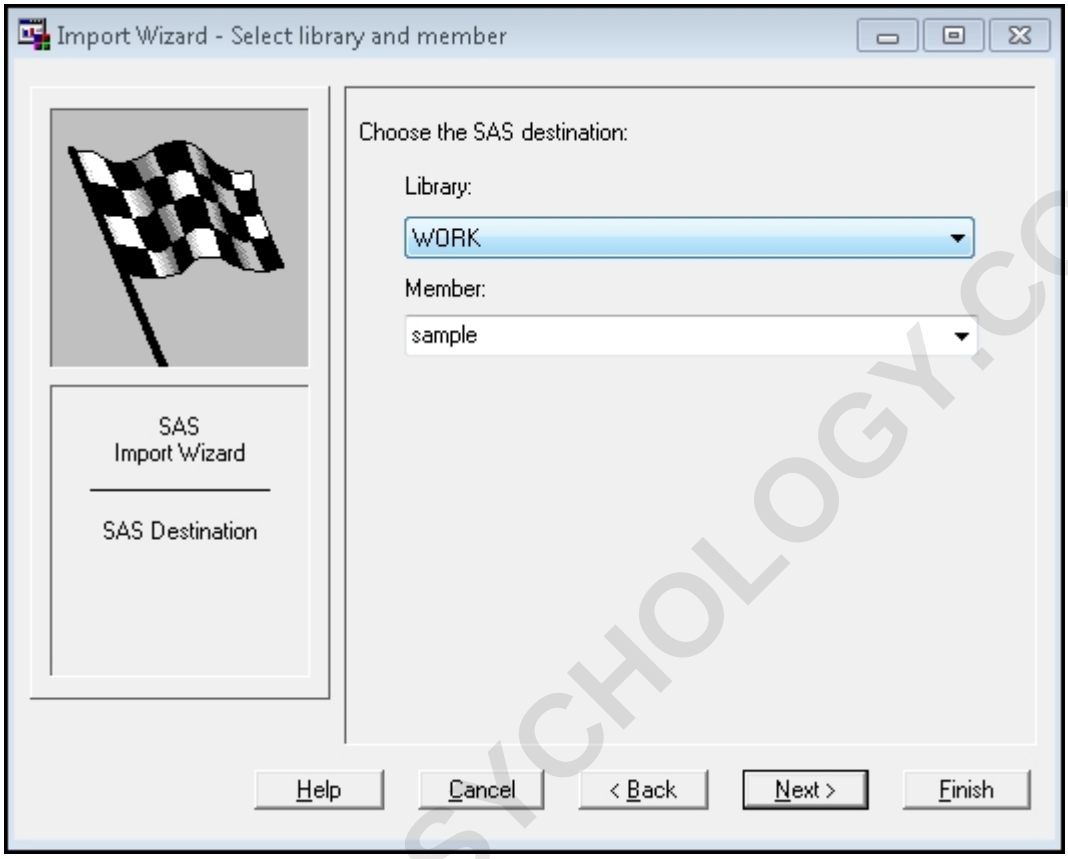
This first screen will ask you to choose the type of data you wish to import. Click **Standard data source** and then choose the program that is the source of your data from the drop down menu. (The second option, specifying a file format, is not covered in this tutorial.)

In our case, the dataset we want to import is an Excel file, so select **Microsoft Excel Workbook**. As you can see, SAS provides you with a large variety of data types to import. Once you've chosen the data source, click **Next**.

Now you need to tell SAS where to find the file you want to import. You can either type the file directory into the text box, or click **Browse** and choose the file to import.



Once you've added the file path to the text box, click **OK**. SAS then asks you what sheet from the file you want to import. In this example we will choose Sheet 1 since our data appears on Sheet 1 in the Excel file. Then click **Options**. Be sure and select the options that are correct for your dataset. The default is for all options to be checked, and that works for our purposes. Click **Next**.



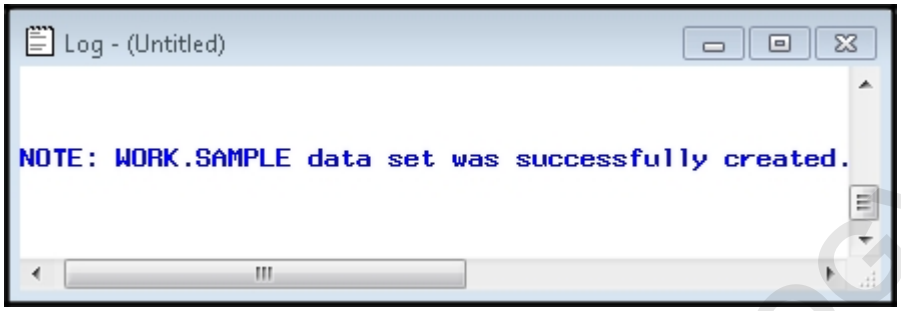
This next step tells SAS where you want to store the newly imported dataset. The first drop-down menu is a list of available libraries that you can choose to store your newly imported SAS dataset in. If you want it to be temporarily stored for now, choose **WORK**. If you've already created a library with a `LIBNAME` statement, you can choose one of those instead.

The next dropdown menu, under **Member**, requires you to name the dataset. You can type in a dataset name here, or choose a dataset from the list. **NOTE: Choosing an existing dataset from the list will over-write that data;** the existing file in the library will be replaced with the file you are importing. Then, click **Next**.

The last step allows you to save the statements that SAS generates while executing the Import Wizard into an *Editor* file. This is recommended. This way you have the import steps saved, and you can go back and re-run it or modify it later if you need to. You can type the file directory directly into the text box, or click on **Browse** to locate a folder to save the program in. Don't forget to name it. The last step is to click **Finish**.

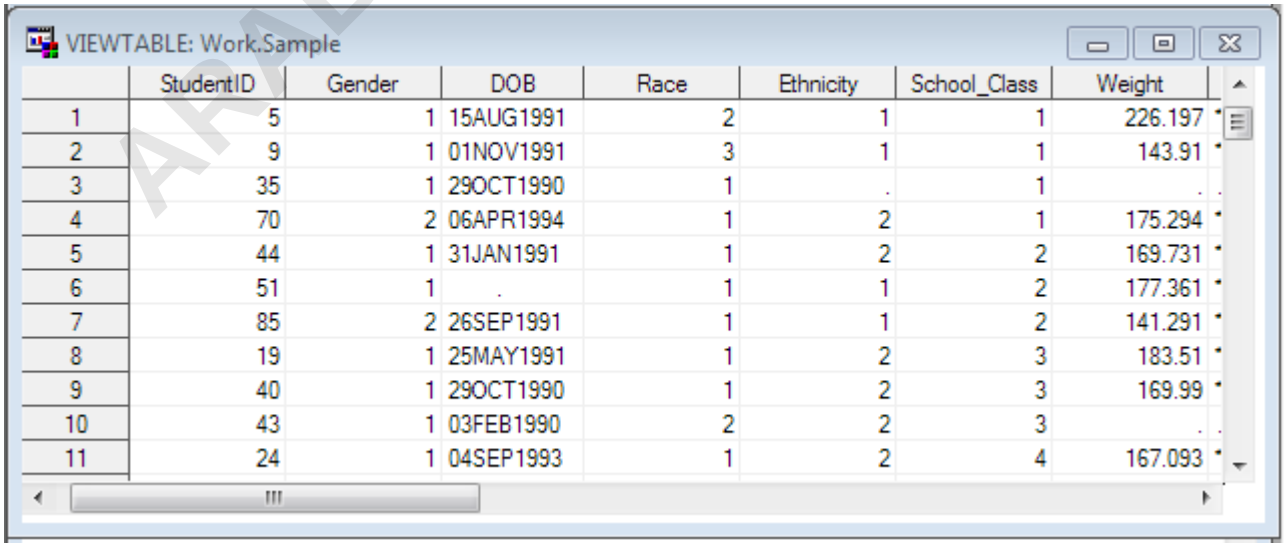
How do I know if it worked?

Clicking on the finish button was probably anticlimactic, because nothing seems to happen. No data appears for you to see and enjoy; no *Editor* file appears for you to manipulate and play with. But let's look a little closer. Remember the earlier tip to look at your *Log* window after you execute any statements in SAS. Running the Import Wizard executes statements in SAS - it's just behind the scenes a bit because the Wizard writes the statements for you - so then the *Log* window is the first place you should look. If you look in the *Log* window you'll see there was some action:



The Log window provides this Note statement that tells you your dataset was successfully created. If there had been a problem with the import, a Warning or Error would have appeared instead.

You will also probably want to look at your data to make sure everything looks right. You can view any of your SAS datasets by finding them in the *Explorer* window. In the *Explorer* window, double-click on **Libraries** to display the Libraries that are available in this session. In this case, the imported dataset is in the temporary *Work* library, so double-click on **Work**. Locate the dataset icon and double-click on it. This will open your data to view in SAS.



	StudentID	Gender	DOB	Race	Ethnicity	School_Class	Weight
1	5	1	15AUG1991	2	1	1	226.197
2	9	1	01NOV1991	3	1	1	143.91
3	35	1	29OCT1990	1	.	1	.
4	70	2	06APR1994	1	2	1	175.294
5	44	1	31JAN1991	1	2	2	169.731
6	51	1	.	1	1	2	177.361
7	85	2	26SEP1991	1	1	2	141.291
8	19	1	25MAY1991	1	2	3	183.51
9	40	1	29OCT1990	1	2	3	169.99
10	43	1	03FEB1990	2	2	3	.
11	24	1	04SEP1993	1	2	4	167.093

Finally, we might want to take a look at the statements generated from the Import Wizard. The Import Wizard saved an *Editor* file, but it did not open it or append it to an already open *Editor* file.

Make sure the *Editor* window is active, and then click **File > Open Program** or click  in the toolbar. Locate the directory that you told the Import Wizard to save your *Editor* file to, highlight the file and click **Open**. It should look like something similar to below.

```
PROC IMPORT OUT=WORK.sample
DATAFILE="C:/mydata/Sample Data.xlsx"
DBMS=EXCEL REPLACE;
RANGE="Sheet1$";
GETNAMES=YES;
MIXED=YES;
SCANTEXT=YES;
USEDATE=YES;
SCANTIME=YES;
```

It starts with a `PROC IMPORT` statement, which triggers the data import action. Note the first semicolon is not until the third line. This is because the syntax includes quite a few options associated with the `PROC IMPORT` statement:

The `OUT` option tells SAS where to put the new SAS dataset it is creating - in this case we told it to put the new SAS dataset "sample" in the *Work* library.The `DATAFILE` option points to the file directory of the dataset you are importing.The `DBMS` option tells SAS the type of data it is importing. Specifically, it tells SAS what *engine* to use to read the data (in this case, the `EXCEL` engine).
The optional `REPLACE` statement says that if there is an existing dataset in SAS's memory with this name, it should be overwritten.

The rest of the lines are statements with further information for SAS - you might recognize these from the list produced after clicking on the *Options* button in the Import Wizard.

`RANGE="Sheet1$"` statement tells SAS what sheet to read; in this case, a sheet named "Sheet1". The dollar sign after the sheet name tells SAS to read the entire sheet. This statement is optional; if omitted, SAS will simply read the entirety of the first sheet in the workbook.`GETNAMES=YES` instructs SAS to use the first row of the file as variable names.`MIXED` controls how SAS "guesses" the appropriate informat for a variable. By default, SAS looks at the first 8 rows of a column, and makes an "educated guess" about what informat is appropriate for the data it encounters. This works well if the values are homogenous, but can fail if the values have been recorded inconsistently. `MIXED=YES` tells SAS that if multiple formats are detected in a column, that column should instead be read in as a string variable. (This ensures that the original information is not lost

during the import process.) (Source: Usage Note 13526: Clarification of MIXED=YES in SAS 9.x SAS/ACCESS Interface to PC Files)`SCANTEXT` applies to columns containing text. If `SCANTEXT=YES`, then SAS will scan the column for the longest string, and uses its length as the column width.`USEDATE=YES` tells SAS to honor any date format settings in the Excel file. That is, if you've added a date format to a column or variable in Excel, SAS will read in that variable using a date format. (If using `USEDATE=NO`, SAS will read it in as a string variable.)`SCANTIME=YES` tells SAS to scan the variables for time-specific formats.

It's important to note that these options are all specific to the `DBMS=EXCEL` engine that is being used. PROC IMPORT can read in other file types, and the options may be different for those file types.

If you are using a 32-bit version of SAS, use `DBMS=EXCEL` in the `PROC IMPORT` statement. If you are using a 64-bit version of SAS, use `DBMS=xls` or `DBMS=xlsx` (whichever is appropriate for the file you're importing). You can determine if you have the 32-bit or 64-bit version of SAS by examining the contents of the Log window when you first open SAS. However, be sure to run PROC CONTENTS to verify that your variables were properly imported - especially long string variables, date variables, and time variables. If any of your variables were misread, you may need to use an alternative method to read the data.

Importing Excel Files into SAS 9.3 or 9.4 (64-bit) using LIBNAME PCFILES

If you've tried to use the Import Wizard to import an Excel file into SAS and have seen the following error message in the Log window:

```
ERROR: Connect: Class not registered
ERROR: Error in the LIBNAME statement.
Connection Failed. See log for details.
```

This error can be especially confusing if you have previously executed PROC IMPORT without issue! The most likely reason you may see this error is because you have a 64-bit version of SAS 9.3 or SAS 9.4 installed. The 64-bit version of SAS uses a different *engine* to read Excel files than 32-bit SAS.

If this is happening, then you most likely will not be able to use the Import Wizard to import Excel files into SAS; you'll have to use an alternative method. One option is to modify the `PROC IMPORT` code above to use `DBMS=XLSX`. However, for many datasets (including the sample dataset), DBMS=XLSX will read the data, but will potentially make several irreversible errors when reading the string and datetime variables:

At least one variable representing a duration in a time-specific format (hh:mm:ss) will be misread using the MONTH2. informatSome string variables may be truncated, since their width exceeds the default length for string variablesThe sheet name contains a space

Instead, the most reliable way to read Excel files into 64-bit SAS is to use `LIBNAME PCFILES`. The approach is slightly different than using `PROC IMPORT`, but is no more difficult to use. The general syntax for `LIBNAME PCFILES` is:

```
/*Step 1: Read in the Excel workbook.*/
LIBNAME myexcel PCFILES PATH="C:/Statistics/Sample Dataset 2014.xlsx"
SCANTIME=YES STRINGDATES=NO DBMAX_TEXT=2000;
```

/*Step 2: Copy the data from the desired spreadsheet into a data set in the WORK library.*/
DATA work.sample;
SET myexcel.'Sample Dataset 2014$'n;
RUN;

In the first line, the `LIBNAME` statement reads the Excel file into a SAS library called `myexcel`, and uses several options to ensure that date, time, and string columns are read properly:

`DBMAX_TEXT=2000` says that the maximum length for string variables is 2000 characters. (By default, this argument is set to 1024, but it can accept any value between 1 and 32767 inclusive.)`SCANTIME=YES` tells SAS to check for time-specific formats in the Excel file, and if they are found, to use a time informat to read the variable.`STRINGDATES=NO` tells SAS to not read in date-format variables as strings. That is, if you have set a variable to have a date format in the Excel file, it will be imported into SAS using a date informat.

If you have Chinese, Korean, or Japanese characters in your datafile, you'll want to add `UNICODE=YES` to your `LIBNAME PCFILES` statement. If you notice that some characters in your string variables are being corrupted, try adding `UNICODE=YES` to your `LIBNAME PCFILES` statement.

At this point, the entire workbook is in a SAS library, but we need to get the data out of a specific sheet before we can use it. In the subsequent data step block, we create a dataset called "sample" in the work library, which is cloned from the sheet named "Sample Dataset 2014" in our Excel file. **Note that in the `SET` statement, the `n` before the semicolon is not a typo.** The `n` is included because the sheet name contains a space. If your sheet's name does not contain any spaces -- e.g., a name like "Sheet1" -- you can omit the n before the semicolon.

SET syntax when Excel sheet name does not contain a space

```
SET myexcel.'Sheet1$';
```

SET syntax when Excel sheet name contains a space

```
SET myexcel.'Sheet 1$'n;
```

Why use the `PCFILES` engine? Isn't there an `XLSX` engine? There is a `LIBNAME XLSX` engine, but it has an extremely limited number of options for reading data. In particular, it does not give the user any control over what informats to use for each variable. This is especially problematic for variables containing dates, times, or strings. Conversely, the `PCFILES LIBNAME` engine is much more flexible: it can read *.xlsx, *.xls, *.xlsb, or *.xlsm files, and it fully supports LIBNAME options for PC files (such as `DBMAX_TEXT` and `SCANTIME`.)

References

SAS/Access 9.4 Interface to PC Files: Reference, Fourth Edition: SAS LIBNAME Statement for PC Files: OptionsSAS/ACCESS 9.4 Interface to PC Files: Comparing SAS LIBNAME Engines for Microsoft Excel DataThe SAS Dummy Blog: The top gotchas when moving to 64-bit SAS for WindowsThe SAS Dummy Blog: Using LIBNAME XLSX to read and write Excel files

Other Ways to Import Data

Using the Import Wizard is an easy and straightforward way to import existing data with well-behaved formatting into SAS. There are other methods for importing data into SAS, or even entering raw observations into SAS itself to create a new dataset. These methods of importing or creating data can give you greater control over how to read variables (the informats), how to write the variables (the formats), how to parse the data (delimited, aligned, repetition, etc.), and more.

Creating a dataset using the Viewtable Window

Data can be manually entered in the Viewtable Window, spreadsheet-style.

Creating a dataset within a data step using instream data

If you have a very small dataset, you can use the CARDS or DATALINES statements within a data step to read manually entered observations. Data entry in this format is text-based, so you will most likely use some kind of delimiter.

Importing data within a data step

The `INFILE` statement in a data step can be used as an alternative to the Import Wizard for importing existing data from file. It is especially useful if the formatting of the data in the file is non-

standard: for example, if you have more than one subject's observations per line.

This tutorial does not cover these methods, but you can find more information about these techniques in these tutorials:

UCLA SAS Learning Module: Inputting Data into SASHow to manually enter delimited data in a data step using the CARDS or DATALINES statements; how to use the INFILE statement in a data step to read data from a file.