How to Easily Import Excel Data into SAS

Authored by

stats writer

December 1, 2025

Introduction to Data Management and SAS

The process of loading external data sources into an analytical environment is a fundamental task for any data scientist or analyst. When dealing with proprietary formats like Excel spreadsheets, efficient conversion is crucial. SAS (Statistical Analysis System) provides powerful tools to manage and analyze massive datasets, but leveraging these capabilities first requires getting the data into a usable SAS dataset format.

Traditionally, importing proprietary spreadsheet data could involve intermediate steps, such as saving the source file as a delimited text format like CSV. While this approach is robust, it often adds unnecessary complexity and steps to the workflow. Analysts seek direct, streamlined methods to preserve data integrity and metadata during the transition from a spreadsheet application to a statistical programming environment.

Fortunately, modern versions of SAS, especially when dealing with **.xlsx** or older **.xls** files, offer native procedures that automate this conversion. The primary tool utilized for this seamless integration is the **IMPORT procedure**, which intelligently reads various file formats and maps the source data directly into a structured SAS dataset, significantly enhancing efficiency and reducing the margin for error associated with manual formatting or conversion.

Understanding the PROC IMPORT Statement

The PROC IMPORT statement stands as the cornerstone of external data ingestion within the SAS programming environment. It is a specialized procedure designed to convert data from common file types--including text files, database tables, and spreadsheet formats like Excel--into a native SAS data file. This procedure eliminates the need for manual data definition, as it attempts to infer the correct variable types (numeric, character, date, etc.) based on the source data content.

The flexibility of PROC IMPORT is derived from its reliance on Access Descriptor files and various engine libraries that enable interaction with diverse data sources. For importing Excel files specifically, PROC IMPORT leverages the capabilities associated with the designated Database Management System (DBMS) engine, such as the **XLSX** engine for modern Excel files. Understanding the correct syntax and parameter usage is vital for successful data transfer and accurate variable mapping.

When preparing to execute PROC IMPORT, the user specifies three key pieces of information: the location of the source file (**DATAFILE**), the format of that file (**DBMS**), and the desired name and location of the output SAS dataset (**OUT**). Furthermore, numerous optional statements allow granular control over how the import handles variable names, sheet selection, and existing file replacement, ensuring that the imported data meets precise analytical requirements.

Detailed Syntax Breakdown of PROC IMPORT

To effectively use PROC IMPORT for high-volume or recurring tasks, it is essential to master its core syntax structure. This structure is relatively simple yet powerful, relying on keyword parameters to define the necessary actions.

The standard syntax for importing data from an Excel workbook uses the following structure:

```
/*import data from Excel file called my_data.xlsx*/
proc import out=my_data
datafile="/home/u13181/my_data.xlsx"
dbms=xlsx
replace;
getnames=YES;
run;
```

Each parameter within this structure serves a specific function critical to the import operation. Mastery of these options ensures data integrity and operational efficiency when dealing with diverse spreadsheet layouts and organizational standards.

Here is a detailed explanation of the essential statements and options utilized within the **PROC IMPORT** procedure:

out: This required keyword specifies the name of the resulting SAS dataset once the data has been successfully imported. If a two-level name is used (e.g., **libref.dataset_name**), the output file will be saved permanently to the specified SAS library.
datafile: This keyword defines the exact path and filename of the source external file, which in this context is the Excel workbook (**.xlsx** or **.xls**) being imported. Using a fully qualified path is crucial to prevent errors related to file location.
dbms: This option specifies the Database Management System (DBMS) type or engine that SAS should use to read the external file. For modern Excel files, **XLSX** is the standard designation. For older formats, **EXCEL** or **PCFILES** might be used, depending on the SAS configuration.
replace: This optional statement instructs SAS to overwrite the output dataset if a dataset with the same name already exists in the specified library. If **REPLACE** is omitted and the dataset exists, the procedure will fail, preventing accidental data loss.
getnames: This is a crucial optional statement that determines whether the first row of the source data should be used to define the variable names in the resulting SAS dataset. Setting **GETNAMES=YES** (which is highly common when importing structured data) uses the header row, while **GETNAMES=NO** assigns generic names like **VAR1**, **VAR2**, etc.

Preparing the Excel Data for Import

While PROC IMPORT is highly capable, the quality and structure of the source Excel file significantly impact the success and accuracy of the resulting SAS dataset. Preparing the data beforehand minimizes potential errors and ensures correct data type inference.

First, ensure that the data intended for import is located within a single, contiguous range on a specific worksheet. Avoid merged cells, extraneous headers, or footers above the actual data table, as these structures can confuse the import procedure when attempting to identify the true beginning of the dataset.

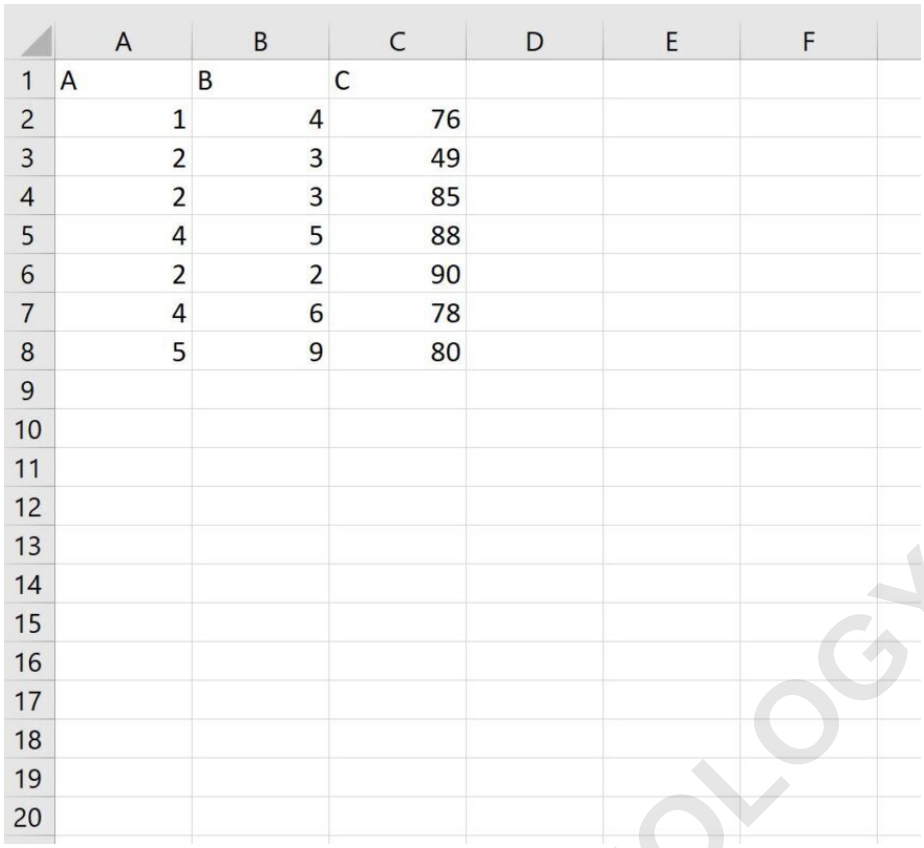
Second, pay close attention to the variable names (the header row). If **GETNAMES=YES** is used, the names in the first row should conform to SAS naming rules (e.g., typically starting with a letter or underscore, and often limited in length depending on the SAS version). Although SAS can often auto-correct invalid characters by converting them to underscores or truncating them, using clean headers in Excel prevents unexpected renaming issues.

Finally, review data types. PROC IMPORT determines variable types based on the content of the first few rows. If a column contains mostly numbers but includes a single text entry in the first 20 rows, SAS might interpret the entire column as character data. Cleaning up mixed data types in Excel prior to import is a best practice to maintain intended variable formats in the SAS environment.

Step-by-Step Example: Importing Excel Data

To demonstrate the practical application of the **PROC IMPORT** procedure, let us consider a typical scenario where an analyst needs to load sales data contained within an Excel spreadsheet into SAS for analysis.

Suppose we have the following sample dataset structured in an Excel file named **my_data.xlsx**. This dataset includes columns for **EmployeeID**, **Region**, and **SalesAmount**, with the first row serving as the clear header:



	A	B	C	D	E	F
1	A	B	C			
2	1	4	76			
3	2	3	49			
4	2	3	85			
5	4	5	88			
6	2	2	90			
7	4	6	78			
8	5	9	80			
9						
10						
11						
12						
13						
14						
15						
16						
17						
18						
19						
20						

To import this data, we must specify the file path, indicate the use of the **XLSX** engine via the DBMS option, and ensure that the header row is captured as variable names. We will name the resulting SAS data file **new_data**, ensuring that if it already exists, it is overwritten using the **replace** option.

We can use the following code to import this dataset into SAS and call it **new_data**:

```
/*import data from Excel file called my_data.xlsx*/
proc import out=new_data
datafile="/home/u13181/my_data.xlsx"
dbms=xlsx
replace;
getnames=YES;
run;

/*view dataset using PROC PRINT to confirm import success*/
proc print data=new_data;
run;
```

Reviewing and Validating the Imported Dataset

Upon execution of the PROC IMPORT procedure, validation is a mandatory step to ensure that the data structure and content accurately reflect the source Excel file. The use of **PROC PRINT**, as shown in the example, provides a quick visual confirmation of the data rows and variable names.

Obs	A	B	C
1	1	4	76
2	2	3	49
3	2	3	85
4	4	5	88
5	2	2	90
6	4	6	78
7	5	9	80

As confirmed by the output displayed above, the data successfully transitioned from the Excel environment into the SAS environment. The structure and content of the SAS output perfectly match the data presented in the original Excel file, confirming that the import was executed without loss or alteration of the records.

It is important to reiterate the critical role of the **GETNAMES=YES** option in this scenario. Since the first row of the source Excel file contained meaningful descriptive titles for the columns, setting **GETNAMES=YES** ensured these titles were correctly utilized as the variable names within the resulting **new_data** SAS dataset. Had the source file lacked headers or contained non-data elements in the first row, **GETNAMES=NO** would have been appropriate, necessitating subsequent renaming using procedures like **PROC DATASETS**.

Handling Specific Import Challenges and Options

While the basic syntax covers most import needs, advanced options within PROC IMPORT allow users to address specific challenges posed by complex Excel workbooks. For instance, when dealing with files containing data that spans across multiple sheets, the **SHEET=** option is indispensable.

The **SHEET=** option allows the user to explicitly name the worksheet (e.g., **SHEET="Q4_Sales_Data"**) or specify the sheet index number (e.g., **SHEET="2"** for the second sheet). Without this specification, SAS typically defaults to importing the first available sheet, which may not always contain the intended analytical data.

Another common challenge is importing only a specific range of data within a worksheet, rather than the entire sheet. The **RANGE=** option is used for this purpose, allowing the analyst to define a standard Excel cell range (e.g., **RANGE="A1:F50"**). This is especially useful if the workbook contains extraneous summary tables or commentary outside the main data block. Using **RANGE=** in conjunction with **GETNAMES=** allows precise control over which rows are interpreted as headers and which constitute the dataset body.

Alternative Import Methods in SAS

While direct use of PROC IMPORT with the **DBMS=XLSX** engine is the most common and robust method for handling Excel files in SAS programming, two other methods offer flexibility depending on the user environment and requirements: the SAS Import Wizard and utilizing ODBC connectivity.

The **SAS Import Wizard** provides a graphical user interface (GUI) alternative for users who prefer point-and-click operation over writing code. Available in environments like SAS Enterprise Guide or SAS Studio, the Wizard walks the user through steps such as selecting the input file, defining the output dataset name, and visually verifying the variable mappings. A key advantage of the Wizard is that it often generates the corresponding **PROC IMPORT** code, allowing the user to save and reuse the script later, bridging the gap between GUI convenience and scripting efficiency.

For highly interconnected environments or scenarios requiring real-time data access, **ODBC (Open Database Connectivity)** provides a powerful alternative. By setting up an ODBC connection to the Excel file, users can treat the spreadsheet as a temporary database. Procedures like **PROC SQL** can then be used with the **LIBNAME** statement to directly query or import data from the Excel sheets. Although more complex to set up, ODBC connections offer superior flexibility for querying specific subsets of data without requiring the entire sheet to be loaded.

Conclusion and Further Resources

Mastering the importation of external data, particularly proprietary formats like Excel, is essential for effective data management and analytical processing within the SAS environment. The **PROC IMPORT** procedure provides a streamlined, reliable, and highly configurable method for achieving this goal.

By correctly defining the source file path (**DATAFILE**), the engine type (**DBMS=XLSX**), and crucial options such as **GETNAMES** and **REPLACE**, analysts can quickly transform spreadsheet data into a structured SAS dataset ready for immediate statistical analysis or advanced data manipulation. Adhering to best practices, such as pre-cleaning the Excel source and validating the imported structure using **PROC PRINT** and **PROC CONTENTS**, ensures the highest quality of imported data.

The skills covered here form the foundation for integrating SAS with the broader data ecosystem. For those seeking to deepen their knowledge, exploring advanced options for handling date and time formats during import, or learning how to utilize the SAS Import Wizard for rapid prototyping, offers valuable next steps in becoming proficient with SAS data management capabilities.