Learning SAS: Importing Text Files Using PROC IMPORT – A Comprehensive Guide

Authored by

Mohammed looti

November 14, 2025

In the complex and demanding environment of advanced statistical computing and data management, the ability to seamlessly integrate external source data into an analytical software system is not merely convenient--it is a foundational requirement. This comprehensive guide is dedicated to mastering the process of incorporating raw information from an external text file into the leading statistical analysis system, SAS. We will provide a deep dive into the usage of the versatile **PROC IMPORT** procedure, a utility meticulously engineered to transform disorganized external data into a structured, analytical dataset ready for rigorous statistical examination.

Raw text files, which include standardized formats such as comma-separated values (CSV) and various delimited structures, remain one of the most common and universally compatible methods for storing data across diverse industries. For any data professional handling empirical, operational, or scientific information, achieving proficiency in the correct procedures for file ingestion is absolutely critical. This article provides an exhaustive, step-by-step methodology, accompanied by clear, executable examples, designed to ensure you can confidently navigate and resolve various text file import challenges within the SAS environment, thereby unlocking the full analytical potential of your collected information.

Streamlining Data Ingestion with the PROC IMPORT Statement

The **PROC IMPORT** statement serves as an indispensable utility within SAS for efficiently reading and structuring data sourced from a multitude of external file types, including plain text, various spreadsheet formats, and database tables. Its primary and most significant advantage lies in its inherent capacity for automatic data structure inference. This intelligence dramatically simplifies the ingestion process by minimizing manual effort, standing in sharp contrast to the labor-intensive requirement of manually defining every data field, length, and data type through a traditional DATA step combined with an INPUT statement.

The fundamental syntax required to initiate a data import from a text source into SAS is remarkably concise and intuitive. It mandates the precise specification of three core parameters: the exact location, known as the file path, of the input source file; the designated name for the resulting output dataset within the SAS library; and the format or structural type of the external file. By intelligently handling common data structures and variations, this automated procedure substantially minimizes the need for extensive, custom programming. The following code snippet illustrates the foundational structure required for successfully invoking **PROC IMPORT**:

```
/*import data from text file called data.txt*/
proc import out=my_data
datafile="/home/u13181/data.txt"
dbms=dlm
replace;
```

```
getnames=YES;
run;
```

This foundational block of code represents the essential building blocks for executing a reliable and structured import operation. Every option specified within the procedure block performs a critical function, instructing SAS precisely on how to interpret, parse, and ultimately store the external data in a standardized, structured format. A thorough comprehension of these options is paramount to leveraging the full efficiency and power inherent in the **PROC IMPORT** utility, ensuring data integrity from the very first step.

Deconstructing the Core Syntax and Essential Options

To execute **PROC IMPORT** effectively, data analysts must possess a clear and granular understanding of the purpose and function behind each command option. These parameters provide the necessary level of customization, ensuring that the source data is interpreted accurately and stored in the desired format within the SAS environment. The following list details the most frequently used and critical options required when importing standard text files:

`out`: This required option designates the specific name assigned to the resulting SAS dataset upon successful completion of the import. For instance, using `out=project_data` instructs SAS to create a new dataset named `project_data` within the currently active working library. Selecting a descriptive and meaningful name is vital for maintaining clarity, organization, and adherence to best practices in the data management workflow.

`datafile`: This crucial parameter requires the specification of the complete system path and file name of the external text file intended for ingestion. It acts as the absolute locator, informing SAS precisely where the source data physically resides on the host system or network. Accuracy in specifying this path is essential to prevent common file access errors and procedure termination.

`dbms`: This is a crucial control option that defines the Database Management System (DBMS) or, more practically, the format type of the external source file. For standard delimited text files, **`dbms=dlm`** is the typical and recommended choice, signaling to SAS that data fields are separated by a delimiter (by default, a space or tab). Alternatives include `CSV` for comma-separated files or `TAB` for explicit tab-separated structures. Specifying the correct `dbms` value is paramount for SAS to accurately parse the structural boundaries and content of the file.

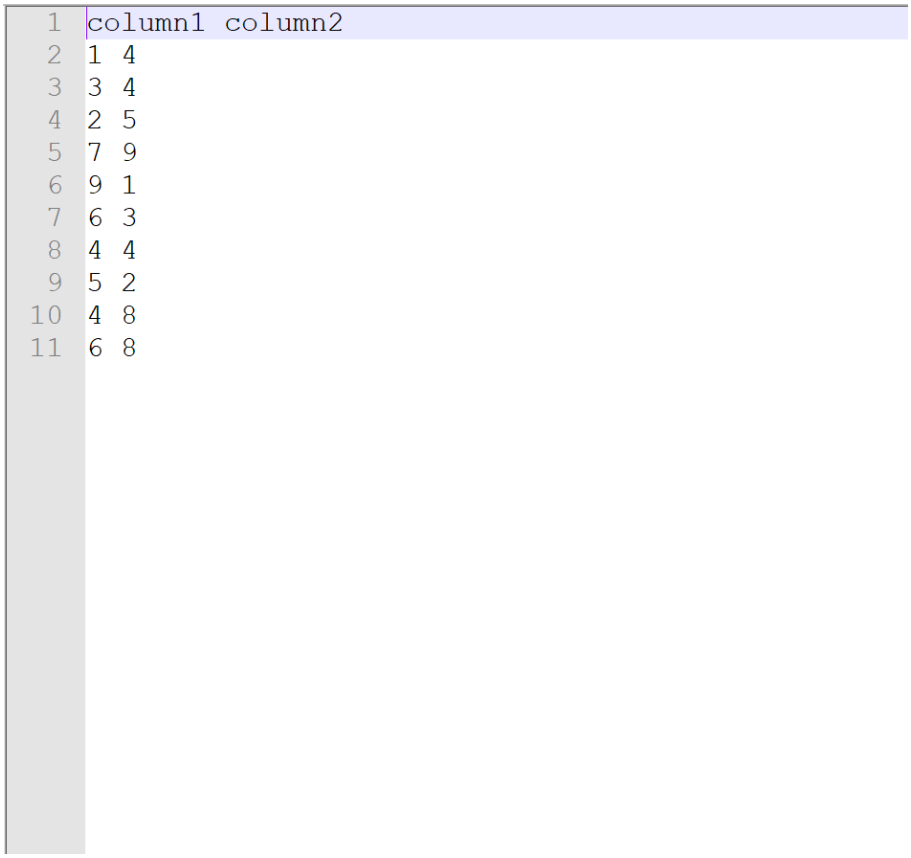
`replace`: When this option is included, the `replace` directive grants SAS explicit permission to overwrite the output dataset if a dataset with the identical name already exists within the specified library. If this option is omitted and a name conflict occurs, the SAS procedure will typically terminate with an error, protecting existing data. Due to its potential for permanent data loss, analysts should exercise extreme caution when utilizing `replace`, particularly within sensitive production or shared data environments.

`getnames`: This binary option controls whether the first physical row of the external source file should be interpreted as the official column headers, or variable names, for the resulting SAS dataset. Setting **`getnames=YES`** ensures that the first row values are used as descriptive headers. Conversely, setting `getnames=NO` instructs SAS to treat the first row as observational data and automatically assign generic variable names, such as VAR1, VAR2, and so forth.

By judiciously configuring and applying these core options, data professionals achieve meticulous and robust control over the initial data ingestion process, thereby establishing an accurate and reliable foundation for all subsequent data manipulation and complex statistical analysis within the SAS environment.

Practical Walkthrough: Importing a Delimited Text File

To solidify the theoretical understanding of the **PROC IMPORT** statement, we will now proceed through a detailed, practical demonstration. Consider a common scenario where we need to integrate data from a raw text file named **`data.txt`**, which is located at the system path `/home/u13181/data.txt`. This file contains transactional sales data, including key indicators such as product identification codes, revenue figures, and associated regional identifiers. The internal structure of this source file is visualized in the image below:



```
column1 column2
1 4
3 4
2 5
7 9
9 1
6 3
4 4
5 2
4 8
6 8
```

As clearly depicted, the inaugural row of **`data.txt`** serves as the header, containing the descriptive variable names (`product`, `sales`, `region`), while the subsequent rows hold the actual sales observations. Crucially, the individual values within each row are separated by single spaces, classifying this document as a space-delimited file. Our specific objective is to import this raw data into the SAS system, assigning the descriptive name **`new_data`** to the final resulting internal dataset.

The following SAS code is constructed to execute this import task efficiently and accurately. This specific programming block not only manages the core ingestion process but also strategically incorporates a subsequent **PROC PRINT** statement. This immediate post-import verification step is considered a critical best practice in data workflow, as it enables rapid visual verification of the newly created SAS dataset, ensuring the structural integrity and data accuracy of the migration process before proceeding to analysis:

```
/*import data from text file called data.txt*/
proc import out=new_data
datafile="/home/u13181/data.txt"
dbms=dlm
replace;
getnames=YES;
run;

/*view dataset*/
proc print data=new_data;
```

Verifying the Import: Using PROC PRINT

Upon successful execution of the code presented in the previous section, SAS processes the source **`data.txt`** file, creating the designated internal dataset named **`new_data`**. Subsequently, the accompanying **PROC PRINT** statement generates a clean, readable output table. This output should precisely reflect the structure and content of the original external text file, serving as conclusive confirmation of the accuracy and integrity of the import routine.

Obs	column1	column2
1	1	4
2	3	4
3	2	5
4	7	9
5	9	1
6	6	3
7	4	4
8	5	2
9	4	8
10	6	8

The resulting SAS output clearly demonstrates that the data has been ingested successfully into the SAS environment. The columns corresponding to `product`, `sales`, and `region` were correctly identified and assigned as the official variable names, and the associated data values were accurately populated into the new table structure. This confirmation validates that the **PROC IMPORT** statement, specifically utilizing the configured options (especially `dbms=dlm` for space-delimited data), effectively converted the raw text input into a structured, analytical SAS dataset.

It is paramount to emphasize the critical and specific role of **`getnames=YES`** in this scenario. Since the initial row of the **`data.txt`** file contained meaningful descriptive labels intended to serve as permanent column headers, this option ensured that SAS correctly interpreted these labels and assigned them as the formal variable names for the **`new_data`** dataset. Had the first row contained observational data rather than headers, specifying `getnames=NO` would have been the necessary instruction, prompting SAS to generate standardized, default variable identifiers (e.g., VAR1, VAR2).

Advanced Considerations and Troubleshooting Best Practices

While the **PROC IMPORT** procedure is designed primarily for simplicity and efficiency, mastering a few advanced considerations and best practices is essential for handling the complexities inherent in diverse text files and preventing common data integrity issues. These insights are crucial for robust and reliable data preparation in professional settings.

A frequent challenge encountered by analysts involves dealing with varying delimiters. Although `dbms=dlm` defaults to recognizing space or tab separation, many files employ commas, semicolons, or pipe symbols (`|`). To manage these variations effectively, the analyst must explicitly use the **`DELIMITER=`** option within the PROC IMPORT statement. For example, `dbms=dlm`

`delimiter=','` specifies a comma separator, while `delimiter='09'x` explicitly designates a tab character using its hexadecimal representation. This necessary flexibility ensures that SAS can accurately parse data fields regardless of the specific character used for field separation.

Another crucial aspect involves the systematic management of missing values. External text files often encode missing observations in inconsistent ways--using empty spaces, periods (`.`), or textual markers like 'NA' or 'NULL'. While **PROC IMPORT** generally handles standard missing value representations well, complex or non-standard indicators may necessitate meticulous preprocessing of the source file. Alternatively, for maximum, granular control over how these specific indicators are treated and converted into standard SAS system missing values, utilizing the fundamental DATA step with INFILE and INPUT statements offers superior precision and customization.

Furthermore, SAS automatically attempts to infer the data types (e.g., numeric, character) for each variable based on scanning the initial rows of data. This inference process is typically accurate but can fail when dealing with complex mixed data or when character fields (like product IDs) contain numeric characters that SAS incorrectly interprets as true numeric variables. Should the inferred data type or length prove unsuitable for analysis, the analyst may need to resort to the more explicit control offered by a custom DATA step. This manual approach allows for the precise definition of data types, lengths, formats, and informats, ensuring complete data integrity before any statistical procedure is initiated.

Finally, data professionals must always treat the SAS log file as the single, authoritative source for troubleshooting and verification. After every run of **PROC IMPORT**, the log provides detailed feedback, including the count of observations and variable names read, any truncation warnings, or critical errors related to data conversion. A diligent and careful review of the log is the single most important step for verifying the completeness and structural soundness of your newly imported analytical dataset.

Conclusion: Mastering Data Ingestion for Analysis

The successful and efficient importation of text files constitutes a non-negotiable prerequisite for effective statistical analysis and data modeling. The **PROC IMPORT** statement provides a highly user-friendly and exceptionally efficient mechanism to accomplish this critical task, simplifying the initial, often complex, stage of data ingestion. By thoroughly understanding its fundamental operational parameters--**`OUT=`**, **`DATAFILE=`**, **`DBMS=`**, **`REPLACE`**, and **`GETNAMES=`**--data professionals gain the essential capability to accurately translate raw external data into a perfectly structured analytical SAS dataset.

Our detailed practical example provided a clear demonstration of importing a space-delimited file, illustrating the seamless workflow from raw external data to a usable SAS table structure. We also

underscored the importance of proactive measures, such as carefully managing various delimiters and explicitly handling missing values, along with the critical practice of reviewing the SAS log file to ensure maximum data integrity. Mastery of the **PROC IMPORT** procedure will substantially increase both productivity and confidence when engaging with diverse and challenging data sources.

We strongly recommend practicing these techniques using your own empirical data to firmly embed this knowledge and develop muscle memory for common import tasks. The proficiency to efficiently ingest data is the crucial precursor to deriving meaningful insights through rigorous statistical methods. For those highly complex scenarios requiring the most granular level of control over data definitions and transformations, continued exploration of the DATA step, utilizing the INFILE and INPUT statements, will provide the necessary additional flexibility and precision.

Additional Resources

To further deepen your expertise and expand your skill set within the SAS environment, consider engaging with the following high-quality resources and tutorials that focus on common data management and analytical tasks:

For comprehensive, authoritative details on **PROC IMPORT**, consult the official SAS documentation.
Tutorials focused on accurately handling various complex and non-standard data formats.
In-depth guides covering data cleaning, validation, and transformation techniques in SAS programming.
Practical examples demonstrating the application of statistical procedures using newly imported datasets.