How do I export a SAS dataset to an external file?

Authored by

stats writer

November 19, 2025

The ability to share and analyze data across different platforms is a core requirement in modern data processing. When utilizing the SAS (Statistical Analysis System) environment, researchers and analysts frequently need to move proprietary SAS datasets into formats that can be easily read by external applications, such as Microsoft Excel or SPSS. Achieving this data migration efficiently relies almost exclusively on employing the powerful PROC EXPORT procedure. This specialized procedure is designed to write a SAS dataset--which is typically stored in a highly optimized binary format--to a file in a specified external format, ensuring seamless compatibility across systems. To successfully execute the PROC EXPORT process, the user must meticulously define three critical parameters: the full path and filename of the target output file, the name of the source SAS dataset being exported, and the precise data format (or DBMS identifier) required for the external file. Once these specifications are provided and the procedure runs, the data is transformed and saved, making it instantly accessible for further analysis or reporting using alternative software tools.

Fundamentals of the PROC EXPORT Statement

In SAS programming, the PROC EXPORT statement serves as the primary mechanism for converting SAS data files into various external file types. This procedure offers robust control over the export process, allowing users to handle everything from simple comma-separated value files to complex database formats. Understanding the fundamental syntax is the first step toward mastering this essential data management skill, as it dictates how SAS interacts with the operating system and the specified output file location.

The general syntax for PROC EXPORT is intuitive yet highly specific regarding parameter placement. It typically begins with the procedure call, followed by necessary keywords that define the source, destination, and format. Although many optional arguments exist for advanced use cases, the following core structure provides the necessary requirements for nearly all basic export operations, utilizing the **PROC EXPORT** statement to export datasets in SAS to external files.

This statement uses the following basic syntax, which demonstrates a complete, operational SAS statement that converts the internal SAS dataset named `my_data` into an external file:

```
proc export data=my_data
outfile="/home/u13181/my_data.csv"
dbms=csv
replace;
run;
```

Every component in the syntax plays a distinct and critical role in ensuring the data transfer is

successful, from identifying the input source to defining the handling of existing files at the destination path. Proper specification of these parameters is crucial; errors in file paths or DBMS definition often lead to immediate execution failure of the data export procedure.

Deconstructing the Core Export Parameters

To use the PROC EXPORT procedure effectively, it is essential to understand the exact function of each required and frequently used parameter. These parameters control the flow of data from the SAS environment to the external file system, offering flexibility in naming conventions, destination paths, and format conversion. Below is a detailed breakdown of the standard arguments used within the procedure:

data: This required keyword specifies the name of the dataset that you intend to export. This dataset must already exist within the current SAS session or library structure. If the dataset name is omitted or misspelled, SAS will halt execution, requiring the user to verify the source data's availability and spelling.
outfile: This parameter defines the complete file path and name for the exported external file. It must include the directory where the file will be saved and the desired filename, including the appropriate extension (e.g., .csv, .xlsx, .txt). The path must be valid and accessible by the SAS environment, especially when operating on server-based systems.
dbms: Standing for Database Management System, this crucial argument tells SAS the exact format the output file should adhere to. It determines the structure of the resulting file, whether it should be a flat file like CSV, a proprietary format like Excel, or a specific database type. The value assigned here dictates the necessary internal conversion process SAS performs on the data.
replace: This is an optional but highly recommended statement. When included, it instructs SAS to automatically overwrite the target file if a file with the same name already exists at the specified **outfile** location. If **replace** is omitted and the file already exists, the procedure will typically terminate with an error, preventing accidental data duplication or conflict.

Mastering these four elements allows users to confidently manage data interchange between SAS and other statistical or analytical programs. It is worth noting that depending on the chosen DBMS, certain additional optional parameters might be necessary, such as specifying sheet names when exporting to multi-tab workbooks in Microsoft Excel.

Configuring Output Formats via the DBMS Option

The versatility of PROC EXPORT largely stems from the flexibility provided by the **dbms** parameter. This parameter acts as a critical translator, instructing SAS how to structure the output data stream to conform to external software standards. The value assigned to **dbms** is always a short, standardized identifier representing the target file format.

You can use this general syntax framework to export SAS datasets to a wide range of popular file types. The only necessary change is adjusting the value for the **dbms** argument based on the desired output format. For instance, exporting to a standard text format requires a different identifier than exporting to a spreadsheet format designed for graphical viewing and formula computation. The specific identifier dictates how delimiters, quoting rules, and data types are handled during the conversion from the internal SAS format.

Below are common examples of how the **dbms** argument is utilized to target different file types:

To export to a standard CSV file, specify the identifier: **dbms=csv**.
To export to an Excel file (using the modern XLSX standard), specify: **dbms=xlsx**.
To export to a Text file (using a tab delimiter), specify: **dbms=tab**.

The following examples show how to use **PROC EXPORT** to export SAS datasets to each of these file formats, demonstrating the specific application of the **dbms** argument in practical scenarios.

Example 1: Exporting a SAS Dataset to a CSV File

Exporting to a CSV file is arguably the most common data export requirement, as CSV files are universally supported by nearly all statistical, spreadsheet, and database software. CSV files represent data as plain text, where values are typically delimited by commas. This example details the creation of a sample SAS dataset and its subsequent export using the **dbms=csv** option.

Suppose we first create the following dataset in SAS using the `DATA` step and confirm its structure using `PROC PRINT`:

```
/*create dataset*/
data my_data;
input A B C;
datalines;
1 4 76
2 3 49
2 3 85
4 5 88
2 2 90
4 6 78
5 9 80
;
run;

/*view dataset*/
```

proc print data=my_data;

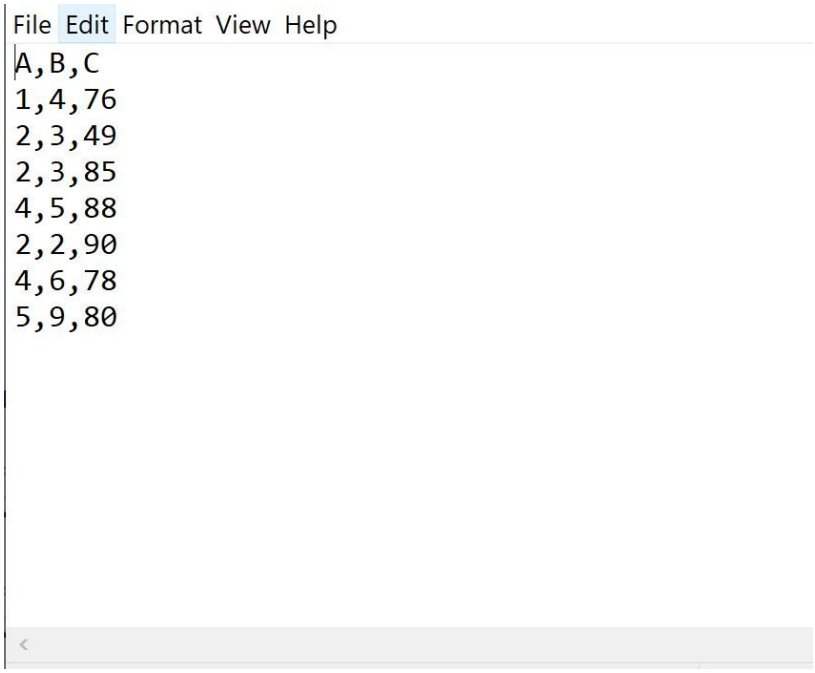
The structured dataset, once created in the SAS session, appears as follows:

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

We can use the following code to export this dataset to a CSV file called **data.csv**. Notice the critical inclusion of `dbms=csv` and the `replace` statement for non-interactive overwriting:

```
/*export dataset*/
proc export data=my_data
outfile="/home/u13181/data.csv"
dbms=csv
replace;
run;
```

Upon successful execution, I can then navigate to the location on my computer where I exported the file and view it in an external application, confirming that the data structure and content align perfectly with the original SAS dataset:



Example 2: Exporting a SAS Dataset to an Excel File

Exporting data directly into a proprietary spreadsheet format like Microsoft Excel (using the modern XLSX standard) is essential when the data needs to be delivered to end-users who rely on Excel for immediate viewing and manipulation. This requires setting the DBMS identifier to `xlsx`. Furthermore, we can utilize the optional **sheet** parameter to name the worksheet within the Excel workbook.

Suppose we reuse the following dataset in SAS:

```
/*create dataset*/
data my_data;
input A B C;
datalines;
1 4 76
2 3 49
2 3 85
4 5 88
2 2 90
4 6 78
5 9 80
;
```

run;

/*view dataset*/
proc print data=my_data;

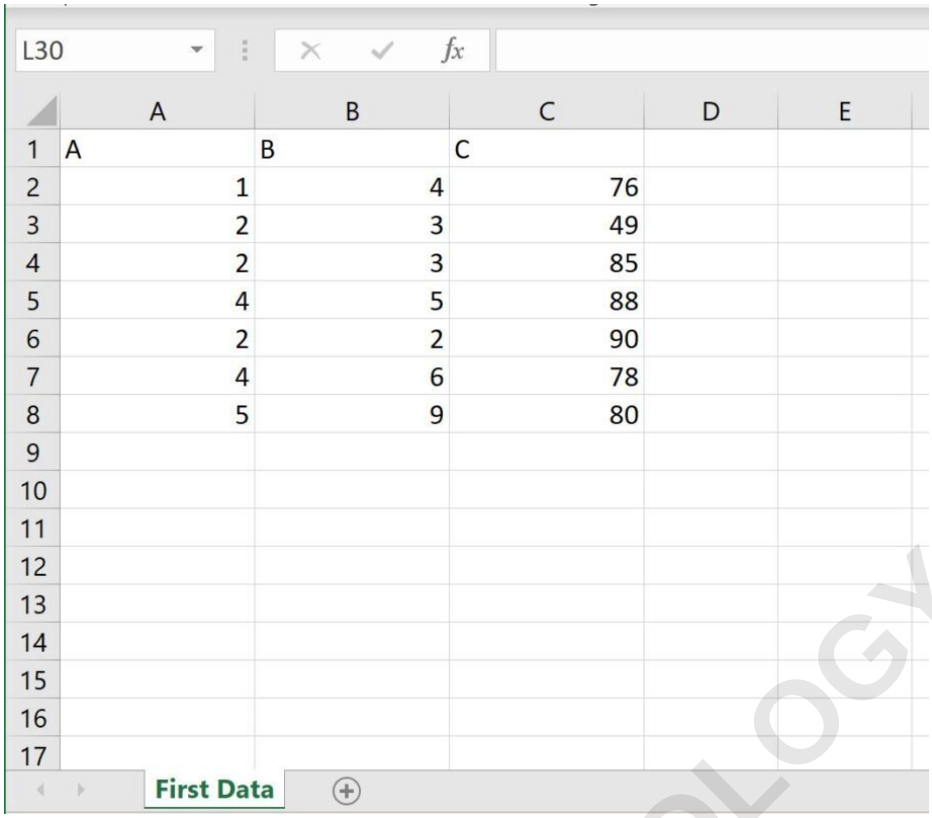
The visual representation of the source data is:

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

We can use the following code to export this dataset to an Excel file called **my_data.xlsx**. Note the inclusion of the **sheet** statement, which is highly useful when exporting to Excel:

/*export dataset*/
proc export data=my_data
outfile="/home/u13181/my_data.xlsx"
dbms=xlsx
replace;
sheet="First Data";
run;

I can then navigate to the location on my computer where I exported the file and view it in Excel. The data in Excel matches the dataset from SAS, and the sheet in the Excel workbook is correctly titled "First Data" just as specified in the proc export statement:



	A	B	C	D	E
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					

Example 3: Exporting a SAS Dataset to a Tab-Delimited Text File

When data contains commas within character fields, exporting as a standard CSV can cause parsing issues for the receiving program. In such cases, using a tab delimiter is a highly reliable alternative for plain text data transfer. The **dbms=tab** option is specifically designed for this purpose, utilizing the ASCII tab character as the field separator.

Suppose we have the following dataset in SAS that contains information about various basketball players:

```
/*create dataset*/
data my_data;
input rating points assists rebounds;
datalines;
90 25 5 11
85 20 7 8
82 14 7 10
88 16 8 6
94 27 5 6
90 20 7 9
```

```
76 12 6 6
75 15 9 10
87 14 9 10
86 19 5 7
;
run;

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

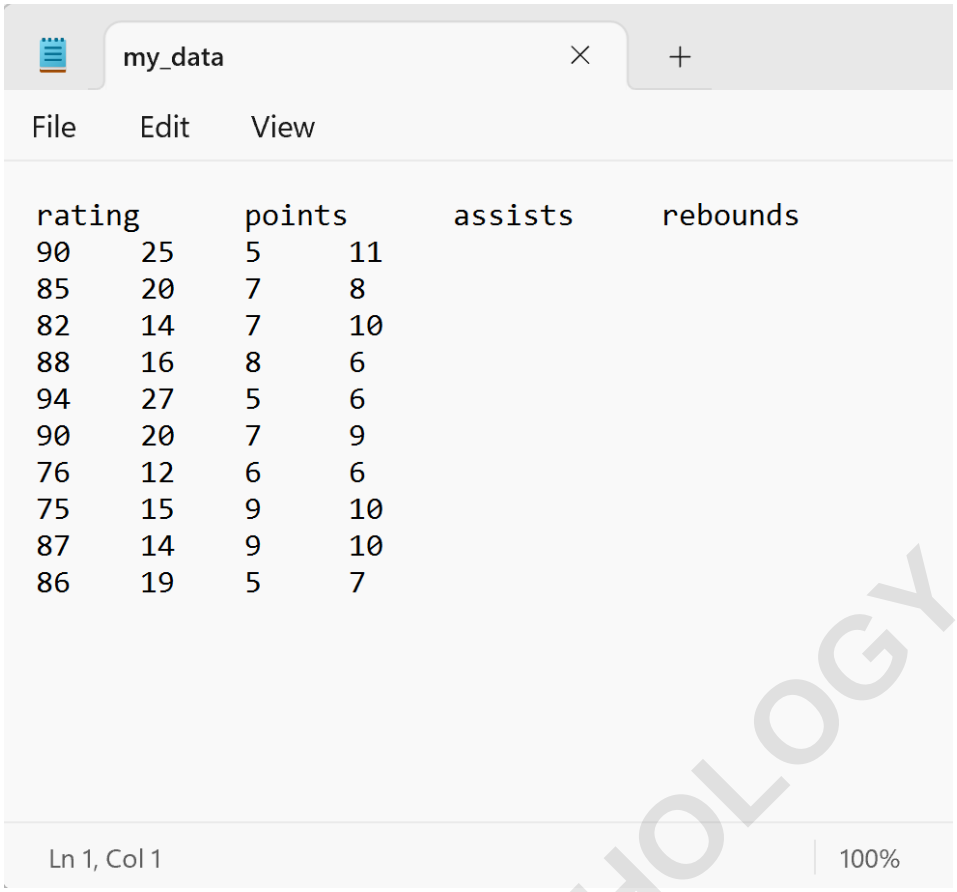
The structured data is confirmed in the SAS output:

Obs	rating	points	assists	rebounds
1	90	25	5	11
2	85	20	7	8
3	82	14	7	10
4	88	16	8	6
5	94	27	5	6
6	90	20	7	9
7	76	12	6	6
8	75	15	9	10
9	87	14	9	10
10	86	19	5	7

We can use the following code to export this dataset to a tab-delimited text file called **my_data.txt** by specifying **dbms=tab** within the PROC EXPORT statement:

```
/*export dataset*/
proc export data=my_data
outfile="/home/u13181/my_data.txt"
dbms=tab
replace;
run;
```

I can then navigate to the location on my computer where I exported the file and view it in a text editor. The output confirms that the data in the text file matches the dataset from SAS, separated cleanly by tabs.



Accessing Advanced Features and Documentation

While the examples provided cover the most frequent use cases for data exporting in SAS, the PROC EXPORT procedure supports numerous additional parameters for highly customized exports. These advanced options allow users to control encoding, specify variable labels, or handle complex delimiters, depending on the requirements of the receiving system. For instance, exporting to databases or systems that require specific character sets might necessitate the use of the `encoding` option, while handling data that requires quotation marks around character fields may involve the `options` statement.

Users who require detailed control over specific aspects of the data export process--such as handling missing values, adjusting date formats, or utilizing alternative delimiters (like semicolons or pipes)--should consult the comprehensive resources provided by SAS. The official documentation serves as the ultimate authoritative source for a complete and up-to-date list of all optional arguments and specific usage nuances applicable to various DBMS identifiers supported by the procedure.

It is always best practice to review the documentation when encountering complex data types or when exporting to less common formats to ensure the output file meets all necessary specifications

for the target system. Relying on validated parameters ensures smooth, reliable data interchange.

Note: Refer to the SAS documentation for a complete list of optional arguments you can use when exporting files and for detailed procedural guidance.

Related Data Management Tutorials in SAS

Mastering data export is one step in comprehensive data management using SAS. The following related tutorials explain how to perform other common tasks necessary for preparing, manipulating, and analyzing data efficiently within the SAS environment:

How to Import External Data into SAS using PROC IMPORT.
Techniques for Merging and Appending SAS Datasets.
Using PROC SQL for Advanced Data Queries and Manipulation.
Methods for Handling Missing Data Values in SAS.