How to Use the TRANSLATE Function in SAS?

Authored by

stats writer

November 19, 2025

The TRANSLATE function in SAS is a core utility designed for the precise manipulation of character data. Its primary function is to replace one or more specific characters within a given string with corresponding replacement characters. This process is governed by two crucial parameters: the FROM parameter, which defines the characters targeted for substitution, and the TO parameter, which defines the characters to be inserted. The TRANSLATE function is highly effective for rigorous data cleansing, especially when standardizing special symbols, punctuation, or inconsistent character entries.

Core Syntax and Positional Parameters

The TRANSLATE function provides a highly efficient means within the SAS Data Step to replace every occurrence of specific characters in a source string with new characters, relying entirely on positional mapping between the characters defined in the TO and FROM arguments.

The function adheres to the following precise syntax structure:

TRANSLATE(source, to, from)

A detailed explanation of the three required positional arguments is provided below:

source: This argument specifies the name of the character variable or string within the dataset that the function will scan and modify.
to: This argument lists the replacement characters. The character at position N in the FROM string is replaced by the character at position N in the TO string.
from: This argument contains the list of characters that the function is instructed to locate and replace within the source variable.

It is essential to remember that TRANSLATE operates on a case-sensitive basis. If you intend to replace both uppercase and lowercase versions of a character, both must be explicitly included in the FROM argument, with corresponding replacements defined in the TO argument.

Setting Up the Demonstration Dataset

To effectively illustrate the application of character manipulation using the TRANSLATE function, we will establish a simple sample dataset. This dataset, named `my_data`, models basketball player statistics and contains a character variable, `position`, which we will target for modification in the subsequent examples.

```
/*create dataset: Defining the structure and population of the input data for demonstration*/
data my_data;
input team $ position $ points assists;
```

```
datalines;
A Guard 14 4
A Guard 22 6
A Guard 24 9
A Forward 13 8
A Forward 13 9
A Guard 10 5
B Guard 24 4
B Guard 22 6
B Forward 34 2
B Forward 15 5
B Forward 23 5
B Guard 10 4
;
run;
```

/*view dataset: Displaying the initial data structure before any character manipulation takes place*/
proc print data=my_data;

The initial output shows the raw character strings in the `position` variable, which will be the subject of our transformation exercises.

Obs	team	position	points	assists
1	A	Guard	14	4
2	A	Guard	22	6
3	A	Guard	24	9
4	A	Forward	13	8
5	A	Forward	13	9
6	A	Guard	10	5
7	B	Guard	24	4
8	B	Guard	22	6
9	B	Forward	34	2
10	B	Forward	15	5
11	B	Forward	23	5
12	B	Guard	10	4

Example 1: Direct Character Substitution

This example demonstrates the fundamental usage of the **TRANSLATE** function by performing a straightforward substitution: replacing every instance of the lowercase letter "r" within the **position** column with the letter "z". This highlights the character-level, global nature of the function, where all matching characters are systematically changed.

We are creating a new dataset, `new_data_1`, where the transformation is applied. The statement `position = translate(position, "z", "r");` instructs SAS to look for the single character in the FROM string ("r") and replace it with the single character in the TO string ("z").

```
/*create new dataset: Applying the TRANSLATE operation*/
data new_data_1;
set my_data;
position = translate(position, "z", "r");
run;

/*view new dataset*/
proc print data=new_data_1;
```

Reviewing the Results of Direct Substitution

The resulting data confirms the successful substitution. The original values like "Guard" and "Forward" have been transformed into "Guazd" and "Fowazd" respectively, demonstrating that the function targeted and replaced all instances of 'r' within the character variable.

Obs	team	position	points	assists
1	A	Guazd	14	4
2	A	Guazd	22	6
3	A	Guazd	24	9
4	A	Fozwazd	13	8
5	A	Fozwazd	13	9
6	A	Guazd	10	5
7	B	Guazd	24	4
8	B	Guazd	22	6
9	B	Fozwazd	34	2
10	B	Fozwazd	15	5
11	B	Fozwazd	23	5
12	B	Guazd	10	4

Observe that every occurrence of the letter "r" in the **position** column has been precisely replaced with the letter "z". This immediate and comprehensive replacement capability makes the TRANSLATE function highly effective for batch updates and data standardization.

Example 2: Using TRANSLATE for Character Deletion

The TRANSLATE function can also be utilized for character deletion, which is achieved by specifying a TO string that is shorter than the FROM string, or in this case, an empty string (""). This setup instructs SAS to replace the targeted characters (in this case, "r") with blank spaces.

The following code demonstrates how to use **TRANSLATE** to replace each occurrence of "r" in the **position** column with a blank, followed by the necessary step to remove the resulting whitespace:

```
/*create new dataset: Applying TRANSLATE for removal, followed by COMPRESS*/
data new_data_2;
set my_data;
position = compress(translate(position, "", "r"));
run;

/*view new dataset*/
proc print data=new_data_2;
```

Integrating TRANSLATE with COMPRESS for Whitespace Management

When `TRANSLATE` replaces characters using an empty TO string, it inserts a single blank space for each character removed. To achieve true character deletion--where the string is compacted without residual spaces--it is best practice to nest the `TRANSLATE` function within the COMPRESS function.

The output image illustrates the result of this combined operation.

Obs	team	position	points	assists
1	A	Guad	14	4
2	A	Guad	22	6
3	A	Guad	24	9
4	A	Fowad	13	8
5	A	Fowad	13	9
6	A	Guad	10	5
7	B	Guad	24	4
8	B	Guad	22	6
9	B	Fowad	34	2
10	B	Fowad	15	5
11	B	Fowad	23	5
12	B	Guad	10	4

The combined use of `TRANSLATE` and COMPRESS function effectively achieves the goal of removing the string "r" from each position name, resulting in compacted strings like "Guad" and "Fowad".

Note #1: We utilized the COMPRESS function wrapped around the **TRANSLATE** function to remove all trailing and internal blanks that were generated by the substitution of 'r' with a space in the **position** column.

Official Documentation and Resources

Note #2: For detailed specifications regarding parameter limitations, character encoding, and advanced usage scenarios of the TRANSLATE function, please consult the complete documentation provided by SAS.

The following resources offer further guidance on performing other common data manipulation and

character handling tasks in SAS: