How to use the TRANWRD Function in SAS (With Examples)

Authored by

stats writer

November 19, 2025

The **SAS** System provides the **TRANWRD function**, an indispensable utility for precise string manipulation. This function is designed to systematically replace every occurrence of a specified word or character sequence (the target) with a new word or character sequence (the replacement) within a given **text string**. Mastering **TRANWRD** is crucial for any programmer dealing with data cleaning, standardization, or text processing within large-scale analytical projects.

The fundamental syntax for the **TRANWRD function** is `TRANWRD(source, target, replacement)`. Here, `source` identifies the variable containing the string to be modified; `target` is the exact sequence of characters to be replaced; and `replacement` is the new sequence inserted in its place. This functional clarity makes **TRANWRD** highly efficient for global substitutions, ensuring consistency across a **data set**, whether you are replacing words, managing character patterns, or removing multiple substrings simultaneously.

The primary purpose of the **TRANWRD function** in **SAS** is to execute a search-and-replace operation across all instances of a specific **character pattern** found within a source string. This allows for powerful and flexible transformation of text variables, providing a key capability for maintaining data quality and consistency in statistical environments.

The function operates strictly based on the following three-part syntax, which must be adhered to precisely during implementation:

TRANWRD(source, target, replacement)

where each parameter fulfills a critical role in the operation:

source: Specifies the variable or **text string** that the function will search within. This is the original data that is subject to modification.
target: Defines the exact **character pattern** or substring that the function must locate for replacement. This search is case-sensitive by default.
replacement: Dictates the sequence of characters that will be substituted for every occurrence of the *target* string found in the source. Utilizing a null string (`""`) here results in the removal of the target text.

The following practical examples demonstrate how to apply this function in a **SAS** environment. We begin by setting up the demonstration **data set** that will be used across all transformations:

Setting Up the Demonstration Data

To effectively illustrate the capabilities of the **TRANWRD function**, we first require a representative sample **data set**. The data below contains team names with intentional prefixes that will serve as our targets for replacement operations. This setup allows us to clearly observe the function's

behavior on multiple observations simultaneously.

The code block utilizes the `DATA` and `DATALINES` statements to create the initial dataset named `original_data`. We define the `team` variable as a character type with a length of 20, accommodating the various team names. This base dataset is critical as it provides the foundation for our subsequent manipulation steps.

Once the data is created, we use `PROC PRINT` to output the contents, ensuring that the input data is correctly structured and available for the transformation examples that follow. Note the repetition of prefixes such as "Fast" and "Wild," which are prime candidates for modification using **TRANWRD**.

```
/*create dataset*/
data original_data;
input team $1-20;
datalines;
Fast Bees
Angry Hornets
Wild Mustangs
Fast Panthers
Fast Cobras
Wild Cheetahs
Wild Aardvarks
;
run;

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

Obs	team
1	Fast Bees
2	Angry Hornets
3	Wild Mustangs
4	Fast Panthers
5	Fast Cobras
6	Wild Cheetahs
7	Wild Aardvarks

Example 1: Using TRANWRD to Replace Characters with New Characters

This example showcases the most common use of **TRANWRD**: substituting a targeted substring with a completely new string. Our objective is to standardize the team names by changing all instances of the prefix "Fast" to "Slow." This is achieved within a new **DATA step** where the `SET` statement reads the original data, and the **TRANWRD** function performs the transformation.

The core transformation statement is `team = tranwrd(team, "Fast", "Slow");`. By assigning the result back to the `team` variable, we overwrite the original value with the transformed **text string**. Because **TRANWRD** executes a global search, all three occurrences of "Fast" across the dataset--in "Fast Bees," "Fast Panthers," and "Fast Cobras"--are simultaneously updated to "Slow."

The procedural output confirms the successful execution of this replacement. Observations not containing the target **character pattern** remain untouched, demonstrating the function's precise conditional behavior. This highlights how **TRANWRD** provides a clean, concise method for executing mass text updates based on specific string matches.

```
/*create new dataset*/
data new_data;
set original_data;
team = tranwrd(team, "Fast", "Slow");
run;

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

Obs	team
1	Slow Bees
2	Angry Hornets
3	Wild Mustangs
4	Slow Panthers
5	Slow Cobras
6	Wild Cheetahs
7	Wild Aardvarks

Notice that each occurrence of "Fast" in the team names have been replaced with "Slow" instead, validating the global replacement functionality.

Example 2: Using TRANWRD to Replace Characters with Blanks (Substring Removal)

The second critical application of the **TRANWRD function** involves using it not for substitution, but for deletion. By setting the replacement argument to a null string (`""`), we instruct **TRANWRD** to remove the target string entirely wherever it appears in the source variable. This technique is highly effective for cleaning up prefixes, suffixes, or internal identifiers that need to be stripped from the data.

In this specific demonstration, we aim to remove the prefix "Fast" from the relevant team names. The code implements this via: `team = tranwrd(team, "Fast", "");`. By replacing the sequence "Fast" with nothing, the function seamlessly concatenates the remaining parts of the string. It is important to remember that if the target string includes trailing spaces (e.g., `"Fast "`), those spaces will also be removed, often leading to cleaner data outputs.

The resulting **data set** shows that the team names previously starting with "Fast" now begin directly with the second word, potentially preceded by any residual space not captured in the target string. This showcases the efficiency of **TRANWRD** for data reduction and normalization, particularly in preparing textual fields for analysis where extraneous descriptors must be eliminated.

```
/*create new dataset*/
data new_data;
set original_data;
team = tranwrd(team, "Fast", "");
run;

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

Obs	team
1	Bees
2	Angry Hornets
3	Wild Mustangs
4	Panthers
5	Cobras
6	Wild Cheetahs
7	Wild Aardvarks

This transformation achieves the goal of simply removing the string "Fast" from each team name that contained it.

Advanced Text Processing Considerations

While **TRANWRD** is a highly accessible function, its reliance on exact, literal matching means that complex text manipulation often requires combined functions or alternative tools. A key limitation is its inherent case sensitivity. If the data contained "fast," "Fast," and "FAST," **TRANWRD** would only match the one specified in the target argument. To overcome this, the source string must first be standardized using functions like `UPCASE` or `LOWCASE`, or one must opt for the specialized function `TRANWRDI`, which performs case-insensitive replacement.

For scenarios demanding pattern matching beyond fixed strings--such as replacing all numbers followed by a specific letter, or handling variable whitespace--the **TRANWRD** function is insufficient. In these situations, **SAS** programmers should leverage the `PRXCHANGE` function, which integrates the power of Perl Regular Expressions (PRX). This allows for dynamic, complex pattern recognition and substitution that **TRANWRD** cannot accommodate.

Furthermore, when dealing strictly with single-character transformations--for instance, changing punctuation marks or swapping vowels--the `TRANSLATE` function is generally preferred. `TRANSLATE` is optimized for character-by-character substitution, often proving more efficient and cleaner than using **TRANWRD** for tasks that do not involve replacing multicharacter substrings. Selecting the right function depends entirely on whether the task involves word-level replacement (**TRANWRD**) or character-level replacement (`TRANSLATE`).

Conclusion on String Manipulation in SAS

The **TRANWRD function** serves as a foundational element in **SAS** programming for high-volume,

global **text string** replacement. Its straightforward structure ensures that developers can quickly and accurately modify textual data in large **data sets**, contributing significantly to the data preparation phase of any analysis. The versatility to either substitute a target string with new characters or simply delete it using a null replacement makes it invaluable.

By adhering to the syntax `TRANWRD(source, target, replacement)` and understanding the case-sensitive nature of the function, users can reliably achieve consistent data standardization. While more advanced functions exist for complex pattern matching, **TRANWRD** remains the ideal choice for literal string substitution due to its simplicity and processing speed. For complete documentation and further details on related string functions, always refer to the official resources.

The following tutorials explain how to perform other common tasks in SAS: