How to Easily Replace Characters in Strings Using SAS

Authored by

stats writer

December 1, 2025

The ability to manipulate and clean textual data is fundamental in statistical programming, and the SAS system provides powerful tools for this purpose. When dealing with character variables, analysts frequently need to substitute specific patterns or words within a string. For this critical task, SAS offers the highly efficient **TRANWRD** function, designed specifically for whole-word or substring replacement.

Unlike simpler character-level translation functions, **TRANWRD** (short for Translate Word) searches for a specific substring and replaces all non-overlapping occurrences of that substring with a new specified replacement string. This function is essential for standardizing data fields, correcting common errors, or dynamically updating text based on business rules within a DATA step environment. Understanding its syntax and application is crucial for advanced data cleaning in SAS.

The core syntax utilizes the **TRANWRD()** function, which requires three primary arguments: the source string, the target substring to be replaced, and the replacement string. Mastering these two common methods allows for complete control over character replacement within your datasets.

The two most common applications of this function involve either substituting the target substring with a completely new set of characters or effectively deleting the target substring by replacing it with a blank value. We will explore both methods in detail using practical examples.

Understanding the TRANWRD Function Syntax

The TRANWRD function follows a clear structure designed for efficiency in the SAS environment. It is important to note that this function is **case-sensitive** by default, meaning that "wild" will not match "Wild." If case-insensitive replacement is needed, analysts should consider preprocessing the variable using functions like `UPCASE` or `LOWCASE`, or utilize the more advanced regular expression capabilities found in PRX functions.

The general format is `TRANWRD(source, target, replacement)`, where the arguments serve distinct, critical roles in the replacement operation:

source: This is the character expression or variable that contains the string where the replacement operation will occur. This is typically an existing variable within your dataset.

target: This is the specific substring or word you wish to find and replace within the source string. It must match the text exactly, including capitalization and spacing, to be recognized.

replacement: This is the new string value that will substitute the target substring whenever it is found. This argument dictates the output of the function.

We will now explore the two primary methods for applying the TRANWRD function, illustrating how to either introduce new text or remove unwanted characters effectively in a DATA step context.

Method 1: Replacing a Substring with a New Value

This method is used when you need to substitute an existing piece of text with a different, meaningful word or phrase. This is vital for tasks like renaming categories, correcting misspelled entries, or updating outdated information within descriptive fields in your dataset. This process preserves the structure of the data while updating the specific content found in the variable.

The syntax below demonstrates how to define a new variable (`new_variable`) by applying TRANWRD to an existing variable (`old_variable`). The function identifies the exact text "OldString" and systematically replaces every occurrence with the provided "NewString." Notice that all operations are enclosed within a standard DATA step, ensuring that the transformation is repeatable and integrated into your data processing pipeline.

```
data new_data;
set original_data;
new_variable = tranwrd(old_variable, "OldString", "NewString");
run;
```

Method 2: Removing a Substring by Replacing with Blanks

The second common application involves using the **TRANWRD** function to effectively delete or remove a substring from a character variable. This is accomplished by providing an empty string (`""`) as the replacement argument. This technique is indispensable for removing unwanted prefixes, suffixes, extra spaces, or specific keywords that might interfere with downstream analysis or reporting, yielding a cleaner resultant variable.

It is critical when removing text to ensure the replacement argument is genuinely an empty string (two quotation marks with nothing in between, `""`), rather than a single space (`" "`). Replacing text with a single space will leave a residual space character, which can sometimes cause unintended formatting issues or complicate subsequent text manipulation tasks. Careful implementation of this method ensures truly clean data where the target substring vanishes entirely.

```
data new_data;
set original_data;
new_variable = tranwrd(old_variable, "OldString", "");
run;
```

Setting Up the Demonstration Dataset in SAS

To fully illustrate the functionality of the TRANWRD function, we will utilize a small, sample dataset. This dataset contains a single character variable, `team`, which holds descriptive names containing prefixes that we intend to modify or remove. This scenario mirrors real-world data cleaning needs where standardization is required across categorical text fields, such as survey responses or descriptive labels.

The following code uses the DATA step along with the `DATALINES` statement to input the sample data directly into the SAS environment, creating the temporary dataset named `original_data`. We then use `PROC PRINT` to verify the structure and content of our starting data before applying any transformations, ensuring our base data is correctly loaded.

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

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

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

Practical Application 1: Substituting Text with a New String

In this first practical example, we implement Method 1 to standardize the prefixes in our team names. Specifically, we want to replace the descriptive word "Wild" with the new word "Fast." This scenario is typical when organizational branding or categorization shifts require updating text fields across thousands of records within a SAS table, demanding bulk textual replacement.

The code below creates a new dataset called `new_data`. Inside the DATA step, we define the new variable `new_team` using the TRANWRD function. We specify `team` as the source variable, "Wild" as the target to be replaced, and "Fast" as the replacement string. Only those observations containing the exact string "Wild" will be affected by this operation, ensuring precision.

```
/*replace "Wild" with "Fast" in team variable*/
data new_data;
set original_data;
new_team = tranwrd(team, "Wild", "Fast");
run;

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

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

Upon reviewing the output, it is clear that only the records containing "Wild" (Mustangs, Cheetahs, Aardvarks) were updated, now appearing as "Fast Mustangs," "Fast Cheetahs," and "Fast Aardvarks." The remaining records that did not contain the exact target substring, such as "Angry Bees" and "Kind Panthers," retained their original values without modification, demonstrating the targeted precision of the **TRANWRD** function when implementing text substitutions.

Practical Application 2: Removing Substrings by Using Blanks

Our second example focuses on data cleansing, specifically the removal of the word "Wild" entirely from the `team` variable. This application uses Method 2, replacing the target substring with an empty string (`""`). This is commonly required when preparing data for merging or analysis where descriptive prefixes or suffixes are deemed unnecessary noise, and the base name of the entity is the only required component.

The implementation is nearly identical to the previous example, but the third argument--the replacement value--is set to the null string `""`. This tells SAS not to substitute the found text with any characters at all, effectively deleting it from the character variable. This is the most efficient way to remove substrings without resorting to complex logic.

```
/*replace "Wild" with a blank in team variable*/
data new_data;
set original_data;
new_team = tranwrd(team, "Wild", "");
run;

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

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

Reviewing the results confirms that "Wild Mustangs" is now displayed as " Mustangs" (note the residual leading space left by the removal of "Wild"), and similarly for the other affected records. The remaining team names, such as "Angry Bees," remain unchanged. If the resulting leading space is undesirable, further manipulation using functions like `TRIM()` or `LEFT()` might be necessary to normalize the string formatting after the replacement operation is complete.

Important Considerations for TRANWRD Usage

While the TRANWRD function is straightforward, users must be aware of its operational characteristics, particularly concerning case sensitivity and overlapping substrings. As previously mentioned, **TRANWRD** performs an exact, case-sensitive match. If your input data contains variations in capitalization (e.g., "WILD," "Wild," and "wild"), you must either apply the function multiple times for each variation, or preferably, standardize the case of the source variable first using functions like `UPCASE()`.

Furthermore, **TRANWRD** handles multiple occurrences by replacing all non-overlapping instances of the target substring. However, it does not support regular expressions for complex pattern matching. If complex, conditional replacements or replacements based on fuzzy patterns (like dates, specific formats, or partial matches) are needed, analysts should consider using the powerful PRX functions, such as `PRXCHANGE`, which offer much greater flexibility in textual manipulation within the DATA step environment.

Finally, remember that the length of the new string resulting from the replacement may change the overall length of the variable. If the replacement string is longer than the target string, ensure that the target variable has sufficient length defined in the DATA step, or SAS may truncate the output, potentially losing critical data, which is a common pitfall for new users unaware of variable length management.

Conclusion and Further Reading

The **TRANWRD()** function is an indispensable utility for any SAS programmer needing to perform exact substring replacements. Its straightforward syntax and efficient operation make it the go-to choice for standardizing names, correcting errors, and performing targeted data cleansing tasks within large datasets. By mastering the distinction between replacing with a new value and replacing with a blank, users gain robust control over their character variables and improve data quality.

For those looking to expand their knowledge beyond simple substring replacement, SAS offers a spectrum of related functions that handle different types of textual manipulation, including character translation (`TRANSLATE`), phonetic matching, and advanced pattern recognition. These tools are often employed in sequence with **TRANWRD** to achieve comprehensive data quality goals, especially when preparing unstructured text for rigorous statistical analysis.

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