What are the options of using PROC STDIZE in SAS?

Authored by

stats writer

November 19, 2025

Welcome to this comprehensive tutorial dedicated to mastering the **PROC STDIZE** procedure in SAS. Data standardization is a foundational step in robust data analysis and machine learning preparation, ensuring fairness across features measured on different scales. This guide offers a deep dive into the purpose and application of this vital tool, including step-by-step examples and critical verification techniques.

By the conclusion of this expert walkthrough, you will possess a clear understanding of when and how to apply **PROC STDIZE**, allowing you to prepare your datasets with confidence for advanced statistical modeling. We will explore both the default settings and advanced options for highly controlled data transformation.

Why Standardization is Essential in Data Preparation

In advanced statistical modeling, particularly algorithms that rely on distance metrics (such as K-means or PCA), input variables must be treated equally regardless of their natural scale or units of measurement. Failing to standardize data means that variables with inherently larger magnitudes will disproportionately influence the model's outcome, leading to biased results. Standardization corrects this by transforming all variables onto a common scale.

The most common form of standardization is the Z-score transformation, which effectively centers the data distribution around zero and scales its variance to one. This transformation is crucial for normalizing distributions, making coefficients in certain models more comparable, and generally improving the convergence speed of optimization algorithms.

To **standardize** a variable means to scale each of the values for the variable such that the mean value is 0 and the standard deviation is 1. This process ensures that every variable contributes equally to the overall statistical model, regardless of its original units.

The Mathematical Foundation of Z-Score Standardization

While **PROC STDIZE** handles the computation automatically, understanding the underlying mathematical transformation is vital for correct interpretation. The Z-score standardization transforms an observation by measuring how many standard deviations it is above or below the mean. This process preserves the shape of the data's distribution but relocates it within a normalized space.

You can use the following formula to standardize a variable, where the numerator calculates the deviation from the mean and the denominator scales this deviation by the variability:

(xi - x) / s

where:

xi: The ith value in the dataset, representing the individual raw observation.
x: The calculated sample mean of the variable.
s: The calculated sample standard deviation of the variable.

The easiest way to perform this transformation on large datasets in SAS is to use the **PROC STDIZE** statement, which is highly optimized for efficiency and accuracy.

Setting Up the Demonstration Dataset in SAS

To illustrate the practical application of PROC STDIZE, we will begin by creating a simple dataset named `my_data`, which tracks key performance indicators for basketball players. This dataset contains both character variables (`player`) and numeric variables (`points`, `assists`, `rebounds`) measured on different scales.

The DATA step ensures that we have a clean, reproducible starting point. We utilize the **INPUT** statement to define the variable names and types, followed by **DATALINES** to supply the raw observation data. Notice the significant variation in the raw scores, which mandates the need for standardization before comparative analysis.

The following code block demonstrates the creation and initial inspection of our data using **PROC PRINT**, allowing us to see the original, unscaled values. This snapshot confirms the initial range and scale of each numeric variable before transformation.

```
/*create first dataset*/
data my_data;
input player $ points assists rebounds;
datalines;
A 18 3 15
B 20 3 14
C 19 4 14
D 14 5 10
E 14 4 8
F 15 7 14
G 20 8 13
H 28 7 9
I 30 6 5
J 0 31 9 4
;
run;
```

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

Obs	player	points	assists	rebounds
1	A	18	3	15
2	B	20	3	14
3	C	19	4	14
4	D	14	5	10
5	E	14	4	8
6	F	15	7	14
7	G	20	8	13
8	H	28	7	9
9	I	30	6	5
10	J	0	31	9

Performing General Standardization Across All Numeric Variables

When the goal is to standardize all continuous variables in the dataset, the default operation of PROC STDIZE is the simplest approach. By omitting the **VAR** statement, the procedure automatically identifies all numeric columns and applies the Z-score transformation. This is the most common use case for preparing data for models like ridge or lasso regression.

We must specify the input dataset using the `DATA=` option and, critically, use the `OUT=` option to create a new dataset (here, `std_data`) to store the standardized results. This practice ensures data integrity by preventing the original source data from being overwritten.

The following example executes the procedure and then prints the resulting dataset. Notice how the values for `points`, `assists`, and `rebounds` are transformed into standardized scores, reflecting their position relative to the mean of their respective original distributions.

```
/*standardize all numeric variables in dataset*/
proc stdize data=my_data out=std_data;
run;

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

Obs	player	points	assists	rebounds
1	A	0.02414	-0.57572	1.16477
2	B	0.26558	-0.57572	0.86611
3	C	0.14486	-0.45578	0.86611
4	D	-0.45872	-0.33584	-0.32852
5	E	-0.45872	-0.45578	-0.92584
6	F	-0.33801	-0.09595	0.86611
7	G	0.26558	0.02399	0.56745
8	H	1.23131	-0.09595	-0.62718
9	I	1.47274	-0.21590	-1.82182
10	J	-2.14876	2.78266	-0.62718

Each of the numeric variables (points, assists, rebounds) have been standardized to have a theoretical mean of 0 and standard deviation of 1. Positive Z-scores indicate observations above average, while negative Z-scores indicate observations below average.

Targeted Standardization Using the VAR Statement

There are situations where only a specific subset of variables requires standardization. Using the **VAR** statement provides the necessary control to specify exactly which columns should undergo the transformation, leaving all unlisted variables (numeric or character) untouched in the output dataset. This precision is vital when some numeric variables represent naturally scaled indices or counts that should retain their original interpretation.

By incorporating the **VAR** statement immediately following the **PROC STDIZE** command, we instruct the procedure to limit its scope. All specified variables are standardized, and the procedure automatically carries forward all other columns into the output dataset.

For example, we can use the following PROC STDIZE statement with the **VAR** statement to only standardize the points variable, preserving the raw values of assists and rebounds for a different type of subsequent data analysis.

```
/*standardize points variable in dataset*/
proc stdize data=my_data out=std_data;
var points;
run;
```

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

Obs	player	points	assists	rebounds
1	A	0.02414	3	15
2	B	0.26558	3	14
3	C	0.14486	4	14
4	D	-0.45872	5	10
5	E	-0.45872	4	8
6	F	-0.33801	7	14
7	G	0.26558	8	13
8	H	1.23131	7	9
9	I	1.47274	6	5
10	J	-2.14876	31	9

The values in the points column have been standardized while all other columns have remained untouched, confirming that the **VAR** statement effectively isolated the transformation to the specified variable only.

Verification of Standardization Using PROC MEANS

Verification is a mandatory step to confirm the integrity of data transformations. After running **PROC STDIZE**, we must formally prove that the standardized variable meets the Z-score criteria: a mean of 0 and a standard deviation of 1. This confirmation is easily achieved using the **PROC MEANS** procedure.

By running **PROC MEANS** on the output dataset (`std_data`), we request descriptive statistics for all numeric variables. This allows us to compare the statistics of the standardized column (`points`) against the unstandardized columns (`assists` and `rebounds`) in the new dataset.

We use the **PROC MEANS** statement without additional options to view the default output, which includes the N (count), Mean, and Standard Deviation. The output should definitively show the expected statistical properties for the transformed data.

```
/*view mean and standard deviation of each variable*/
proc means data=std_data;
```

The MEANS Procedure

Variable	N	Mean	Std Dev	Minimum	Maximum
points	10	-4.44089E-17	1.0000000	-2.1487585	1.4727446
assists	10	7.8000000	8.3373324	3.0000000	31.0000000
rebounds	10	11.1000000	3.3482997	5.0000000	15.0000000

We can see that the points variable indeed has a mean value extremely close to 0 (often reported as 0 due to rounding or minute floating-point discrepancies) and a standard deviation of 1. This result validates the successful execution of the standardization process on the selected variable.

Exploring Alternative Standardization Methods

The flexibility of PROC STDIZE extends beyond the default Z-score method. The **METHOD=** option allows analysts to select alternative scaling techniques suitable for different data conditions or modeling requirements. The default method is `METHOD=STD` (using mean and standard deviation), but others are equally important.

One powerful alternative is `METHOD=RANGE`, which performs Min-Max scaling. This technique rescales the data so that all values fall strictly between 0 and 1. Range scaling is calculated by dividing the centered value by the range (Max - Min) and is often required for specific machine learning algorithms, such as those involving probability estimates or constrained input layers.

For datasets containing significant outliers, using the mean and standard deviation can skew the standardization constants. In such scenarios, robust scaling is preferred. PROC STDIZE offers robust methods like `METHOD=MEDIAN`, which centers the data around the median and scales it using the interquartile range (IQR). This provides a transformation less sensitive to extreme values.

Conclusion and Further Exploration

The **PROC STDIZE** procedure is indispensable in the SAS toolkit for any analyst performing preparatory data manipulation. Its ability to quickly and accurately perform Z-score transformations, alongside options for robust scaling and range normalization, makes it flexible enough to handle diverse datasets across various scientific and business applications. By standardizing variables, we ensure that our statistical inferences and model predictions are reliable and unbiased by differences in measurement units.

We have successfully demonstrated the core functionality, including standardizing all numeric variables and selectively standardizing a subset using the **VAR** statement, followed by critical verification using **PROC MEANS**. Mastery of these techniques is essential for transitioning from

basic data handling to advanced statistical modeling and complex data analysis projects.

We encourage readers to explore additional documentation on **PROC STDIZE** to learn about advanced options such as centering without scaling (using `METHOD=CENTER`) or saving the standardization parameters for application to future test datasets (using the **OUTSTAT=** option).

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