SAS: Using PROC MEANS with CLASS Statement

Authored by

stats writer

November 19, 2025

Introduction to Data Aggregation and PROC MEANS in SAS

In the realm of statistical analysis, calculating descriptive or summary statistics is often the essential first step toward understanding the underlying distribution of data. The SAS software package provides a robust suite of tools for this purpose, with the **PROC MEANS** procedure standing out as the primary utility for generating these metrics. This procedure is designed to efficiently calculate metrics such as mean, standard deviation, minimum, and maximum for every numeric variable present within a specified dataset. When used in its most basic form, **PROC MEANS** provides statistics for the entire dataset as a single unit of observation.

However, real-world data analysis frequently requires deeper insights, demanding that these summary calculations be applied not globally, but to specific subsets of the data. This process, known as data aggregation, is crucial for comparing different groups within the same dataset. This is where the **CLASS** statement becomes indispensable. By incorporating the **CLASS** statement within PROC MEANS documentation, analysts can instruct SAS to partition the data based on one or more categorical variables, generating separate, meaningful summary statistics for each unique combination of these grouping variables.

Mastering the proper use of the **CLASS** statement is foundational for anyone performing statistical reporting or exploratory data analysis in **SAS**. This guide will demonstrate, through practical examples, how to effectively leverage the **CLASS** statement to transform simple descriptive output into powerful, segmented reports. We will begin by establishing a sample dataset that captures relevant metrics about basketball players, providing a clear context for our statistical demonstrations.

Setting Up the Analysis Environment: The Basketball Dataset

To illustrate the utility of the **CLASS** statement, we will utilize a small, manageable dataset containing key performance indicators (KPIs) for basketball players. This dataset includes both numeric variables (quantifiable data like points and assists) and categorical variables (descriptive characteristics like team and position), which are perfect candidates for use in the grouping process. Understanding the structure of this input data is vital before proceeding to the statistical calculations.

The dataset, named `my_data`, is created using standard **SAS** data steps. It includes four primary variables: `team` (categorical identifier, A or B), `position` (categorical role, Guard or Forward), `points` (numeric score), and `assists` (numeric count). The structure ensures we have sufficient variation across the categorical fields to demonstrate effective segmentation using the **CLASS** statement in subsequent examples.

The following code block shows the creation of this dataset and the subsequent use of `PROC`

`PRINT` to display the raw data, allowing for visual inspection of the values that will be summarized:

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

After executing the above data step and print procedure, the resulting output table confirms the successful creation and structure of the dataset. This table serves as the foundation for all subsequent analyses where we calculate the mean, count, and deviation of the `points` and `assists` variables.

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

Baseline Summary: Calculating Global Statistics Without Grouping

Before we delve into segmented analysis, it is essential to understand the default behavior of **PROC MEANS** when executed without any grouping instructions. When the procedure is called only with the `DATA=` option, it treats all observations within the dataset as a single unit, producing overall summary statistics for every numeric column. This gives us a baseline understanding of the central tendency and dispersion of the entire population of basketball players.

This global summary provides crucial context. For instance, knowing the overall average points scored across all teams and positions helps determine if subsequent group averages are significantly higher or lower than the grand mean. This initial step confirms that the numeric variables (`points` and `assists`) are correctly identified and processed by **SAS**.

The following simple code block demonstrates this fundamental application of **PROC MEANS**:

```
/*calculate summary statistics for numeric variables for the entire dataset*/
proc means data=my_data;
run;
```

Upon execution, the output reflects the descriptive statistics generated for all 12 observations combined. Note that **PROC MEANS** defaults to calculating the N (count), Mean, Standard Deviation (Std Dev), Minimum, and Maximum unless otherwise specified using the `VAR` and `TYPES`

statements.

The MEANS Procedure

Variable	N	Mean	Std Dev	Minimum	Maximum
points	12	18.6666667	7.2780284	10.0000000	34.0000000
assists	12	5.5833333	2.1514618	2.0000000	9.0000000

As clearly shown in the resulting table, by default, **PROC MEANS** successfully computes these aggregate metrics for each numeric variable (`points` and `assists`) across the entirety of the `my_data` dataset, treating it as a single homogeneous group.

The Power of Segmentation: Introducing the CLASS Statement

While the global statistics derived from the previous example are useful, analysts often require a more granular view of the data. To compare performance between different groups--such as comparing Team A players to Team B players--we must employ segmentation. This is precisely the function of the **CLASS** statement within **PROC MEANS documentation**. The **CLASS** statement allows the user to nominate one or more categorical variables that SAS should use to define subgroups.

The procedure works by first sorting the dataset internally (or relying on a pre-sorted state, depending on the data structure and specific **SAS** version) according to the variables listed in the **CLASS** statement. It then calculates the requested **summary statistics** independently for each unique combination of values found within those categorical variables. This powerful feature facilitates comparative analysis and helps identify variations or anomalies between specified groups, moving beyond simple total averages to provide contextualized metrics.

The ability to segment data based on categories like geography, demographic information, or, in our case, team and position, is crucial for actionable data insights. Without the **CLASS** statement, achieving this level of detail would require multiple separate data steps or tedious manual filtering, underscoring its efficiency and importance in the **SAS** programming environment.

Detailed Grouping: Analyzing Statistics by a Single Categorical Variable

Our first application of the **CLASS** statement involves grouping the summary statistics solely by the `team` variable. This will generate two distinct sets of results: one for Team A and one for Team B. This segmentation allows us to directly compare the offensive output (points and assists) of players on Team A versus those on Team B, revealing potential performance differences that were

obscured in the overall global summary.

The process is remarkably straightforward. We simply invoke **PROC MEANS** and then include the **CLASS** statement followed by the variable `team`. **SAS** handles the rest, automatically recognizing the two unique values (A and B) and executing the statistical calculations for each subgroup. This segmentation is fundamental for producing team-specific performance reports.

Observe the implementation of this technique in the following **SAS** code:

```
/*calculate summary statistics for numeric variables, grouped by team*/
proc means data=my_data;
class team;
run;
```

The resulting output is dramatically different from the baseline example. Instead of one set of results, we receive multiple rows categorized by the `team` variable. Each row provides the count (N), Mean, Standard Deviation, Minimum, and Maximum for `points` and `assists`, but only for the observations corresponding to that specific team.

The MEANS Procedure

team	N Obs	Variable	N	Mean	Std Dev	Minimum	Maximum
A	6	points	6	16.0000000	5.6213877	10.0000000	24.0000000
		assists	6	6.8333333	2.1369761	4.0000000	9.0000000
B	6	points	6	21.3333333	8.2381228	10.0000000	34.0000000
		assists	6	4.3333333	1.3662601	2.0000000	6.0000000

Crucially, the output verifies that for Team A, there were 6 observations, and the mean points scored was 17.17. Conversely, Team B also had 6 observations, but their mean points scored was significantly higher at 21.33. This direct comparison, enabled by the **CLASS** statement, provides the analytical insight that Team B appears to have a higher scoring average than Team A.

Multi-Dimensional Analysis: Grouping by Combined Categorical Variables

The true power of the **CLASS** statement is realized when multiple grouping variables are employed simultaneously. By adding a second categorical variable, such as `position`, we instruct **SAS** to calculate statistics for every unique combination of `team` and `position`. This moves the analysis from a simple team comparison to a nuanced positional analysis within each team structure. This method allows us to see not just which team is scoring more, but which roles within those teams are contributing to that scoring.

When defining multiple variables in the **CLASS** statement (e.g., `CLASS team position;`), **SAS** creates cross-classification groups. In our basketball dataset, this results in four distinct groups: Team A Guards, Team A Forwards, Team B Guards, and Team B Forwards. The procedure calculates the descriptive metrics independently for each of these four subgroups, providing a highly detailed segmentation of the player performance data.

This level of multi-dimensional data aggregation is essential for complex reporting where factors interact. For instance, if Team B scores higher overall, is it due to outstanding guard play or exceptional forward performance? Grouping by both variables answers such intricate questions.

Here is the **SAS** code demonstrating the use of two variables within the **CLASS** statement:

```
/*calculate summary statistics for numeric variables, grouped by team and position*/
proc means data=my_data;
class team position;
run;
```

The resulting output now features multiple rows detailing the cross-classified statistics. We can clearly observe the performance of each specific positional group within their respective teams.

The MEANS Procedure

team	position	N Obs	Variable	N	Mean	Std Dev	Minimum	Maximum
A	Forward	2	points	2	13.0000000	0	13.0000000	13.0000000
			assists	2	8.5000000	0.7071068	8.0000000	9.0000000
	Guard	4	points	4	17.5000000	6.6080759	10.0000000	24.0000000
			assists	4	6.0000000	2.1602469	4.0000000	9.0000000
B	Forward	3	points	3	24.0000000	9.5393920	15.0000000	34.0000000
			assists	3	4.0000000	1.7320508	2.0000000	5.0000000
	Guard	3	points	3	18.6666667	7.5718778	10.0000000	24.0000000
			assists	3	4.6666667	1.1547005	4.0000000	6.0000000

For instance, the first row of the output corresponds to Team A Forwards, showing their mean points (13.00) and mean assists (8.50), based on 2 observations. Comparing this to Team B Forwards (mean points 24.00), it becomes evident that the high overall scoring average of Team B is strongly driven by their Forward position. This depth of analysis is only possible through the effective application of the multi-variable **CLASS** statement.

Interpreting Results and Advanced Applications of Grouped Data

The output generated by **PROC MEANS** with the **CLASS** statement provides not just the

numerical results, but also structured insights into data distribution. When interpreting these results, it is crucial to pay attention to the observation count (N) for each group. A small N indicates less reliable statistics for that specific subgroup. Furthermore, the standard deviation (Std Dev) reveals the variability within the group--a large standard deviation suggests the players within that category (e.g., Team B Guards) have widely varying performance metrics.

Beyond simple viewing, the results from grouped means calculations are often used as inputs for subsequent statistical models. For example, the mean points calculated per team and position could be saved to a new dataset using the `OUTPUT` statement within **PROC MEANS**. This new dataset of aggregated means can then be utilized for comparative tests, visualizations (like bar charts comparing group means), or as explanatory variables in a regression model.

Advanced users can also leverage options like `NWAY`, `MAXDEC`, and `VARDEF` in conjunction with the **CLASS** statement to fine-tune the output. The `NWAY` option, for example, restricts the output dataset to only the deepest level of classification (the combinations of all variables listed in **CLASS**), which is particularly useful when dealing with multiple levels of hierarchical data aggregation, ensuring that only the most detailed segment means are reported or saved.

Conclusion: Best Practices for Effective Use of the CLASS Statement

The **PROC MEANS** procedure is a cornerstone of descriptive analysis in **SAS**, and the **CLASS** statement elevates its utility by enabling powerful, segmented analysis. By properly specifying categorical variables for grouping, analysts can transition from understanding overall trends to pinpointing specific performance drivers or deviations across organizational units or demographic segments. This capacity for granular detail is indispensable in business intelligence and academic research alike.

When employing the **CLASS** statement, always ensure the variables listed are truly categorical and have a manageable number of unique levels. Using variables with hundreds or thousands of unique values in the **CLASS** statement can lead to extremely long outputs and inefficient processing. Furthermore, carefully review the output structure to ensure the intended cross-classification has occurred, particularly when using multiple grouping variables, as demonstrated by our team and position analysis.

To summarize the utility demonstrated in these examples, using **PROC MEANS** with the **CLASS** statement is the most efficient and robust way to calculate statistics on subsets of your data in **SAS**. We have shown how to calculate global summary statistics, segment the data by one variable, and perform complex multi-dimensional aggregation, providing a complete methodology for data exploration using this powerful procedure.

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