Learning Percentiles Calculation in SAS: A Step-by-Step Guide

Authored by

Mohammed looti

October 31, 2025

Understanding Percentiles and the Role of SAS

In data analysis, percentiles are crucial measures that indicate the value below which a given percentage of observations in a group of observations falls. For example, the 70th percentile is the value where 70% of the data falls below it. Calculating these values efficiently is fundamental for understanding data distribution, identifying benchmarks, and detecting potential outliers. The SAS statistical software environment provides robust tools for these calculations, primarily through the use of the **PROC UNIVARIATE** procedure. This guide outlines the three most effective and common ways to calculate percentiles within a SAS dataset, moving from simple single calculations to complex grouped analyses.

The foundation of percentile calculation in SAS rests upon **PROC UNIVARIATE**. This procedure is designed to provide detailed descriptive statistics for numeric variables, including measures of location, variability, and distribution shape. When calculating percentiles, we leverage several key options within this procedure to direct the output precisely. The following methods demonstrate how to tailor **PROC UNIVARIATE** to meet specific analytical needs, whether you are interested in a single cut-off point or a comprehensive set of distribution markers.

Method 1: Calculating a Single Specific Percentile Value

Often, analysts only require a single percentile value--perhaps the median (50th percentile) or a specific benchmark like the 75th percentile (the third quartile). This method is the simplest application of **PROC UNIVARIATE**. By utilizing the **PCTLPTs** option, we instruct SAS exactly which percentile point to calculate. This result is then saved to a new output dataset, allowing for seamless integration into further analyses or reporting.

The structure below shows the necessary syntax to calculate one specific percentile. It is essential to specify the input dataset (using the **DATA** statement) and the variable of interest (using the **VAR** statement). The **OUTPUT OUT** statement defines the name of the new dataset containing the results, while **PCTLPTs** specifies the desired percentile rank (e.g., 70). The **PCTLPRE** option assigns a customizable prefix to the new percentile variable created in the output dataset, ensuring clarity in the results.

```
/*calculate 70th percentile value for var1*/
proc univariate data=original_data;
var var1;
output out=percentile_data
```

```
pctlpts = 70
pctlpre = P_;
run;
```

Method 2: Calculating Multiple Specific Percentile Values

When a comprehensive view of the data distribution is required, calculating multiple percentile values simultaneously is far more efficient than running separate procedures. This is particularly useful for segmenting data into deciles (10th, 20th, 30th, etc.) or for detailed quartile analysis. The PROC UNIVARIATE procedure handles this seamlessly by allowing multiple values within the **PCTLPTs** statement, separated by spaces.

This approach minimizes processing time and consolidates the output into a single, easily interpretable dataset. For example, if we need to identify the 70th, 80th, and 90th percentile values for a variable named `var1`, we simply list these points within the **PCTLPTs** option. The output dataset will subsequently include three new variables, each prefixed according to the **PCTLPRE** specification (e.g., P_70, P_80, P_90), facilitating straightforward interpretation of the key distribution markers.

```
/*calculate 70th, 80th, and 90th percentile value for var1*/
proc univariate data=original_data;
var var1;
output out=percentile_data
pctlpts = 70 80 90
pctlpre = P_;
run;
```

Prerequisite: Sorting Data for Grouped Analysis

Before proceeding to calculate percentiles for data segmented by groups (e.g., calculating separate percentiles for Team A and Team B), a critical prerequisite in SAS is sorting the input dataset. When using the **BY** statement within any SAS procedure, the input data must be sorted according to the variables listed in the **BY** statement. Failure to sort the data first will result in an error or, potentially worse, misleading results.

We employ **PROC SORT** for this task. The **PROC SORT** step takes the original dataset and

creates a new version (or overwrites the existing one) where observations are ordered sequentially based on the grouping variable, in this case, `var2`. This structured approach ensures that **PROC UNIVARIATE** can correctly identify and process each distinct group independently when calculating the required statistics.

```
/*sort original data by var2*/
proc sort data=original_data;
by var2;
run;
```

Method 3: Calculating Percentiles by Group

Segmenting data is often necessary to compare subgroups within a population. For instance, comparing the distribution of scores between different teams or departments provides deeper insights than looking at the overall distribution alone. Once the data has been sorted (as shown above using **PROC SORT**), we can use the **BY** statement in PROC UNIVARIATE to calculate percentiles separately for each group defined by the sorting variable.

In this method, the **OUTPUT** dataset will contain multiple rows for the percentile results--one set of percentile values corresponding to each unique value in the grouping variable (`var2`). This provides a powerful comparative tool, allowing analysts to quickly determine how the distribution of `var1` differs across the various categories defined by `var2`. The combined use of **PROC SORT** and the **BY** statement within **PROC UNIVARIATE** is foundational for effective subgroup data analysis in SAS.

```
/*calculate percentiles for var1 grouped by var2*/
proc univariate data=original_data;
var var1;
by var2;
output out=percentile_data
pctlpts = 70, 80, 90
pctlpre = P_;
run;
```

Important Note on Options: The **pctlpts** statement is used exclusively to specify which percentiles (e.g., 70, 95) should be calculated and stored. The **pctlpre** statement is crucial for assigning a consistent prefix (e.g., P_) to the new variables representing the percentile values in

the output dataset, ensuring easy identification.

Practical Examples Using Sample Data

To illustrate these methods, we will use a small sample dataset containing scores (`points`) achieved by two different teams (`team`). The following SAS code block generates this demonstration data, which will serve as the input for all subsequent examples.

```
/*create dataset*/
data original_data;
input team $ points;
datalines;
A 12
A 15
A 16
A 21
A 22
A 25
A 29
A 31
B 16
B 22
B 25
B 29
B 30
B 31
B 33
B 38
;
run;

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

Obs	team	points
1	A	12
2	A	15
3	A	16
4	A	21
5	A	22
6	A	25
7	A	29
8	A	31
9	B	16
10	B	22
11	B	25
12	B	29
13	B	30
14	B	31
15	B	33
16	B	38

Example 1: Calculating a Single Specific Percentile Value

In this first example, we calculate only the 70th percentile for the **points** variable across the entire combined dataset. This tells us the score below which 70% of all observations fall. We use the **PCTLPTs = 70** option within the **proc univariate** output statement, storing the result in a new dataset called `percentile_data`.

The subsequent **PROC PRINT** step is used to inspect the newly created dataset, which contains the calculated percentile value. This method is fast and precise when only a single statistical cutoff point is required for benchmarking or filtering operations.

```
/*calculate 70th percentile value for points*/
proc univariate data=original_data;
var points;
output out=percentile_data
pctlpts = 70
pctlpre = P_;
run;
```

```
/*view results*/
proc print data=percentile_data;
```

Obs	P_70
1	30

As shown in the output, the value calculated at the 70th percentile point is **30**. This means that 70% of all scores in the `original_data` dataset are less than or equal to 30.

Example 2: Calculating Multiple Specific Percentile Values

To gain a more granular understanding of the data distribution, this example demonstrates calculating the 70th, 80th, and 90th percentile values simultaneously. This is achieved by listing all desired percentile ranks (70, 80, 90) in the **PCTLPTs** option. This approach is highly efficient and provides immediate insight into the upper tail of the distribution of `points`.

The resulting output dataset will feature three new variables, P_70, P_80, and P_90, reflecting the scores that correspond to these statistical thresholds. Analyzing these multiple cutoffs is essential when defining performance tiers or identifying potential high-value observations within the data distribution.

```
/*calculate 70th, 80th, and 90th percentile value for points*/
proc univariate data=original_data;
var points;
output out=percentile_data
pctlpts = 70 80 90
pctlpre = P_;
run;
```

Obs	P_70	P_80	P_90
1	30	31	33

The interpretation of this consolidated output is straightforward:

The value at the 70th percentile is **30**.
The value at the 80th percentile is **31**.
The value at the 90th percentile is **33**.

Example 3: Calculating Percentiles by Group

This final example showcases the strength of SAS in performing comparative statistical data analysis. Here, we calculate the 70th, 80th, 90th, and 95th percentiles for the **points** variable, but critically, these calculations are segmented by the **team** variable. This allows us to compare the score distributions of Team A versus Team B directly.

As discussed in the prerequisite section, we must first use **PROC SORT** to order the data by `team`. We then execute **PROC UNIVARIATE**, incorporating the **BY team** statement. This instructs the procedure to reset the percentile calculation for every new team it encounters, producing two distinct sets of percentile values in the output dataset, one for each team. This is invaluable for performance benchmarking across organizational units.

```
/*sort original data by team*/
proc sort data=original_data;
by team;
run;

/*calculate percentiles for points grouped by team*/
proc univariate data=original_data;
var points;
by team;
output out=percentile_data
pctlpts = 70, 80, 90 95
pctlpre = P_;
run;
```

Obs	team	P_70	P_80	P_90	P_95
1	A	25	29	31	31
2	B	31	33	38	38

The resulting output table clearly presents the values for the 70th, 80th, 90th, and 95th percentiles for the **points** variable, calculated separately for both Team A and Team B. For example, while the 70th percentile for Team A is 29, the 70th percentile for Team B is 31, immediately highlighting a higher overall score distribution for Team B.

Additional Resources for SAS Procedures

Mastering percentile calculation using **PROC UNIVARIATE** is a fundamental step in advanced SAS usage. For those looking to expand their procedural knowledge, the following tutorials explain how to perform other common statistical and data manipulation tasks in the SAS environment.