Learning to Create Line Plots in SAS with PROC SGPLOT

Authored by

Mohammed looti

November 1, 2025

A line plot, often referred to as a line chart, is an indispensable tool in data analysis. It serves the critical function of displaying quantitative data points connected by line segments, primarily used to visualize trends over continuous intervals, most commonly time. In the statistical software environment of SAS, generating professional and high-quality graphical output is best achieved by employing the modern and robust **proc sgplot** procedure.

Unlike older SAS graphical procedures, **proc sgplot** offers streamlined control and flexibility, allowing analysts to create complex statistical visualizations, including detailed line plots, with minimal code. This comprehensive guide is designed to walk you through the essential components of **proc sgplot**, providing clear instructions and practical examples for generating both single-series and comparative multiple-series line plots effectively within the SAS system.

Understanding the Basic `PROC SGPLOT` Syntax

To successfully execute a standard line plot using **proc sgplot**, two core elements are mandatory: the specification of the input **dataset** and the utilization of the `SERIES` statement. The `SERIES` statement is the workhorse of line plotting, as it explicitly defines which variables are mapped to the horizontal (X) and vertical (Y) axes, thereby establishing the relationship being visualized.

Adhering to correct syntax is crucial for efficient programming in SAS. The fundamental structure for generating any line plot follows a standardized, simple pattern, ensuring that the procedure knows where the data resides and how the axes should be constructed. This basic framework is the foundation upon which all subsequent customizations are built.

The core structure for generating any line plot in SAS follows this simple pattern:

```
/*create dataset*/
proc sgplot data=my_data;
series x=x_variable y=y_variable;
run;
```

The subsequent examples provided in this guide will demonstrate how to apply this essential structure and introduce various options to tailor your visualizations, allowing you to effectively communicate insights derived from different types of data.

Example 1: Creating a Line Plot with a Single Series

Our initial example focuses on visualizing a single trend over time. We will analyze the daily sales performance of a single retail outlet across a continuous period of 10 days. This scenario is perfectly suited for a basic line plot, which clearly illustrates the fluctuations and overall trajectory of sales figures over the specified time frame.

Before plotting can occur, we must establish our input **dataset**. We name this dataset `my_data`. It contains two essential variables: `day`, which represents the independent variable (time interval), and `sales`, which represents the dependent variable (the metric being tracked).

The following SAS code block first creates and populates the `my_data` dataset using `DATALINES`, and then utilizes `PROC PRINT` to display the data in the output window, ensuring data integrity before proceeding to the plotting stage:

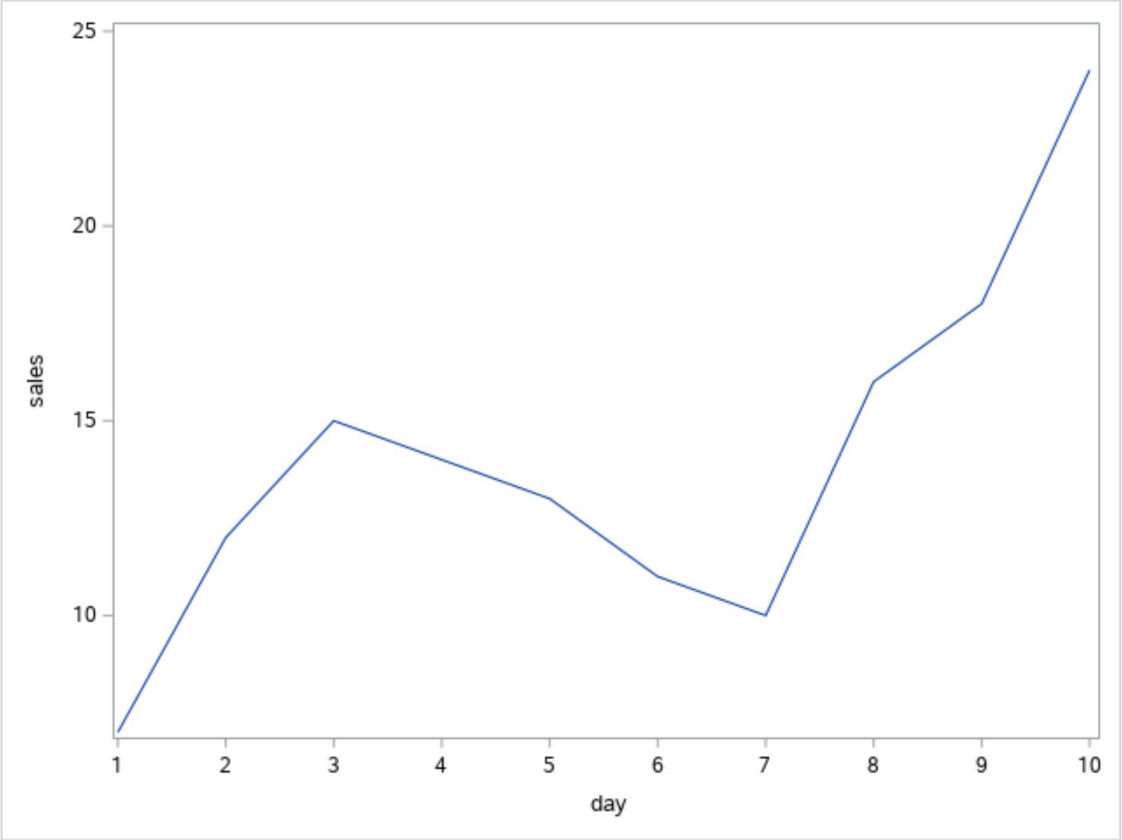
```
/*create dataset*/
data my_data;
input day $ sales;
datalines;
1 7
2 12
3 15
4 14
5 13
6 11
7 10
8 16
9 18
10 24
;
run;

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

Obs	day	sales
1	1	7
2	2	12
3	3	15
4	4	14
5	5	13
6	6	11
7	7	10
8	8	16
9	9	18
10	10	24

With the data successfully loaded and verified, we proceed to apply the plotting procedure. We invoke **proc sgplot**, specifying the `my_data` dataset. The critical step is using the `SERIES` statement, which maps the `day` variable to the X-axis and the `sales` variable to the Y-axis. This simple command generates the initial, unformatted line plot, clearly illustrating the daily sales trend.

```
/*create line plot that displays sales by day*/
proc sgplot data=my_data;
series x=day y=sales;
run;
```



Enhancing Visual Appearance and Customization

While the basic plot generated in the previous step is technically functional, real-world data visualization often demands enhanced aesthetics and greater clarity. **proc sgplot** provides extensive options for visual customization, allowing analysts to significantly modify the chart's appearance. These modifications typically include adding a descriptive chart title, changing the line color, adjusting the line pattern (e.g., from solid to dashed), and setting the line thickness for improved emphasis.

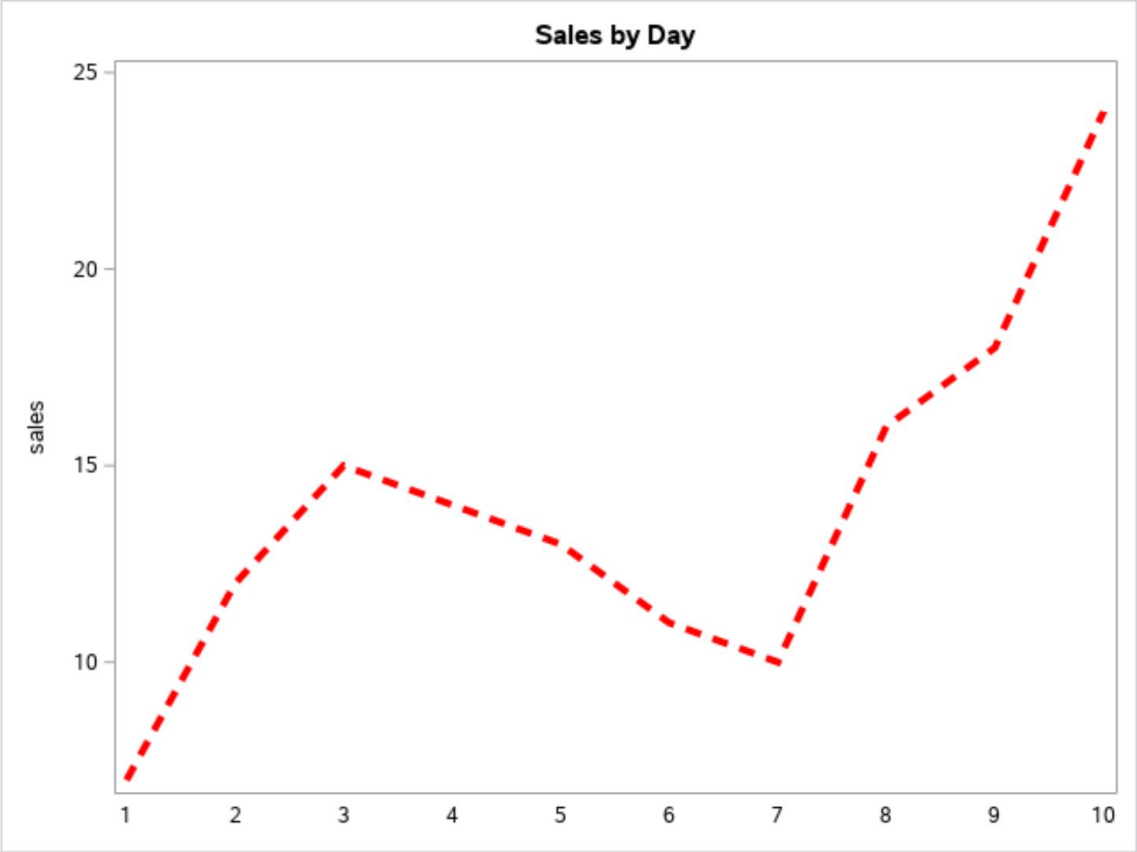
The primary mechanism for customizing the line itself is the `LINEATTRS` option, which is nested within the `SERIES` statement. This option accepts parameters such as `COLOR`, `PATTERN`, and `THICKNESS`. Furthermore, the `XAXIS` statement allows for granular control over the axis display, enabling the suppression of redundant elements like labels, lines, or ticks, thereby decluttering the visualization and focusing the viewer's attention on the trend data.

The following syntax demonstrates how to incorporate these customization options. Note the use of the `TITLE` statement to provide context and the `XAXIS` options to refine the display of the horizontal axis:

/*create custom line plot*/

```
title "Sales by Day";
proc sgplot data=my_data;
series x=day y=sales / lineattrs=(color=red pattern=dash thickness=4);
xaxis display=(nolabel noline noticks);
run;
title;
```

The resulting plot is significantly cleaner and more visually impactful, highlighting the data trend using a distinct color, thicker line, and a dashed pattern, as requested by the `LINEATTRS` options.



Example 2: Visualizing Trends with Multiple Lines

A common analytical requirement is the comparison of multiple trends simultaneously, which is necessary when tracking similar metrics across different categories or groups. In SAS, generating a multi-line plot is straightforward and requires the introduction of a grouping variable within the `SERIES` statement.

For this comparative example, we will track and compare the sales performance of three separate retail stores (labeled A, B, and C) over a period of five consecutive days. To accommodate this

comparison, our input **dataset** must now include a third categorical variable, `store`. This grouping variable allows SAS to draw distinct lines for each unique store location.

Below is the revised data step, which includes the grouping variable `store`, followed by the print procedure to verify the data structure:

```
/*create dataset*/
data my_data;
input store $ day $ sales;
datalines;
A 1 13
A 2 18
A 3 20
A 4 25
A 5 26
B 1 3
B 2 7
B 3 12
B 4 12
B 5 11
C 1 6
C 2 12
C 3 19
C 4 20
C 5 21
;
run;

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

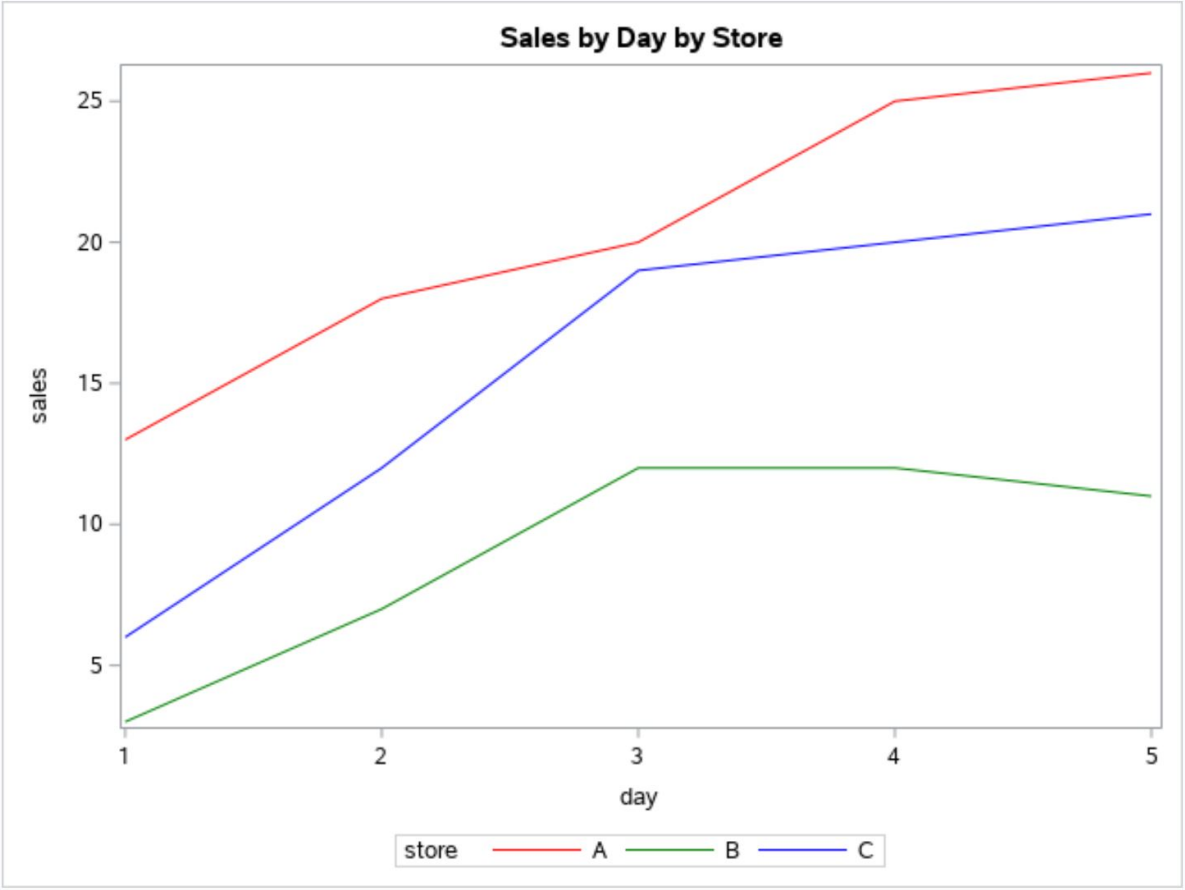
Sales by Day

Obs	store	day	sales
1	A	1	13
2	A	2	18
3	A	3	20
4	A	4	25
5	A	5	26
6	B	1	3
7	B	2	7
8	B	3	12
9	B	4	12
10	B	5	11
11	C	1	6
12	C	2	12
13	C	3	19
14	C	4	20
15	C	5	21

To generate the visualization comparing all three stores, we again use **proc sgplot**. The key difference here is the inclusion of the `GROUP=` option within the `SERIES` statement, setting it equal to the `store` variable. This instruction directs SAS to automatically segment the data and draw a separate, distinct line for each unique value found in the grouping variable.

For even greater control over the visual presentation, we employ the `STYLEATTRS` statement. By using `STYLEATTRS DATACONTRASTCOLORS=`, we explicitly define the colors that SAS will use for each of the three data series, ensuring maximum visual separation and clarity:

```
/*create line plot that displays sales by day for each store*/
title "Sales by Day by Store";
proc sgplot data=my_data;
styleattrs datacontrastcolors=(red green blue);
series x=day y=sales / group=store;
run;
title;
```



The resulting plot effectively maps the consecutive days on the X-axis and the total sales volume on the Y-axis. The powerful combination of the `GROUP` parameter and the explicit color definition in `STYLEATTRS` successfully separates the data into three distinct, color-coded lines, facilitating an immediate and clear visual comparison of the performance trends among the three store locations.

Next Steps and Related Visualizations

The **proc sgplot** procedure stands as the essential, modern framework for generating professional-grade line plots in SAS. By developing proficiency in utilizing the `SERIES` statement, particularly through its customization options (like `LINEATTRS`) and grouping capabilities (`GROUP=`), you gain the ability to accurately and effectively visualize temporal trends, comparative data, and multiple series within a single, coherent graphic.

To further advance your data visualization expertise in SAS, we highly recommend exploring the versatility of **proc sgplot** beyond simple line charts. The procedure supports a wide array of other crucial statistical graph types using different statements (such as `VBAR` for bar charts, `SCATTER` for scatter plots, and `HISTOGRAM` for distribution analysis). Mastering these additional graph types will significantly broaden your analytical toolkit.

For continued learning, consider reviewing tutorials that focus on related visualizations. The following resources explain how to create other common visualizations in SAS using this powerful procedure: