PharmaSUG 2022 - Paper QT-169

A Beginner's Guide to Create Series Plots Using SGPLOT Procedure: From Basic to Amazing

Aakar Shah, Neoleukin Therapeutics
Tracy Sherman, Ephicacy Consulting Group, Inc.

ABSTRACT

As a beginner SAS programmer, it can be intimidating to work on graph assignments. SAS® ODS Graphics provides an avenue for SAS users to create visually pleasing graphs very quickly. However, ODS Graphics documentation is extensive and cumbersome. This paper can help beginner SAS programmers on their journey from creating a basic series plot using a simple SGPLOT statement to amazing, formatted graphs which can be shared with upper management or in a conference presentation. Knowledge learned from this paper can be further applied to other types of graphs.

INTRODUCTION

The goal of this paper is to describe the journey from producing very basic scatter plots to the most amazing looking scatter plot. And along the way, it shines light on various options, tips and tricks associated with the SGPLOT procedure. This paper is one-stop shop to find what you need to provide best graphical value in your organization.

This paper starts with giving you a basic understanding of this procedure using SAS v9.4 and will explain and show the result of adding each statement and specific options included in the SGPLOT syntax to produce the most basic to advanced figures. Readers will learn skills to produce figures that are very eye-catching with the use of attributes such as backgrounds, markers and color schemes. Not only will the reader learn to produce these figures but will also understand each component of the syntax the figure is built-upon. The result are plots that can be used for quality presentations to upper management for safety monitoring and conference presentations.

The SGPLOT procedure can create one or more figures and can overlay them on a single axis. You can also use the SGPLOT procedure to create other graphics such as histograms, scatter plots and many other types of plots. In this paper, the focus will be on producing high quality laboratory series plots with all the bells and whistles.

The syntax of SGPLOT looks something like:

```
proc sgplot <data=data-set> <options>;
   plot-statement(s) required-parameters </options>;
   <styleattrs statement(s)>;
   <refline-statement(s)>;
   <inset-statement(s)>;
   <axis-statement(s)>;
   <keylegend-statement(s)>;
run;
```

In the discussion that follows, the syntax of SGPLOT will be broken down and the relationships between each statement will be explained and demonstrated.

SOURCE DATA

Source data (see Appendix I for full code) used throughout the paper consists of dummy laboratory data that was created as follows:

```
data lab;
   infile datalines delimiter=',';
   input Subject $ Cohort $ Visitnum Visit $ Test $ Result Unit $;
   datalines;
```

```
    100-101, A, 1, C01 D01, Monocyte, 0.11, 10^3/uL
    100-101, A, 2, C01 D08, Monocyte, 0.22, 10^3/uL
    100-101, A, 3, C02 D01, Monocyte, 0.23, 10^3/uL
    100-101, A, 4, C02 D08, Monocyte, 0.51, 10^3/uL
…
run;
```

MANAGING YOUR GRAPHICS WITH ODS

ODS supports multiple outputs for procedure destination. LISTING destination's default output is SAS output listing. For the LISTING destination, GPATH= option is used to tell SAS where to save the graph files. PNG file extension is its default image format.

```
ods listing gpath = "<path>";
```

The Output Delivery System (ODS) manages all output created by procedures and enables you to display the output in a variety of formats, such as HTML, PDF, and RTF. The SAS ODS Graphics procedures such as PROC SGPLOT and many SAS Analytical procedures use ODS Graphics for creation of their graphs.

You can use the ODS GRAPHICS statement options to control many aspects of your graphics. The settings that you specify remain in effect for all graphics in the current session until you change or reset these settings with another ODS GRAPHICS statement. When you use the SAS ODS Graphics procedures in the SAS Windowing environment, ODS GRAPHICS is ON by default. In batch mode, ODS GRAPHICS is OFF by default.

The basic syntax for the ODS GRAPHICS statement is as follows:

```
ods graphics < off | on> </ options>;
```

You can use the ODS GRAPHICS statement to control many features, including the size of the image, the type and name of the image created. WIDTH option can be specified in many units such as centimeters (cm), inches (in), millimeters (mm), percentage (pct or %), point size (72 points – 1 inch) (pt), pixels (px). By only specifying WIDTH = 10in, SAS will adjust the height using a default aspect ratio of 4:3 (Slaughter and Delwiche, 2015). IMAGENAME= option specifies the image name. A file extension for filename is automatically generated based on the OUTPUTFMT= file-type, where file-type can be PNG, BMP, EMF, JPG, JPEG, TIFF and more. RESET option will reset all ODS GRAPHICS option to its default.

```
* Open the ODS destination and reference the custom style if applicable *;
ods listing gpath = "&output.";

* Define the attributes of the image file *;
ods graphics / reset imagename = "fig_1" outputfmt = jpg width = 10in;

title1 "Absolute Monocytes";
proc sgplot data = lab;
   series x = visit y = result / group = subject;
run;

* Close the ODS destination *;
ods listing close;
```

BASIC SGPLOT WITH SERIES STATEMENT

One of the most significant statements in the SGPLOT procedure is the SERIES statement. When you specify the GROUP= option, you specify a variable such as subject that is used to group the lines of your series plot (see Figure 1). The plot lines for each group value are automatically distinguished by different visual attributes (SAS Institute 2022a).

```
title1 "Absolute Monocytes";
proc sgplot data = lab;
   series x = visit y = result / group = subject;
run;
```

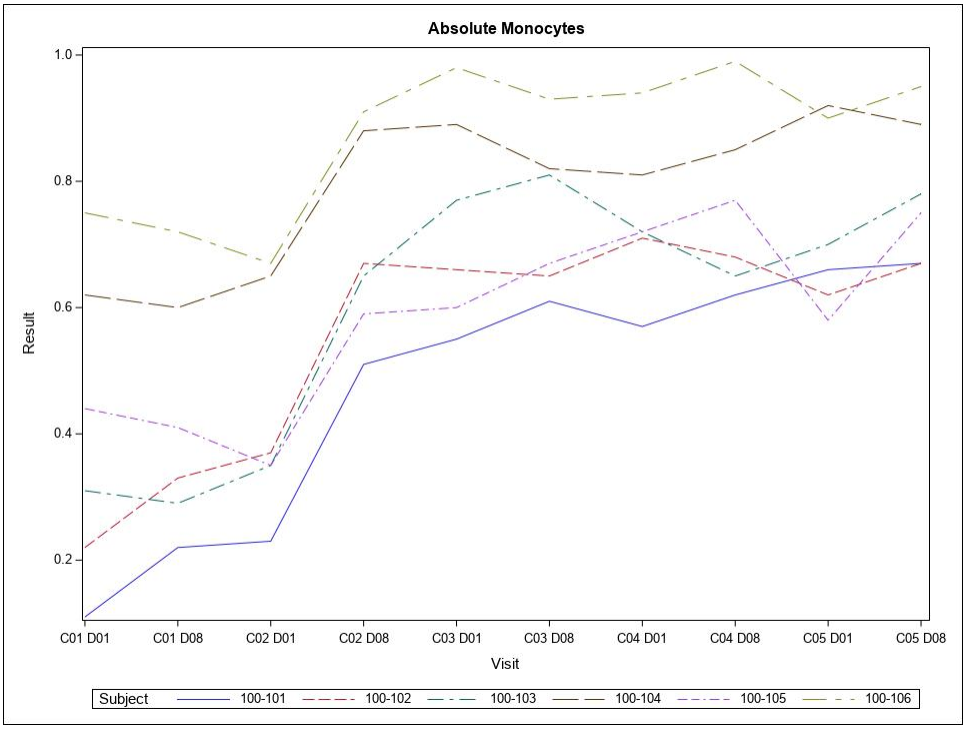


Figure 1. Basic Series Plot by Visit with Option GROUP=subject

THE AXIS STATEMENT WITH LABEL= OPTION

Using the XAXIS and YAXIS statements, you can specify options such as LABEL=<>. 'VISIT' was used for the xaxis and 'x10^{9}/L' for the yaxis (see Figure 2). Anytime ODS encounters the (*ESC*), it will interpret whatever follows (in this case it is the unicode value of '2079' to create a superscript of 9) to be a string of control characters that will change the value in the final ODS output.

LABEL OPTION USING (*ESC*) AS THE ODS ESCAPE CHARACTER

```
proc sgplot data = lab;
   series x = visit y = result / group = subject;
   xaxis label = "VISIT";
   yaxis label = "x10(*ESC*){unicode '2079'x}/L";
run;
```

LABEL OPTION USING THE ODS ESCAPE STATEMENT

Another solution for adding an escape character is adding the ODS ESCAPECHAR='^' statement and then replacing the '(*ESC*)' text with the caret '^'.

```
ods escapechar = '^';
proc sgplot data = lab;
   series x = visit y = result / group = subject;
   xaxis label = "VISIT";
   yaxis label = "x10^{unicode '2079'x}/L";
run;
```

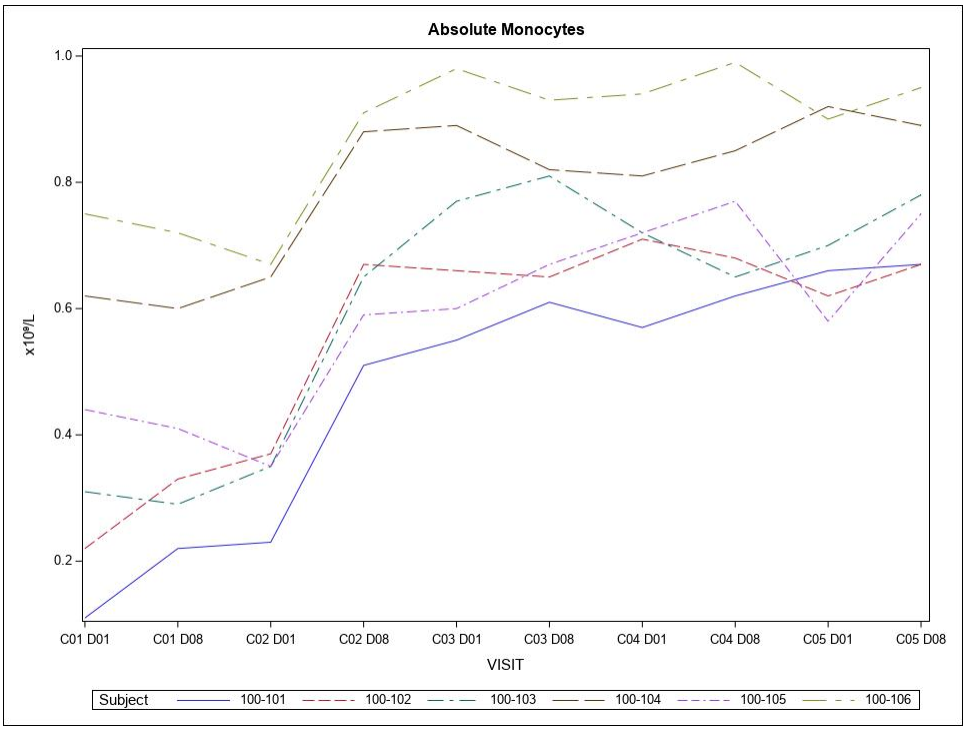


Figure 2. Using the XAXIS and YAXIS Statements and Adding Superscript Values with the LABEL= Option

THE AXIS STATEMENT WITH VALUES= OPTION

With the VALUES= options you can specify which values you want displayed for each tick on either axis. In Figure 3 and shown in the code below, the values of the yaxis are updated to show a tick mark from 0 to 1 by 0.1.

```
proc sgplot data = lab;
   series x = visit y = result / group = subject;
   xaxis label = "VISIT";
   yaxis values = (0 to 1 by 0.1) label = "x10(*ESC*){unicode '2079'x}/L";
run;
```

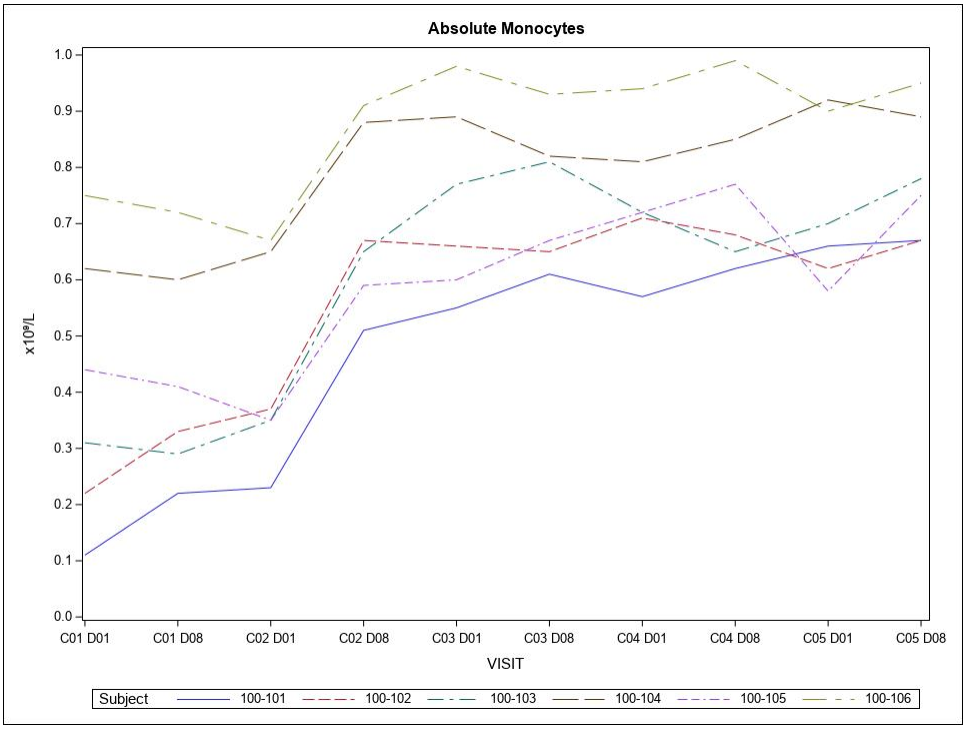


Figure 3. Using the VALUES= Option to Adjust the Tick Values of the Yaxis

THE AXIS STATEMENT WITH GRID OPTION

The GRID option creates faint grid lines at each tick on the axis. In Figure 4, you can see the grid lines that enable the viewer to trace the value on the yaxis to each point on the graph.

```
proc sgplot data = lab;
   series x = visit y = result / group = subject;
   xaxis label = "VISIT";
   yaxis grid values = (0 to 1 by 0.1)
         label = "x10(*ESC*){unicode '2079'x}/L";
run;
```

THE AXIS STATEMENT WITH DISPLAY= OPTION

To make the x and yaxis look cleaner, you can use the DISPLAY= option to remove the tick marks with DISPLAY= (NOTICKS) (see Figure 5). Also note removal of xaxis label with DISPLAY=(NOLABEL) as the xaxis values are self-explanatory.

```
proc sgplot data = lab;
   series x = visit y = result / group = subject;
   xaxis display = (nolabel noticks) label = "VISIT";
   yaxis display = (noticks) grid values = (0 to 1 by 0.1)
         label = "x10(*ESC*){unicode '2079'x}/L";
run;
```

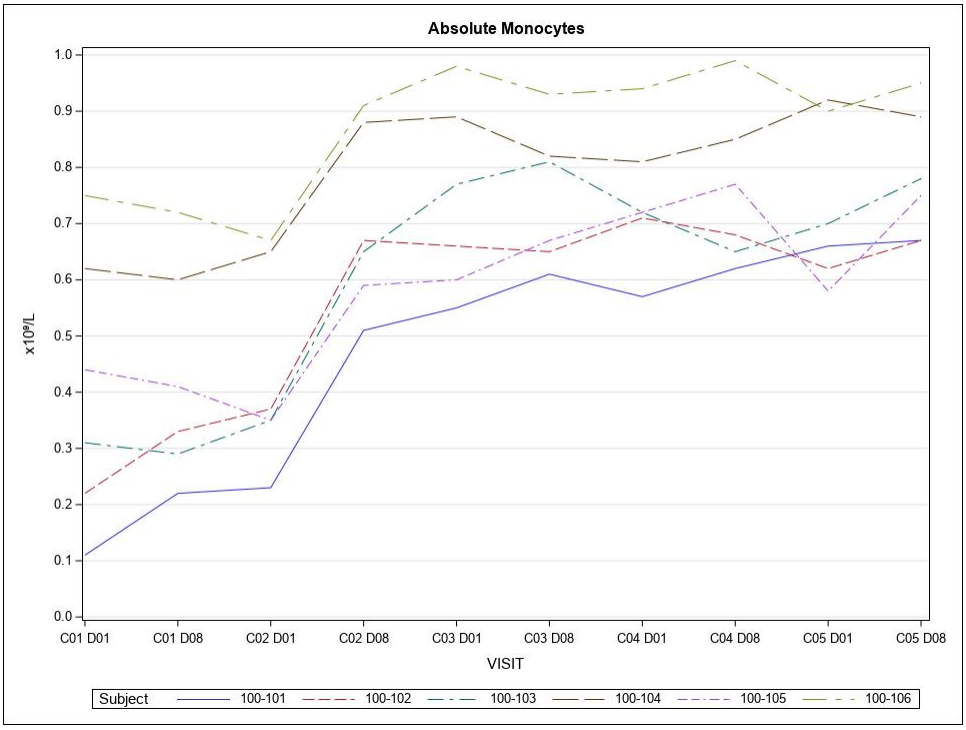


Figure 4. The GRID Option of YAXIS Statement Creates a Grid Line at Each Tick Mark on the Yaxis

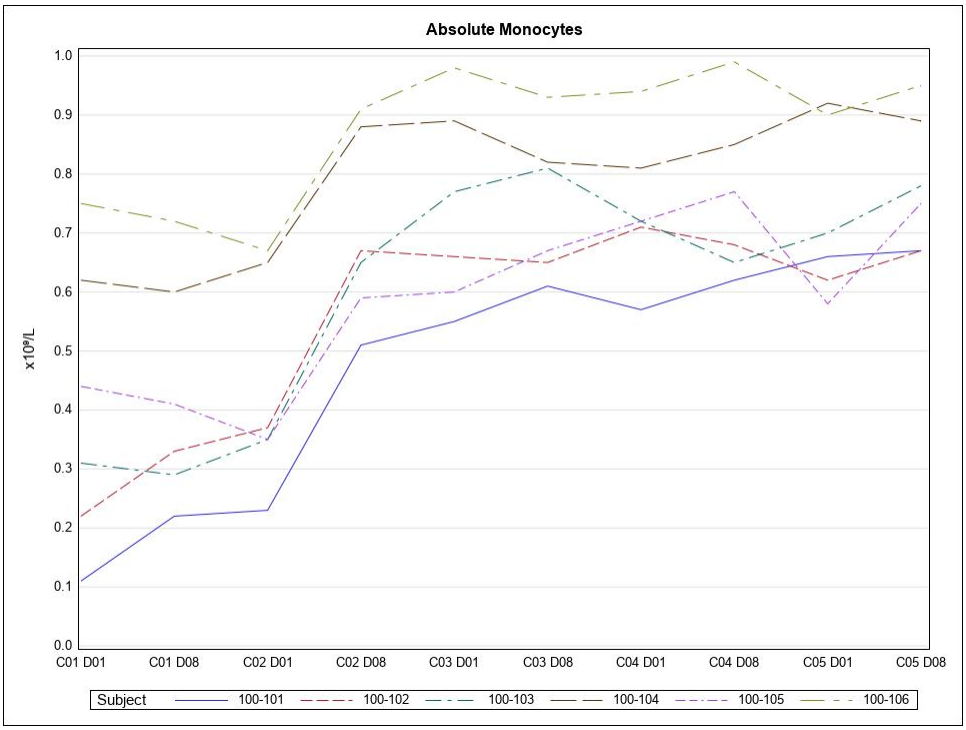


Figure 5. DISPLAY= Option with NOTICKS to Remove Tick Marks on the X and Yaxis and NOLABEL to Remove Xaxis Label.

THE SERIES STATEMENT WITH LINEATTRS= AND MARKERS OPTIONS

You can change the line thickness and pattern with the LINEATTRS= option within the SERIES statement. Here we increased the thickness to 3 and changed the line patterns to solid for all lines. In addition, the option MARKERS was used to add data point markers to the series plot data points (Figure 6). As you can see, SAS will select default markers for you (star, triangle, cross, circle, plus sign, and square) and these markers can be adjusted with the MARKERATTRS= option that will be shown later in the paper.

```
proc sgplot data = lab;
   series x = visit y = result / group = subject
          lineattrs = (thickness = 3 pattern = solid) markers;
   xaxis display = (nolabel noticks) label = "VISIT";
   yaxis display = (noticks) grid values = (0 to 1 by 0.1)
                label = "x10(*ESC*){unicode '2079'x}/L";
run;
```

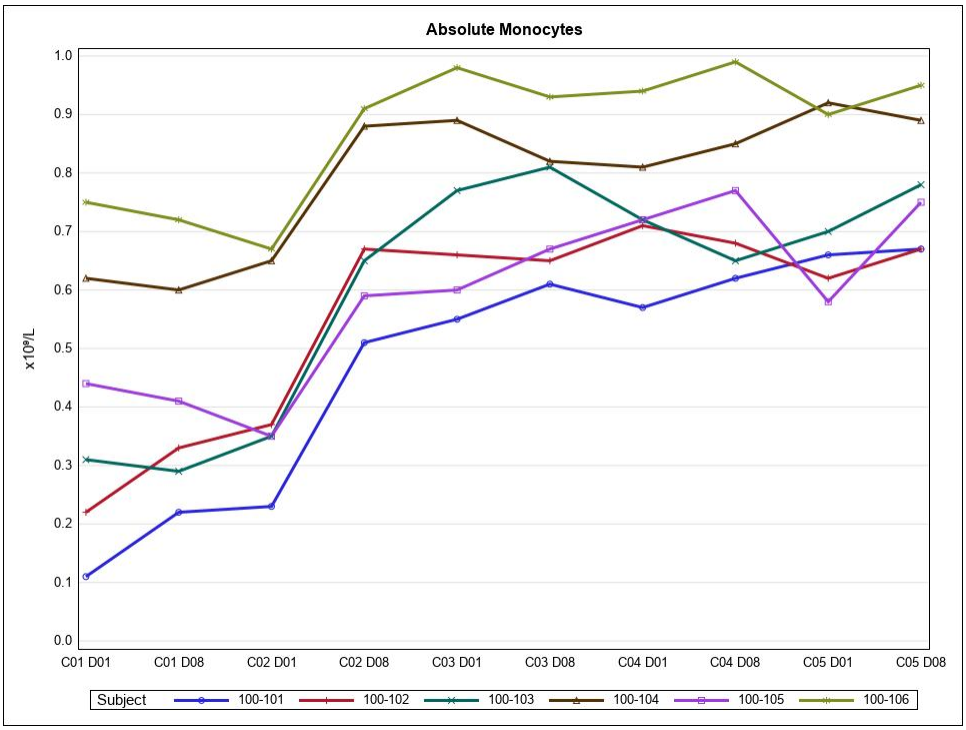


Figure 6. SERIES Statement with LINEATTRS= and MARKERS Option

PROC SGPLOT STATEMENT WITH NOWALL AND NOBORDER

With the NOWALL and NOBORDER options added to the SGPLOT statement, you can remove the walls and borders to create a less cluttered and cleaner looking figure (Figure 7).

```
proc sgplot data = lab nowall noborder;
   series x = visit y = result / group = subject
          lineattrs = (thickness = 3 pattern = solid) markers;
   xaxis display = (noline nolabel noticks) label = "VISIT";
   yaxis display = (noline noticks) grid values = (0 to 1 by 0.1)
         label = "x10(*ESC*){unicode '2079'x}/L";
run;
```

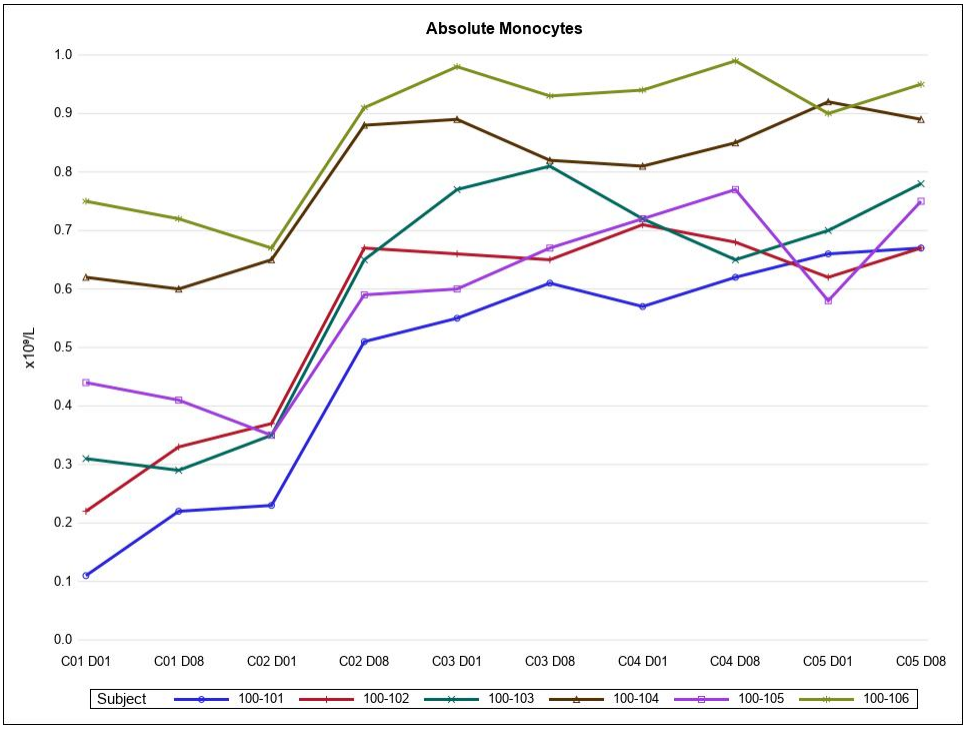
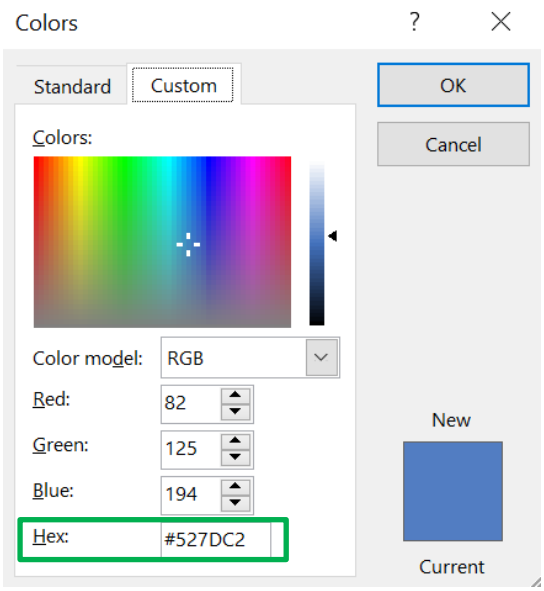


Figure 7. Using NOWALL and NOBORDER to Remove the Walls and Borders

STYLEATTRS STATEMENT AND BACKCOLOR OPTION

Using the STYLEATTRS statement, you can add any background color to the graph area. Hexadecimal colors are the visual language of the graphs. A hex color code is a 6-symbol code of up to three 2-symbol elements. One for each R (Red), G (Green) and B (Blue). You can obtain the code of any color by selecting the color in the color selection tool available in any popular Microsoft Office software. Display 1 shows the hex code “527DC2” for the selected color. In Figure 8, you can see that to make the data points more prominent, we chose a vivid blue background by adding BACKCOLOR = CX527DC2.



Display 1. Microsoft Office Color Selection Tool

```
proc sgplot data = lab nowall noborder;
   series x = visit y = result / group = subject
          lineattrs = (thickness = 3 pattern = solid) markers;
   styleattrs backcolor = CX527DC2;
   xaxis display = (noline nolabel noticks) label = "VISIT";
   yaxis display = (noline noticks) grid values = (0 to 1 by 0.1)
         label = "x10(*ESC*){unicode '2079'x}/L";
run;
```

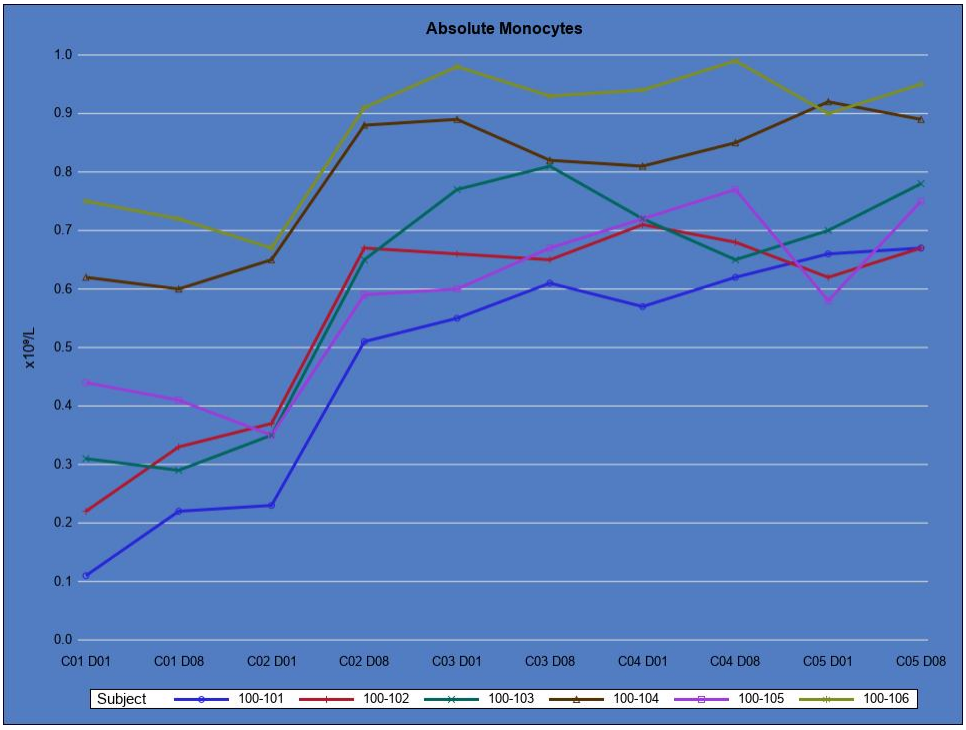


Figure 8. BACKCOLOR=CX527DC2 in the STYLEATTRS Statement Creates a Vivid Blue Background

AXIS STATEMENTS USING VALUEATTRS AND LABELATTRS OPTIONS

To change the axis labels and value to white, you can add the options VALUEATTRS = (color = white weight = bold) and LABELATTRS = (color = white weight = bold). Now in Figure 9, the axis labels and values are more notable against the newly added blue background.

```
proc sgplot data = lab nowall noborder;
   series x = visit y = result / group = subject
         lineattrs = (thickness = 3 pattern = solid) markers;
   styleattrs backcolor = CX527DC2;
   xaxis display = (noline nolabel noticks) label = "VISIT"
         valueattrs = (color = white weight = bold)
         labelattrs = (color = white weight = bold);
   yaxis display = (noline noticks) grid values = (0 to 1 by 0.1)
         valueattrs = (color = white weight = bold)
         labelattrs = (color = white weight = bold)
         label = "x10(*ESC*){unicode '2079'x}/L";
run;
```

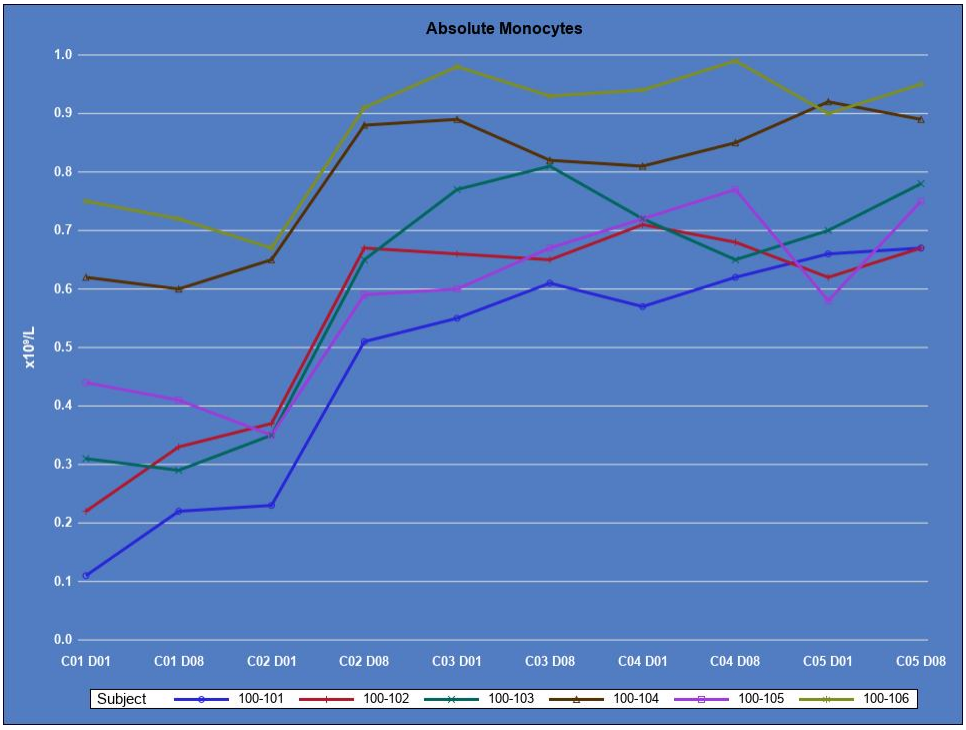


Figure 9. Change Axis Labels and Values to White to be More Notable Against a Blue Background

REFLINE STATEMENT

The REFLINE statement creates a horizontal or vertical reference line. In our case, we needed a reference line drawn at Cycle 2 Day 1 (labelled 'C02 D01' on the xaxis) which is date of first dose of study medication (see Figure 10). AXIS = x specifies you want the reference line drawn vertically from the xaxis. Using the LINEATTRS option, any color and pattern can be selected.

```
proc sgplot data = lab nowall noborder;
   series x = visit y = result / group = subject
         lineattrs = (thickness = 3 pattern = solid) markers;
   styleattrs backcolor = CX527DC2;
   refline 'C02 D01'/ axis = x lineattrs = (color = white pattern = dash);
   xaxis display = (noline nolabel noticks) label = "VISIT"
         valueattrs = (color = white weight = bold)
         labelattrs = (color = white weight = bold);
   yaxis display = (noline noticks) grid values = (0 to 1 by 0.1)
         valueattrs = (color = white weight = bold)
         labelattrs = (color = white weight = bold)
         label = "x10(*ESC*){unicode '2079'x}/L";
run;
```

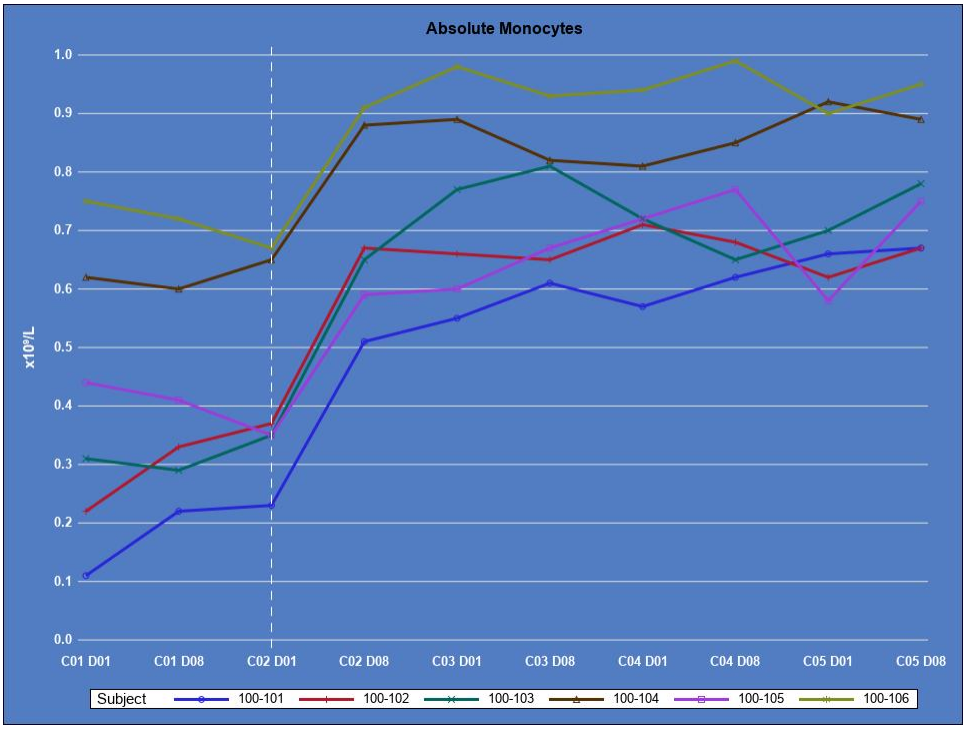


Figure 10. Vertical Reference Line Added at Cycle 2 Day 1 Using the REFLINE Statement

KEYLEGEND STATEMENT

To customize the default legend that SAS produces, add a KEYLEGEND statement to remove the border (NOBORDER), change the background of legend to transparent (NOOPAQUE), remove the title (TITLE = " "), change the attributes of the legend values (VALUEATTRS = (color = white size = 10 weight = bold).

In Figure 11, the legend title 'Subject' has been removed as this is self-evident that these are subject numbers, and values are now bold, white and the background is transparent.

```
proc sgplot data = lab nowall noborder;
   series x = visit y = result / group = subject
         lineattrs = (thickness = 3 pattern = solid) markers;
   styleattrs backcolor = CX527DC2;
   refline 'C02 D01'/ axis = x lineattrs = (color = white pattern = dash);
   xaxis display = (noline nolabel noticks) label = "VISIT"
         valueattrs = (color = white weight = bold)
         labelattrs = (color = white weight = bold);
   yaxis display = (noline noticks) grid values = (0 to 1 by 0.1)
         valueattrs = (color = white weight = bold)
         labelattrs = (color = white weight = bold)
         label = "x10(*ESC*){unicode '2079'x}/L";
   keylegend / noborder noopaque title=" "
         valueattrs = (color = white size = 10 weight = bold);
run;
```

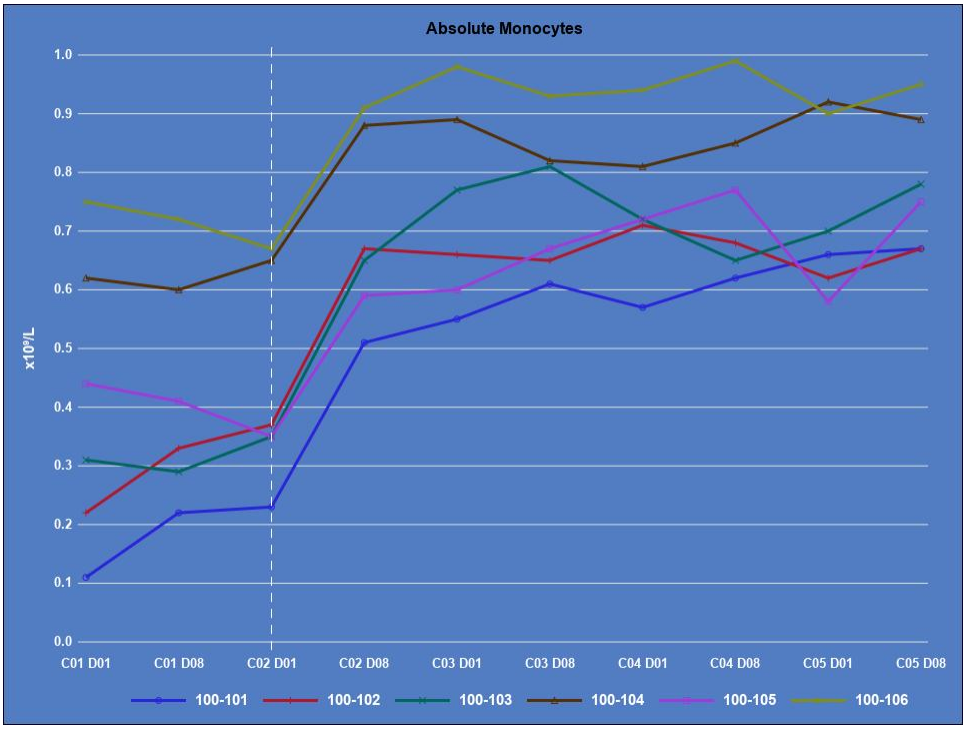


Figure 11. KEYLEGEND Statement to Remove the Legend Title, Values are Bold, White, Size=10 and Legend Background is Transparent.

STYLEATTRS STATEMENT USING DATACONTRASTCOLORS= AND SERIES STATEMENT USING MARKERATTRS=

To change SAS assigned colors of plot lines for each group value, use DATACONTRASTCOLORS in STYLEATTRS statement to assign colors of your choice. We recommend maintaining an external file of color assignments for each group value and macro variable assignment for selected colors so that colors can be consistent across different types of graphs and becomes easier to follow a subject across graphs.

MARKERATTRS option within SERIES statement changes markers attributes such as shape and color. In Figure 12, custom colors are assigned for plot lines and markers now have the symbols “circlefilled” with a bright yellow for further enhancement.

```
proc sgplot data = lab nowall noborder;
   series x = visit y = result / group = subject
         lineattrs = (thickness = 3 pattern = solid) markers
         markerattrs = (symbol = circlefilled color = yellow);
   styleattrs datacontrastcolors = (white brown charcoal purple cyan gold)
         backcolor = CX527DC2;
   refline 'C02 D01'/ axis = x lineattrs = (color = white pattern = dash);
   xaxis display = (noline nolabel noticks) label = "VISIT"
         valueattrs = (color = white weight = bold)
         labelattrs = (color = white weight = bold);
   yaxis display = (noline noticks) grid values = (0 to 1 by 0.1)
         valueattrs = (color = white weight = bold)
         labelattrs = (color = white weight = bold)
         label = "x10(*ESC*){unicode '2079'x}/L";
   keylegend / noborder noopaque title=" "
         valueattrs = (color = white size = 10 weight = bold);
run;
```

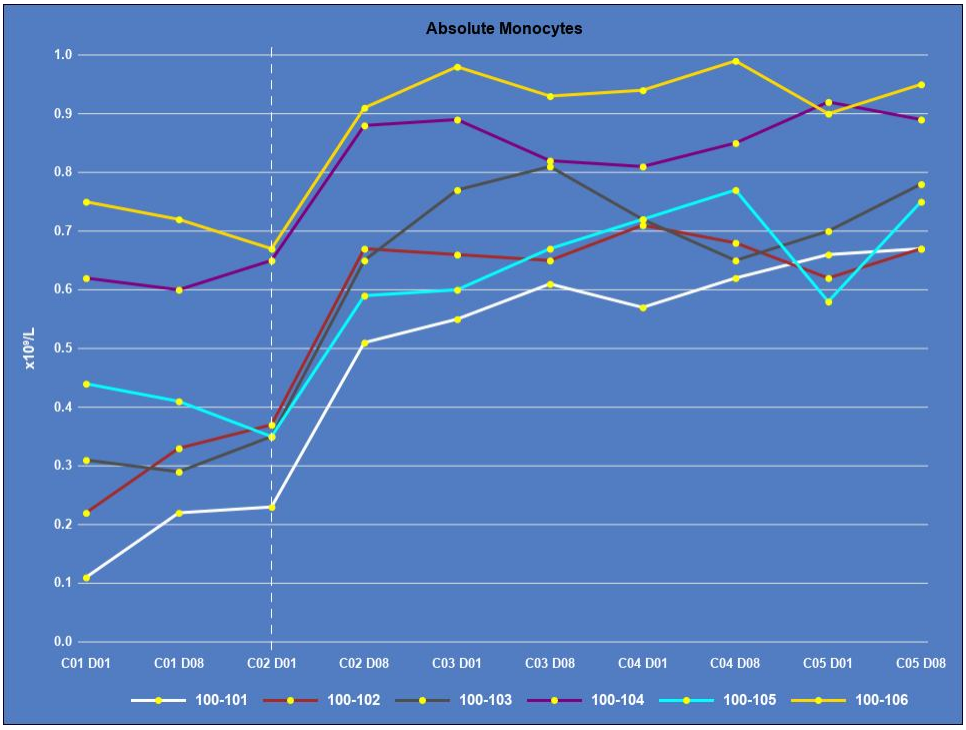


Figure 12. STYLEATTRS DATACONTRASTCOLORS= Option Changes the Line Color and SERIES MARKERATTRS= Option Changes Each Symbol to "circlefilled" and Yellow

SERIES STATEMENT USING OPTIONS DATALABEL= AND DATALABELATTRS=

DATALABEL displays a label for each datapoint. DATALABELATTRS assigns attributes such as color, weight etc. to the labels. We chose to assign DATALABEL = TI to the last datapoint of each plotline to indicate name of the cohort a particular subject was enrolled. The variable, TI, has values such as 'A, B'.

Also note the change in GROUP = COHSUB option in the SERIES statement to make the cohort assignments more visible in the legend. We used DATALABELATTRS = (color = white weight = bold). Full code of the source data, lab3, which includes the derivation of two new variables, TI and COHSUB, are included in Appendix II.

```
proc sgplot data = lab3 nowall noborder;
   series x = visit y = result / group = cohsub
         lineattrs = (thickness = 3 pattern = solid)
         markers markerattrs = (symbol = circlefilled color = yellow)
         datalabel = ti datalabelattrs = (color = white weight = bold);
   styleattrs datacontrastcolors = (white brown charcoal purple cyan gold)
         backcolor = CX527DC2;
   refline 'C02 D01'/ axis = x lineattrs = (color = white pattern = dash);
   xaxis display = (noline nolabel noticks) label = "VISIT"
         valueattrs = (color = white weight = bold)
         labelattrs = (color = white weight = bold);
   yaxis display = (noline noticks) grid values = (0 to 1 by 0.1)
         valueattrs = (color = white weight = bold)
         labelattrs = (color = white weight = bold)
         label = "x10(*ESC*){unicode '2079'x}/L";
   keylegend / noborder noopaque title=" "
         valueattrs = (color = white size = 10 weight = bold);
run;
```

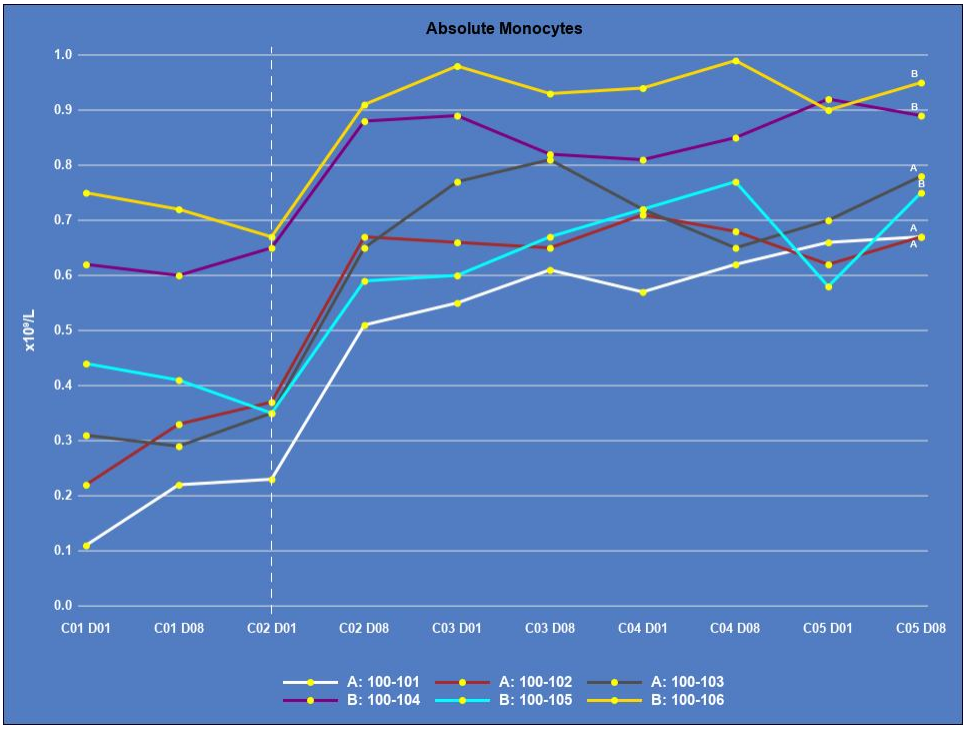


Figure 13. DATALABEL= Displays a Label for each Datapoint and DATALABELATTRS= Changes the Datapoints to Circlefilled and Yellow

AMAZING SGLOT – COHORT AVERAGES ADDED WITH INSET STATEMENT

The final plot includes all the PROC SGPLOT statements with specific options that creates an amazing display of laboratory values by visit and cohort to be used by upper management to look for trends in these values or can be used as a template for figures used in conference presentations.

The INSET statement adds the averages to the bottom right corner of Figure 14. You can change the position with POSITION= option, and attributes such as color (COLOR=), size (SIZE=), font (FAMILY=) and value alignment (VALUEALIGN=). For the averages, PROC MEANS was used to get the mean value by cohort (see Appendix III for SAS code). These values were placed in macro variables: &COHORTA and &COHORTB and used in the INSET statement ("Cohort A" = "&cohorta" "Cohort B" = "&cohortb"). SAS Institute 2022b has additional code and description of the attributes that can be used for INSETs.

```
proc sgplot data = lab3 nowall noborder;
   series x = visit y = result / group = cohsub
          lineattrs = (thickness = 3 pattern = solid)
          markers markerattrs = (symbol = circlefilled color = yellow)
          datalabel = ti datalabelattrs = (color = white weight = bold);
   styleattrs datacontrastcolors = (white brown charcoal purple cyan gold)
         backcolor = CX527DC2;
   refline 'C02 D01'/ axis = x lineattrs = (color = white pattern = dash);
   inset ("Cohort A" = "&cohorta" "Cohort B" = "&cohortb")
         / title = "Averages" position = bottomright
         textattrs = (color = white size = 10 Family=Arial)
         titleattrs = (color = white size = 10 Family=Arial)
         valuealign = left;
   xaxis display = (noline nolabel noticks) label = "VISIT"
         valueattrs = (color = white weight = bold)
         labelattrs = (color = white weight = bold);
```

```
    yaxis display = (noline noticks) grid values = (0 to 1 by 0.1)
          valueattrs = (color = white weight = bold)
          labelattrs = (color = white weight = bold)
          label = "x10(*ESC*){unicode '2079'x}/L";
    keylegend / noborder noopaque title=" "
          valueattrs = (color = white size = 10 weight = bold);
run;
```

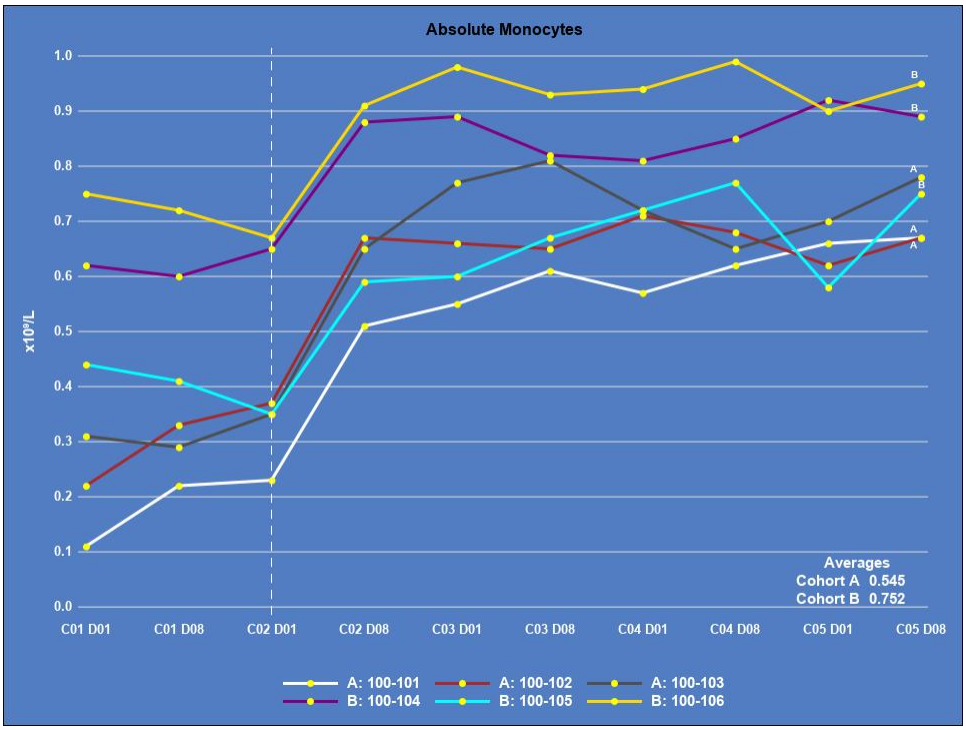


Figure 14. Final Amazing SGPLOT with Inset Statement for Including Cohort Averages

CONCLUSION

The SGPLOT procedure is a powerful and efficient tool to create exceptionally tailored figures. In this paper, each statement and multiple options within the procedure is demonstrated to take a built-upon approach to figure generation for laboratory data.

To produce the last most amazing, eye-catching, yet minimalistic figure, some of the important tips and tricks used (thanks to the exceptional capabilities of SGPLOT), were the following: grouping by cohort; adding the cohort label at the end of each line; background, line, and marker color selection; no walls or borders; grid selection; removing the title of the legend and the x-axis; including a reference line; value attribute selection such as using the color white and arial font; and finally adding averages to the bottom right-hand corner. Any beginner programmer can implement these suggestions to produce similar high-quality plots.

As shown, the SGPLOT is part of the Output Delivery System that not only allows the creation of high-quality graphics but also creates an easy method to output multiple formats.

The ideas presented can be used for different types of continuous data such as vital signs, PK, and ECG. With SGPLOT, you can also create regression plots, histograms, heatmaps, horizontal boxplots, hi-lo plots and more that are not presented here. Consult the recommended reading section to find more information of the various plots that SGPLOT can produce.

REFERENCES

SAS Institute Inc. 2022a. “PROC SGPLOT SERIES Statement.” Accessed March 27, 2022. SAS Help Center: Syntax: PROC SGPLOT SERIES Statement.

SAS Institute Inc. 2022b “Example: Panel with Insets.” Accessed March 29, 2022. SAS Help Center: Syntax: PROC SGPANEL INSET Statement

Slaughter, Susan J. and Delwiche, Lora D., 2015. “Paper 2441-2015 Graphing Made Easy with SGPLOT and SGPANEL Procedures”. *Proceedings of the SAS Global Forum 2015 Conference,* Dallas, TX: The SAS Institute, Inc., Cary, NC. Available at https://support.sas.com/resources/papers/proceedings15/2441-2015.pdf

ACKNOWLEDGMENTS

The authors would like to thank Eric Song, Ganesh Gopal and Syamala Schoemperlen for their ongoing support and encouragement in conference participation.

RECOMMENDED READING

- *SAS 9.4 ODS Graphics: Procedures Guide, Sixth Edition (SAS Institute, 2016)*

CONTACT INFORMATION

Your comments and questions are valued and encouraged. Contact the authors at:

Aakar Shah
Neoleukin Therapeutics
ashah@neoleukin.com
www.neoleukin.com

Tracy Sherman
Ephicacy Consulting Group, Inc.
tracy.sherman@ephicacy.com
www.ephicacy.com

APPENDIX I SOURCE DATA: LAB

Dummy laboratory data for Figures 1-12 was created as follows:

```
data lab;
   infile datalines delimiter=',';
   input Subject $ Cohort $ Visitnum  Visit $ Test $ Result Unit $ ;
   datalines;
   100-101, A, 1, C01 D01, Monocyte, 0.11, 10^3/uL
   100-101, A, 2, C01 D08, Monocyte, 0.22, 10^3/uL
   100-101, A, 3, C02 D01, Monocyte, 0.23, 10^3/uL
   100-101, A, 4, C02 D08, Monocyte, 0.51, 10^3/uL
   100-101, A, 5, C03 D01, Monocyte, 0.55, 10^3/uL
   100-101, A, 6, C03 D08, Monocyte, 0.61, 10^3/uL
   100-101, A, 7, C04 D01, Monocyte, 0.57, 10^3/uL
   100-101, A, 8, C04 D08, Monocyte, 0.62, 10^3/uL
   100-101, A, 9, C05 D01, Monocyte, 0.66, 10^3/uL
   100-101, A, 10,C05 D08, Monocyte, 0.67, 10^3/uL
   100-102, A, 1, C01 D01, Monocyte, 0.22, 10^3/uL
   100-102, A, 2, C01 D08, Monocyte, 0.33, 10^3/uL
   100-102, A, 3, C02 D01, Monocyte, 0.37, 10^3/uL
   100-102, A, 4, C02 D08, Monocyte, 0.67, 10^3/uL
   100-102, A, 5, C03 D01, Monocyte, 0.66, 10^3/uL
   100-102, A, 6, C03 D08, Monocyte, 0.65, 10^3/uL
   100-102, A, 7, C04 D01, Monocyte, 0.71, 10^3/uL
   100-102, A, 8, C04 D08, Monocyte, 0.68, 10^3/uL
   100-102, A, 9, C05 D01, Monocyte, 0.62, 10^3/uL
   100-102, A, 10,C05 D08, Monocyte, 0.67, 10^3/uL
   100-103, A, 1, C01 D01, Monocyte, 0.31, 10^3/uL
   100-103, A, 2, C01 D08, Monocyte, 0.29, 10^3/uL
   100-103, A, 3, C02 D01, Monocyte, 0.35, 10^3/uL
   100-103, A, 4, C02 D08, Monocyte, 0.65, 10^3/uL
   100-103, A, 5, C03 D01, Monocyte, 0.77, 10^3/uL
   100-103, A, 6, C03 D08, Monocyte, 0.81, 10^3/uL
   100-103, A, 7, C04 D01, Monocyte, 0.72, 10^3/uL
   100-103, A, 8, C04 D08, Monocyte, 0.65, 10^3/uL
   100-103, A, 9, C05 D01, Monocyte, 0.70, 10^3/uL
   100-103, A, 10,C05 D08, Monocyte, 0.78, 10^3/uL
   100-104, B, 1, C01 D01, Monocyte, 0.62, 10^3/uL
   100-104, B, 2, C01 D08, Monocyte, 0.60, 10^3/uL
   100-104, B, 3, C02 D01, Monocyte, 0.65, 10^3/uL
   100-104, B, 4, C02 D08, Monocyte, 0.88, 10^3/uL
   100-104, B, 5, C03 D01, Monocyte, 0.89, 10^3/uL
   100-104, B, 6, C03 D08, Monocyte, 0.82, 10^3/uL
   100-104, B, 7, C04 D01, Monocyte, 0.81, 10^3/uL
   100-104, B, 8, C04 D08, Monocyte, 0.85, 10^3/uL
   100-104, B, 9, C05 D01, Monocyte, 0.92, 10^3/uL
   100-104, B, 10,C05 D08, Monocyte, 0.89, 10^3/uL
   100-105, B, 1, C01 D01, Monocyte, 0.44, 10^3/uL
   100-105, B, 2, C01 D08, Monocyte, 0.41, 10^3/uL
   100-105, B, 3, C02 D01, Monocyte, 0.35, 10^3/uL
   100-105, B, 4, C02 D08, Monocyte, 0.59, 10^3/uL
   100-105, B, 5, C03 D01, Monocyte, 0.60, 10^3/uL
   100-105, B, 6, C03 D08, Monocyte, 0.67, 10^3/uL
   100-105, B, 7, C04 D01, Monocyte, 0.72, 10^3/uL
   100-105, B, 8, C04 D08, Monocyte, 0.77, 10^3/uL
   100-105, B, 9, C05 D01, Monocyte, 0.58, 10^3/uL
```

```
      100-105, B, 10,C05 D08, Monocyte, 0.75, 10^3/uL
      100-106, B, 1, C01 D01, Monocyte, 0.75, 10^3/uL
      100-106, B, 2, C01 D08, Monocyte, 0.72, 10^3/uL
      100-106, B, 3, C02 D01, Monocyte, 0.67, 10^3/uL
      100-106, B, 4, C02 D08, Monocyte, 0.91, 10^3/uL
      100-106, B, 5, C03 D01, Monocyte, 0.98, 10^3/uL
      100-106, B, 6, C03 D08, Monocyte, 0.93, 10^3/uL
      100-106, B, 7, C04 D01, Monocyte, 0.94, 10^3/uL
      100-106, B, 8, C04 D08, Monocyte, 0.99, 10^3/uL
      100-106, B, 9, C05 D01, Monocyte, 0.90, 10^3/uL
      100-106, B, 10,C05 D08, Monocyte, 0.95, 10^3/uL
run;
```

APPENDIX II – SOURCE DATA: LAB3

For Figures 13 and 14, two variables, COHSUB (Cohort: Subject Number) and TI was added to the dummy data generated in Appendix I as follows:

```
data lab1;
   set lab;
   cohsub = strip(cohort)||": "||strip(subject);
run;

proc sort data = lab1 out = lab2;
   by cohsub test visitnum;
run;

data lab3;
   set lab2;
   by cohsub test visitnum;
   if last.test then ti = cohort;
run;
```

APPENDIX III – PROC MEANS AND MACRO VARIABLE CREATION FOR INSET STATEMENT

For the averages, PROC MEANS was used to get the mean value by cohort. These values were placed in macro variables, &COHORTA and &COHORTB and used in the INSET statement (“Cohort A” = “&cohorta” “Cohort B” = ”&cohortb”).

```
proc means data = lab3;
  var result;
  by cohort;
  output mean = mn out = mnlab3;
run;

data _null_;
  set mnlab3;
  mean = strip(put(mn,8.3));
  if cohort = 'A' then call symput ('cohorta', mean);
  if cohort = 'B' then call symput ('cohortb', mean);
run;

%put Cohort A = &cohorta Cohort B = &cohortb;
```