PharmaSUG 2021 - Paper EP-123

Intelligent Axis Scaling in SAS plots: %IAS

Sidian Liu, Genentech Inc.;

Toshio Kimura, Regeneron Pharmaceuticals Inc.

ABSTRACT

When creating graphs to visualize statistical analysis, the intuitiveness of the axis boundary and interval values can considerably impact the ease of interpretation. The default SAS plotting algorithm focuses primarily on structure where the figure is centered and covers a substantial proportion of the canvas. However, the SAS default approach may compromise interpretability.

We generated figures from simulated data to compare SAS defaults and the proposed algorithm. In some cases, the SAS default axis values were outside of the axis labels being printed: for example, when the actual values range from 0.7 to 4.1, the printed tick marks by the SAS default algorithm were 1 to 4 whereas a more preferable range would have been 0 to 5.

We propose an alternative intelligent axis scaling algorithm %IAS, implemented through a SAS macro that addresses this concern by focusing on the interpretability of the Y-axis. The algorithm has 3 main properties: 1) Boundary and interval values are multiples of power of 10; 2) Number of intervals is between 5 to 10; and 3) Figure is still centered and covers a substantial proportion of the canvas. The algorithm places a high emphasis on selecting easily interpretable critical values such as 0 and multiples of the power of 10 as the min/max boundary values and more logical intervals as reference tick marks. The values selected by this alternative algorithm as opposed to the SAS defaults will facilitate the interpretation of the graphs being generated.

INTRODUCTION

Often times when directly using the SAS SGPLOT procedure, we find that the Y-axis values are hard to interpret because the default tick marks may not cover the full range of the graph and the intervals can be fractional numbers. Although the SGPLOT procedure allows the user to input the axis limit and interval values, this is not feasible if we do not know the ranges beforehand. This is particularly common when producing a large number of figures.

For example, Figure 1A is generated with simulated data using SGPLOT without any user input. The limits of the data go above 4 and below 1, but the tick marks do not provide a good reference on how much it goes beyond this axis range. Comparatively, Figure 1B would be a better plot with Y-axis ranging from 0 to 5. It is easier on the eyes and more importantly, provides the reviewers a good estimate of the overall range since 0 and 5 are more intuitive to understand compared to 1 and 4.

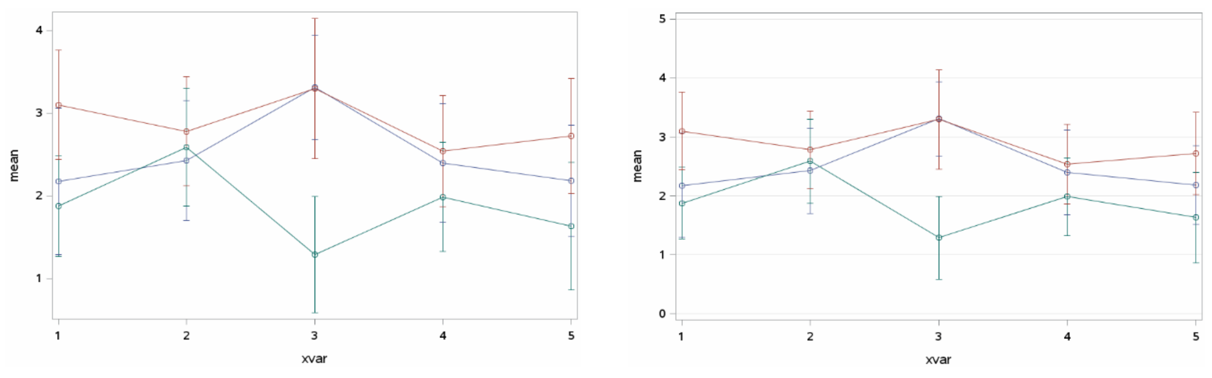


Figure 1A: Using PROC SGPLOT default (no user input) **Figure 1B: Using %IAS**

Figure 1: Line Plot Displaying Means with Standard Error, Example 1

The lack of intuitiveness of Y-axis scale can also lead to longer review time especially when we generate a large number of graphs, since it's difficult to read the range values and come to quick conclusions for each graph. Therefore, we would like to create a smart and automated plotting algorithm that finds the optimal axis tick marks and aids interpretability of the Y-axis values.

METHODOLOGY

With the above goal in mind, we designed a macro program called %IAS in SAS that determines the appropriate values for Y-axis tick mark based on the range of the datasets to be plotted. The derived min, max and interval values are then fed into SGPLOT for batch plots generation. Below are the rules we created.

STEP 1: SELECTING BOUNDARY VALUES

As the first step, the algorithm finds the best-fit upper and lower limit values by starting at an approximately large range, then iteratively reducing the range to the nearest multiples of 5 and 10. This makes sure the range of the dataset is always within the upper and lower tick marks. In addition, the boundary values are rounded based on the most intuitive numbers: 5 and 10.

a. Obtain the minimum and maximum values of the dataset (this should all values being plotted including the error bars) and scale the values to be between [-10, 10] by dividing the minimum and maximum values by a base 10 scaling factor. The scaling factor being used will be denoted as *scaling_factor* (and will be used later in Step 4) and the minimum and maximum values will be denoted as *min, max*. For example, if the minimum and maximum values were 125 and 575. The values would be scaled to *min* = 1.25 and *max* = 5.75 and the scaling factor would be 100.

b. Round down *min*, round up *max* to the nearest integer, denote as *lower, upper.*

c. Ensure that the range of the observed data covers a minimum proportion (defined by Proportion 1) of the canvas relative to the range of the upper and lower boundaries. If the actual range of the dataset accounts for less than a predefined proportion of the boundary values of the canvas as calculated and evaluated by $\frac{range\ of\ the\ observed\ data}{range\ of\ the\ boundaries} = \frac{max-min}{upper-lower}$ <= *proportion1*, then *max* and *min* are rounded up or down to the nearest 0.5, 0.1, 0.05, 0.01… until the proportion1 criteria is satisfied. Note that the resulting *upper* and *lower* values will always cover the range of *min, max.*

   Below are two examples. In the first example, 4.85 and 5.78 are first rounded to the near integers 4 and 6 as explained in Step 1b. The ratio of $\frac{5.78-4.85}{6-4}$ = calculated which is 0.465 and is evaluated against Proportion1 of 0.6. Since 0.475 < 0.6, the 4.85 and 5.78 are then rounded to the nearest 0.5 which would be 4.5 and 6. The ratio of $\frac{5.78-4.85}{6-4}$ is calculated which is 0.62 and is evaluated again against Proportion 1 of 0.6. Since the ratio (0.62) is greater than or equal to Proportion 1 of 0.6, the algorithm is completed, and the lower and upper boundaries are set to 4.5 and 6 and the divisor to 0.5. In the second example, the max and min values are 4.85 and 5.68. The initial boundaries rounded to the nearest integer results in a ratio of 0.415 which is less than Proportion 1 of 0.6. When the values are rounded to the nearest 0.5, the ratio is 0.533 which is still less than Proportion 1 of 0.6. The min and max values are rounded to the near 0.1 which results in 4.8 and 5.7 at which point the ratio is 0.922 which is greater than Proportion1 of 0.6. Therefore, the algorithm is completed, and the lower and upper boundaries are set to 4.8 and 5.7

and the divisor to 0.1.

		Divisor			
	Examples	1	0.5	0.1	…
lower	4.85	4	4.5		
upper	5.78	6	6		
	$\frac{max - min}{upper - lower}$	$\frac{5.78-4.85}{6-4}$ = 0.465	$\frac{5.78-4.85}{6-4.5}$ = 0.62		
	$\frac{max - min}{upper - lower}$ >= 0.6?	N	Y		
lower	4.85	4	4.5	4.8	
upper	5.68	6	6	5.7	
	$\frac{max - min}{upper - lower}$	$\frac{5.68-4.85}{6-4}$ = 0.415	$\frac{5.68-4.85}{6-4.5}$ = 0.553	$\frac{5.68-4.85}{5.7-4.8}$ = 0.922	
	$\frac{max - min}{upper - lower}$ >= 0.6?	N	N	Y	

As described in the examples, the last rounding divisor number used for the boundary values is denoted as *divisor*. (This variable will be used later in Step 3a). *Proportion1* is a tunable parameter between 0 and 1 that determines the minimum proportion of space the range of the data will fill with respect to the range of the Y-axis. If the value is larger, the boundaries will be shrunk more in order to satisfy the criteria, and the range will fill in larger space of the canvas. If the value is smaller, the criteria is more likely to be satisfied and the range will fill in smaller space. The default value of *Proportion1* is set to 0.6.

STEP 2: FINE-TUNING BOUNDARY VALUES

After obtaining the upper and lower limits using the heuristic approach, we further evaluate and potentially adjust them to more intuitive anchor points, such as 0, 5, 10. Specifically, if they are close to these anchor points, and the current range takes up a reasonable proportion of the canvas, we enlarge the boundaries to a further step as long as it does not compromise aesthetics. This proportion is tunable parameter between 0 and 1 and referred to as *proportion2*. *Proportion 2* determines whether or not the upper or lower limit is close enough to one of the defined anchor points. If the value is larger, it requires the upper and lower limit to be closer to the anchor point before relaxation, whereas if the value is smaller, they don't need to be as close in order to be approximated to the anchor points. The larger the value, the less likely this relaxation is performed. The default value of Proportion 2 is set to 0.6. This step maximizes the readability of the tick marks.

a. Apply the following rules

- If 0 < *lower* <= 2 and $\frac{upper-lower}{10}$ >= *proportion2*, then assign *lower* to 0;
- If 0 < *lower* <= 1 and $\frac{upper-lower}{5}$ >= *proportion2*, then assign *lower* to 0;
- If *upper* >= 8 and $\frac{upper-lower}{10}$ >= *proportion2*, then assign *upper* to 10;
- If 4 <= *upper* < 5 and $\frac{upper-lower}{5}$ >= *proportion2*, then assign *upper* to 5;
- If -10 < *lower* <= -8 and $\frac{upper-lower}{10}$ >= *proportion2*, then assign *lower* to -10;
- If -5 < *lower* <= -4 and $\frac{upper-lower}{5}$ >= *proportion2*, then assign *lower* to -5;
- If 0 > *upper* >= -2 and $\frac{upper-lower}{10}$ >= *proportion2*, assign *upper* to 0;
- If -6 <= *upper* < -5 and $\frac{upper-lower}{5}$ >= *proportion2*, then assign *upper* to -5;

STEP 3: SELECTING INTERVAL VALUES

After the upper and lower limits are determined, we then define the number of tick marks and the interval values. To enhance readability, intervals are always multiples of 5 and 10, and the number of tick marks should be neither too sparse nor too dense.

b. Starting with the resulting *divisor* used in Step 1c as *initial_interval* (as noted in Step 1c, *divisor* and hence *initial_interval* can take the values of 1, 0.5, 0.1, 0.05, 0.01 etc.), calculate the number of tick marks as $\frac{upper-lower}{initial\_interval}$ and denote this value as *number_of_intervals*.

c. Apply the following rules to set the number of intervals to between 5 and 10:

- If *number_of_intervals* = 1, set *interval_width* to $\frac{initial\_interval}{10}$ and multiply the *number_of_intervals* by 10;
- If *number_of_intervals* = 2, set *interval_width* to $\frac{initial\_interval}{5}$ and multiply the *number_of_intervals* by 5;
- If *number_of_intervals* = 3 or 4, set *interval_width* to $\frac{initial\_interval}{2}$ and multiply the *number_of_intervals* by 2;
- If *number_of_intervals* >10, set *interval_width* to *intitial_interval * 2* and divide the *number_of_intervals* by 2;
- If none of the conditions are satisfied, set *interval_width* to *intitial_interval*

STEP 4: TRANSFORM VALUES BACK TO THE ORIGINAL SCALE

Since the values are still in the scaled -10 to 10 range. The values will need to be converted back to the original scale. The *scaling_factor* from Step 1a will be used to perform this calculation.

a. Transform the *lower, upper* and *interval* back to their original scale using the *scaling_factor* if the original values were not within [-10,10].

EXAMPLE: STEP-BY-STEP IMPLEMENTATION

This section provides two examples of how the algorithm is implemented. The figure being plotted will display means with standard error bars over time.

EXAMPLE 1

In the first example (Figure 2), the minimum and maximum values of the data are 4.9 and 5.85. In addition, the tunable parameters are defined as *proportion1* = 0.6 and *proportion2* = 0.6.

Step 1: Selecting the Boundary Values

Step 1a: The first step is to scale the values to be between [-10,10]. Since the values already fall within this range, the values do not need to be scaled. The *scaling_factor* in this case would be 1, and *min* = 4.9 and *max* = 5.85.

Step 1b: Round down the *min* and round up the *max* to the nearest integer and denote these values as *lower* and *upper*, thus *lower* = 4 and *upper* = 6.

Step 1c: Ensure that the graph fills at least 60% of the canvas (defined by *proportion1*=0.6). Perform this evaluation by the following calculation: $\frac{max-min}{upper-lower}$ = $\frac{5.85-4.9}{6-4}$ = 0.475. Since 0.475 <= 0.6, the Proportion 1 criteria has not been met, and the range of the boundaries must be adjusted. This perform the second rounding by reducing the *divisor* from 1 to 0.5. As a result, (4,6) is rounded down and up to the nearest number divisible by 0.5 to (4.5, 6), and the Proportion 1 criteria is re-evaluated: $\frac{5.85-4.9}{6-4.5} = 0.6333$. Since 0.6333 >= 0.6, the Proportion 1 criteria has been met, and we can stop the iteration, obtain the lower, upper limit as (4.5, 6) with a *divisor* = 0.5. Note that if the proportion criteria were still not satisfied, the algorithm will continue the rounding using the smaller divisor: 0.1, 0.05, 0.01 until the proportion is larger than 60%.

Step 2: Fine-tuning the boundary values

Step 2a: The lower and upper boundaries are not close to any of the anchor point values. For example, the lower boundary of 4.5 is not between 0 and 2. The upper boundary value of 6 is not greater or equal to 8. Since none of rules are applicable, the lower and upper boundaries do not need to be re-assigned to a different value, and, thus, this step is skipped.

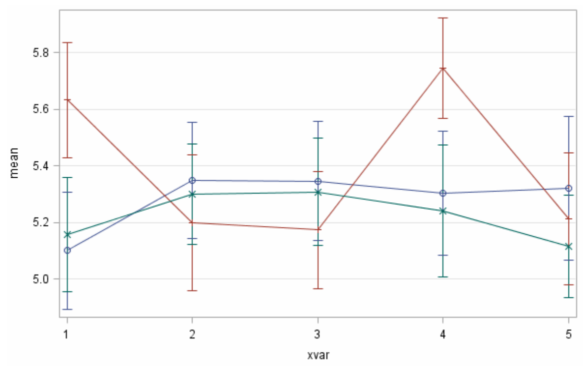
Step 3: Selecting the number of tick marks and interval values

Step 3a: Starting with the *divisor* value defined in Step 1c where *divisor* = 0.5, we obtain the *number_of_intervals* as $\frac{6-4.5}{0.5} = 3$. This results in only 4 tick marks which is would result in tick marks sparsely distributed along the Y-axis.

Step 3b: Following the rules for when the *number_of_intervals* = 3, we divide the *initial_interval* (which will initially be the value of *divisor*) by 2: $\frac{initial\_interval}{2} = \frac{0.5}{2} = 0.25$. The *number_of_intervals* will be multiplied by 2, so that there are now 6 intervals.

Step 4: Transform values back to the original scale

Step 4a: Since the original *min* and *max* values were already within [-10,10], a scaling factor was not used, and, therefore, the values do not need to transformed back to the original scale. Figure 2 below shows the SGPLOT default Y-axis in Figure 2A and the %IAS Y-axis in Figure 2B.



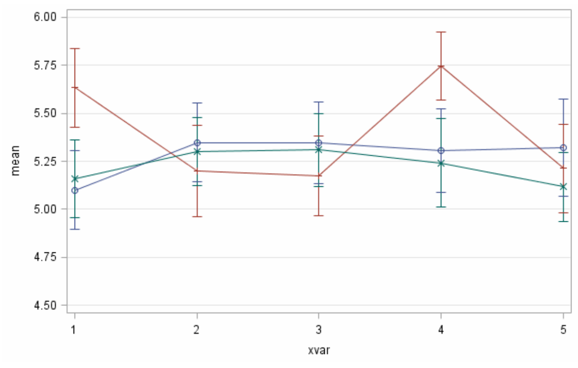


Figure 2A: Using PROC SGPLOT default (no user input) **Figure 2B: Using %IAS**

Figure 2: Means with Standard Error Plot, Example 2

EXAMPLE 2

The second example below (Figure 3) shows some other features of the algorithm. The min and max values of the data are -398 and 307. Same as Example 1, the tunable parameters are defined as proportion1 = 0.6 and proportion2 = 0.6.

Step 1: Selecting the Boundary Values

Step 1a: The first step is to scale the values to be between [-10,10]. Since the values of (-398, 307) fall outside of the [-10,10] range, the value must be scaled by a factor of 100. Therefore, the *scaling_factor* in this case would be 100, and *min* = -3.98 and *max* = 3.07.

Step 1b: Round down the *min* and round up the *max* to the nearest integer and denote these values as *lower* and *upper*, thus *lower* = -4 and *upper* = 4.

Step 1c: Ensure that the graph fills at least 60% of the canvas (defined by *proportion1*=0.6). Perform this evaluation by the following calculation: $\frac{max-min}{upper-lower} = \frac{3.07-(-3.98)}{4-(-4)}$ = 0.88. Since 0.88 >= 0.6, the Proportion 1 criteria has been met. No further adjustments to the boundaries are necessary. The selected values are as follows: *lower* = -4, *upper* = 4 and *divisor* = 1.

Step 2: Fine-tuning the boundary values

Step 2a: The lower and upper boundaries need to evaluated with respect to various anchor points. Both the upper and lower boundaries are near key anchor points; therefore, they will need to be adjusted.

Lower: The lower boundary is near -5, and, therefore, the following rule applies:

- If -5 < *lower* <= -4 and $\frac{upper-lower}{5}$ >= *proportion2*, then assign *lower* to -5;

Upper: The upper boundary is near 5, and, therefore, the following rule applies:

- If 4 <= *upper* < 5 and $\frac{upper-lower}{5}$ >= *proportion2*, then assign *upper* to 5;

Based on the above rules, the lower and upper boundaries are fine-tuned and adjusted so that *lower* = -5 and *upper* = 5.

Step 3: Selecting the number of tick marks and interval values

Step 3a: Starting with the *divisor* value defined in Step 1c where *divisor* = 1 and setting that as the *initial_interval*, we obtain the *number_of_intervals* as $\frac{upper-lower}{initial\_interval} = \frac{5-(-5)}{1} = 10.$

Step 3b: Since 10 falls within the set of acceptable number of intervals (between 5 and 10) no further adjustments to the *interval_width* or *number_of_intervals* are necessary. The *number_of_intervals* = 10 *and interval_width* = 1.

Step 4: Transform values back to the original scale

Step 4a: Since the original minimum and maximum values were not initially within [-10,10], a *scaling_factor* of 100 was used (as defined in Step 1a). Therefore, the *lower* and *upper* as well as the *interval_width* will need to be scaled back to the original scale. The *lower* and *upper* boundary values as well as the *interval_width* value will be multiplied by the *scaling_factor*:

lower = -5 * 100 = -500

upper = 5 * 100 = 500

interval_width = 1 * 100

Figure 3 below shows the SGPLOT default Y-axis in 3A and the %IAS Y-axis in 3B.

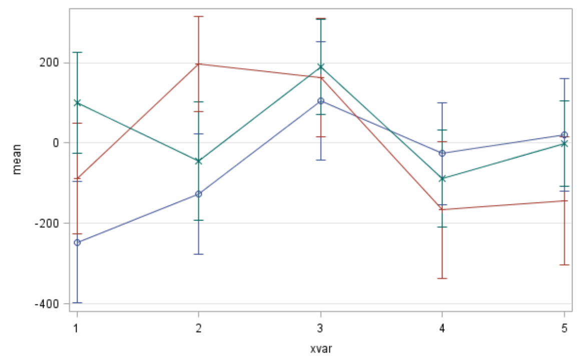


Figure 3A: Using PROC SGPLOT default (no user input)

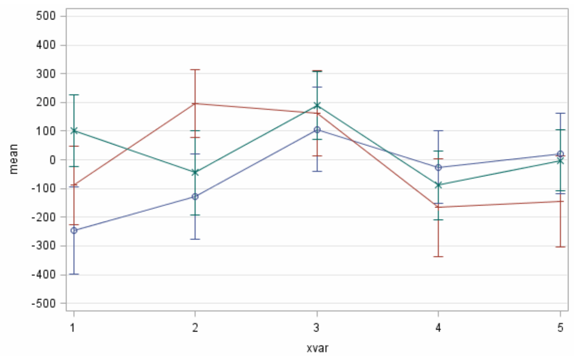


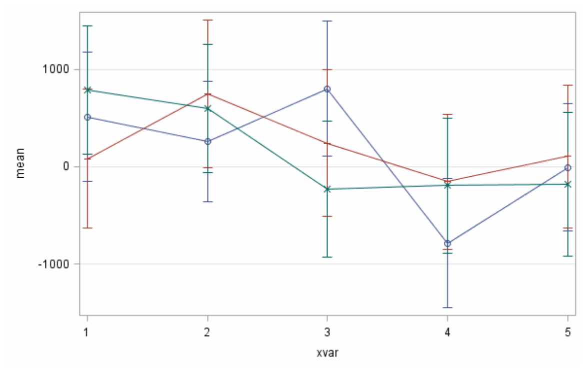
Figure 3B: Using %IAS

Figure 3: Means with Standard Error Plot, Example 3

DISCUSSION

The %IAS macro solves several problems the default SAS plot has, including non-intuitive tick mark values, upper and lower limits out of the maximum and minimum tick marks, and sparsity. By choosing the interval as a multiple of powers of 5 and 10, including common anchor points, and controlling the number of intervals, it allows easy interpretation and understanding of graphs which helps to improve efficiency significantly when there is a large number of graphs to review.

We selected 2 example plots to compare the difference between %IAS algorithm and SAS default SGPLOT. In Figure 4A and 4B, we can see that in the default plot the values go beyond the biggest and smallest tick marks, making it difficult to interpret the range, whereas in the %IAS plot, it is clear that the data ranges from -2000 to 2000. Another difference is that SAS default plot has 4 intervals and the tick marks are by 1000. %IAS plot has 8 intervals and the interval value is 500. More intervals help to see the granular level of details better. In this case where values are centered around -1000 to 1000, the addition of 500 and -500 marks allows you to approximate the values of the means and standard errors more easily.



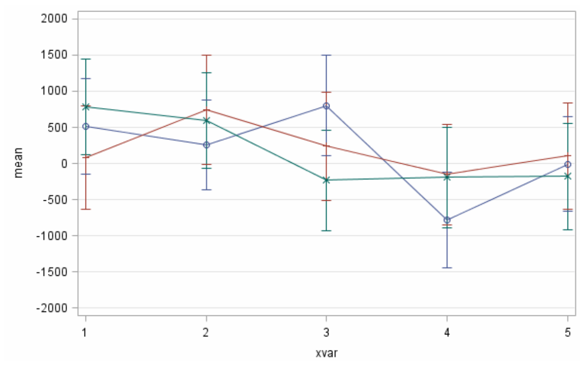
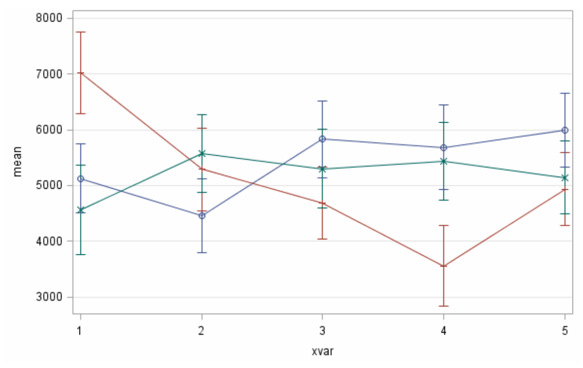


Figure 4A: Using PROC SGPLOT default (no user input) **Figure 4B: Using %IAS**

Figure 4: Means with Standard Error Plot, Example 4

The aim of the fine-tuning step is to allow even more intuitive tick marks to be displayed so that users can quickly interpret the range of the plots. However, the compromise is that the data can be compressed into a smaller range of the canvas with a lot of white space unused. For example, below are the comparison between the SAS default (Figure 5A) and %IAS (Figure 5B) outputs when the dataset satisfies the criterion in the fine-tuning step given the tunable parameter *proportion2=0.6*. From the original range (2000, 8000) the algorithm widens it to (0, 10000) so that 0 and 10000 which facilitate users to interpret the results better are included as tick marks. On the other hand, SAS does not consider these anchor points and the graph is displayed as a "zoomed-in" manner compare to %IAS. To balance interpretability and aesthetic, we designed proportion1 and proportion2 as user defined parameters with a default value = 0.6 for both. Higher values mean the graph should take up higher proportion of the canvas but common anchor points are less likely to appear because of the more stringent criteria, lower values mean the tick marks are more intuitive to read, but the graph may be compressed and less easy on eyes. By adjusting both parameters, users can select their own proportions based on preference.



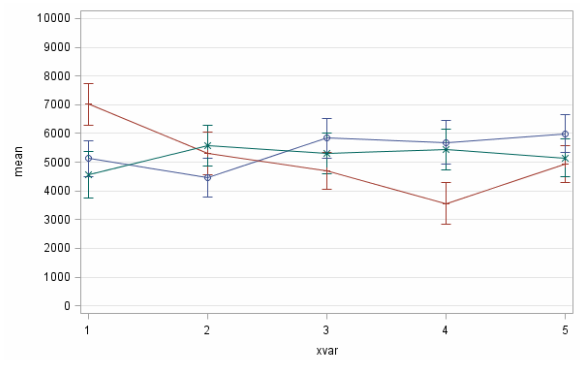


Figure 5A: Using PROC SGPLOT default (no user input) **Figure 5B: Using %IAS**

Figure 5: Means with Standard Error Plot, Example 5

CONCLUSION

In conclusion, %IAS is an alternative to SAS default plot that accommodates both functionality and readability of plots. With %IAS, the figures will utilize more intuitive to readable ranges allowing for easier and faster interpretation. %IAS also gives users the flexibility to tune parameters in order to adjust the proportion of graph against canvas based on preferences to customize the look. Another advantage is its automation feature. When graphs are generated in batches and it’s hard to specify axis values manually for each graph, %IAS is able to help the user determine the optimal Y-axis automatically for all graphs in one step.

Contact Information:
liusidianplus@gmail.com