Paper SAS2180-2018

Advanced Graphs using Axis Tables

Sanjay Matange, SAS Institute Inc.

ABSTRACT

An important feature of graphs used for the analysis data or for clinical research is the inclusion of textual data in the graph, usually aligned with the x or y axis. The axis table statements available with the SGPLOT procedure make it easy to add such data to the graphs. Axis tables can also be used for creating custom axes or custom graphs. This presentation will describe how to use axis tables to create complex custom graphs.

INTRODUCTION

Analysis of the data for clinical research and many other domains is made easier by presenting the data in graphical form. This makes it easier to compare responses for different treatments or visualize the laboratory results over time. Such graphs often require the display of data in textual form aligned with a numerical or discrete data on the x or y axis. A popular example is the Survival Plot shown in Figure 1.

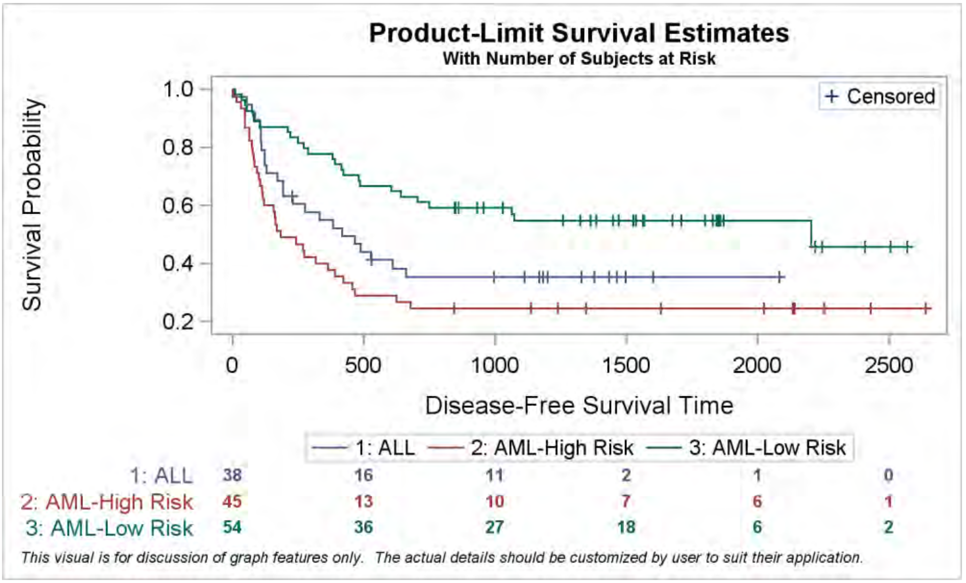


Figure 1 - Survival Plot.

The survival plot displays the survival probabilities over time by different groups, in this case, the type of Leukemia. An important feature of the graph is the display of the number of patients in the study by cancer type at the bottom of the graph. The numerical values are displayed at the corresponding locations along the time axis representing the number of subjects in the study at that time.

Traditionally, such information was inserted into the graph as a post process, after the creation of the plot. Some space is reserved at the bottom of the graph by the user and then an “annotation” process is used to display the tabular data below the legend. This requires the programmer to estimate the space needed for the table, and reserve this space. Then the user has to code annotation instructions for each text string to be displayed in the reserved space, including the position of the string along the x-axis and its position from the bottom of the graph. When the information is classified by type, the values need to be displayed as a stack of multiple rows, with the class value for each row displayed on the left of the y-axis.

In addition to scripting the instructions needed to display the values, this process does not scale well to changing data, as the space that needs to be reserved can change. This process becomes even more tenuous when adding textual data columns to the left or right of the data as we do for a forest plot.

With SAS 9.4, the new AXISTABLE statement was introduced in GTL to make such graphs easier to create. This new statement was specifically designed to support the need to insert textual data aligned with numeric or discrete x or y axes. Similarly, XAXISTABLE and YAXISTABLE statements were introduced in the SGPLOT and SGPANEL procedures for the same purpose. This paper will describe the key features of these new statements and how you can use them to create popular graphs like the survival plot or the forest plot.

In addition to creating these popular graphs, you can also use these features creatively to customize your graphs in other interesting ways. This paper will provide some such examples to show you other ways in which these statements. In this paper, we will discuss examples using the SGPLOT version of the statement for ease of use. You can also use this with GTL template based graphs.

THE AXISTABLE STATEMENT SYNTAX

The syntax for usage of the axis table statements is shown below. You can use the XAXISTABLE to place rows of textual data aligned with the x or x2 axis as in the survival plot shown in Figure 1. You can use the YAXISTABLE to place columns of textual data aligned with the y or y2 axis as in the forest plot shown later in this paper.

```
  xaxistable variable <variable> / <x=variable> <options>;
  yaxistable variable <variable> / <y=variable> <options>;
```

Normally, the XAXISTABLE or YAXISTABLE statement is used in conjunction with plot statements such as a SCATTER or VBAR or HBAR. The variable assigned to the primary axis role of such statements (such as X or CATEGORY) is used to align the values from the variables in the axis table statement(s). One or more variables can be specified to create stacks of rows or columns. Or, the table can be classified by a variable to display multiple rows or columns. You can also align the values on the axis using a different variable (of the same type), using the optional X= or Y= parameters as we will do in the survival plot example in Figure 5 and 6.

The axis table statements support a large number of data and appearance options. Some of the common options are listed below. Refer to the product documentation for all the details.

CLASS=var	Multiple rows or columns of data are displayed, one per unique value.
COLORGROUP=var	Text color is based on the unique values of this variable.
INDENTWEIGHT=num-var	Values are indented based on the value of this variable.
TEXTGROUP=var	Text attributes are displayed based on the unique values of this variable.

Table 2 - Data options

CLASSDISPLAY	The values per class level are displayed stacked or side by side.
INDENT=value	Specifies the indent value to be used as multiplier with INDENTWEIGHT
TEXTGROUPID=Id	Attribute id in associated attribute map for text attributes.
LOCATION	Location inside or outside the plot area.
POSITION	Position of the row (above or below), or column (left or right) of the plot area
LABEL | NOLABEL	Display or hide the variable name or label.
STAT	Aggregation statistics for repeated values for a specific category / group value

Table 3 - Appearance options

The INDENT and INDENTWEIGHT options can be used to indent the values in a column for the YAXISTABLE. TEXTGROUP and TEXTGROUPID with an attribute map can be used to apply different text attributes for the subgroups and values. These are useful when creating a subgrouped forest plot.

USING THE XAXISTABLE FOR SURVIVAL PLOT

Creating the popular survival plot becomes easier using the new XAXISTABLE statement. The code and results are shown below. In this case, I have extracted the data needed for the graph by using the ODS OUTPUT statement to save the data from the LIFETEST procedure step using the BMT data set.

```
ods output Survivalplot=data.SurvivalPlotData;
proc lifetest data=sashelp.BMT plots=survival(atrisk=0 to 2500 by 500);
   time T * Status(0);
   strata Group / test=logrank adjust=sidak;
run;
```

Figure 4 – Creating the data set for the graph

```
title 'Product-Limit Survival Estimates';
title2  h=0.8 'With Number of Subjects at Risk';
proc sgplot data=data.SurvivalPlotData;
  step x=time y=survival / group=stratum name='s';
  scatter x=time y=censored / markerattrs=(symbol=plus) name='c';
  scatter x=time y=censored / markerattrs=(symbol=plus) GROUP=stratum;
  xaxistable atrisk / x=tatrisk class=stratum colorgroup=stratum;
  keylegend 'c' / location=inside position=topright;
  keylegend 's' / linelength=20;
run;
```

Figure 5 – Creating the survival plot

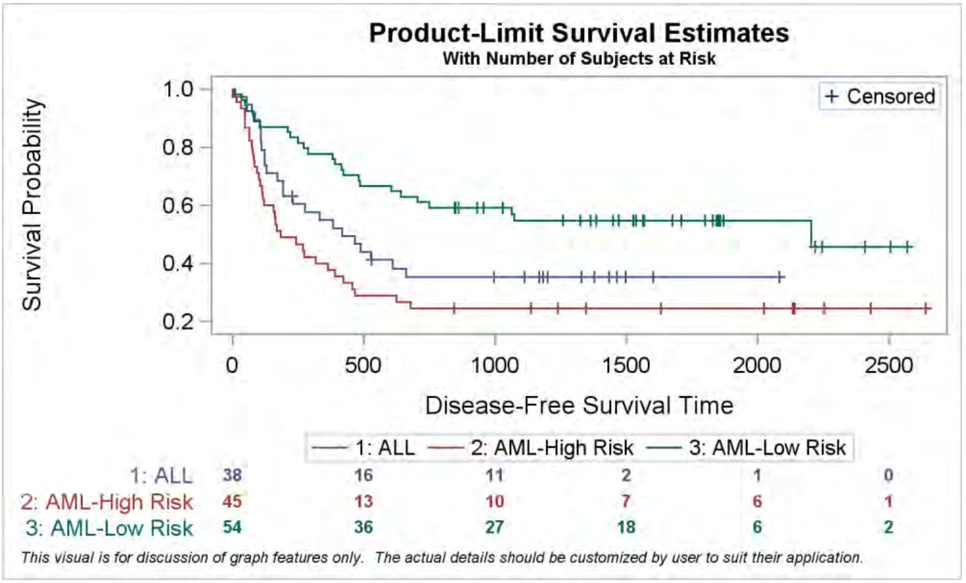


Figure 6 - Survival Plot with outer table of subjects at risk.

Note the following aspects of this program and graph.

- The STEP plot is used to display the survival probabilities by type over time.
- The XAXISTABLE statement is used to display the table of subjects at risk at the bottom.
- The default alignment variable for the axis table would be the "time" variable used with the STEP plot.
- In this case, the option X=tatrisk is used to align the values with this variable.
- The values are displayed from the "atrisk" column, classified by CLASS=stratum.
- Since the "stratum" variable has three unique values, three rows of values are displayed.
- Each row displays the value of the class variable on the left.
- The values and the class values in the table are colored using the COLORGROUP option.
- The appropriate amount of space is allocated to display the values and rows.
- The appropriate amount of space is allocated to the left of the graph for the row headers.
- The graph will adjust automatically for changing number of class values or string lengths.
- Note, the placement of the y-axis label farther away from the y-axis values. This is due to the placement of the table in a separate cell below the graph. See Figure 7 for an alternative layout.
- No annotation is needed to create this graph.

Another benefit of using the axis table for displaying the table of subjects at risk is that it can be easily moved to other places in the graph by changing one option. Setting LOCATION=inside, creates the same graph, with the table positioned above the axis, closer to the data as shown in Figure 7 below.

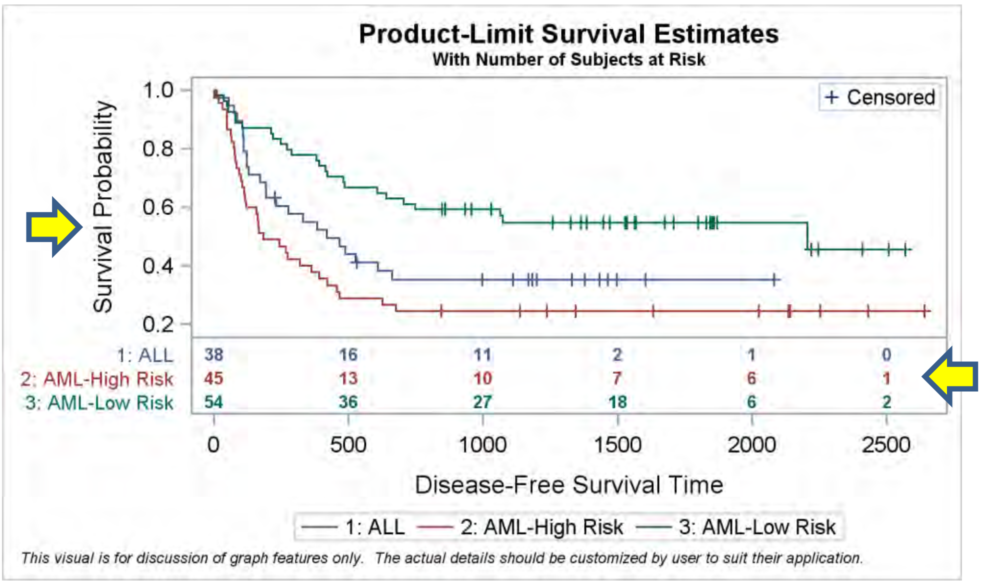


Figure 7 - Survival Plot with inner table of subjects at risk.

In the graph in Figure 7, the table of subjects at risk is displayed closer to the survival curves, above the x-axis. This reduces the clutter in the lower port of the graph in Figure 6. Also note, placing the table inside the data area allows the y-axis label to be placed closer to the y-axis values.

USING THE YAXISTABLE FOR FOREST PLOT

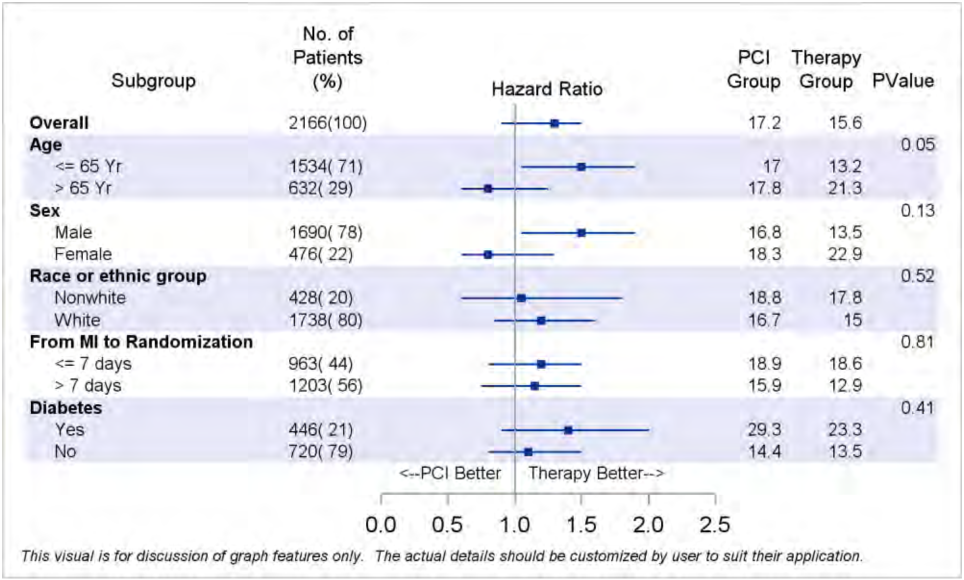
Another popular graph used for clinical research is the forest plot. In this example, the requirement is to display the study data by various subgroups as shown in the data in Figure 8. The program to create this graph is shown in Figure 9, and the graph is shown in Figure 10.

ObsId	Id	Subgroup	indentWt	countpct	Low	Mean	High	PCIGroup	Group	PValue	ref
1	1	Overall	0	2166(100)	0.90	1.3	1.50	17.2	15.6	.	.
2	1	Age	0		.	.	.	.	.	0.05	2
3	2	<= 65 Yr	1	1534(71)	1.05	1.5	1.90	17.0	13.2	.	3
4	2	> 65 Yr	1	632(29)	0.60	0.8	1.25	17.8	21.3	.	4
5	1	Sex	0		.	.	.	.	.	0.13	.

Figure 8 - Data for forest plot.

```
proc sgplot data=forest_subgroup_2 nowall noborder nocycleattrs
dattrmap=attrmap noautolegend;
  styleattrs axisextent=data;
  refline ref / lineattrs=(color=cxe7e7f7) discretethickness=1;
  highlow y=obsid low=low high=high;
  scatter y=obsid x=mean / markerattrs=(symbol=squarefilled size=4);
  scatter y=obsid x=mean / markerattrs=(size=0) x2axis;
  refline 1 / axis=x;
  text x=xl y=obsid text=text1 / position=left contributeoffsets=none;
  text x=xl y=obsid text=text2 / position=right contributeoffsets=none;
  yaxistable subgroup  / location=inside position=left textgroup=id
             textgroupid=text indentweight=indentWt;
  yaxistable countpct / location=inside position=left
  yaxistable PCIGroup group pvalue / location=inside position=right;
  yaxis reverse display=none type=discrete
        colorbands=odd colorbandsattrs=(transparency=1);
  xaxis display=(nolabel) values=(0.0 0.5 1.0 1.5 2.0 2.5);
  x2axis label='Hazard Ratio' display=(noline noticks novalues);
run;
```

Figure 9 – SGPLOT procedure code for the forest plot



Subgroup	No. of Patients (%)	PCI Group	Therapy Group	PValue
Overall	2166(100)	17.2	15.6	
Age				0.05
<= 65 Yr	1534(71)	17	13.2	
> 65 Yr	632(29)	17.8	21.3	
Sex				0.13
Male	1690(78)	16.8	13.5	
Female	476(22)	18.3	22.9	
Race or ethnic group				0.52
Nonwhite	428(20)	18.8	17.8	
White	1738(80)	16.7	15	
From MI to Randomization				0.81
<= 7 days	963(44)	18.9	18.6	
> 7 days	1203(56)	15.9	12.9	
Diabetes				0.41
Yes	446(21)	29.3	23.3	
No	720(79)	14.4	13.5	

This visual is for discussion of graph features only. The actual details should be customized by user to suit their application.

Figure 10 – Forest plot

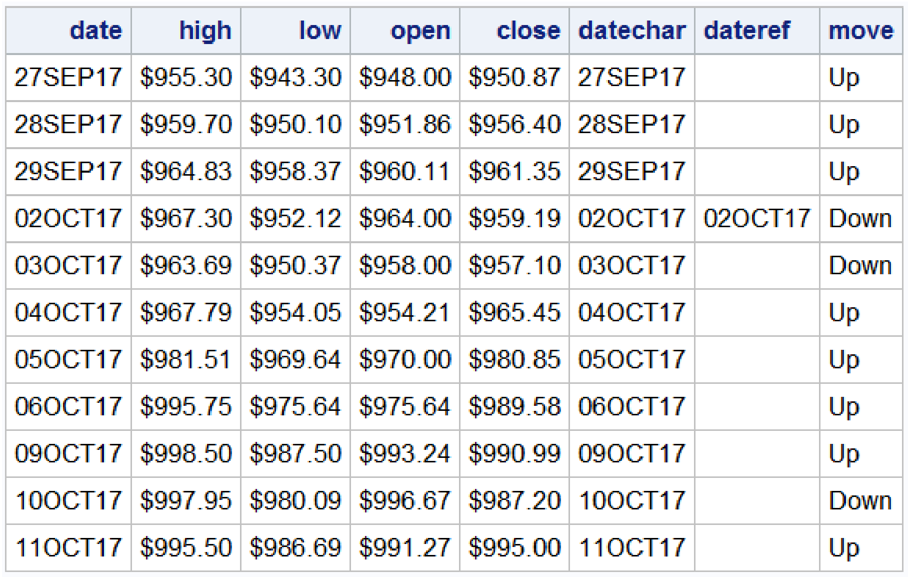
Note the following aspects of this program and graph.

- A HIGHLOW plot and a SCATTER plot are used to display the hazard ratios in the middle.
- One YAXISTABLE statement is used to display the sub grouped study names on the left.
  - The TEXTGROUP and TEXTGROUPID options are used for the bold font for the subgroups.
  - The INDENTWEIGHT option is used to indent the values.
- The second YAXISTABLE statement is used to display the count + percent values on the left.
- The last YAXISTABLE statement is used to display three statistics on the right.
- In a wide graph like this one, it helps to use the horizontal bands as a visual aid.
  - The REFLINE statement is used to display the bands using the “ref” variable.
  - Wide reference lines are drawn wherever the “ref” variable is not missing.
  - DISCRETETHICKNESS=1 is used to set the refline thickness equal to the midpoint spacing.
- The YAXISTABLE automatically displays the variable label above the column, split to fit the space.
- The dataneeded to display the “PCI Better” indicators on the x-axis is in the data set. TEXT statements are used to display these values.
- Note the use of AXISEXTENT=data to display the axis line only over the data extent
- No annotation is needed to create this graph.

USING THE XAXISTABLE FOR CUSTOM GRAPHS

The survival plot and the forest plots were instrumental in the design of the AXISTABLE statements. These graphs were often requested by users and building them using the traditional annotation process was very cumbersome and the process did not easily scale to different data. However, you can also use these statements in other creative ways to build custom graphs.

For the first example, let us review a common stock plot shown in Figure 13. To create this graph, I downloaded the historical stock data for Amazon for the last 6 months from the Nasdaq web site as a CSV file. I used a Data step to clean the data and transform it into a format suitable for SAS, with a few additional columns that will be useful later. A portion of the data is shown in Figure 11.



date	high	low	open	close	datechar	dateref	move
27SEP17	$955.30	$943.30	$948.00	$950.87	27SEP17		Up
28SEP17	$959.70	$950.10	$951.86	$956.40	28SEP17		Up
29SEP17	$964.83	$958.37	$960.11	$961.35	29SEP17		Up
02OCT17	$967.30	$952.12	$964.00	$959.19	02OCT17	02OCT17	Down
03OCT17	$963.69	$950.37	$958.00	$957.10	03OCT17		Down
04OCT17	$967.79	$954.05	$954.21	$965.45	04OCT17		Up
05OCT17	$981.51	$969.64	$970.00	$980.85	05OCT17		Up
06OCT17	$995.75	$975.64	$975.64	$989.58	06OCT17		Up
09OCT17	$998.50	$987.50	$993.24	$990.99	09OCT17		Up
10OCT17	$997.95	$980.09	$996.67	$987.20	10OCT17		Down
11OCT17	$995.50	$986.69	$991.27	$995.00	11OCT17		Up

Figure 11 – Stock data

Note in this data set, the original character date is retained. Also, at every start of a new month, the character date is copied into the “dateref” column. For other observations, this value is missing. Now, let us create a stock plot from this data using the “date” variable. The program is shown in Figure 12 and the resulting graph is shown in Figure 13.

```
title 'Stock Price for Amazon by Date';
proc sgplot data=amzn noborder;
  highlow x=date low=low high=high / open=open close=close
            lineattrs=(thickness=2) y2axis;
  y2axis display=(noline noticks nolabel) grid;
  xaxis display=(nolabel);
run;
```

Figure 12 – SGPLOT procedure code for the stock plot

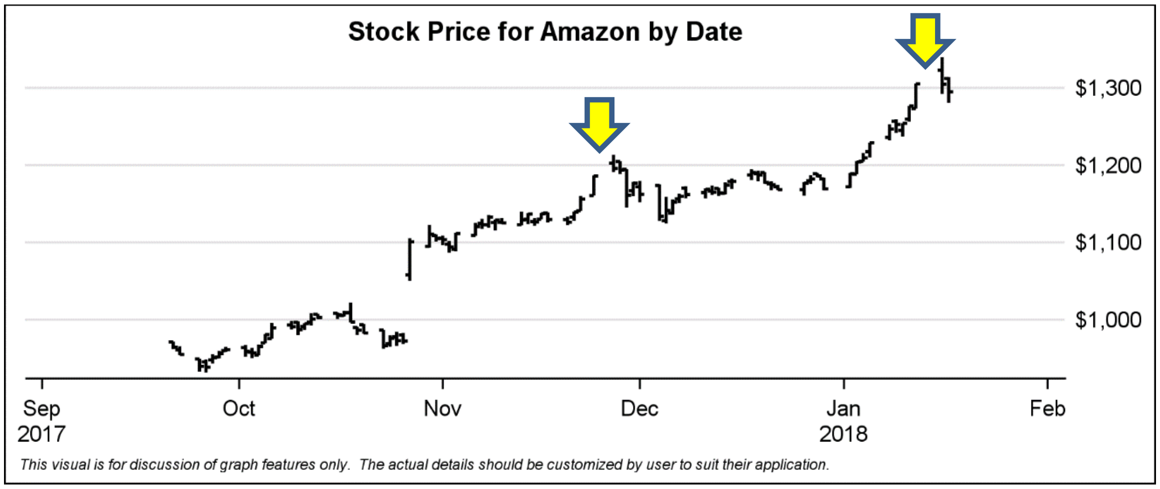


Figure 13 – Stock plot

This graph uses a SAS date axis since the “Date” variable has a SAS date format. The values displayed in the graph are correctly positioned on a correctly scaled date axis. This results in gaps where there is no data on a valid date, such as for weekends and holidays when the stock market is closed.

While the above graph is technically correct, this is not how industry standard stock charts are plotted. The standard practice in the industry is to plot the data in a continuous manner, where the values for Monday are shown immediately after Friday, without any gaps. The same is true for all holidays.

You can create such a graph by first sorting the data by ascending date, and then plotting the data with a discrete x-axis, so that all the values are shown equally spaced on the axis. This will achieve our goal except the x-axis will display all the date value at a 45^o angle, creating a very cluttered axis.

The code for a solution is shown in Figure 14, with the resulting graph in Figure 15.

```
title 'Stock Price for Amazon by Date';
proc sgplot data=amzn noborder noautolegend dattrmap=attrmap;
  refline dateref / axis=x lineattrs=graphgridlines;
  highlow x=datechar low=low high=high / open=open close=close
          group=move lineattrs=(thickness=2) y2axis attrid=Move;
  xaxistable dateref / nolabel valueattrs=(weight=bold);
  y2axis display=(noline noticks nolabel) grid;
  xaxis type=discrete display=(nolabel novalues noticks)
        discreteorder=data;
run;
```

Figure 14 – SGPLOT procedure code for the stock plot

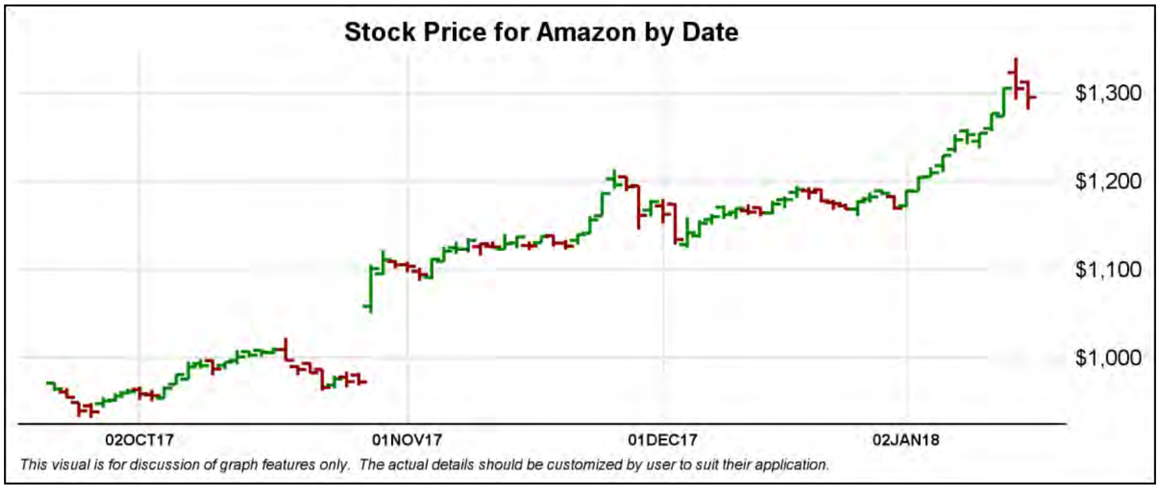


Figure 15 – Stock plot

The stock plot in Figure 15 shows the data in the industry preferred form, where the data for each day is displayed next to the previous day when the stock market is open. There are no gaps for weekends or holidays. We have achieved this result by using the XAXISTABLE to display the date values from the "dateref" column.

- In the data, the "dateref" column contains text string for only the first working day of each month.
- The stock data is sorted by ascending date and plotted on a discrete x-axis.
- The x-axis is fully hidden.
- A XAXISTABLE is used to display the associated "dateref" values at the bottom of the graph.
- Reference lines are drawn at each non-missing dateref value using the GraphGridLines style.
- Note in this graph, the tick values are at uneven intervals, based on the number of observations per month. The bars are colored green when the stock closes above previous close.

USING THE YAXISTABLE FOR CUSTOM GRAPHS

Sometimes there is a need to display a custom axis where the tick values on the axis need to be something different from the default. Often, you can do that by providing alternate tick values for the axis by using the VALUES option. If the values need to be modified further, you can use the VALUESDISPLAY option to provide the text strings for the specific values to be replaced.

The data for this example is shown in Figure 17 on the right. The "y2" column contains the value from the column y for every other observation. Also, a "group" column is computed that will be used later.

```
title "Default Y axis for Y=4*X~{unicode '00b2'x}";
proc sgplot data=trans noborder noautolegend;
  needle x=x y=y / displaybaseline=auto);
  yaxis grid display=(nolabel);
  xaxis display=(nolabel);
run;
```

Figure 16 – SGPLOT procedure code for y=4*x^2 plot

x	y	y2	group
0	0	0	1
10	400	.	1
20	1600	1600	1
30	3600	.	1
40	6400	6400	1
50	10000	.	1
60	14400	14400	2

Figure 17 - Data

The output graph is shown in Figure 18. The default y-axis displays the tick values and grid lines at a regular interval.

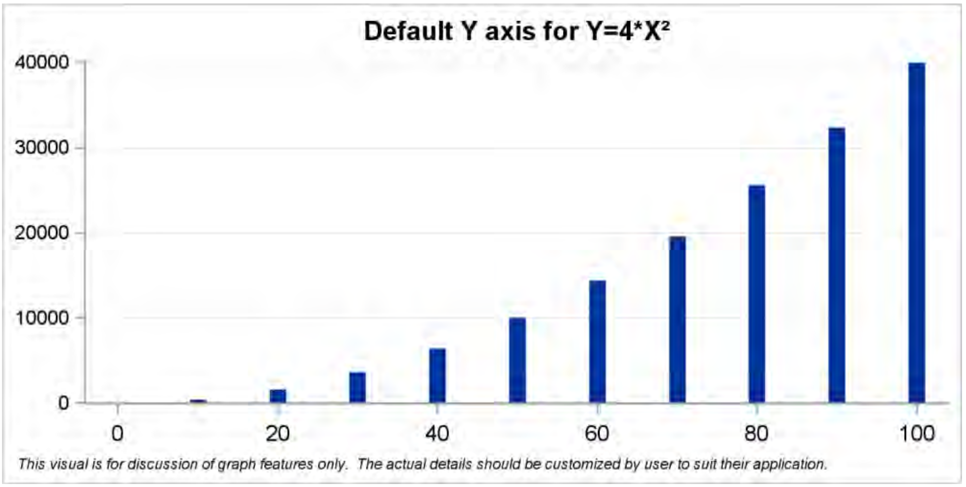


Figure 18 – Default plot of y=4*x^2

Now, what if we want to display the y-axis tick values at custom location, also with highlighting of some tick values? As mentioned earlier, you can change the axis values displayed on the y-axis by using the VALUES option. Then, the individual value string can be replaced using the VALUESDISPLAY option. However, this requires us to populate these values into the procedure syntax. But, there is no way to change the color of some of the tick values through axis options.

The program in Figure 19 uses the YAXISTABLE to display the values on the left side from the "y" column, with COLORGROUP=group. This displays every value in the "y" column on the left, and also displays the "14400" value in red. The output graph is shown in Figure 20.

```
title "Custom Y axis for Y=4*X~{unicode '00b2'x} with color";
proc sgplot data=trans noborder noautolegend;
  styleattrs datacontrastcolors=(black red);
  refline y / lineattrs=graphgridlines;
  needle x=x y=y / displaybaseline=auto  lineattrs=(thickness=10);
  yaxistable y / location=outside position=left valueattrs=(size=8)
        nomissingchar nolabel colorgroup=group;
  yaxis display=(nolabel noticks novalues) offsetmin=0;
  xaxis display=(nolabel);
run;
```

Figure 19 – SGPLOT procedure code for y=4*x^2 plot

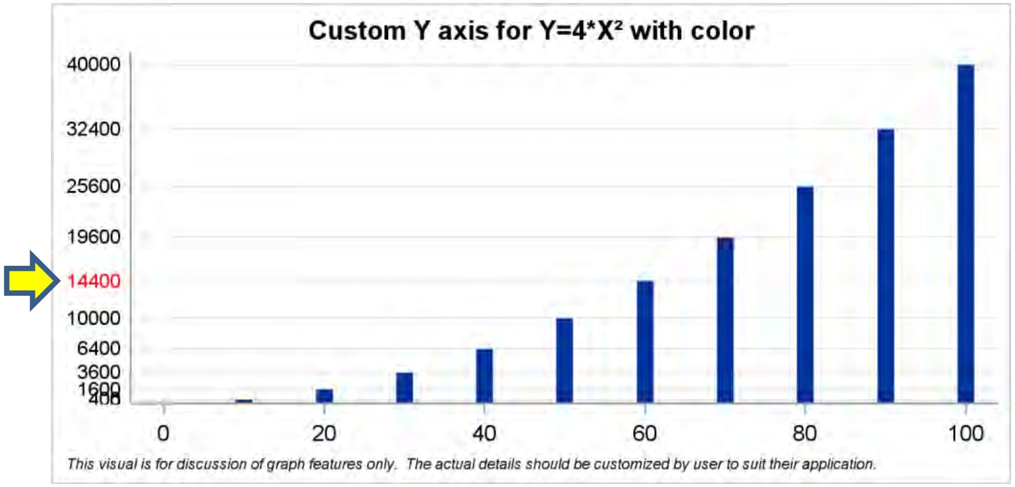


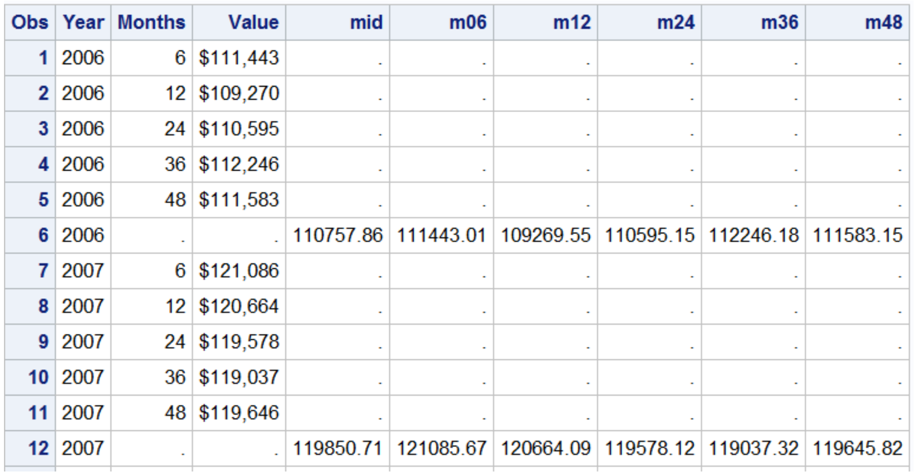
Figure 20 – Custom axis for plot of y=4*x^2

We could have used the “y2” variable in the YAXISTABLE statement to display alternate values instead of every one. If needed, you can plot the values at different locations, or even put text strings instead of numbers. The key benefit is that the axis table allows you to control this from your data, instead of having to set up the options in the procedure syntax.

USING THE YAXISTABLE FOR SPARKLINE PANEL

Finally, let us see how you can build a graph to view inpatient claims over 6 to 48 months and also view the data as a spark line that can give you a quick visual of the trend. Such a graph was presented at the SESUG Conference 2017 in Cary by Rick Andrews in the presentation "Methods for creating Sparklines using SAS".

Rick used the traditional SAS/GRAPH procedures and annotation to create the graph. A similar graph can be created using the SGPANEL procedure with the YAXISTABLE statements. The data for this graph is simulated by me using random functions and is shown in Figure 21.



Obs	Year	Months	Value	mid	m06	m12	m24	m36	m48
1	2006	6	$111,443	.	.	.	.	.	.
2	2006	12	$109,270	.	.	.	.	.	.
3	2006	24	$110,595	.	.	.	.	.	.
4	2006	36	$112,246	.	.	.	.	.	.
5	2006	48	$111,583	.	.	.	.	.	.
6	2006	.	.	110757.86	111443.01	109269.55	110595.15	112246.18	111583.15
7	2007	6	$121,086	.	.	.	.	.	.
8	2007	12	$120,664	.	.	.	.	.	.
9	2007	24	$119,578	.	.	.	.	.	.
10	2007	36	$119,037	.	.	.	.	.	.
11	2007	48	$119,646	.	.	.	.	.	.
12	2007	.	.	119850.71	121085.67	120664.09	119578.12	119037.32	119645.82

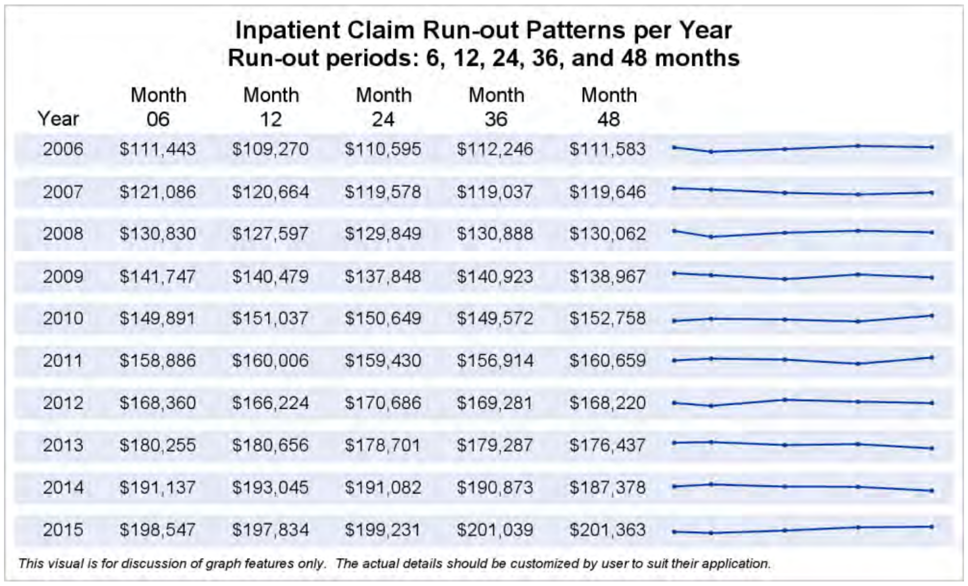
Figure 21 – Inpatient claim run-out data

This data set contains the claim runout values by year and period. For each year, I have also put the monthly values in separate columns such as “m06”, “m12” and so on. I have also computed an average of all these columns in the column “mid”.

Now, I can use the SGPANEL program shown in Figure 22 to display the data as a panel by year, including the monthly values, and a “SparkLine” graph showing the trend over the monthly periods by Year. The output graph is shown in Figure 23.

```
title 'Inpatient Claim Run-out Patterns per Year';
title2 'Run-out periods: 6, 12, 24, 36, and 48 months';
proc sgpanel data=final;
  format value m06 m12 m24 m36 m48 dollar8.0;
  styleattrs wallcolor=cxe7ebf7;
  panelby year / layout=rowlattice onepanel uniscale=column
          noheader noborder spacing=10;
  series x=months y=value / markers
          markerattrs=(symbol=circlefilled size=5);
  rowaxistable year m06 m12 m24 m36 m48 / y=mid nomissingchar
         valueattrs=(size=8) position=left pad=10 labelpos=top;
  rowaxis display=none;
  colaxis display=none;
run;
```

Figure 22 – SGPANEL procedure code for the “SparkLine” panel.



Inpatient Claim Run-out Patterns per Year
Run-out periods: 6, 12, 24, 36, and 48 months

Year	Month 06	Month 12	Month 24	Month 36	Month 48
2006	$111,443	$109,270	$110,595	$112,246	$111,583
2007	$121,086	$120,664	$119,578	$119,037	$119,646
2008	$130,830	$127,597	$129,849	$130,888	$130,062
2009	$141,747	$140,479	$137,848	$140,923	$138,967
2010	$149,891	$151,037	$150,649	$149,572	$152,758
2011	$158,886	$160,006	$159,430	$156,914	$160,659
2012	$168,360	$166,224	$170,686	$169,281	$168,220
2013	$180,255	$180,656	$178,701	$179,287	$176,437
2014	$191,137	$193,045	$191,082	$190,873	$187,378
2015	$198,547	$197,834	$199,231	$201,039	$201,363

This visual is for discussion of graph features only. The actual details should be customized by user to suit their application.

Figure 23 – "SparkLine" Panel graph

Here are the key features of this graph:

- We use the SGPANEL procedure with PANELBY year and LAYOUT=RowLattice.
- We display the SERIES plot of runout value by month. The axes display is suppressed.
- We use the ROWAXISTABLE to display Year, and the 5 monthly values.
- Note, we use Y=mid option to set the y axis value to the mean of the yearly values.
    - If we use the default y=value, then the values in the table will not be evenly displayed.

CONCLUSION

With SAS 9.4, the SGPLOT, SGPANEL procedures and GTL support the new axis table feature. This allows you to easily include textual data in your graph, aligned with the x or y axis. These statements were initially motivated by the popular survival and forest plots. Now, you can use axis table statements to include such data without the need for annotation.

Additionally, you can also use the axis table statements to create custom graphs. The examples show how to use these for creating custom x or y-axes. You can also use these to create complex graphs such as the "SparkLine" panel.

REFERENCES

Matange, Sanjay 2016. *Clinical Graphs Using SAS®*. 1st ed. Cary, NC: SAS Institute.

Andrews, Rick, Hadden, Louise and Allison, Robert. 2017. "Methods for creating Sparklines using SAS." *Proceedings of SESUG,* Cary, NC.

Matange, Sanjay, "Graphically Speaking."
https://blogs.sas.com/content/graphicallyspeaking/2017/11/14/spark-table/.

RECOMMENDED READING

- *Base SAS® Procedures Guide*
- *Clinical Graphs Using SAS®*

CONTACT INFORMATION

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

Sanjay Matange
SAS Institute Inc.,
100 SAS Campus Dr.
Cary, NC 27513
Sanjay.Matange@sas.com
http://blogs.sas.com/content/graphicallyspeaking