PharmaSUG 2011 – Paper TT01

Scatter Charts of Serial Observations with Proc SGPLOT and Graphics Template Language

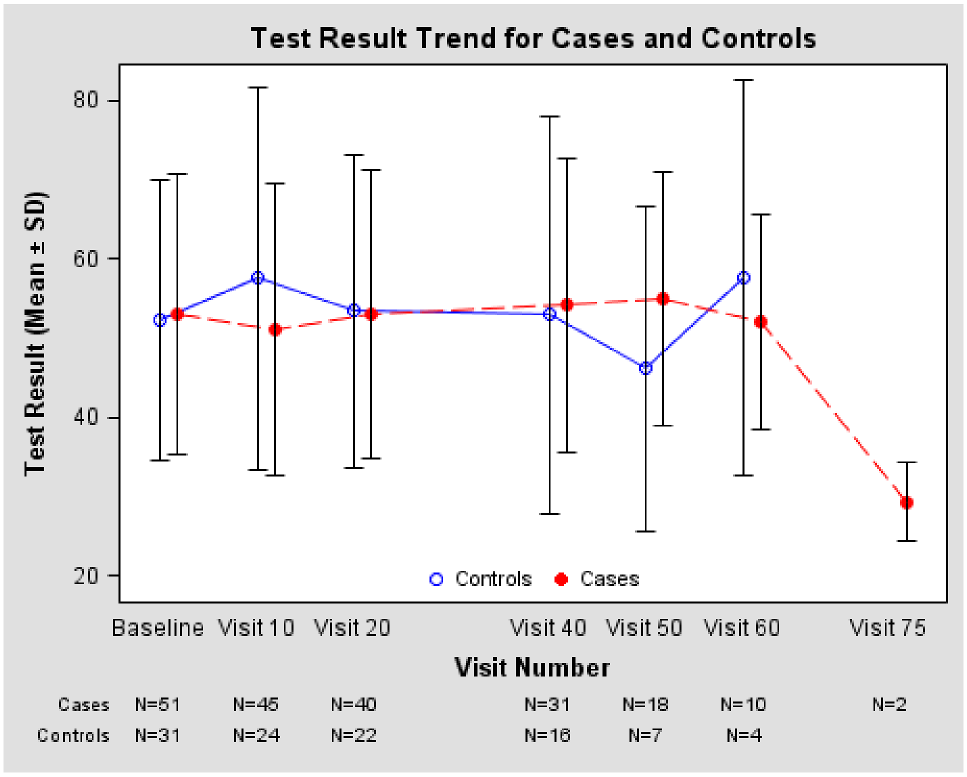
Anthony L. Feliu, Genzyme Corporation, Cambridge, Massachusetts

ABSTRACT

The new statistical graphics procedures and underlying Graphics Template Language (GTL) in SAS® 9.2 offer programmers the opportunity to create sophisticated, publication–ready charts using less code than with the traditional graphics procedures. This paper selects one chart common to clinical trials—scatter chart to trend observational data—and presents a step-by-step analysis of the code to create this output, including how sample counts can be presented below the chart image.

INTRODUCTION

Clinical trials are conducted to assess the safety and efficacy of drug products and devices. Patient data are routinely collected at intervals prescribed in the protocol, and analyzed by comparing findings for study subjects vs. a comparator group, or by comparing pre– vs. post–treatment findings for a single cohort.



Most trials collect voluminous quantities of data, particularly laboratory test results. Therefore, graphical data presentation is an important technique to comprehend patterns that might not be easily discerned from tabular or summarized outputs. Among the charts used for this purpose is one presenting the mean test result at serial visits, as shown at left.

With the traditional graphics procedures, most charting requires data preprocessing and painstaking configuration of both GOPTIONS and procedure options.

However, the SAS 9.2 statistical graphics functionality considerably simplifies the task of creating this chart. Firstly, some computational capability is encapsulated with the procedure, allowing the observational file to be charted without prior summarization. Secondly, the overall chart appearance leverages the Output Delivery System (ODS), conferring consistency across a set of outputs. Finally, customizations can be implemented by plot statement options within the procedure or by preparing a specific GTL template.

In this paper, four techniques to create a mean–test–result chart with SGPLOT and GTL programming are discussed.

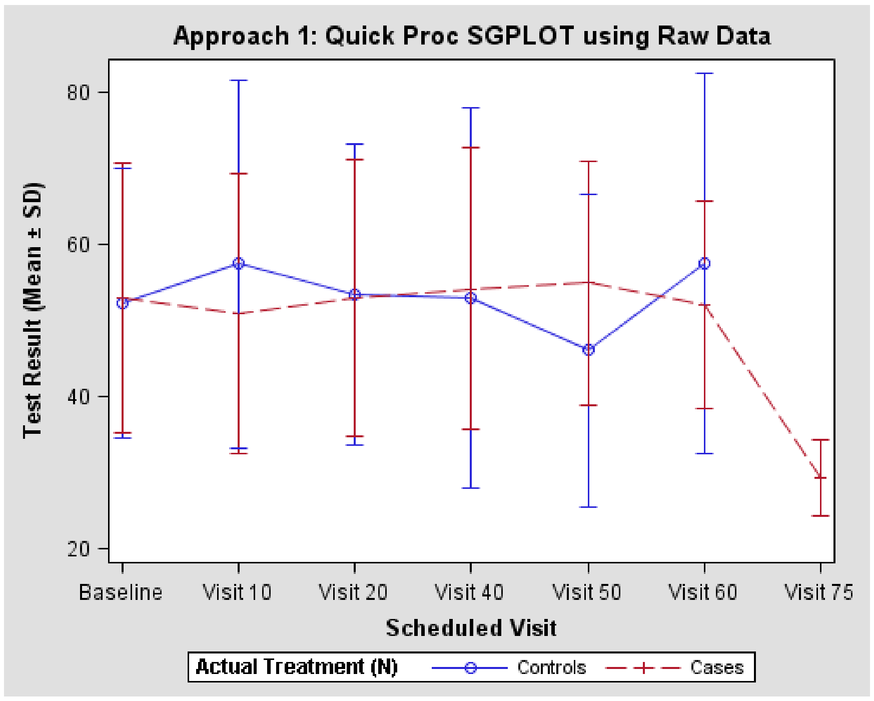
APPROACH 1: PROC SGPLOT WITH SOURCE DATA

Given raw data with one observation per record, such as the CDISC ADaM Basic Dataset Structure (BDS), a trending chart can be created with one concise call to the SGPLOT procedure:

```
proc sgplot data = work.adam ;
   vline avisitn / response  = aval
                   group     = trtan
                   stat      = mean
                   limitstat = stddev
                   limits    = both
                   markers ;
   where paramcd = 'MYTEST' ;
run ;
```

Behind the scenes, observations are grouped by visit number AVISITN, similar to the “class” statement in Proc MEANS.

Means and standard deviations of the result variable AVAL for each patient treatment group TRTAN are then computed on the fly before Proc SGPLOT creates the chart. The GROUP=TRTAN option thus serves as another “class” variable.



By default, the X-axis scale of a VLINE plot is categorical. By adding an XAXIS statement this could be changed to a proportional scale:

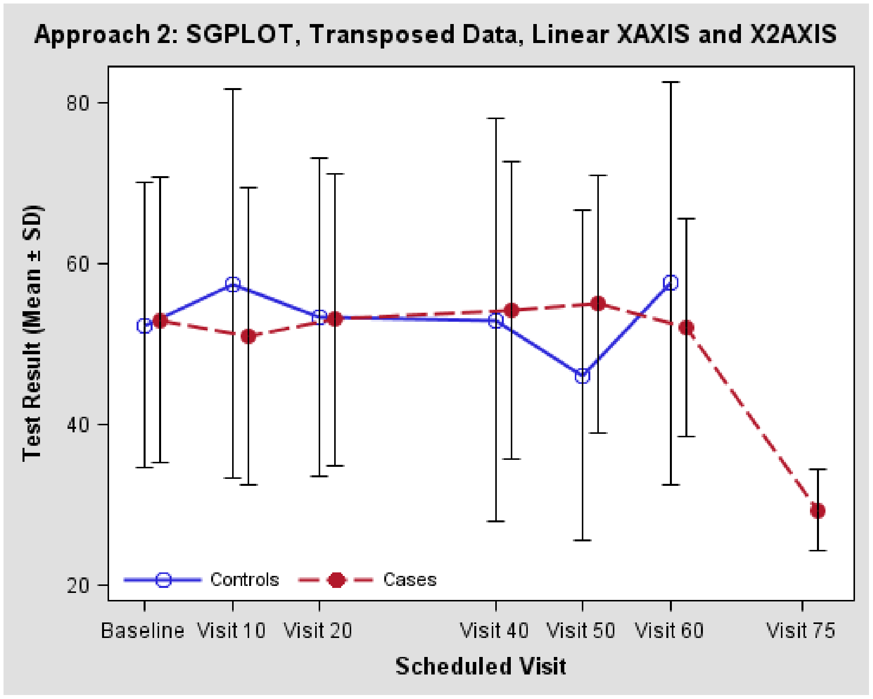
```
xaxis type = linear ;
```

Colors, markers, and line styles are set by ODS. The default ODS style was used here.

For customizations, create a new ODS style with Proc TEMPLATE, or add options to the VLINE statement (see below).

In many situations, the resulting output is entirely adequate. One disadvantage, however, is overlap of the error bars which partly obscures group differences.

APPROACH 2: PROC SGPLOT WITH PREPROCESSED DATA



Visual separation of the data series can be achieved by configuring Proc SGPLOT to create a secondary X-axis with identical scale but offset from the primary X-axis.

To accomplish this, our original dataset will have to be transposed to place result values for cases and controls in different variables. At the same time, we will run off a list of visit numbers to configure the X-axis ticks.

```
proc transpose data = work.adam
               out  = work.transpose
               prefix = aval ;
   by  subjid avisitn ;
   var aval ;
   id  trtan ;
   format trtan ;
   where paramcd = 'MYTEST' ;
run ;

proc print data = work.transpose ;
run ;
```

```
------------------------------------
Obs  SUBJID  AVISITN  AVAL0  AVAL1
------------------------------------
  1  01011       0    52.75
  2  01011      10    70.71
  3  01011      20    70.54
  4  01011      40    77.84
  5  01011      50    70.23
  6  01011      60    70.10

  7  01012       0            69.68
  8  01012      10            69.53
  9  01012      20            63.28
 10  01012      40            63.14
-------------------------------------
```

```
proc sql noprint ;
   select distinct avisitn
       into :vis_list separated by ' '
   from work.transpose
   order by 1 ;
quit ;

%put VIS_LIST: &vis_list ;
```

```
VIS_LIST: 0 10 20 40 50 60 75
```

With this dataset—which remains observational—two VLINE plot statements can be arranged.
Don't worry !! It's not nearly as complicated as it looks !

```
proc sgplot data = work.transpose ;
   yaxis  label = 'Test Result (Mean %sysfunc(byte(0177)) SD)' ;
                                      /* Primary axis (bottom) */
   xaxis  type      = linear
          values    = (&vis_list)
          offsetmin = 0.05
          offsetmax = 0.07
          label     = 'Scheduled Visit' ;
                                      /* Secondary axis (top) */
   x2axis type      = linear
          values    = (&vis_list)
          offsetmin = 0.07
          offsetmax = 0.05
          display   = (nolabel noticks novalues) ;
```

Axis TYPE=LINEAR plots numeric values on proportional scale.

Primary and secondary X axes are given a tick list so they have identical ranges regardless of the data.

Options OFFSETMIN, OFFSETMAX define margins which SAS will maintain clear of data points:

"0.05" is 5% margin.
"0.07" is 7% margin.

Our "secret" to achieve visual separation without programming is to tinker with the margins, while maintaining the same range.

- Both axes devote 12% to margins. The primary XAXIS has a rather wider right margin, while the secondary X2AXIS has a wider left margin.
- At the same time, the value list synchronizes both axis ranges, even though the Controls have data to Week 60, while the Cases have data through Week 75.

With these axis definitions in place, it remains merely to plot the data for each treatment group.
Several plot options have been added to enhance the presentation.

```
   vline avisitn / response    = aval1
                   stat        = mean
                   limits      = both
                   limitstat   = stddev
                   x2axis               /* note x2axis option */
                   markers
                   markerattrs = (size = 9 symbol = circle)
                   lineattrs   = (thickness = 2px)
                   limitattrs  = (thickness = 1px)
                   legendlabel = "Cases" ;
```

This VLINE statement will summarize and plot data for the cases (RESPONSE=AVAL1).

It references the secondary X2AXIS.

```
   vline avisitn / response    = aval0
                   stat        = mean
                   limits      = both
                   limitstat   = stddev
                   markers
                   markerattrs = (size = 9 symbol = circle)
                   lineattrs   = (thickness = 2px)
                   limitattrs  = (thickness = 1px)
                   legendlabel = "Controls" ;
```

This VLINE statement will summarize and plot data for the controls (RESPONSE=AVAL0).

There is no mention of an axis, so the primary XAXIS will apply.

```
   keylegend / location = inside
               position = bottomleft
               noborder
               title = '' ;
```

Place legend within the axes.

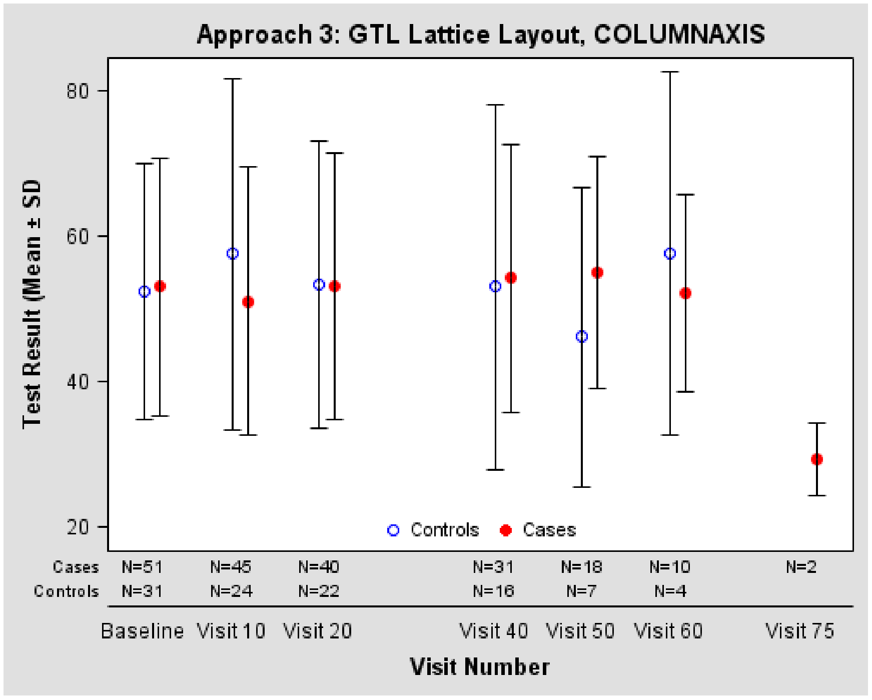
```
   format avisitn vis. ;
run ;
```

X-axis uses numeric visits to have linear scaling, but the tick label display will be visit names.

For a modest programming investment, we improved the appearance of our chart.

APPROACH 3: GTL WITH COLUMNAXES BLOCK

When there is need for sample counts to accompany the chart—a common request from statisticians and medical writers—we must go beyond Proc SGPLOT functionality.



In the old days, programmers using SAS/GRAPH would have to make an annotate file, basically placing each count by hand.

With SAS 9.2 and Graphics Template Language (GTL), presentation of data points and sample counts can be entirely automated.

Let's plan for a pair of charts, one above the other:

- The upper chart will duplicate what we already have.
- A lower chart will place treatment group on the Y-axis, visit number on the X-axis, and use pre-calculated sample counts as the "markers."
- X-axis scaling will be common to both charts.

SAS CAN WRITE GTL CODE FOR YOU

GTL syntax is different from Proc SGPLOT. However, crafting graphs with GTL is not difficult because Proc SGPLOT uses GTL behind the scenes.

```
proc sgplot data     = work.transpose
            tmplout = 'mygraph.txt' ;  /* look */
   yaxis label = 'Test Result ...' ;
   xaxis ... ;
   vline avisitn / response = aval0 ... ;
   vline avisitn / response = aval1 x2axis ... ;
   keylegend ... ;
run ;
```

With the TMPLOUT= procedure option, SAS shows you how it is rendering a graph.

SAS–generated GTL is the perfect starting point when building out a custom template.

DIGRESSION

SGPLOT and GTL work within the Output Delivery System. When you compile GTL with Proc TEMPLATE, by default, SAS writes to a permanent item store, SASUSER.TEMPLAT. Especially when you are developing—or even thereafter !!—you might want to redirect your code to a temporary item store. Keep your hard drive clean! After all, out of sight, out of mind!

Find something that works for you and do it consistently. Here's my approach:

```
proc datasets library = sasuser nowarn nolist ;
   delete templat (memtype = itemstor) ;
run ;
```

In case something was saved and forgotten, delete personal item store.

```
proc datasets library = work nowarn nolist ;
   delete templat (memtype = itemstor) ;
run ;
quit ;
```

When developing code, running and rerunning things, sometimes it's good to clear even the working item store.

```
ods path reset ;
ods path (prepend) work.templat (update) ;
```

Put temporary item store first in the ODS path. Any subsequent GTL will be saved there.

Lastly, I'll print what I actually have to the SAS log.

```
ods path show ;
```

```
Current ODS PATH list is:
1. WORK.TEMPLAT(UPDATE)
2. SASUSER.TEMPLAT(UPDATE)
3. SASHELP.TMPLMST(READ)
```

WORK.TEMPLAT is the new temporary item store. It's showing first. Good.

SASUSER.TEMPLAT is the default permanent item store (generally found in the "My SAS Files" folder if running SAS on Windows platform).

SASHELP.TMPLMST is part of the SAS software distribution, containing all built–in ODS styles and templates.

GTL SYNTAX

Graphic template language has two syntactical constructs:

- GTL Blocks are units of code identified by opening and closing tags. They serve as wrappers for GTL statements and for other blocks.
- In the example at right, a layout block is nested within a graph block.
- GTL Statements are programming instructions. They consist of a keyword, followed by arguments, ending with a semicolon.
- In the example at right, two statements are shown within the layout block.

```
BEGINGRAPH ;
   LAYOUT OVERLAY / {optional arguments} ;
      KEYWORD {required arguments} /
              {options} ;
      KEYWORD {required arguments} /
              {options} ;
   ENDLAYOUT ;
ENDGRAPH ;
```

GTL is feature rich. Both blocks and statements have many possible options.
Keep the SAS documentation at hand to find your way.

GTL PLANNING

To begin constructing this template, we prepare a drawing canvas with provision for two charts:

```
proc template ;
   define statgraph approach3 ;
      begingraph ;

         dynamic _ti ;
         entrytitle _ti ;
```

Create a graph template named "approach3".

Create a "dynamic" variable for the graph title. Dynamic variable values can be assigned at run time.

This definition allows changing the title without recompiling the template every time.

```
      layout lattice /
             columns          = 1
             rows             = 2
             rowweights       = (0.9 0.1)
             rowgutter        = 0
             columndatarange = union ;
```

A LATTICE layout block creates a virtual grid wherein the plot areas and axes are aligned among the several graphs.

Layout options prepare for two charts, one above the other. The upper chart (for the data) will occupy 90% of the canvas. The lower chart (for the sample counts) will occupy 10% of the canvas.

There will be no spacing between the two chart areas (ROWGUTTER=0), and the column data range (i.e. X-axis) will be consistent (COLUMNDATARANGE=UNION).

```
                /* This is an outline.                  */
                /* COLUMNAXES and LAYOUT blocks will    */
                /* be fully developed in subsequent     */
                /* sections of this paper.              */

         columnaxes ;
            columnaxis / ... ;
         endcolumnaxes ;
```

The COLUMNAXES block supports the unified X-axis we specified with the COLUMNDATARANGE=UNION option.

Notice this block is at the same level of hierarchy as the two graphs because it exerts control on all charts in that column.

```
         layout overlay  / ... ;
            scatterplot x = avisitn  ... ;
            scatterplot x = avisitn  ... ;
            discretelegend ... ;
         endlayout ;
```

An OVERLAY layout block creates a single chart. It is called “overlay” because more than one data series can be overlaid on the given X–Y axis pair.

This is the first layout block, so it will be sized 90% of the canvas area.

In this chart we will plot the data.

```
         layout overlay  / ... ;
            scatterplot x = avisitn  ... ;
            scatterplot x = avisitn  ... ;
         endlayout ;
```

This is the second layout block, sized 10% of the available canvas area.

We will use this chart to present the sample counts. Taking advantage of the unified X-axis, sample counts for each visit will automatically align with the corresponding data point.

```
      endlayout ;     /* End lattice layout block. */
   endgraph ;         /* End graph block.   */
end ;                 /* End define block.  */
run ;
```

Block closing tags.

GTL COLUMNAXES BLOCK

The COLUMNAXES block will have options to configure the unified X-axis we requested with the LATTICE layout.

```
columnaxes ;
```

Opening tag of COLUMNAXES block.

```
   columnaxis /
```

COLUMNAXIS statement has no required arguments, but the forward slash signals that optional arguments follow.

```
                offsetmin   = 0.05
                offsetmax   = 0.07
```

Offsets define margins clear of data points. Unequal margins (5% left, 7% right) borrow the idea used in Approach 2.

```
                type       = linear
```

Axis TYPE=LINEAR results in a proportional numeric scale. (TYPE=DISCRETE would give equally–spaced ticks. We rejected those after reviewing Approach 1).

```
                linearopts = (tickvaluelist    = (&vis_list)
                              tickvalueformat = vis.)
```

Further options input the tick list. Without this, SAS would automatically decide on the tick intervals. In other data, this is fine, but here we want to see every visit label.

```
                label      = 'Visit Number'
                display    = (line label tickvalues)
```

Option DISPLAY= specifies to draw the axis line, to label the axis, and to print tick values. Tick marks will be omitted.

```
                displaysecondary = none ;
```

Drawing of a secondary axis at the top of the canvas is suppressed.

```
endcolumnaxes ;
```

Block closing tag.

GTL UPPER LAYOUT OVERLAY BLOCK

In the main chart, we are plotting the two data series, with mean values and standard deviations. GTL does not have the built–in computational ability that Proc SGPLOT does, so we will have to summarize the data before using the template. For now, let's continue writing code.

There's a lot going on within this block, but it's quite straightforward.

```
layout overlay /
```

Opening tag of LAYOUT block.

Remember, an OVERLAY layout specifies one X–Y chart.

Optional arguments follow the forward slash.

```
   x2axisopts = (offsetmin = 0.07
                 offsetmax = 0.05
                 linearopts = (tickvaluelist = (&vis_list))
                 display    = none)
```

The primary X–axis was already defined at the LATTICE level.

Here, a secondary X–axis is created. The margin offsets are opposite to the primary X–axis.

Providing the tick list at this point is crucial to have identical scaling between primary and secondary axes.

Option DISPLAY=NONE suppresses display of labels and tick marks which would appear at the top of the chart.

```
   yaxisopts  = (offsetmin = 0.05
                 offsetmax = 0.03
                 label =
        "Test Result (Mean (*ESC*){unicode 'B1'x} SD)"
                )
```

The Y–axis is for result values. By default, it is displayed on numeric scale (TYPE=LINEAR). We'll let SAS will determine the range.

(Logarithmic scaling with base 10, 2, or E is also supported.)

Marginal offsets anticipate clearance for a legend inside the chart at the bottom.

```
   ;
```

End of layout block options.

```
   scatterplot x           = avisitn
               y           = mean_1 /
               xaxis       = x2
               name        = 'G1'
               legendlabel = "%scan(&trt_list, 2, %str(|))"
               markerattrs = (size   = 8
                              symbol = circlefilled
                              color  = red)
               yerrorupper = eval(mean_1 + std_1)
               yerrorlower = eval(mean_1 - std_1)
               ;
```

This SCATTERPLOT statement charts mean values with error bars for the cases (Y=MEAN_1).

The secondary axis is referenced (XAXIS=X2).

Standard deviations must be derived in advance. But use of the GTL function EVAL avoids the need to create high and low limit variables.

```
   scatterplot x           = avisitn
               y           = mean_0 /
               xaxis       = x
               name        = 'G0'
               legendlabel = "%scan(&trt_list, 1, %str(|))"
               markerattrs = (size   = 8
                              symbol = circle
                              color  = blue)
               yerrorupper = eval(mean_0 + std_0)
               yerrorlower = eval(mean_0 - std_0)
               ;
```

This SCATTERPLOT statement charts data for the controls (Y=MEAN_0).

The primary axis is referenced (XAXIS=X).

```
   discretelegend "G0" "G1" / location  = inside
                              autoalign = (bottom)
                              border    = false ;
```

To position the legend with respect to this chart, the legend statement is included within this OVERLAY layout.

(Incidentally, a DISCRETELEGEND is for serial data. Another legend type CONTINUOUSLEGEND is available for color stripes.)

```
endlayout ;
```

Block closing tag.

GTL LOWER LAYOUT OVERLAY BLOCK

The lower chart is a convenient "device" to align automatically sample counts and visit labels. The Y values will be constants, the X values our visit numbers. This forms a grid that will adjust to the schedule of visits in our data.

Instead of plotting symbols, we plot a variable which holds the count of subjects for each treatment group at each visit. Naturally, we must prepare these variables in advance of generating the chart. Since counting is easy, I hope you will agree it is fair for SAS/GTL do the hard work of placing them on the page.

```
layout overlay /
```

Opening tag of LAYOUT block with arguments to follow.

```
   walldisplay = none
```

Chart wall refers to a box around the plot area. It's turned off because we're interested in a table of numbers.

```
   yaxisopts    = (offsetmin  = 0.3
                   offsetmax  = 0.3
                   type       = linear
                   linearopts = (tickvaluelist = (0 1)
                                 tickvalueformat = trtn. )
                   display    = (tickvalues)
                   tickvalueattrs = (size = 7pt )
                   label = ' ')
```

There are two treatment groups, so we arrange Y margin offsets to evenly space two rows of numbers.

The tick list consists of our treatment group constants.

The tick values are formatted so treatment group labels are displayed instead of zero and one.

No mention is made of the X-axis because we already configured this at the layout lattice level. Indeed, SAS would ignore XAXISOPTS because a COLUMNAXES block exists.

```
   ;
```

End of layout block options.

```
   scatterplot x               = avisitn
               y               = tx_1 /
               name            = 'N1'
               markercharacter = cnt_1 ;
   scatterplot x               = avisitn
               y               = tx_0 /
               name            = 'N0'
               markercharacter = cnt_0 ;
```

Two scatter plot statements for cases and controls respectively.

The plot option MARKERCHARACTER is the crucial bit. SAS will display variable values instead of symbols.

```
endlayout ;
```

Block closing tag.

This concludes the code walk through for Approach 3. See Appendix for the clean, ready–to–compile program code.

PREPROCESSING THE DATA

In elaborating the template, the needs and constraints of GTL have driven our assumptions of the input dataset. Yet, in all this programming, we have not strayed as far from our original data as you might have thought.

That's surely a good thing, lest the burden of reworking the data outweigh any benefits of GTL. How about this ?

```
proc summary data = work.adam nway missing ;
   class avisitn trtan ;
   var   aval ;
   output out = work.summary (drop = _:)
          mean = mean
          std  = std
          n    = n ;
run ;

%macro tr(v) ;
   proc transpose data   = work.summary
                  out    = work.tr&v
                  prefix = &v._ ;
      by avisitn ;
      id trtan ;
      var &v ;
      format trtan ;
   run ;
%mend ;

%tr(mean)
%tr(std)
%tr(n)
```

```
data work.mean_sd ;
   merge work.trmean work.trstd work.trn ;
   by avisitn ;

   retain tx_0 0 tx_1 1 ;       /* Treatment group constants for lower chart.     */
   length cnt_0 cnt_1 $10 ;     /* Counts translated to "N=0" formatted strings.  */
   array n n_: ;
   array c cnt_: ;

   do over n ;
      if not missing(n) then c = 'N=' || trim(left(put(n, best.))) ;
   end ;
run ;
```

```
proc print data = work.mean_sd ;
run ;
```

```
    WORK.MEAN_SD: Source data after summary and transpose. For GTL/Proc SGRENDER.
-----------------------------------------------------------------------------------
Obs  AVISITN   MEAN_0   MEAN_1   STD_0    STD_1   N_0  N_1  TX_0  TX_1  CNT_0  CNT_1
-----------------------------------------------------------------------------------

 1       0    52.3919  53.0188  17.6863  17.6976   31   51    0     1   N=31   N=51
 2      10    57.5246  50.9804  24.2356  18.4387   24   45    0     1   N=24   N=45
 3      20    53.4100  53.0490  19.7771  18.2700   22   40    0     1   N=22   N=40
 4      40    53.0113  54.2000  25.1030  18.5240   16   31    0     1   N=16   N=31
 5      50    46.1257  55.0194  20.5661  16.0268    7   18    0     1   N=7    N=18
 6      60    57.6200  52.1300  25.0103  13.5643    4   10    0     1   N=4    N=10
 7      75             29.3800            5.0063         2    0     1          N=2

-----------------------------------------------------------------------------------
```

Gee, that was quick. Now for the fun stuff !

RENDERING THE CHART

Once the GTL code has been complied using Proc TEMPLATE, we may bring together template and data to generate the chart image.

Proc SGPLOT automatically enables ODS graphics. But with GTL, we must do this explicitly.

```
ods graphics on / imagefmt   = png
                  imagename  = "approach3"
                  border     = off
                  height     = 600px
                  width      = 800px
                  ;
```

Initialize ODS graphics.

Statement options include file format, image dimensions, and file name.

```
ods listing     / image_dpi  = 200
                  style      = default
                  ;
```

Open one (or several) ODS destinations. The “listing” destination produces an image file only.

Statement option STYLE= permits selection among several built–in color schemes (ANALYSIS, DEFAULT, JOURNAL, STATISTICAL), or a user–defined style. Sample images in this paper have all referenced the DEFAULT style.

```
proc sgrender template = approach3
              data     = work.mean_sd ;
   dynamic _ti = "My dynamic title" ;
run ;
```

Merge data with template using Proc SGRENDER to generate the chart.

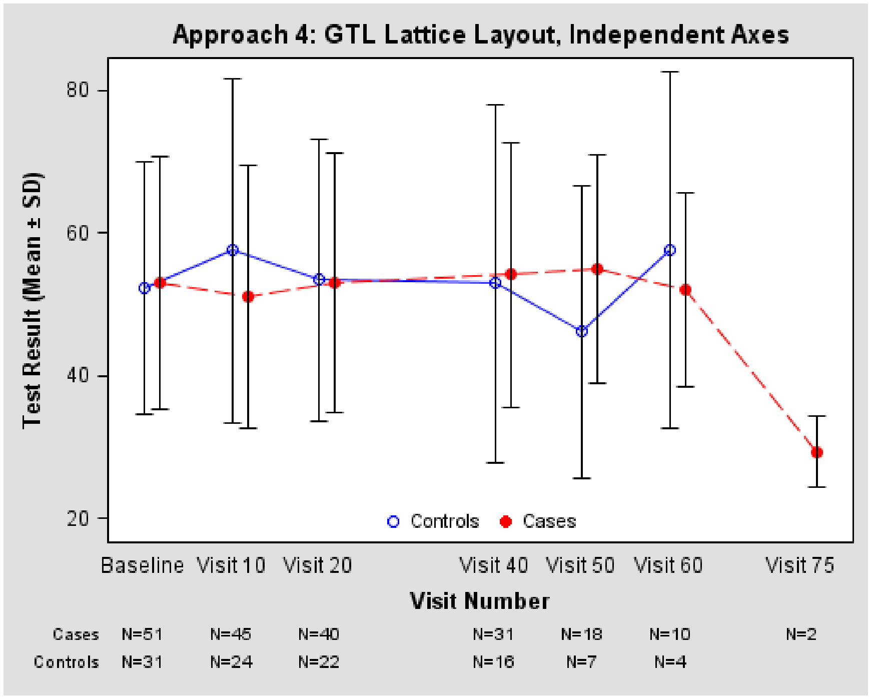
```
ods graphics off ;
```

Lastly, close the ODS destination.

```
ods listing close ;
```

Voila !

APPROACH 4: GTL WITHOUT COLUMNAXES BLOCK



There remain few details to complete our work.

In approach 3, counts and data series came together nicely with the unified X-axis. But the resulting presentation was non–standard. The visit axis came below the table of counts, when normally counts would be at the very bottom.

Compare the arrangement at left.

Also, we neglected to connect the visits. This is something Proc SGPLOT does automatically, but in GTL we must code for it.

In order to place the axis and tick labels at the edge of the chart, we have to relinquish the convenience of a unified X-axis. Think of it this way. The COLUMNAXES definition applied to the pair of charts, so the display attributes (axis line, tick labels, and axis label) would necessarily be at the bottom of the stack. Conversely, neither chart described by LAYOUT OVERLAY blocks could have an X–axis definition because the COLUMNAXES block held precedence.

Well then, what is the impact of omitting the COLUMNAXES block? . . . . More responsibility for the programmer.

We now have three axis definitions to maintain in synchrony—primary and secondary axes on upper chart plotting the data series, and primary axis on lower chart with the counts. Look at these revisions ...

```
proc template ;
define statgraph approach4 ;
   begingraph ;

      dynamic _ti ;
      entrytitle _ti ;

      layout lattice /
             columns          = 1
             rows             = 2
             rowweights       = (0.9 0.1)
             rowgutter        = 0
             ~~columndatarange = union~~ ;

      ~~columnaxes~~ ;
         ~~columnaxis~~ ... ;
      ~~endcolumnaxes~~ ;
                                                 /* Upper chart plots means with SD.    */
      layout overlay /
/* NEW CODE: X-axis options with value list added to LAYOUT OVERLAY block. */
/***/    xaxisopts = (offsetmin  = 0.05
/***/                 offsetmax  = 0.07
/***/                 type       = linear
/***/                 linearopts = (tickvaluelist   = (&vis_list)
/***/                               tickvalueformat = vis.)
/***/                 label      = 'Visit Number'
/***/                 display    = (label line tickvalues) )
```

LAYOUT LATTICE block options are similar, except the COLUMNDATARANGE option is no longer applicable. This option had signaled the GTL compiler to expect a COLUMNAXES block.

The COLUMNAXES block is entirely eliminated.

```
          /* No change to options for the secondary X2-axis or Y-axis.         */
          /* Compare margin offsets for XAXISOPTS and X2AXISOPTS.               */

x2axisopts = (offsetmin  = 0.07
              offsetmax  = 0.05
              linearopts = (tickvaluelist = (&vis_list))
              display    = none)
yaxisopts  = (offsetmin  = 0.05
              offsetmax  = 0.03
              label      = "Test Result (Mean (*ESC*){unicode 'B1'x} SD)" )

;         /* End of block options for upper chart. */

          /* No change to SCATTERPLOT statements drawing means and error bars. */

   scatterplot x           = avisitn
               y           = mean_1 /
               xaxis       = x2
               name        = 'G1'
               legendlabel = "%scan(&trt_list, 2, %str(|))"
               markerattrs = (size   = 8
                              symbol = circlefilled
                              color  = red)
               yerrorupper = eval(mean_1 + std_1)
               yerrorlower = eval(mean_1 - std_1)
               ;

          /* Of course, XAXIS = X now refers to the axis definition             */
          /* within this chart rather than the column axis.                     */

   scatterplot x           = avisitn
               y           = mean_0 /
               xaxis       = x          /* note */
               name        = 'G0'
               legendlabel = "%scan(&trt_list, 1, %str(|))"
               markerattrs = (size   = 8
                              symbol = circle
                              color  = blue)
               yerrorupper = eval(mean_0 + std_0)
               yerrorlower = eval(mean_0 - std_0)
               ;
```

When Proc SGPLOT generates a vertical line VLINE chart, it uses a combination of GTL SCATTERPLOT and SERIESPLOT statements. Check this for yourself with use of TMPLOUT= procedure option.

The scatter plot is category oriented, and has the functionality to draw error bars. The series plot is longitudinal, and joins consecutive points with straight line segments.

To connect the X–Y pairs, we need to add SERIESPLOT statements.

```
/* NEW CODE. Connect the points. */
/***/      seriesplot  x         = avisitn
/***/                  y         = mean_0 /
/***/                  xaxis     = x
/***/                  lineattrs = (pattern = solid
/***/                               color   = blue)
/***/                  ;

/***/      seriesplot  x         = avisitn
/***/                  y         = mean_1 /
/***/                  xaxis     = x2
/***/                  lineattrs = (pattern = mediumdash
/***/                               color   = red)
/***/                  ;

   discretelegend "G0" "G1" / location  = inside
                              autoalign = (bottom)
                              border    = false ;

endlayout ;
```

Series plots do not display markers (unless requested with the MARKERS plot option). Because our scatter plots already draw markers, we don't need them here.

Series plot lines are color coordinated with the scatter plot markets using the LINEARATTRS option.

Block closing tag.

In the lower layout block for sample counts, we add matching X–axis options.

```
                                                /* Lower chart has table of counts.    */
         layout overlay /

/* NEW CODE: X-axis options match scaling in the upper chart. */
/***/                     xaxisopts  = (
/***/                                   offsetmin  = 0.05
/***/                                   offsetmax  = 0.07
/***/                                   type       = linear
/***/                                   linearopts = (tickvaluelist   = (&visit_numlist))
/***/                                   display    = none  /* No axis line or tick labels */
/***/                                  )

                                       /* No change to Y-options and wall display.      */
                         walldisplay = none
                         yaxisopts   = (offsetmin  = 0.3
                                        offsetmax  = 0.3
                                        type       = linear
                                        linearopts = (tickvaluelist = (0 1)
                                                      tickvalueformat = trtn. )
                                        display    = (tickvalues)
                                        tickvalueattrs = (size = 7pt )
                                        label      = ' ')

         ;                             /* End of block options for lower chart.         */

                                       /* SCATTERPLOT statements not changed.           */

            scatterplot x                = avisitn
                        y                = tx_1 /
                        name             = 'N1'
                        markercharacter = cnt_1 ;

            scatterplot x                = avisitn
                        y                = tx_0 /
                        name             = 'N0'
                        markercharacter = cnt_0 ;

         endlayout ;                            /* Close overlay layout block.        */

      endlayout ;                               /* Close lattice layout block.        */
   endgraph ;
end ;
run ;
```

With these additions to the GTL template, we are ready to generate the target graphic using summarized data as in Approach 3. See Appendix for the clean, ready–to–compile program code.

CONCLUSIONS

Proc SGPLOT and GTL can create high quality, customizable graphs. Proc SGPLOT is quick, easy to use, and easy to remember. GTL is more verbose but highly structured and flexible.

All SAS programmers are urged to try these tools on their next assignment. Any investment in learning will be well rewarded by the end results.

FURTHER READING

These papers are worthwhile for beginner and experienced programmers alike:

- "Effective Graphics Made Simple Using SAS/GRAPH® SG Procedures"
  D. Heath, PharmaSUG 2008, Paper SA06.
- "Getting Started with ODS Statistical Graphics in SAS® 9.2"
  R.N. Rodriguez, SAS Global Forum 2008, Paper 305.
- "Modifying ODS Statistical Graphics Templates in SAS® 9.2"
  W.H. Kuhfeld, SAS Global Forum 2009, Paper 323.
- "Using PROC SGPLOT for Quick High-Quality Graphs"
  L.D. Delwiche, SAS Global Forum 2009, Paper 158.
- "When Simpler is Better – Visualizing Laboratory Data Using "SG Procedures"
  W. Cheng, PharmaSUG 2009, Paper PO22.
- "An efficient way to create graphs in SAS 9.2: Utilizing SG procedures and GTL"
  Y.Z. Ling, NESUG 2010, Paper GR10.
- "The Graph Template Language and the Statistical Graphics Procedures: An Example-Driven Introduction"
  W.F. Kuhfeld, PharmaSUG 2010, Paper TU-SAS01.

The SAS documentation is indispensible to make full use of the many statement options and their syntax. Each volume is nicely hyperlinked for easy navigation. Find them at http://support.sas.com/documentation.

- SAS/GRAPH 9.2: Statistical Graphics Procedures Guide
- SAS/GRAPH 9.2: Graph Template Language Reference
- SAS/GRAPH 9.2: Graph Template Language User's Guide

CONTACT INFORMATION

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

Anthony L. Feliu
Genzyme Corporation
500 West Kendall Street
Cambridge, MA 02142
Tel: (617) 768-9296
E-mail: ANTHONY.FELIU@GENZYME.COM

APPENDIX: FINAL CODE FOR APPROACH 2

```
proc transpose data = work.adam        /* Create AVAL0, AVAL1 with result values for    */
               out  = work.transpose   /* treatment groups 0 and 1, repectively.        */
               prefix = aval ;
   by  subjid avisitn ;
   var aval ;
   id  trtan ;
   format trtan ;
run ;

proc sql noprint ;                     /* Get all visit numbers for tick list.          */
   select distinct avisitn
       into :vis_list separated by ' '
   from work.transpose
   order by 1 ;
quit ;

proc sgplot data = work.transpose ;
   yaxis  label = 'Test Result (Mean %sysfunc(byte(0177)) SD)' ;

   xaxis  type      = linear                 /* Primary axis (bottom).             */
          values    = (&vis_list)            /* Tick list to label every visit.    */
          offsetmin = 0.05
          offsetmax = 0.07
          label     = 'Scheduled Visit' ;

   x2axis type      = linear                 /* Secondary axis (top).              */
          values    = (&vis_list)            /* X2AXIS synchronizeed with XAXIS.   */
          offsetmin = 0.07
          offsetmax = 0.05
          display   = (nolabel noticks novalues) ;

   vline avisitn / response    = aval1       /* Plot visit means and SD for cases.  */
                   stat        = mean
                   limits      = both
                   limitstat   = stddev
                   x2axis
                   markers
                   markerattrs = (size = 9 symbol = circle)
                   lineattrs   = (thickness = 2px)
                   limitattrs  = (thickness = 1px)
                   legendlabel = "Cases" ;

   vline avisitn / response    = aval0       /* Plot visit means and SD for controls.*/
                   stat        = mean
                   limits      = both
                   limitstat   = stddev
                   markers
                   markerattrs = (size = 9 symbol = circle)
                   lineattrs   = (thickness = 2px)
                   limitattrs  = (thickness = 1px)
                   legendlabel = "Controls" ;

   keylegend / location = inside
               position = bottomleft
               noborder
               title = '' ;

   format avisitn vis. ;
run ;
```

APPENDIX: FINAL CODE FOR APPROACH 3

```
ods path reset ;

ods path (prepend) work.templat (update) ;

proc template ;
define statgraph approach3 ;                    /* GTL template name is "APPROACH3".   */
   begingraph ;

      dynamic _ti ;                             /* Chart title passed at run time.     */
      entrytitle _ti ;

                                                /* Create canvas for two charts.       */
      layout lattice /
             columns         = 1
             rows            = 2
             rowweights      = (0.9 0.1)        /* Relative chart sizing 90%-10%.      */
             rowgutter       = 0
             columndatarange = union ;

                                                /* One X-axis definition will apply    */
                                                /* across both charts.                 */
         columnaxes ;
            columnaxis / offsetmin  = 0.05
                         offsetmax  = 0.07
                         type       = linear
                         linearopts = (tickvaluelist   = (&vis_list)
                                       tickvalueformat = vis.)
                         label      = 'Visit Number'
                         display    = (line label tickvalues)
                         displaysecondary = none ;
         endcolumnaxes ;

                                                /* This chart displays data series.    */
                                                /* X2AXIS applies to this chart only.  */
         layout overlay / x2axisopts = (offsetmin = 0.07
                                        offsetmax = 0.05
                                        linearopts = (tickvaluelist = (&vis_list))
                                        display   = none)
                          yaxisopts  = (offsetmin = 0.05
                                        offsetmax = 0.03
                                 label = "Test Result (Mean (*ESC*){unicode 'B1'x} SD)" )
                         ;

            scatterplot x           = avisitn
                        y           = mean_1 /
                        xaxis       = x2
                        name        = 'G1'
                        legendlabel = "%scan(&trt_list, 2, %str(|))"
                        markerattrs = (size   = 8
                                       symbol = circlefilled
                                       color  = red)
                        yerrorupper = eval(mean_1 + std_1)
                        yerrorlower = eval(mean_1 - std_1)
                        ;

            scatterplot x           = avisitn
                        y           = mean_0 /
                        xaxis       = x
                        name        = 'G0'
                        legendlabel = "%scan(&trt_list, 1, %str(|))"
                        markerattrs = (size   = 8
                                       symbol = circle
                                       color  = blue)
                        yerrorupper = eval(mean_0 + std_0)
                        yerrorlower = eval(mean_0 - std_0)
                        ;
            discretelegend "G0" "G1" / location  = inside
                                       autoalign = (bottom)
                                       border    = false ;

         endlayout ;

                                                /* This chart displays counts.         */
         layout overlay / walldisplay = none
                          yaxisopts   = (offsetmin   = 0.3
                                         offsetmax   = 0.3
                                         type        = linear
                                         linearopts  = (tickvaluelist = (0 1)
                                                        tickvalueformat = trtn. )
                                         display     = (tickvalues)
                                         tickvalueattrs = (size = 7pt )
                                         label       = ' ')
                          ;

            scatterplot x               = avisitn
                        y               = tx_1 /
                        name            = 'N1'
                        markercharacter = cnt_1 ;

            scatterplot x               = avisitn
                        y               = tx_0 /
                        name            = 'N0'
                        markercharacter = cnt_0 ;

         endlayout ;                            /* Close overlay layout block.         */

      endlayout ;                               /* Close lattice layout block.         */
   endgraph ;
end ;
run ;
```

HOW TO GENERATE CHART WITH TEMPLATE

```
                                                /* Activate ODS graphics feature.      */
ods graphics on / imagefmt  = png
                  imagename = "approach3"
                  border    = off
                  height    = 600px
                  width     = 800px ;

                                               /* Open ODS destination. Choose style.  */
ods listing / image_dpi = 200
              style     = default ;

/* FOR DATA SUMMARIZATION, SEE MAIN NARRATIVE OF THIS PAPER. */
                                               /* Create the chart image.              */
proc sgrender template = approach3
              data     = work.mean_sd ;
   dynamic _ti = "My dynamic title" ;
run ;

                                               /* Close ODS destination.               */
ods graphics off ;

ods listing close ;
```

APPENDIX: FINAL CODE FOR APPROACH 4

```
ods path reset ;

ods path (prepend) work.templat (update) ;

proc template ;
define statgraph approach4 ;                         /* GTL template name is "APPROACH4".   */
   begingraph ;

      dynamic _ti ;                                  /* Chart title passed at run time.     */
      entrytitle _ti ;

                                                     /* Canvas prepared for two charts.     */
     layout lattice /
            columns     = 1
            rows        = 2
            rowweights  = (0.9 0.1)
            rowgutter   = 0 ;

                                                     /* This chart plots means with SD.     */
                                                     /* Axis definitions are self-contained.*/
      layout overlay /
         xaxisopts = (offsetmin  = 0.05
                      offsetmax  = 0.07
                      type       = linear
                      linearopts = (tickvaluelist   = (&vis_list)
                                    tickvalueformat = vis.)
                      label      = 'Visit Number'
                      display    = (label line tickvalues) )
         x2axisopts = (offsetmin  = 0.07
                       offsetmax  = 0.05
                       linearopts = (tickvaluelist = (&vis_list))
                       display    = none)
         yaxisopts  = (offsetmin  = 0.05
                       offsetmax  = 0.03
                       label      = "Test Result (Mean (*ESC*){unicode 'B1'x} SD)" )
                      ;

                                                     /* Plot markers and error bars.        */
            scatterplot x           = avisitn
                        y           = mean_1 /
                        xaxis       = x2
                        name        = 'G1'
                        legendlabel = "%scan(&trt_list, 2, %str(|))"
                        markerattrs = (size   = 8
                                       symbol = circlefilled
                                       color  = red)
                        yerrorupper = eval(mean_1 + std_1)
                        yerrorlower = eval(mean_1 - std_1)
                        ;
            scatterplot x           = avisitn
                        y           = mean_0 /
                        xaxis       = x           /* note */
                        name        = 'G0'
                        legendlabel = "%scan(&trt_list, 1, %str(|))"
                        markerattrs = (size   = 8
                                       symbol = circle
                                       color  = blue)
                        yerrorupper = eval(mean_0 + std_0)
                        yerrorlower = eval(mean_0 - std_0)
                        ;
```

```
                                                    /* Draw lines to connect the points.   */
          seriesplot  x          = avisitn
                      y          = mean_0 /
                      xaxis      = x
                      lineattrs = (pattern = solid
                                   color   = blue)
                      ;

          seriesplot  x          = avisitn
                      y          = mean_1 /
                      xaxis      = x2
                      lineattrs = (pattern = mediumdash
                                   color   = red)
                      ;

          discretelegend "G0" "G1" / location  = inside
                                     autoalign = (bottom)
                                     border    = false ;

       endlayout ;

                                               /* Lower chart has table of counts.        */
                                               /* Axis definitions also self-contained.   */
                                               /* X-scaling and margins match upper chart. */
       layout overlay / xaxisopts = (offsetmin  = 0.05
                                     offsetmax  = 0.07
                                     type       = linear
                                     linearopts = (tickvaluelist   = (&visit_numlist))
                                     display    = none )
                        walldisplay = none
                        yaxisopts = (offsetmin  = 0.3
                                     offsetmax  = 0.3
                                     type       = linear
                                     linearopts = (tickvaluelist = (0 1)
                                                   tickvalueformat = trtn. )
                                     display    = (tickvalues)
                                     tickvalueattrs = (size = 7pt )
                                     label      = ' ')
                        ;

          scatterplot x               = avisitn
                      y               = tx_1 /
                      name            = 'N1'
                      markercharacter = cnt_1 ;

          scatterplot x               = avisitn
                      y               = tx_0 /
                      name            = 'N0'
                      markercharacter = cnt_0 ;

       endlayout ;                          /* Close overlay layout block.         */

     endlayout ;                            /* Close lattice layout block.         */
   endgraph ;
end ;
run ;

/* TO USE THIS TEMPLATE, FOLLOW "HOW TO" AS IN APPROACH 3. */
```