PharmaSUG 2013 - Paper DG03

SAS GTL: Improving Patients Safety and Study Efficiency

Masaki Mihaila, Medivation, Inc, San Francisco, CA

ABSTRACT

Due to the high cost and time required for clinical trials, optimizing data processing is highly desirable. A graphical presentation of clinical data and relationships in Programmed Assisted Patient Narratives (PANs) is more efficient and easier to understand. Graphical tools can shorten the lengthy drug development process while focusing attention on the review and communication of salient safety information across all company areas and functions. This paper describes how to achieve end-user friendly graphical integration into PANs by using SAS 9.3 ODS and Graphical Template Language (GTL) for 64-bit Windows. This paper only focuses on the RTF output destination.

INTRODUCTION

Traditionally, tabular outputs are used to ensure clinical data integrity and drug safety. This approach is time and resource intensive. More cost-effective way is naturally demanded. As a SAS programmer, a fundamental task is to support other functions by providing innovative tools using programming techniques to improve efficiency. An well-designed graphical patient profile can transform enormous amount of clinical data to a single concise report containing all relevant information, thus optimizing the time and effort that the reviewers put into to ensuring clinical data integrity and drug safety. The use of standardized SDTM data is necessary to enable the reuse of these tools for multiple studies.

This paper primarily focuses on how to create a highly customized, integrated visuals for safety clinical data such as EX (drug exposure), CM (Concomitant Medications), and AE (Adverse Events), which are summarized and presented together in a single display (Figure 1.). Each vector represents one event and events are arranged in chronological order summarized by unique term (e.g. EXTRT for EX; AETERM for AE; CMDECOD for CM). Reviewers can cross-check and identify the study drug relationship to adverse events and concomitant medications much more quickly than using traditional methodologies. Annotation tool is utilized to improve the visual representation of the clinical data.

Secondly, this paper explains the GTL template for other finding domains (VS, Figure 2 and LB, Figure 3). Through the use of multiple cells (one cell=one test), individual tests changes can be visualized regardless of differences in variance. This paper also discusses the appropriate procedure to display critical information such as low/high and toxicity grade for lab results.

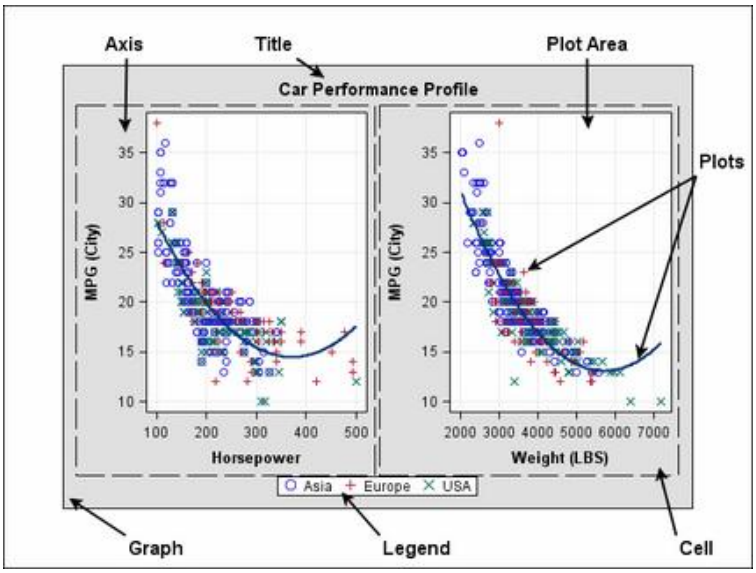
TOOLS

The graphs presented in the following paragraphs are generated by the SGRENDER ODS GRAPHICS procedures, which uses templates to produce graphical output. These templates are created with GTL, a comprehensive language that can create a variety of custom graphs more effortlessly than SAS/GRAPH and other ODS GRAPHICS procedures. It is flexible enough to accommodate almost any type of graph by varying layout features. By knowing the structure and appearance of a desired graph, a template can be created by using the following steps:

Here are the steps:

1. Define the structure using the GTL syntax and compile it, which will be stored in a STATGRAPH template library.
2. Create the appropriate input data source for the graph that can be rendered by SGRENDER procedure.
3. Run the SGRENDER procedure.

It is important to understand the graph area to use GTL to build a user-defined template. The graph area is the overall output area produced from all of the statements that are nested in a BEGINGRAPH statement block. This overall area is segmented into one or more titles, one or more footnotes, and one or more cells. A cell contains plot area, axes, plots, and legends. Here is the basic anatomy of ODS graphics from the SAS online help.



The below syntax is the simplified version of the templates used to create Figure 1, 2 and 3. The layout of each graph is different but the concept is the same. Notice that GTL basic syntax has a specific way of ending each statement. For example, BEGINGRAPH ends with ENDGRAPH, LAYOUT ends with ENDLAYOUT, and so on. Each statement comes with options, some options will be explained in this paper. A complete list of the options can be found at the SAS website.

The box within LAYOUT LATTICE/ENDLAYOUT is the segment unique in each template. All three are multi-cell graphs; in other words, multiple LAYOUT OVERLAYs are nested within LAYOUT LATTICE.

GTL Basic Syntax (LAYOUT LATTICE is the outermost template in the layout):

```
Proc Template;
Define statgraph _TemplateName_;
Mvar;
Nmvar;
Dynamic;
Begingraph/_options_;
Entrytitle;
  Layout Lattice/_options_;
    Layout Overlay/_options_;

     ---GTL Statements---
          ex. Scatterplot, Vectorplot, Referenceline, Entry

    Endlayout;                    Multiple layout overlay can be nested here
  Endlayout;
Entryfootnote;
Endgraph;
End;
Run;
```

GTL supports the DYNAMIC statement for declaring dynamic variables, and the MVAR and NMVAR statements for declaring macro variables. These are available to you to avoid repeating to redefine and recompile the template for the same type of graph with different variables. MVAR resolves to strings; NMVAR converts the supplied value to a numeric token. Dynamic and macro variables are initialized differently. For macro variables, use any macro variable in the current macro table (local or global) at run time. For dynamics, use DYNAMIC statement inside PROC SGRENDER. Values for dynamics that resolve to column names or strings should be quoted. Numeric values should not be quoted. (Note that when a macro variable reference is preceded with ampersand (&), the reference will be resolved when the template is compiled, not at execution.) Following convention, dynamic variable is capitalized followed by underscore and macro variables are capitalized in the template.

```
Layout Lattice/Columns=1 Rows=3 ROWWEIGHTS=_ROWWGT ROWGUTTER=0 BORDER=FALSE;
Layout Overlay/Xaxisopts=(Linearopts=(Tickvakuesequence=(Start=0 End=XMAXVAL Increment=10)
                                      Viewmin=XMINVAL
                                      Viewmax=XVIEWMAX)
                          Offsetmin=0.05
                          Offsetmax=0
                          Display=None)
                  Yaxisopts=(Linearopts=(Tickvakuesequence=(Start=0 End=100 Increment=1)
                                      Viewmin=_YMIN
                                      Viewmax=_YMAX)
                          Offsetmin=0.1
                          Offsetmax=0.05
                          Display=None);
```

LAYOUT LATTICE can create advanced multi-cells and its options define features of the overall layout and its cells.

- COLUMNS and ROWS specifies the number of columns and rows in the layout.
- ROWWEIGHTS specifies the fractional proportion of each cell relative to the overall grid height, not including headers, sidebars, and row axes.
- ROWGUTTER specifies amount of empty space between the rows. If you have multi-columns layout, use COLUMNGUTTER instead.
- BORDER specifies whether a border is drawn around the layout.

LAYOUT OVERLAY builds a composite from one or more GTL statements. In this paper, this layout is nested in a LATTICE layout, but it could be an entire graph. The composite provides contents for one cell in the parent layout. The axes can be customized using the following:

- XAXISOPTS and YAXISOPTS specify its axis option.
- LINEAROPTS specifies the features for a standard numeric interval axis.
- OFFSETMIN and OFFSETMAX reserve areas at the maximum and minimum end of the axis respectively.
- DISPLAY controls which axis features are displayed on the primary axis.

When the GTL template code is submitted and stored properly, it is ready for the SGRENDER procedure. In order to produce the graph, *_InputData_* must be bound to *_TemplateName_* template. The following example code reads in all observations in _InputData_ then passes the data object and template definition to the graph renderer. The renderer will produce an image file which is automatically sent to the ODS destination.

```
Ods Listing Close;
Ods Rtf File='xxx.rtf';
Ods Graphics On/Width=15in Height=5.5in;
Proc Sgrender Data=_InputData_ Template=_TemplateName_;
     Dynamics _Rowwgt="xxxxx" _Ymin=xxxxx _Ymax=xxxxx;
Run;
Ods Graphics Off;
Ods Rtf Close;
Ods Listing;
```

As seen in the above code, ODS graphics needs to be enabled . It is typically disabled by default. With this invocation, ODS Graphics produces graphs in standard image file formats, and appearance and layout of the graphs are controlled by ODS styles and templates, respectively. In this regard, creating graphical output is as easy as creating tables and listings, and integrating tables and graphs in one output becomes easier.

SAFETY DATA INTEGRATED TO ONE GRAPH

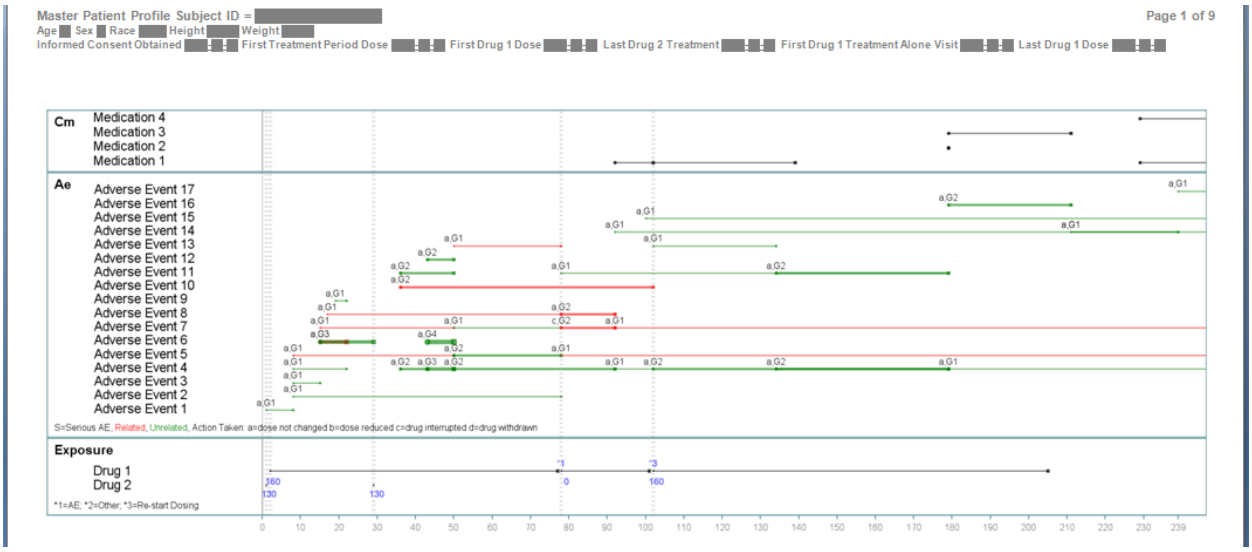


Figure 1. Integrated Safety Data Graphical PAN (Three-cell graph produced using LAYOUT LATTICE statement as the outermost template in the layout)

Figure 1 is the integrated graph that summarizes all safety-regarding exposure-response information in one place. In an ongoing clinical trial, monitoring patient safety is substantially essential. Since this is a time-consuming procedure, it would be ideal to see exposure, concomitant medications and adverse events under the same timeline. The goal is to present this information such a way that end-users can easily understand the relationship between exposure and AE and CM responses, and to translate data/tables as much as possible into graphical presentation. In that sense, annotation tool is of great use. The information annotated into the graph is listed below and will be explained for more details later. (Notice that the AE toxicity levels are distinguished by thickness of lines along with annotation.)

Annotated information:

- [SDTM.AE.AEREL] AE relation to Exposure
- [SDTM.AE.AESER] Serious AE
- [SDTM.AE.AEACN] AE Action Taken
- [SDTM.AE.AETOXGR] AE toxicity level
- [SDTM.EX.EXDOSE] Dose of study drug exposure
- [SDTM.EX.EXADJ] Reason of dose change

The LAYOUT OVERLAY statement builds a 2-D, single-cell graph by overlaying the results of the statement that are contained in the layout cell. The three layouts are nested in a LATTICE LAYOUT. In this example, each cell consists of combinations of contained statements such as VECTORPLOT, SCATTERPLOT, and REFERENCELINE. VECTORPLOT draws a line from the start to end day. SCATTERPLOT displays a symbol at start day and end day. All the annotations are created by using options from the SCATTERPLOT statement. All axes settings are specified using the options on the LAYOUT OVERLAY statement. The ENTRY statement is used to insert 'Cm', 'Ae', and 'Exposure' as texts in each cell, AE and EX footnotes, and text within the plot area.

AEYN	AEDECOD	AEST1	AEST2	AEST3	AEEN1	AEEN2	AEEN3	AETXT1	AETXT2	AETXT3
1	Adverse Event 1	1			8			a,G1		
2	Adverse Event 2	8			78			a,G1		
3	Adverse Event 3	8			15			a,G1		
4	Adverse Event 4	8	36	43	22	43	50	a,G1	a,G2	a,G3
5	Adverse Event 5	8	50	78	50	78		a,G1	a,G2	a,G1
6	Adverse Event 6	15	15	43	22	29	50	a,G3	a,G3	a,G4
7	Adverse Event 7	15	50	78	50	78	92	a,G1	a,G1	c,G2

Display 1. Partial AE input data

Using the AE domain as an example. Display 1 contains the partial data of AE part that feeds into SGRENDER procedure. The data structure must be horizontal. SDTM.AE needs to be transposed, so AE days are in columns. The total number of columns created in the transposed AE data is equal to the maximum number of AE events. The image below should describe how the data was transposed more clearly.

AESEQ	AEDECOD	AESTDY	AEENDY
1	NAUSEA	1	8
2	NAUSEA	8	15
3	NAUSEA	29	41

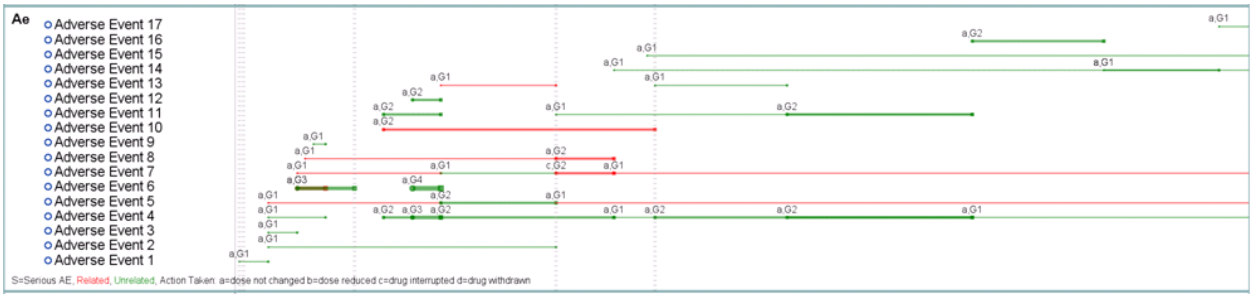
Transposed to

AEDECOD	AEST1	AEST2	AEST3	AEEN1	AEEN2	AEEN3
Nausea	1	8	29	8	15	41

Display 2

In the post-transposed data [Display 1], pair of AESTx (xth AE start day) and AEENx (xth AE end day) comprise the xth event vector for each AEDECOD (dictionary-derived term). AETXTx summarizes the characteristics of the xth pair regarding serious AE, action taken, and toxicity grade, and is annotated to xth vector. A footnote, "*S=Serious AE, Related, Unrelated, Action Taken: a=dose not changed b=dose reduced c=drug interrupted d=drug withdrawn*" is provided using the ENTRY statement. AEYN becomes Y axis value in the graph and AEDECOD texts are inserted as marker labels at positions of (-40, AEYN) using the SCATTERPLOT DATALABEL option. (The details of options are noted in Table 1.) The AEDECOD text insertion is done by writing the following code. Note that data marker color needs to be set to white or set SIZE=0pt for data point markers to be invisible. If forgotten, then default symbol for data markers appears as shown in Display 3. The value of X could be adjusted to any appropriate constant value in order to accommodate the length of text strings.

```
SCATTERPLOT X=-40 Y=AEYN  /Datalabel=AEDECOD
                           Datalableposition=Right
                          Markerattrs=(Color=White Size=0pt)
                           Datalabelattrs=(Color=Black Size=7);①
```



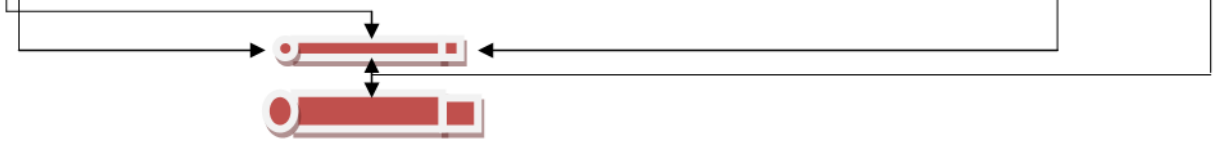
Display 3

The code to display each event is shown below. Note that in order to distinguish color and thickness in line and markers, we need to create the corresponding variables for each case listed:

- Unrelated & AE Toxicity grade 1
- Unrelated & AE Toxicity grade 2
- Unrelated & AE Toxicity grade 3
- Unrelated & AE Toxicity grade 4
- Unrelated & AE Toxicity grade 5
- Related & AE Toxicity grade 1
- Related & AE Toxicity grade 2
- Related & AE Toxicity grade 3
- Related & AE Toxicity grade 4
- Related & AE Toxicity grade 5

The COLOR option is used to distinguish each AE.AEREL value. The LINEARATTRS.THICKNESS option is used to distinguish the line feature in terms of AE.AETOXGR.

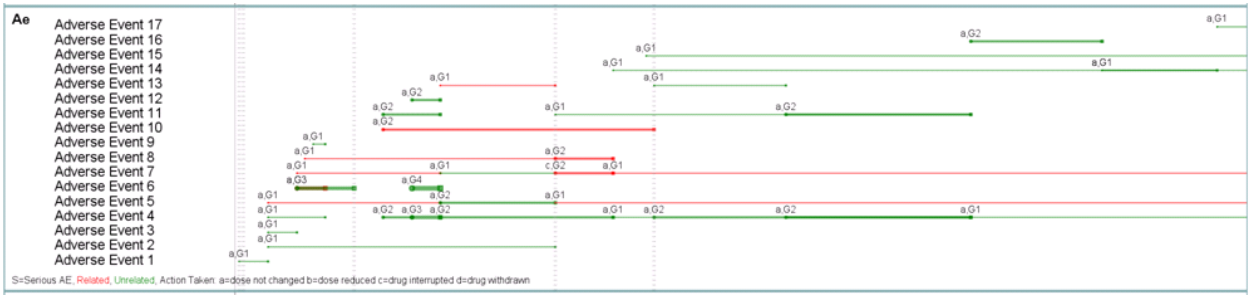
```
VECTORPLOT X=AEEN1 Y=AESEQ XOrigin=AEST1 YOrigin=AESEQ /Arrowheads=False
                                                       Lineattrs=(Color=Red Thickness=3px)
                                                       Datatransparency=0.4;②
SCATTERPLOT X=AEST1 Y=AESEQ /Makerattrs=(Color=Red Size=3pt Symbol=Circlefilled)
                             Datatransparency=0.4;
SCATTERPLOT X=AEEN1 Y=AESEQ /Makerattrs=(Color=Red Size=3pt Symbol=Squarefilled)
                             Datatransparency=0.4;
```



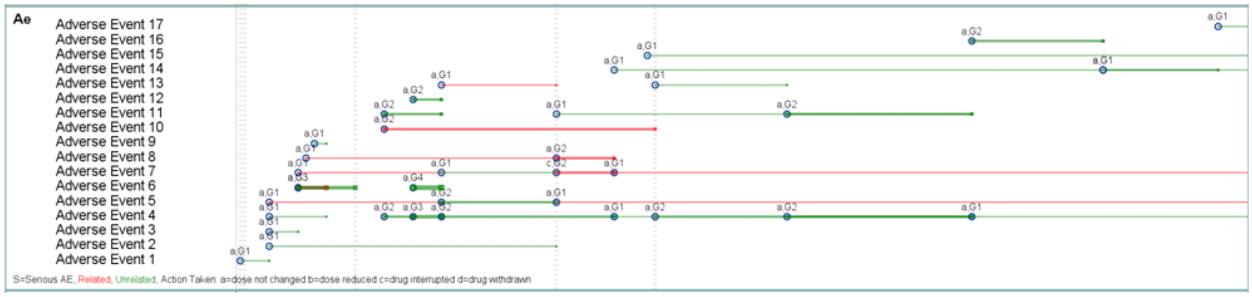
Display 4

The following code puts the annotation (AETXTx values) for each AE event, such as 'a, G1', 'a, G2, and so on. As in AEDECOD text, the DATALABEL option is used to display these texts on the top of data points. The attributes of data markers are controlled by the MARKERATTRS option. Display 5 shows the result if the following code runs, which sets the attributes such way that the data points are invisible. Otherwise, the result will be Display 6, where the default symbols show up on the data points.

```
SCATTERPLOT X=AEST1 Y=AESEQ /Datalabel=AETXT1 Datalabelposition=Top
                            Markerattrs=(Size=0pt Color=White)
                            Datalabelattrs=(Color=Black Size=5pt);
```



Display 5



Display 6

PLOTTING VITAL SIGNS AND LABORATORY TESTS

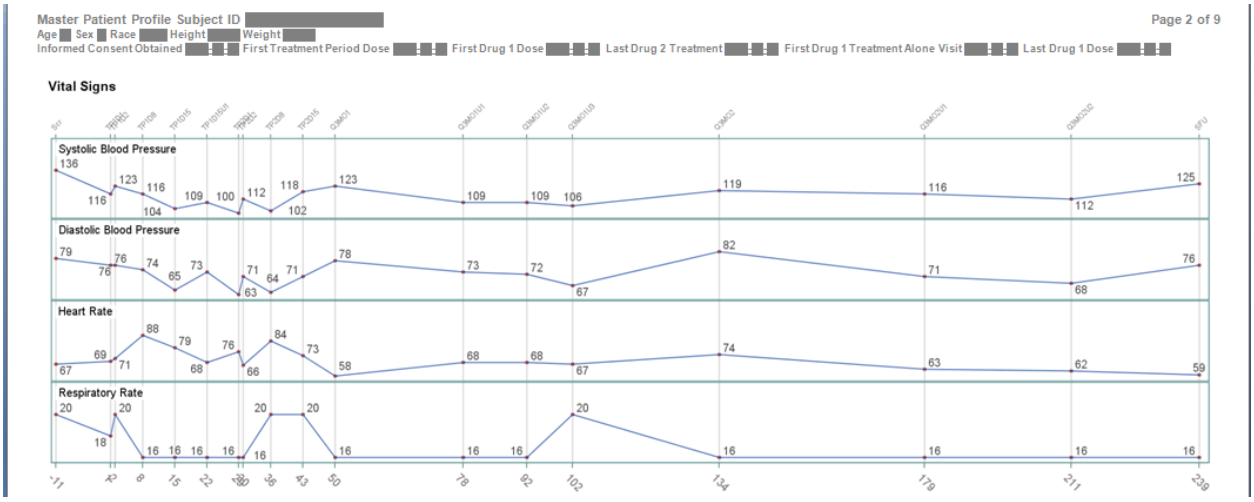


Figure 2. Vital Signs

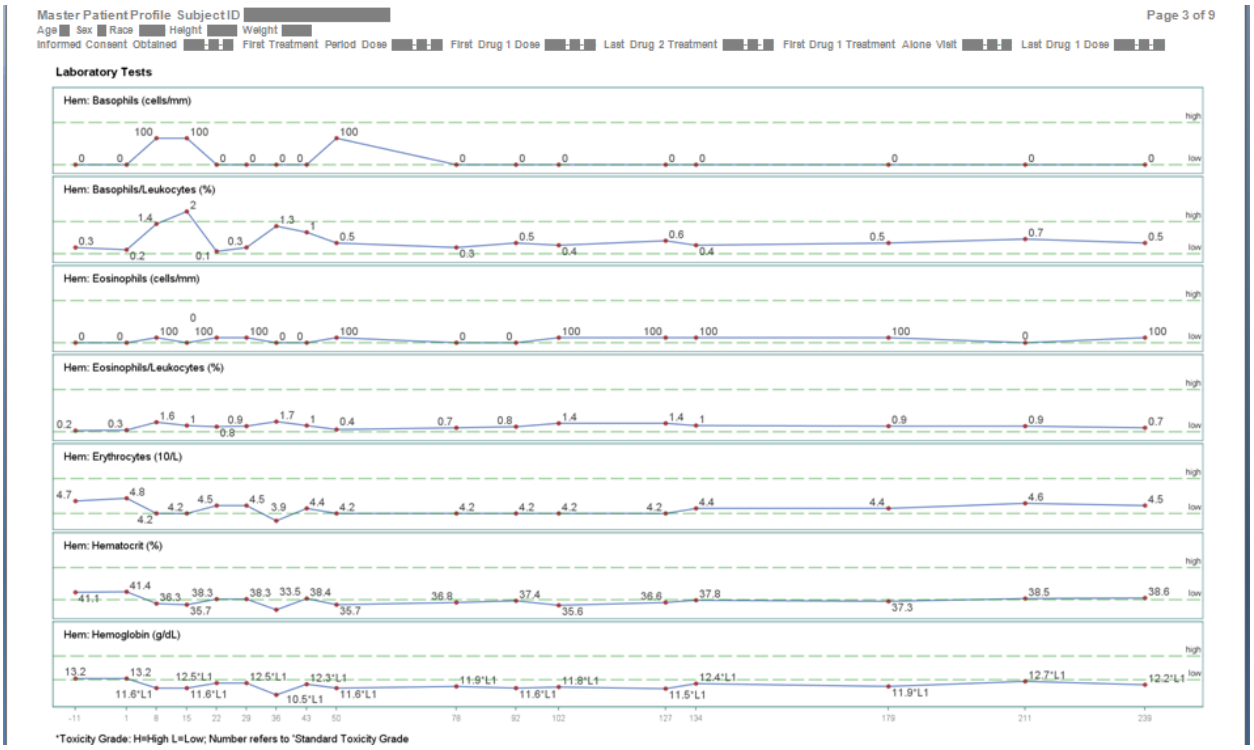


Figure 3. Laboratory Tests

The secondary object of this paper is demonstrating a VS and LB finding domain plot. Both Figure 2 and 3 are constructed upon the same concept. One significant difference from Figure 1 is these plots use the SERIESPLOT to draw and connect lines between data points. The numeric results are annotated on each data point instead of using the Y axis. Values are precisely specified and data reviewers can view all of this data on one page.

As seen in Figure 2, a secondary X axis of visit can be added in addition to the primary X axis (days) to enhance the graph. This can be done easily using the DISPLAYSECONDARY option, which has the same syntax as the DISPLAY option. It takes NONE|STANDARD|ALL or (LABEL, LINE, TICKS, TICKVALUES).

- LABEL : displays the axis label
- LINE : displays the axis line
- TICKS : displays the tick marks
- TICKVALUES : displays the values that are represented by tick marks

```
LAYOUT OVERLAY /Xaxisopts=(Linearopts=(Tickvaluelist=XTICKVAL
                                       Tickvalueformat=SVISF.
                                       Tickvaluefitpolicy=Rotate
                                       Viewmax=VSXMAX
                                       Displaysecondary=(Ticks Tickvalues)
                                       Display=None
                                       Tickvalueattrs=(Color=Gray Size=5pt)
```

Like the AE domain, the LB domain needs to be pre-processed. The table below shows how the SDTM.LB dataset (left) is transposed into the SGRENDER input dataset (right). LBTESTCD=HGB has a column for numeric result, lower limit, higher limit, and a test summary that becomes a data label for each data point (X_HGB). After this data and the GTL template code below are passed to the graph renderer, the final cell in Figure 3 will be created.

LBTESTCD	LBDY	LBSTRESN	LBSTTNLO	LBSTNRHI	X_LBNVAL
HGB	-2	12.3	13	17	12.3"L1
HGB	1	11.7	13	17	11.7"L1
HGB	8	11	13	17	11"L1
HGB	15	12	13	17	12"L1
HGB	22	12.1	13	17	12.1"L1
HGB	29	10.5	13	17	10.5"L1
HGB	41	11.1	13	17	11.1"L1
HGB	71	11.3	13	17	11.3"L1
HGB	106	11.8	13	17	11.8"L1
HGB	134	12.2	13	17	12.2"L1
HGB	162	10.4	13	17	10.4"L1
HGB	169	10.1	13	17	10.1"L1

Transposed

LBDY	HGB	L_HGB	H_HGB	X_HGB
-2	12.3	13	17	12.3"L1
1	11.7	13	17	11.7"L1
8	11	13	17	11"L1
15	12	13	17	12"L1
22	12.1	13	17	12.1"L1
29	10.5	13	17	10.5"L1
41	11.1	13	17	11.1"L1
71	11.3	13	17	11.3"L1
106	11.8	13	17	11.8"L1
134	12.2	13	17	12.2"L1
162	10.4	13	17	10.4"L1
169	10.1	13	17	10.1"L1

Display 7

After SDTM.LB is transposed, each test (*TESTx*) has its own variable named after the test code (LBTESTCD) and comes with additional variables for low (LB.LBSTNRLO → L_ *TESTx)* and high (LB.LBSTNRHI → H_ *TESTx)* from normal ranges and X_ *TESTx,* which is a pre-defined variable to annotate each data point. To compose X_ *TESTx,* the numeric values are concatenated with toxicity (L or H) and its grade from SDTM.LB domain (LBTOX and LBTOXGR). Data point labeling aids end-users to see the test results and whether they are abnormal.

```
Layout Overlay / Xaxisopts=(Linearopts=(Tickvaluelist=_XTICKVAL
                 Tickvalueformat=Best3.
                 Tickvaluefitpolicy=Rotate)
                 Offsetmin=0.02
                 Offsetmax=0.05
                 Display=(Tickvalues Ticks)
                 Tickvalueattrs=(Color=Gray Size=5.7pt)
                 Display=None)

                  Yaxisopts=(Offsetmin=0.1
                             Offsetmax=0.4
                             Display=None);
```

⇐ Used for the very last block

⇐ Used for the rest blocks

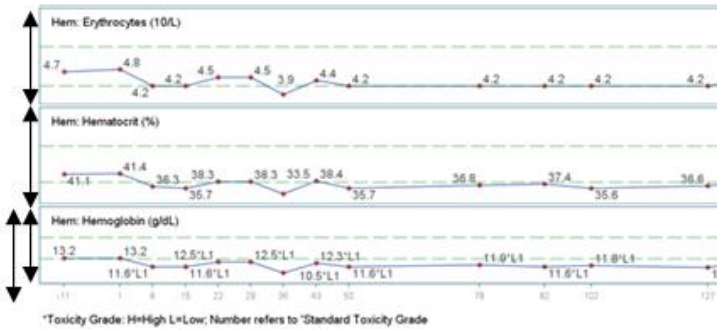
```
Entry LBTEST/Autoalign=(Topleft) Textattrs=(Size=6pt) Opaque=True; ③ [a]
Seriesplot X=Lbdy Y=Testx/Display=All
                         Smoothconnect=False
                         Datalabel=X_Test1
                         Datalabelattrs=(Size=6pt)
                        Markerattrs=(Size=2pt Color=Brown Symbol=Circlefilled)
                         Datatransparency=0.4;④

Referenceline Y=Eval(Mean(L_Testx))/Datatransparency=0.5 Clip=False
                                    Lineattrs=(Color=Green Pattern=4)
                                    Curvelabel='low'
                                    Curvelabelattrs=(Size=5pt Weight=Normal);⑤ [b]

Referenceline Y=Eval(Mean(H_Testx))/Datatransparency=0.5 Clip=False
                                    Lineattrs=(Color=Green Pattern=4)
                                    Curvelabel='high'
                                    Curvelabelattrs=(Size=5pt Weight=Normal); ⑤ [b]
Endlayout;
```

a. The ENTRY statement inserts the name labels of vital signs and lab tests at '*top left'* corner in each cell.
b. The REFERENCE statement draws low and high reference lines (dot lines in Figure 3).
   The lines instantaneously deliver the information and achieve good quality of data checking tool as compared to line-by-line data checking or generating listing output.

Notice that GTL summary statistics function, eval(mean()) is used to construct reference lines for 'low' and 'high', which are supplied at run time. Because all the time points share the same low and high reference values for the same LBTESTCD (Display 7) and it would be trivial to use all the repeated records. Without using the summary function, the reference lines get much thicker as shown below due to the fact that so many lines are repeatedly displayed for no reason. Also notice that somehow high and low reference lines disappeared from the plot area.

Display 8

Here is a small technique; as explained in the TOOL paragraph, the fractional proportions (defined in ROWWEIGHTS option) relative to the overall grid height does not include row axes area (ticks, tickvalues, and labels). So whenever there are cells that come with row axis, the fractional proportion in ROWWEIGHTS=() needs to be adjusted in order to have them evenly drawn by the same size in plot area. If not appropriately adjusted, Display 9 will be the result. The plot area in the last cell is shorter than that of the others.



Display 9

- TICKVALUELIST option displays a list of lab test days that is attached to X axis and can be called because the list of days varies from patient to patient.
- DISPLAY option contains a list of options if the cell comes with axis information; otherwise DISPLAY is set to NONE to suppress any axis feature to appear.
- SERIESPLOT is a multi-functional GTL statement. It can draw lines, add markers, and insert labels. The lines can be smoothly connected for appropriate cases using SMOOTHCONNECT option. In this paper, all the vertices are simply connected with straight lines using the default setting.
- CURVELABEL option labels the reference lines, and the attributes are defined by CURVELABELATTRS option.

As mentioned in the INTRODUCTION, there is an advantage of having multiple cells for these finding domains; that is, by separating each test from others, individual changes (increases and decreases) become quite apparent. There are so many lab tests in the dataset and they are varied in terms of their amount and unit. It would be very difficult to capture each individual changes and characteristic over time if they are on the same axes. Although some lab tests are known to behave in a certain way; in the big scheme of things, it is harder to study each lab's behavior using the graph with all the lab tests displayed together. Thus, this type of display is more desirable for data reviewers' standpoint.

Table 1. Options used in Figure 1 ~ 3

	Parental Statement	Options	Description
①	SCATTERPLOT	DATALABEL	Specifies a column for marker labels
		DATALABELPOSITION	Specifies the location of the data labels relative to the markers • AUTO (default) | TOPRIGHT | TOP | TOPLEFT | LEFT | CENTER | RIGHT | BOTTOMLEFT | BOTTOM | BOTTOMRIGHT
		MARKERATTRS	Specifies the attributes of the data markers
		DATALABELATTRS	Specifies the color and font attributes of the data labels
②	VECTORPLOT	ARROWHEADS	Specifies whether arrowheads are displayed on the vectors • False • True=default
		LINEATTRS	Specifies the properties of the series line
		DATATRANSPARENCY	Specifies the degree of the transparency of the vector line and arrow • 0=default • 0 (opaque) to 1 (entirely transparent)
③	ENTRY	AUTOALIGN	Specifies whether the entry is automatically aligned within its parent when nested within an overlay-type layout • None • Auto • Topleft, Top, Topright, Left, Center, Right, Bottomleft, Bottom, Bottomright
		OPAQUE	Specifies whether the entry background is opaque (TRUE) or transparent (FALSE=default)
④	SERIESPLOT	DISPLAY	Specifies additional feature to display with the series line • Standard: displays a series line without markers • All: displays a series line with markers • Markers: displays a series line with markers
		SMOOTHCONNECT	Specifies that a smoothed line passes through all vertices • False=default • True

	Parental Statement	Options	Description
⑤	REFERENCELINE	CLIP	Specifies whether the data for the reference line or lines are considered when determining the data ranges for the axes • False=default (The data for the line contributes to data range for each axis. Each axis might be extended to force the display of the line.) • True
		CURVELABEL	Specifies a label for the reference line or lines
		CURVELABELATTRS	Specifies the color and font attributes of the reference line label(s)

CONCLUSION

The tools presented in this paper for patient profiles demonstrate a new, powerful graphical aid for clinical trials. SAS GTL is able to create customized templates to summarize a large amount of information at a glance. The ODS Graphics facility allows us to use ODS styles to control the general appearance and consistency of all graphs and tables, which is a very useful feature since most of the patient profiles contain both graphs and tables. Effective graphical images can promote clear and concise communication within and across the company functions. The rich content and visual clarity offered by graphics helps end-users quickly access information needed for their investigation, and identify data relationship across the domains such as revealing patterns, identifying differences, and expressing uncertainty. As a result, with this tool, the time and effort for conducting clinical trial should decrease significantly.

REFERENCES

SAS® 9.3 2011. Graph Template Language User's Guide Cary, NC, USA: SAS Institute Inc.

RECOMMENDED READING

- http://support.sas.com/resources/papers/proceedings09/323-2009.pdf
- http://support.sas.com/resources/papers/proceedings12/280-2012.pdf
- http://www.pharmasug.org/proceedings/2012/TA/PharmaSUG-2012-TA09.pdf

CONTACT INFORMATION

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

Name: Masaki Mihaila
Enterprise: Medivation Inc.
Address: 525 Market St.
City, State ZIP: San Francisco, CA 94105
Work Phone: 415-829-4134
E-mail: masaki.mihaila@medivation.com