PharmaSUG 2018 - Paper EP-17

Creating and Customizing Graphics using Graph Template Language

Yanmei Zhang, Saihua Liu, Titania Dumas-Roberson, Grifols Inc

ABSTRACT

Graph Template Language (GTL) is a powerful language in SAS 9.4. PROC TEMPLATE in GTL can create customized graphs. The Graphics Template Language can also be used in conjunction with special DATA step features to produce graphs independently. This paper introduces the GTL with clinical trial examples. In addition, it provides some methods to solve problems we often meet when we create graphs. Such as, label is too long to fit in one line, and legend is truncated, and so on. Creating complicated graphs by using an annotate data set is discussed in the paper, too.

INTRODUCTION

The need for informative graphs becomes important in the clinical trial or manuscript development process. Statistical programmers and statisticians decide what kind of data should be displayed in the graph to represent more information together. GTL provides powerful syntax for programmers to create the graphs what they desire. Programmers have more options when they use the GTL. This paper represents some problems we meet and solutions for these problems when we create graphs. Specifically this paper includes how to use the ROWHEADERS block to split the long axis label and how to fix the truncated legend. In addition, it introduces how to use the annotate data set to display text above the bar chart.

STRUCTURE OF A BASIC GRAPH TEMPLATE

Two basic step processes are needed to create a graph using GTL. They involve the TEMPLATE and SGRENDER procedures. The code below outlines the basic template definition.

```
proc template;
  define statgraph template-name;
   begingraph;
     GTL-statements;
   endgraph;
  end;
run;
```

PROC TEMPLATE statement compiles and saves an ODS template for GRAPH. The DEFINE statement starts the template definition block, and specifies the template type STATGRAPH and the template name. The END statement terminates the DEFINE block. The BEGINGRAPH statement starts the graph statement block. The ENDGRAPH statement terminates the graph statement block. GTL statements are placed inside the graph statement block.

You can generate a graph from a graph template by utilizing the SGRENDER procedure. The code below is to create the graph in the end.

```
proc sgrender data = data-source template = template-name;
run;
```

These two steps above are necessary when generating the graph using the GTL.

MAKING LONG AXIS LABELS FIT

The label is truncated if it is too long to fit in the allotted space. SAS 9.4 user's guide says LABELFITTPOLICY = SPLIT or the SHORTLABEL=option can remedy this issue. A shortened label is

displayed no matter the label is. However, in clinical studies, we need long labels to express more information for reviewer. LABELFITTPOLICY = SPLIT does not work using SAS 9.4, but the ROWHEADER block can make it possible.

The SAS code below combines the steps above to generate a series figure. LABEL = option in the YAXISOPTS below is too long and LABEL is truncated. Figure 1 did not display all of labels.

```
  proc template;
   define statgraph  out;
    begingraph / border=false designwidth=1300 designheight=500;
     entrytitle textattrs=(size=10pt);
      layout lattice / columns=2 rows=1 border=false opaque=false
                       columngutter=.25in
         layout overlay/ yaxisopts=( label="Total IgG Concentration (mg/dL) for subject 1022002"
                                     labelfitpolicy = split  )
                         xaxisopts=(label="Actual time (hrs post start of infusion)");
            seriesplot  x=x1 y=aval / name ="life1" group = trt break=false datalabel=n display=all;
          endlayout;
          layout overlay/ yaxisopts=(type = log logopts=(base=10 tickintervalstyle=LOGEXPAND
                                     minorticks=true viewmin=100 viewmax=10000)  label=" "  )
                          xaxisopts=( label="Actual time (hrs post start of infusion)");
            seriesplot   x=x1 y=aval / name ="life" group = trt break=false datalabel=n display=all;
         endlayout;
         sidebar;
            discretelegend "life" / border=false;
         endsidebar;
        endlayout;
   endgraph;
  end;
run;
```

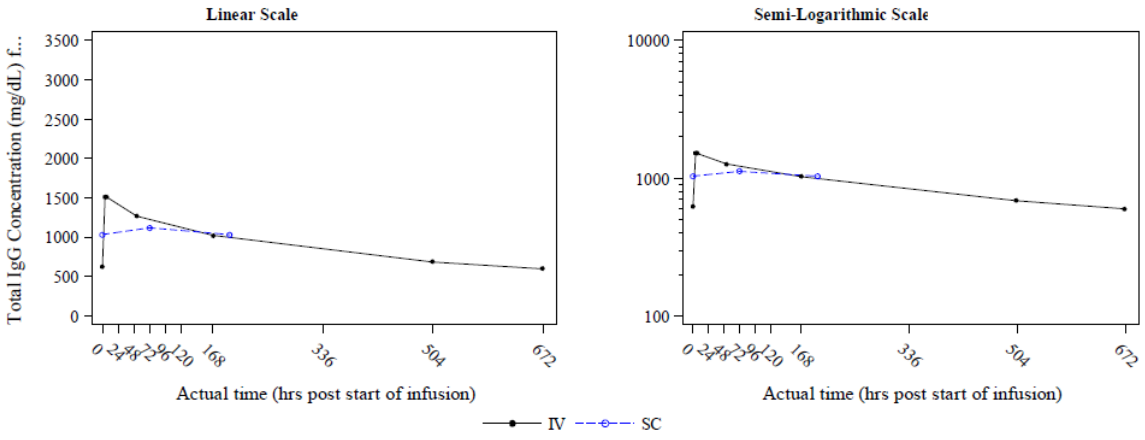

Figure 1. Serial Plot with Truncated Label

One way to display all labels in the row axes is to add the ROWHEADERS (like below) in the PROC TEMPLATE and remove the label information from the row axes. ROWHEADERS syntax is simple. It starts with ROWHEADERS and ends with ENDROWHEADERS. By nesting the ENTRY statements in the GRIDDED layouts, we can have multiple lines of text split exactly where we want and in any text style that we desire. Without the GRIDDED layouts, only one ENTRY statement could be used per row. The ROWHEADERS code below is inserted into the above SAS code after SIDEBAR but before

DISCRETELEGEND. Label of row axes is displayed correctly in the Figure 2. In this example, the use of row headers provided the desired flexibility over row axis labels.

```
rowheaders;
        layout gridded/columns = 2;
          entry "Total IgG Concentration (mg/dL)"/ textattrs=Graphlabeltext rotate = 90;
          entry "for Subject 1022002"/ textattrs=Graphlabeltext rotate = 90;
        endlayout;
endrowheaders;
```

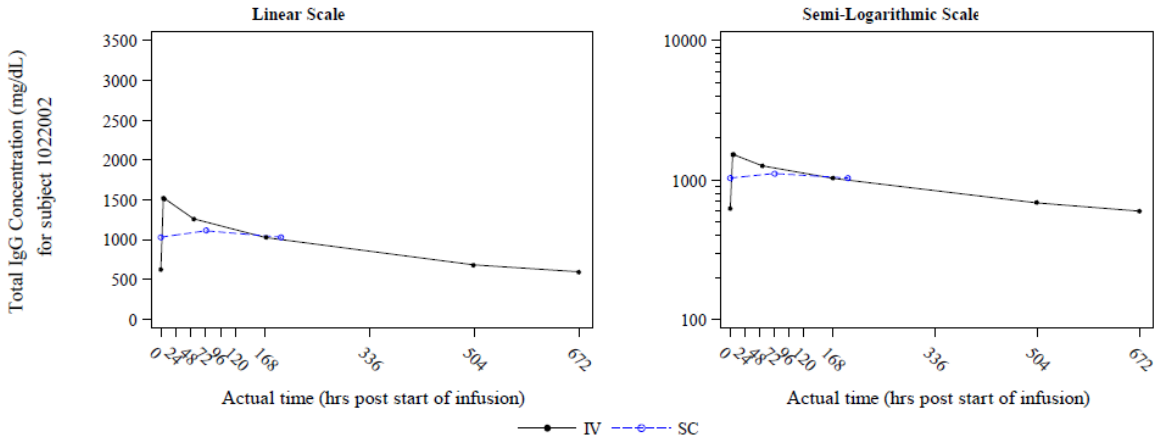

Figure 2. Serial Plot with completed Label

Adding legend to graph

Legend provides a key to the marker symbols, lines, and other data elements in a graph. PROC TEMPLATE does not automatically generate legends, but the mechanism for creating legends is simple and flexible. There are several legend placement options: LOCATION, HALIGN and VALIGN. LOCATION determines whether the legend is drawn inside the plot wall of cell, or outside the plot wall. LOCATION has two values, INSIDE and OUTSIDE. HALIGN determines horizontal alignment. It has TOP, CENTER and BOTTOM values to choose from. VALIGN determines vertical alignment. There are three options: TOP, CENTER and BOTTOM.

The code below generated the inside, top and right legend in the Figure 3. From this figure we can see Legend is truncated. The legend is not truncated when I checked figure using PROC SGRENDER procedure. The problem is from style in the ODS PDF.

```
proc template;
    define statgraph  out;
        begingraph / border=false designwidth=1500 designheight=600;
          layout lattice / columns=1 rows=1;
            layout overlay/ yaxisopts=(  label="Total IgG Concentration (mg/dL)")
                            xaxisopts=(linearopts=(viewmax=9 viewmin=1 tickvaluesequence=(start=0
                            end=9 increment=1)) label="Visit");
                seriesplot  x=visitn y=mean /group =trt name = "trt" break=false datalabel=n
```

```
                    display=all ;
                discretelegend "trt"/location =inside autoalign=(topright bottomleft) ;
            endlayout;
        endlayout;
    endgraph;
end;
run;
```

```
ods pdf file="H:\test\paper_graph3.pdf" bookmarkgen=Yes dpi=300  style=Styles.temp;
  ods listing close;
   proc sgrender data=gfinal template=out ;
   run;
 ods pdf close;
ods listing;
```

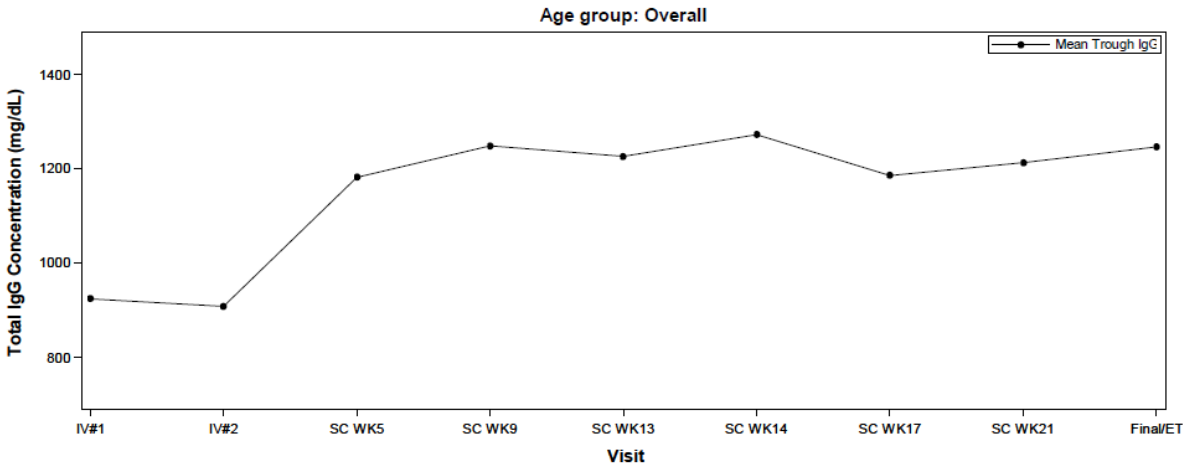

Figure 3. Serial Plot with Truncated Legend

I must update the style in the ODS PDF. Customizing styles and template is to avoid the truncated legend. The code below solved the truncated legend issue. Especially MARGINRIGHT and MARGINLEFT are set to 0.5 inches. The updated PROC TEMPLATE created Figure 4.

```
proc template;
   define style Styles.temp;
      parent = Styles.custom;
      style Body /
      marginright = 0.5in
      marginleft = 0.5in
      marginbottom = 1in
      fontfamily = "Times New Roman";
      style GraphData1 / ContrastColor=black MarkerSymbol="CircleFilled"    markersize=4px
                         Linestyle=1;
      style GraphBackground /background = colors('docbg');
   end;
run;
```

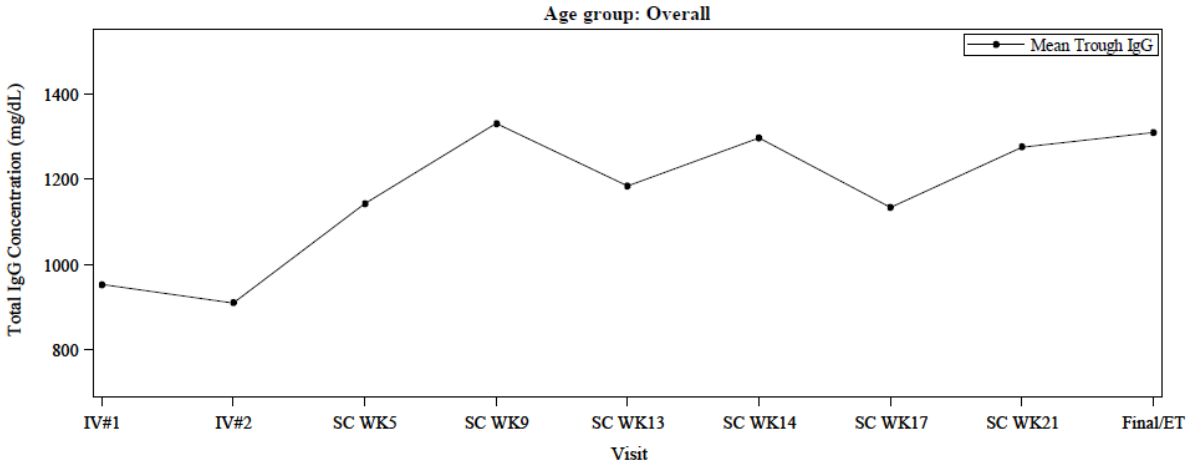

Figure 4. Serial Plot without Truncated Legend

Creating graph template with an annotate statement

A bar chart is generated by the BARCHART or BARCHARTPARM statement. These statements can draw vertical or horizontal bar charts. Since figures need more information in the graph, sometimes the BARCHART statement is not enough. The text or arrow or line is needed to add into graph. ANNOTATE statement can realize these. Annotation drawing space is DATAVALUE, the ANNOTATE statement is placed in the BARCHART/BARCHARTPARM statement layout block in the template. In this location, the DATAVALUE values are in the context of the BARCHART/BARCHARTPARM data. To render the graph with the annotations, the SGANNO=ANNO option is included in the SGRENDER statement.

The code below generated text above the bar chart in Figure 5.

```
data anno;
   retain function "text" drawspace "datavalue"
     textfont "Arial" textsize 8
     width 50 wifthunit "pixel"
     anchor "center"
     discreteoffset 0.1;
  set test1;
   x1=xvar;
   y1=yvar+2;
run;


proc template;
    define statgraph plotfreq;
    begingraph / border = false;
     layout lattice / rows=1  rowgutter=10;
           cell;
            layout overlay / xaxisopts=(label="Treatments"    labelattrs=(size=8pt))
                             yaxisopts=(label="% of Subjects in each Treatment Group"
                                        labelattrs=(size=8pt)
                                        linearopts = (tickvaluesequence=(start=0 end=100 increment =10)
                                       viewmin=0 viewmax=100)  display =(label ticks tickvalues));
              barchart x=xvar y=yvar /  orient=vertical;
              annotate;
            endlayout;
```

```
            endcell;
        endlayout;
      endgraph;
    end;
  run;

proc sgrender data=test1 template=plotfreq sganno = anno;
run;
```

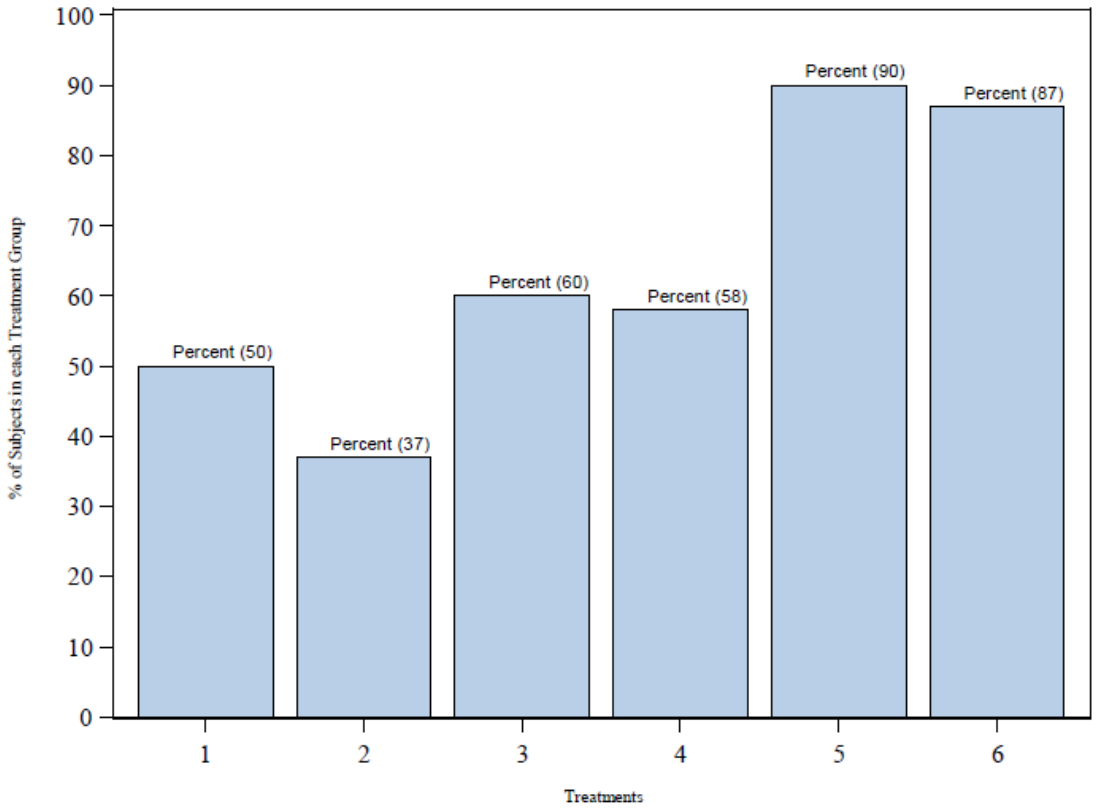

Figure 5. Bar Chart with Annotation above Bar

GTL is easy to fill the bar with the different colors using the attribute map. A discrete attribute map is to associate visual attributes to specific group values, which enables you to make plots that are consistent, regardless of the data order. The attribute maps allow you to map the data values to specific visual style options. This allowed to specify all of the facilities and define the color scheme associated with that facility from within the GTL. Shown below is only a subset of the code used to define the map. Adding the option IGNORECASE = True allows the map to match regardless of the case of the text, which is very handy if you work with inconsistent data. The DISCRETEATTRVAR statement is required for the attribute map to be used, ATTVAR will be used as the group option on the horizontal bar to call the map, VAR is the name of the variable where the data resides, and ATTRMAP is the name that you assigned the map. The code below can generate the Figure 6.

```
proc template;
    define statgraph plotfreq;
    dynamic  _maxy;
    begingraph / border = false;
```

```
        discreteattrmap name = "colors"/ignorecase = true;
                value "1" /fillattrs=(color=grey) ;
                value "2" /fillattrs=(color=black );
                value "3" /fillattrs=(color=purple);
                value "4" /fillattrs=(color=red);
                value "5" /fillattrs=(color=green);
                value "6" /fillattrs=(color=blue);
         enddiscreteattrmap;
        discreteattrvar attrvar=xcolor var=xvar attrmap='colors';
      layout lattice / rows=1  rowgutter=10;
       cell;
         layout overlay / xaxisopts=(label="Treatments" labelattrs=(size=8pt))
                          yaxisopts=(label="% of Subjects in each Treatment Group"
                                     labelattrs=(size=8pt) linearopts = (tickvaluesequence=(start=0 end=100
                                     increment =10)  viewmin=0 viewmax=100)  display =(label ticks tickvalues));
             barchart x=xvar y=yvar / group =xcolor orient=vertical barwidth = 0.7 ;
            annotate;
          endlayout;
       endcell;
      endlayout;
    endgraph;
  end;
run;
```

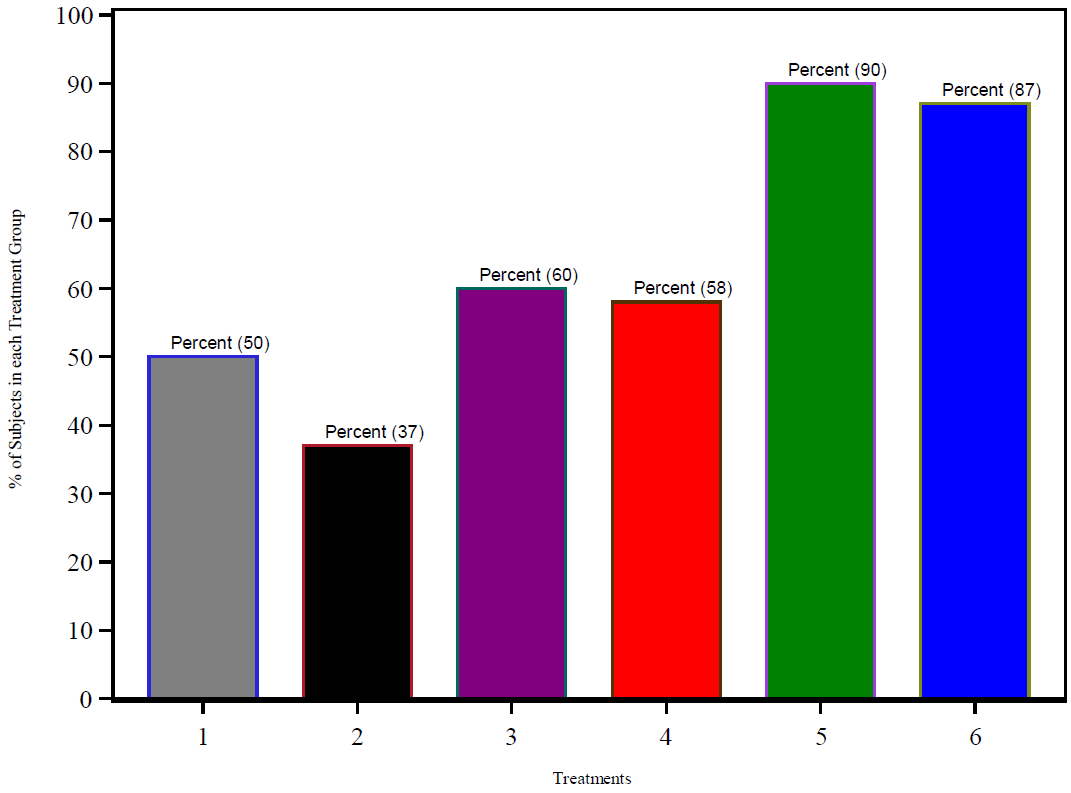

Figure 6. Bar Chart with Different Color for Each Bar

CONCLUSION

In summary, GTL is more powerful and flexible than the PROG GPLOT and PROC SGPLOT. PROC TEMPLATE in GTL can create customized graphs. The Graphics Template Language can also be used

in conjunction with special DATA step features to produce graphs independently. This paper is very useful for programmers beginning to learn to use GTL. This paper provided methods for some of problems programmers may meet, such as, label is too long to fit in one line, legend is truncated, etc. In addition, this paper introduces GTL combines with the annotated data set to create ideal graphs.

REFERENCES

SAS® 9.4Graph Template Language User's Guide

Pankhil Shah. "FILLPATTERNS in SGPLOT Graphs". *Proceedings of the PharmaSUG 2015 Conference.*

https://www.pharmasug.org/proceedings/2015/QT/PharmaSUG-2015-QT30.pdf

http://www.scsug.org/wp-content/uploads/2012/11/Picture-Perfect-Graphing-with-Statistical-Graphics-Procedures.pdf

Kristen Much, Kaitlyn McConville," Creating Sophisticated Graphics using Graph Template Language". *Proceedings of the PharmaSUG 2015 Conference.*
https://www.pharmasug.org/proceedings/2015/DV/PharmaSUG-2015-DV02.pdf

RECOMMENDED READING

- *Base SAS® Procedures Guide*

  SAS® For Dummies®

CONTACT INFORMATION

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

Name: Yanmei Zhang
Enterprise: Grifols Inc
Address: 79 TW Alexander Dr
City, State ZIP: Durham, NC 27709
Work Phone: 919-316-2278
E-mail: Yanmei.zhang@Grifols.com
Web: http://www.grifolsusa.com/en/web/eeuu#

Name: Saihua Liu
Enterprise: Grifols Inc
Address: 79 TW Alexander Dr
City, State ZIP: Durham, NC 27709
Work Phone: 919-316-2078
E-mail: saihua.liu@Grifols.com
Web: http://www.grifolsusa.com/en/web/eeuu#

Name: Titania Dumas-Roberson
Enterprise: Grifols Inc
Address: 79 TW Alexander Dr
City, State ZIP: Durham, NC 27709
Work Phone: 919-316-6153
E-mail: Titania.Dumas-Roberson@grifols.com
Web: http://www.grifolsusa.com/en/web/eeuu#