PT-05

Customizing a Multi-Cell Graph Created with SAS ODS Graphics Designer

Yanhong Liu, Cincinnati Children's Hospital Medical Center, Cincinnati, OH

ABSTRACT

Combining multiple graphs and/or statistical data tables into one graph is an effective way to compare research data side by side or to present a summarized report of related information together. The SAS® Graph Template Language (GTL) provides the ability to create such "multi-cell" graphs by using its powerful "building-block" syntax. In SAS® 9.3, GTL provides a new feature, "discrete attribute maps", which enables us to map visual attributes such as shape and color to input data values. You can add this statement block to the GTL code and customize the appearance of each cell of the multi-cell graph, thereby improving the overall graphical visualization, and allowing better interpretation of the graph. To avoid writing GTL code from scratch, we can use the ODS Graphics Designer which is based on Graph Template Language to create the graph, then copy the GTL code generated by the ODS Graphics Designer into the program editor for further customization.

INTRODUCTION

For one of our longitudinal studies, the investigator requested an enrollment/retention report of the patients' visit status. You can picture this as a multi-cell graph consisting of a stack bar graph to display the visit status (Completed, Missed, or Scheduled) at each visit point (1-year, 2-year, 3-year, 4-year and 5-year), a vertical bar graph showing the visit types (Home Visit, In Person, Online) within the completed visits at each visit point, and tables for statistical data as well. For improved graphical visualization and interpretation of the graph, we colored "Missing Visits" red, and also colored the vertical bar graph differently from the ones in the stack bar graph. (The final graph is in figure 6.)

The SAS® Graph Template Language (GTL) is very extensive and provides the ability to create such "multi-cell" graphs by using the LATTICE layout. Then the SGRENDER procedure applies the template to your data to create the output graphs. In SAS® 9.3, GTL introduces a new feature, "discrete attribute maps". By defining a discrete attribute map in the DISCRETEATTRMAP block, you have the ability to control the style element attributes for each possible value of a variable across graphs. In our GTL code, the discrete attribute map assigns a specific fill color for each value of the Status and Type variables.

However, Graph Template Language (GTL) involves a large number of statements and options. Instead of writing the code from scratch, we obtained the GTL code from the ODS Graphics Designer. The ODS Graphics Designer, which became available as part of Base SAS® in SAS 9.3, provides a GUI-based, interactive interface for designing graphs easily by using point-and-click technology. ODS Graphics Designer is based on the Graph Template Language; it generates GTL code once the graphic is made. The GTL code was copied and pasted into the SAS program editor, and the DISCRETEATTRMAP block and DISCRETEATTRVAR statement were added to the GTL code in order to create a customized multi-cell graph.

INPUT AND DERIVED SAS DATA SETS

The input data set was imported from a tracking database. For demonstration purposes, we kept 4 variables of interest and also modified the data values.

Table 1 presents information about the Input SAS table.

Variable Name	Variable Type	Description	Typical Values
ID	Numeric	Patient ID	1-200
Visit	Numeric	Patient annual visits	1-5
VisitStatus	Character	Visit Status	Completed; Missed; Scheduled
VisitType	Character	How was the completed visit done?	In Person; Home Visit; Online

Table 1. Data values in the input SAS data set

The final graph should include 4 “cells”: a stack bar graph for the Visit Status, a table of the descriptive statistic data for the Visit Status, a vertical bar graph for the Visit Type of completed visits, and a table of that descriptive statistic data. The 4 cells share the same x-axis (VISIT). Since the SGRENDER procedure doesn’t support multiple data sets, we have to use a single data set to hold all the data for those 4 graphs. By applying PROC FREQ and PROC TRANSPOSE and other procedures, along with DATA steps, we prepared the derived data set for the ODS GRAPH DESIGNER.

Table 2 shows the derived data set for the graph

Visit	VisitStatus	StatusCount	StatusStat	VisitType	TypeCount	TypeStat
1	Completed	169	169 (84.5%)	Online	40	40 (23.7%)
1	Missed	26	26 (13.0%)	Home Visit	1	1 (0.6%)
1	Scheduled	5	5 (2.5%)	In Person	128	128 (75.7%)
2	Completed	151	151 (75.5%)	Online	26	26 (17.2%)
2	Missed	22	22 (11.0%)	Home Visit	5	5 (3.3%)
2	Scheduled	27	27 (13.5%)	In Person	120	120 (79.5%)
3	Completed	101	101 (50.5%)	Online	20	20 (19.8%)
3	Missed	14	14 (7.0%)	Home Visit	10	10 (9.9%)
3	Scheduled	85	85 (42.5%)	In Person	71	71 (70.3%)
4	Completed	73	73 (36.5%)	Online	27	27 (37.0%)
4	Missed	12	12 (6.0%)	Home Visit	11	11 (15.1%)
4	Scheduled	115	115 (57.5%)	In Person	35	35 (47.9%)
5	Completed	26	26 (13.0%)	Online	4	4 (15.4%)
5	Missed	6	6 (3.0%)	Home Visit	7	7 (26.9%)
5	Scheduled	168	168 (84.0%)	In Person	15	15 (57.7%)

Table 2. Derived SAS data set “Allvisit_graph”

STEPS FOR CREATING A MULTI-CELL GRAPH WITH ODS GRAPHICS DESIGNER

ODS Graphics Designer became available as part of Base SAS® with SAS 9.3. You can start the application from the Tools > ODS Graphics Designer or run %sgdesign in SAS. Figure 1 shows the designer application interface.

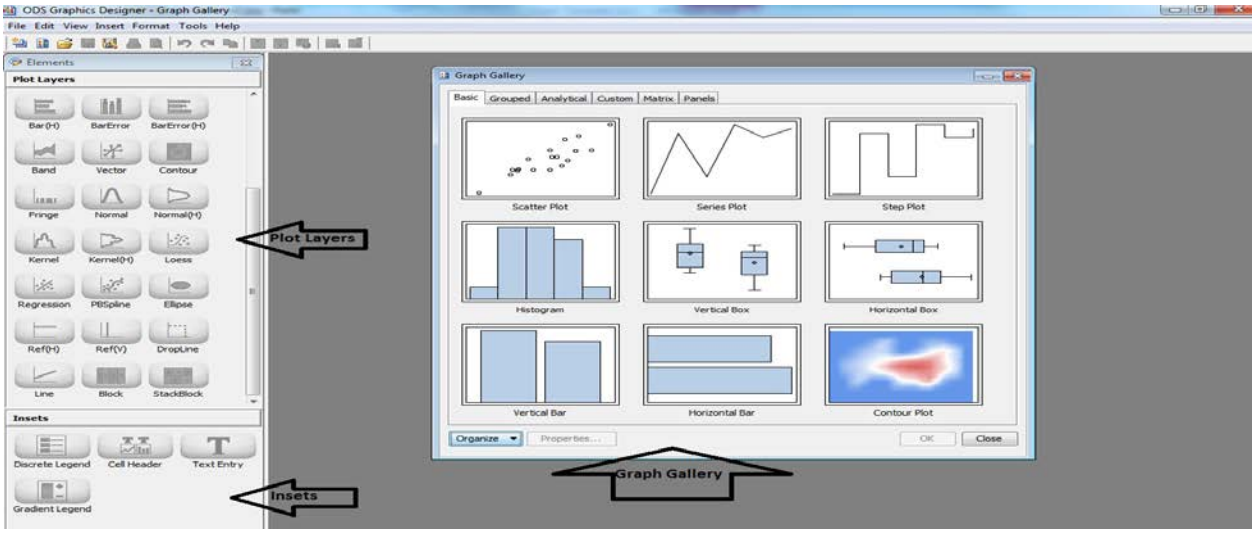

Figure 1. ODS Graphics Designer Application Interface

Below are the steps to create our multi-cell graph by using its point-and-click interaction.

1. Select Grouped Vertical Bar in Graph Gallery and click OK to launch the Assign Data dialog box
2. In the Assign Data dialog box, assign the data for the Visit Status stack bar graph (Figure 2.), then click OK to create our first cell graph (Figure 3.)

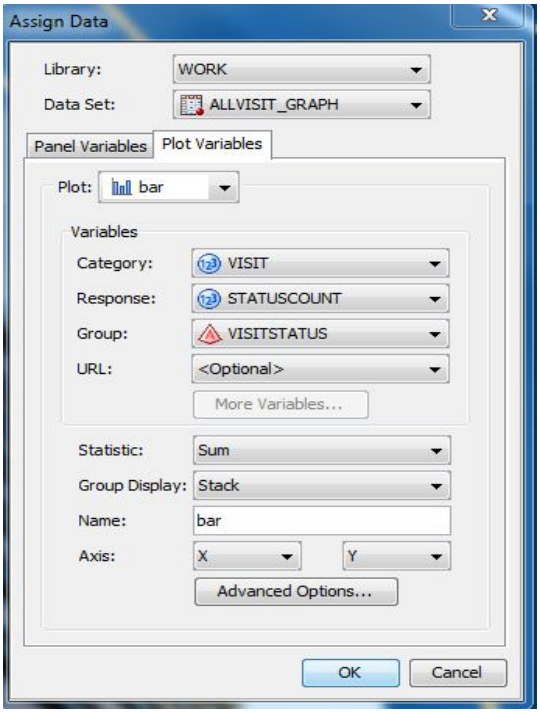

Figure 2. ODS Graphics Designer Assign Data Dialog Box

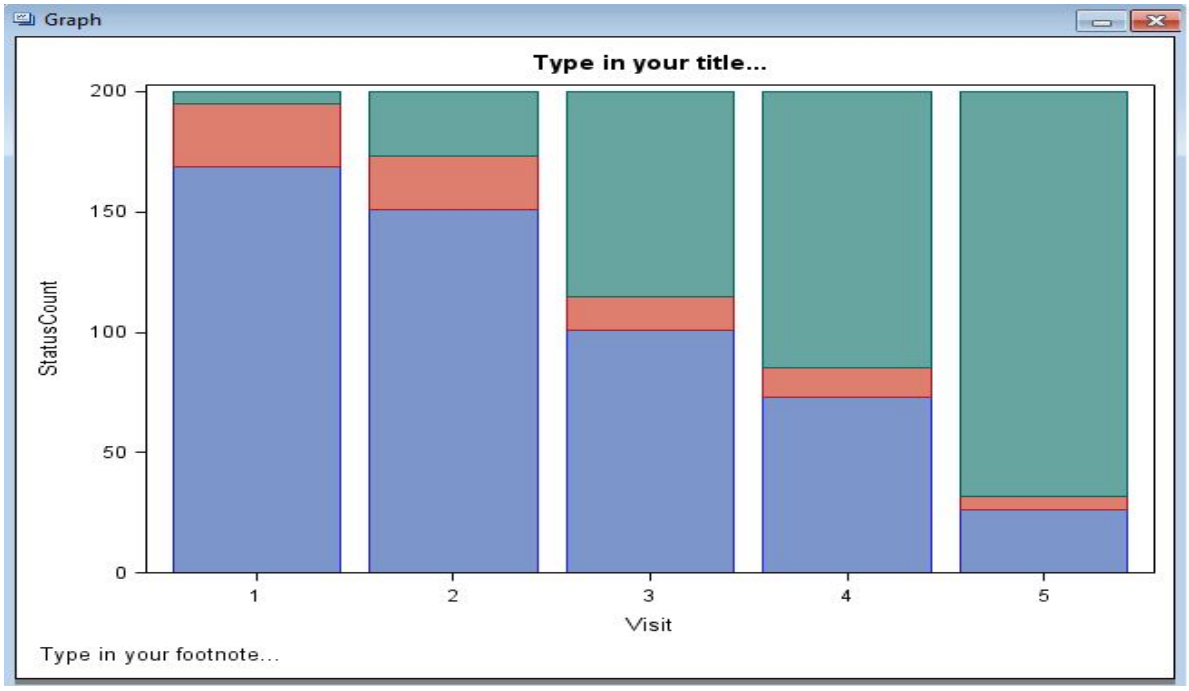

Figure 3. The First Cell Graph

3. Right click in the stack bar graph and select Axis Property, then uncheck all the options in the Display
4. Right click in the stack bar graph and select Add Row, an extra cell is added (Figure 4.)

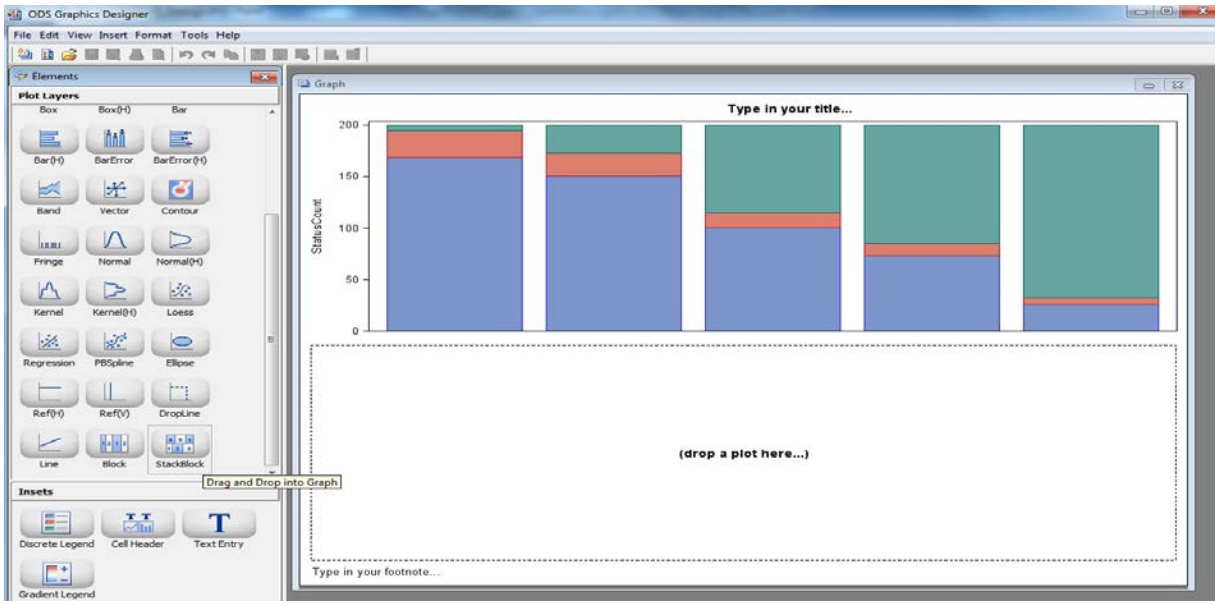

Figure 4. Add an Extra Cell

5. Drag and drop a StackBlock from the Plot Layers to create the table for visit status statistic data. In the Assign Data dialog box, assign the data for visit status statistic data table

   Library : WORK

   Data Set: ALLVISIT_GRAPH

   X: VISIT

   Block: STATUSSTAT

   Group: VISITSTATUS

6. Right click the table in the graph and select Axis Property, then uncheck all the options in the Display
7. Right click the table in the graph and select Add Row
8. Repeat steps 2-6 to add the bar graph (Response: TYPECOUNT, Group: VISITTYPE, Group Display: Cluster) for visit type of completed visits and StackBlock for visit type statistic data table (Block: TYPESTAT, Group: VISITTYPE)
9. Drag and drop the “Discrete Legend” from Insets to both the bar graph cells
10. Add Titles to the graph and edit the label of axis Y
11. Adjust the size of each cell by dragging down the border between the cells

Here is the multi-cell graph created by using ODS Graphics Designer (Figure 5).

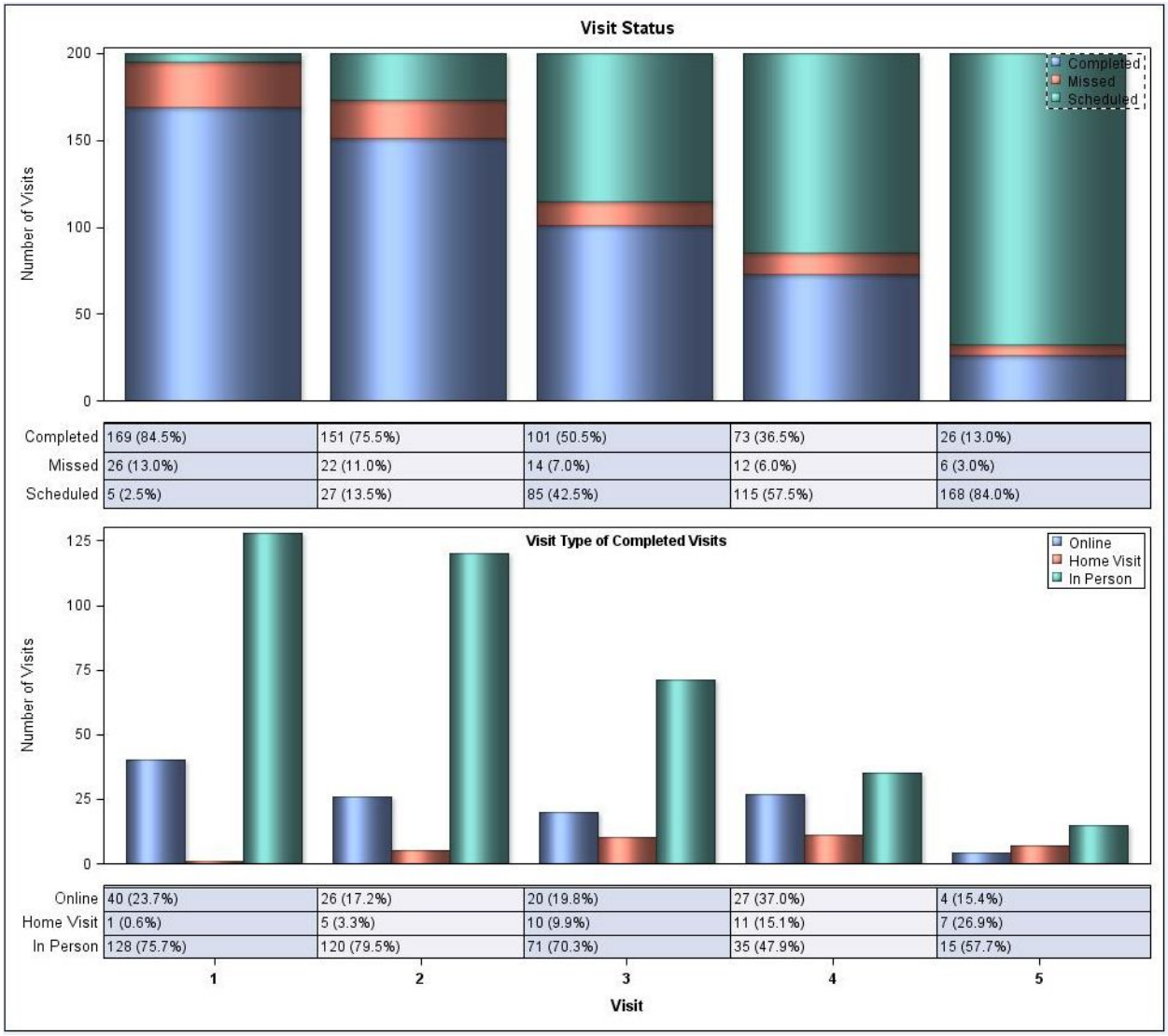

Figure 5. A Multi-cell Graph Created in the ODS Graphics Designer

CUSTOMIZING THE MULTI-CELL GRAPH

Mostly, we got the graph that we wanted, except the coloring. We want to arrange the color for those bar graphs in a manner that is most intuitive for the reviewer to understand. Since the vertical bar graph represents those Completed visits, we want to color both this group and the Completed in the stack bar graph similarly. Also, we want to color the Missed in red as requested.

ODS Graphics Designer is not able to accomplish this. We have to modify the GTL code generated by ODS Graphics Designer to get the color customized graph. By default, when a GROUP= option is used, colors are assigned to groups based on the current ODS style. Prior to SAS 9.3, we may have needed to write our own ODS style by using PROC TEMPLATE to assign colors to group values. Now we can use “Discrete Attribute Map” in the GTL syntax without interfering with the style.

COPY THE GTL CODE GENERATED BY ODS GRAPHICS DESIGNER

As we mentioned, the designer generates Graph Template Language (GTL) code once the graph is created. To copy the code, click on View -> Code, and then click Edit -> Select All -> Copy.

The GTL Code Generated From ODS Graphics Designer

```
proc template;
define statgraph sgdesign;
dynamic _VISIT _STATUSCOUNT _VISITSTATUS _VISIT3 _TYPECOUNT _VISITTYPE _VISIT4
_TYPESTAT _VISITTYPE2 _VISIT2 _STATUSSTAT _VISITSTATUS2;
begingraph;
   entrytitle halign=center 'Visit Status';

       layout lattice / rowdatarange=data columndatarange=data rows=4 rowgutter=10
columngutter=10 rowweights=(0.39 0.1 0.37 0.14);
      layout overlay / xaxisopts=( display=(LINE )) yaxisopts=( label=('Number of
Visits'));
         barchart x=_VISIT y=_STATUSCOUNT / group=_VISITSTATUS name='Bar_Status'
barlabel=false dataskin=sheen discreteoffset=-0.01 clusterwidth=1.0;
         discretelegend 'Bar_Status' / opaque=false border=true halign=right
valign=top displayclipped=true across=1 order=rowmajor location=inside
autoalign=(topright topleft bottomright bottomleft top bottom right left);
      endlayout;
      layout overlay / xaxisopts=( type=discrete display=(LINE ));
         blockplot x=_VISIT2 block=_STATUSSTAT / class=_VISITSTATUS2
name='Block_Status' display=(FILL OUTLINE VALUES LABEL ) filltype=alternate;
      endlayout;
      layout overlay / xaxisopts=( display=(LINE )) yaxisopts=( label=('Number of
Visits'));
         barchart x=_VISIT3 y=_TYPECOUNT / group=_VISITTYPE name='Bar_Type'
dataskin=sheen barwidth=1.0 groupdisplay=Cluster clusterwidth=0.85;
         entry halign=center 'Visit Type of Completed Visits' / valign=top
textattrs=(style=normal weight=bold );
         discretelegend 'Bar_Type' / opaque=false border=true halign=right
valign=top displayclipped=true across=1 order=rowmajor location=inside;
      endlayout;
      layout overlay / xaxisopts=( type=discrete labelattrs=(style=NORMAL
weight=BOLD ) tickvalueattrs=(style=NORMAL weight=BOLD ));
         blockplot x=_VISIT4 block=_TYPESTAT / class=_VISITTYPE2 name='Block_Type'
display=(FILL OUTLINE VALUES LABEL ) filltype=alternate;
      endlayout;
   endlayout;
endgraph;
end;
run;

proc sgrender data=WORK.ALLVISIT_GRAPH template=sgdesign;
dynamic _VISIT="VISIT" _STATUSCOUNT="STATUSCOUNT" _VISITSTATUS="VISITSTATUS"
_VISIT3="VISIT" _TYPECOUNT="TYPECOUNT" _VISITTYPE="VISITTYPE" _VISIT4="VISIT"
_TYPESTAT="TYPESTAT" _VISITTYPE2="VISITTYPE" _VISIT2="VISIT"
_STATUSSTAT="STATUSSTAT" _VISITSTATUS2="VISITSTATUS";
run;
```

ADD DISCRETEATTRMAP BLOCK AND DISCRETEATTRVAR STATEMENT TO GTL CODE

We use the DISCRETEATTRMAP block to create the discrete attribute map in GTL, which assigns the fill color for each value of both variables, Status and Type. In our discrete attribute map definition, the NAME=option assigns the name of the attribute definition, the “ignorecase=true” option ensures that the data mapping is not case sensitive. The VALUE statements define the colors for all the possible values of the discrete variables VisitStatus and VisitType.

Then, two DISCRETEATTRVAR statements are used to create links between the discrete map definition and the discrete variables. The ATTRVAR= option specifies the name to associate between the attribute map and the input column. The VAR= option points to the input column (in our example, it is VisitStatus or VisitType) that is associated with the map. The ATTRMAP= option points to the name of the discrete attribute map to use in the program.

Finally, set the group= option in the barchart statements to the name of the attribute variable that we created in the DISCRETEATTRVAR statement.

The code changes compared to the previous code are shown in yellow below.

Using Discrete Attribute Maps to Control the Color of Group Values

```
proc template;
define statgraph sgdesign;
dynamic _VISIT _STATUSCOUNT _VISITSTATUS _VISIT3 _TYPECOUNT _VISITTYPE _VISIT4
_TYPESTAT _VISITTYPE2 _VISIT2 _STATUSSTAT _VISITSTATUS2;
begingraph;

   entrytitle halign=center 'Visit Status';

      DiscreteAttrMap name="_Fills_" / ignorecase=true;
            Value "Completed" / fillattrs=( color=Blue)
                  lineattrs=( color=Blue);
            Value "Missed" / fillattrs=( color=darkred)
                      lineattrs=( color=darkred);
            Value "Scheduled" / fillattrs=( color=cadetblue)
                    lineattrs=( color=cadetblue);
            Value "In Person" / fillattrs=( color=BIGB)
                    lineattrs=( color=BIGB);
            Value "Online" / fillattrs=( color=RoyalBlue)
                    lineattrs=( color=RoyalBlue);
            Value "home visit" / fillattrs=( color=LightBlue)
                    lineattrs=( color=LightBlue);
      EndDiscreteAttrMap;
      DiscreteAttrVar attrvar=StatusFills var=VisitStatus attrmap="_Fills_";
      DiscreteAttrVar attrvar=TypeFills var=VisitType attrmap="_Fills_";

       layout lattice / rowdatarange=data columndatarange=data rows=4 rowgutter=10
columngutter=10 rowweights=(0.39 0.1 0.37 0.14);
      layout overlay / xaxisopts=( display=(LINE )) yaxisopts=( label=('Number of
Visits'));
         barchart x=_VISIT y=_STATUSCOUNT / group=StatusFills name='Bar_Status'
barlabel=false dataskin=sheen discreteoffset=-0.01 clusterwidth=1.0;
         discretelegend 'Bar_Status' / opaque=false border=true halign=right
valign=top displayclipped=true across=1 order=rowmajor location=inside
autoalign=(topright topleft bottomright bottomleft top bottom right left);
      endlayout;
      layout overlay / xaxisopts=( type=discrete display=(LINE ));
         blockplot x=_VISIT2 block=_STATUSSTAT / class=_VISITSTATUS2
name='Block_Status' display=(FILL OUTLINE VALUES LABEL ) filltype=alternate;
      endlayout;
      layout overlay / xaxisopts=( display=(LINE )) yaxisopts=( label=('Number of
Visits'));
         barchart x=_VISIT3 y=_TYPECOUNT / group=TypeFills name='Bar_Type'
dataskin=sheen barwidth=1.0 groupdisplay=Cluster clusterwidth=0.85;
         entry halign=center 'Visit Type of Completed Visits' / valign=top
textattrs=(style=normal weight=bold );
         discretelegend 'Bar_Type' / opaque=false border=true halign=right
valign=top displayclipped=true across=1 order=rowmajor location=inside;
      endlayout;
      layout overlay / xaxisopts=( type=discrete labelattrs=(style=NORMAL
weight=BOLD ) tickvalueattrs=(style=NORMAL weight=BOLD ));
         blockplot x=_VISIT4 block=_TYPESTAT / class=_VISITTYPE2 name='Block_Type'
display=(FILL OUTLINE VALUES LABEL ) filltype=alternate;
      endlayout;
   endlayout;
endgraph;
end;
```

```
run;
proc sgrender data=WORK.ALLVISIT_GRAPH template=sgdesign;
dynamic _VISIT="VISIT" _STATUSCOUNT="STATUSCOUNT" _VISITSTATUS="VISITSTATUS"
_VISIT3="VISIT" _TYPECOUNT="TYPECOUNT" _VISITTYPE="VISITTYPE" _VISIT4="VISIT"
_TYPESTAT="TYPESTAT" _VISITTYPE2="VISITTYPE" _VISIT2="VISIT"
_STATUSSTAT="STATUSSTAT" _VISITSTATUS2="VISITSTATUS";
run;
```

FINAL GRAPH

We can run this modified GTL code along with other program code in the SAS program editor to have the graph output as part of our final PDF report by using the SAS Output Delivery System (ODS). Or we can print a stand-alone JPG file by adding the statements below.

```
ods listing gpath="C:\MWSUG2013";
ods graphics on / imagefmt=jpeg imagename='MWSUG_graph';
proc sgrender data=WORK.ALLVISIT_GRAPH template=sgdesign;
… …
ods graphics off;
```

Here is our final graph (Figure 6.)

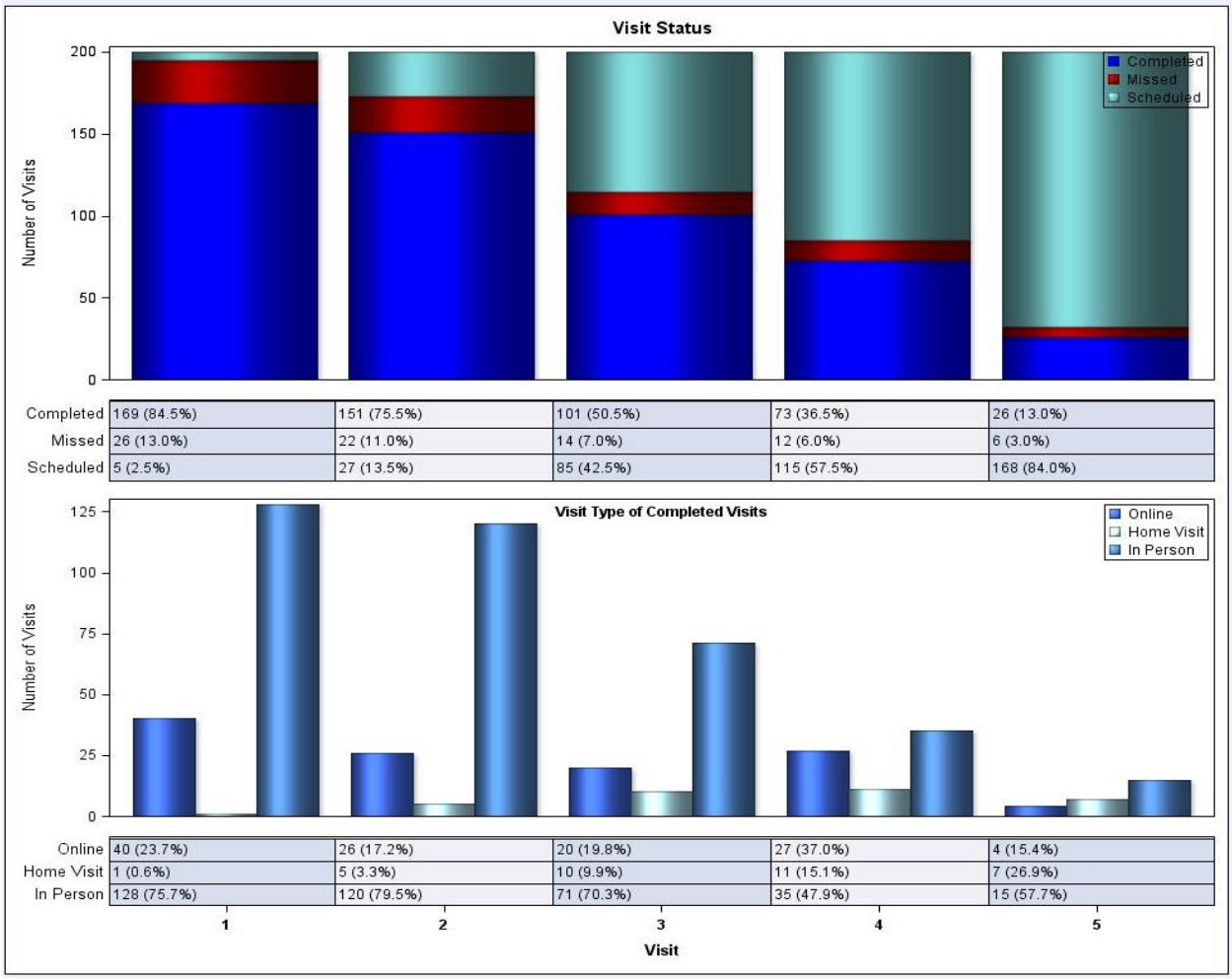

Figure 6. A Customized Multi-cell Graph

CONCLUSION

A multi-cell graph is an effective way to present a summarized report of related information together. With ODS GRAPHICS DESIGNER and new features in SAS®, we can create a customized multi-cell graph easily and efficiently.

REFERENCES

- 2011, SAS Institute, Inc., Cary, NC: SAS® 9.3 ODS Graphics Designer User's Guide
- 2011, SAS Institute, Inc., Cary, NC: SAS® 9.3 Graph Template Language User's Guide

ACKNOWLEDGMENTS

The author would like to thank Robert Tamer and Trish Herbers who reviewed and edited this paper and poster. The author would also like to thank the Data Management Center at Cincinnati Children's Hospital Medical Center for all the support.

CONTACT INFORMATION

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

Yanhong Liu
Data Management Center
Division of Biostatistics & Epidemiology
Cincinnati Children's Hospital Medical Center
3333 Burnet Ave.
Cincinnati, OH 45239
513-803-2614
Yanhong.liu@cchmc.org