PharmaSUG 2023 Paper DV-226

The Flexible Ways to Create Graphs for Clinical Trial

Lily Zhang, Merck & Co., Inc.;
Kaijun Zhang, Merck & Co., Inc.

ABSTRACT

Without a doubt, SAS® programming can create high-quality graphics. In SAS language, PROC TEMPLATE with SGRENDER procedure is a powerful and commonly used tool to create a customized figure. PROC TEMPLATE allows any figure type to overlay in one plotting space, lattice any figure type side by side, and control every visual aspect of the graphical field. However, the R language's more flexible tools have gained popularity recently by using the `ggplot2` package, especially combined with the `patchwork` package. The combination of `patchwork` and the `ggplot2` package can provide you with a more powerful and effective tool to create all kinds of graphs in clinical trial reports[1,2].

This paper will introduce the R package `patchwork` combined with the `ggplot2` using typical graphs in the clinical trial as examples. This paper will also include PROC TEMPLATE with SGRENDER procedure of SAS, as the comparison. In summary, `patchwork` combined with the `ggplot2` package in R opens a new window for you to customize your graphs, especially combining different graphs or text content on one page or across the page.

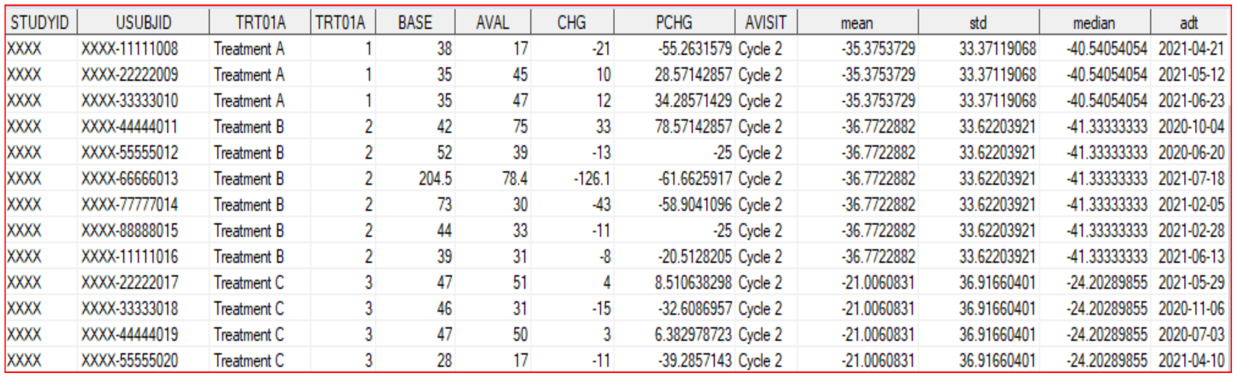
KEYWORDS

Graphics, R language, Graph Template Language (GTL), `ggplot2`, `patchwork`, clinical trial, visualization

INTRODUCTION

When we work with a large and diverse amount of data for clinical trials, data visualization can help us to make data more understandable. A well-designed, informative graph should be essential to analyze these complex data better and have a concise visual presentation to show statistical results for reporting. High-quality graphs help us analyze and understand the data from different clinical trial domains.

In this paper, the SAS data set named "adtl" is the example data set to generate graphs by SAS langue and the R package. The data set contains made-up data with one patient record per test date, to simulate a clinical patient study where the outcome of interest is the evaluation of target lesion tumor size in the solid tumor during a study. The variable 'BASE' is the baseline of tumor size, 'AVAL' is the tumor size value at that test date, 'CHG' is the difference between 'AVAL' and 'BASE', 'PCHG' is the percentage change, and the 'mean', 'std', 'median' is based on the treatment. Below are the first 12 observations in the example data ADTL.



STUDYID	USUBJID	TRT01A	TRT01A	BASE	AVAL	CHG	PCHG	AVISIT	mean	std	median	adt
XXXX	XXXX-11111008	Treatment A	1	38	17	-21	-55.2631579	Cycle 2	-35.3753729	33.37119068	-40.54054054	2021-04-21
XXXX	XXXX-22222009	Treatment A	1	35	45	10	28.57142857	Cycle 2	-35.3753729	33.37119068	-40.54054054	2021-05-12
XXXX	XXXX-33333010	Treatment A	1	35	47	12	34.28571429	Cycle 2	-35.3753729	33.37119068	-40.54054054	2021-06-23
XXXX	XXXX-44444011	Treatment B	2	42	75	33	78.57142857	Cycle 2	-36.7722882	33.62203921	-41.33333333	2020-10-04
XXXX	XXXX-55555012	Treatment B	2	52	39	-13	-25	Cycle 2	-36.7722882	33.62203921	-41.33333333	2020-06-20
XXXX	XXXX-66666013	Treatment B	2	204.5	78.4	-126.1	-61.6625917	Cycle 2	-36.7722882	33.62203921	-41.33333333	2021-07-18
XXXX	XXXX-77777014	Treatment B	2	73	30	-43	-58.9041096	Cycle 2	-36.7722882	33.62203921	-41.33333333	2021-02-05
XXXX	XXXX-88888015	Treatment B	2	44	33	-11	-25	Cycle 2	-36.7722882	33.62203921	-41.33333333	2021-02-28
XXXX	XXXX-11111016	Treatment B	2	39	31	-8	-20.5128205	Cycle 2	-36.7722882	33.62203921	-41.33333333	2021-06-13
XXXX	XXXX-22222017	Treatment C	3	47	51	4	8.510638298	Cycle 2	-21.0060831	36.91660401	-24.20289855	2021-05-29
XXXX	XXXX-33333018	Treatment C	3	46	31	-15	-32.6086957	Cycle 2	-21.0060831	36.91660401	-24.20289855	2020-11-06
XXXX	XXXX-44444019	Treatment C	3	47	50	3	6.382978723	Cycle 2	-21.0060831	36.91660401	-24.20289855	2020-07-03
XXXX	XXXX-55555020	Treatment C	3	28	17	-11	-39.2857143	Cycle 2	-21.0060831	36.91660401	-24.20289855	2021-04-10

Graph Template Language (GTL) in SAS and `ggplot2`, together with the `patchwork` package in R, provides many possible combinations of statements to create graphs. This paper will discuss the visual presentation of data in clinical trials using two tools, especially for combining different plots into one page using `ggplot2` together with the `patchwork` package in R. We will take a couple of boxplot graphs as examples to describe how to accomplish the goal. Through the illustrations, we will recommend a few valuable options that help the customization. We will also demonstrate how to generate the same graphs using SAS language for comparison. The three boxplots as examples for combination in one page include as following:

- Percent changes for tumor size (Sum of diameter, SOD) from baseline by treatment over time.
- The baseline value of tumor size (Sum of diameter, SOD).
- Percent changes for tumor size (Sum of diameter, SOD) from baseline by treatment.

CREATING GRAPHS BY USING GTL

Key Syntax of GTL

In the SAS application, we can define the graph template with the TEMPLATE procedure as follows:

```
proc template;
    define statgraph <template-name>;
      beginGraph / <options>;
            <GTL statements>;
      endgraph;
    end;
run;
```

The TEMPLATE procedure defines and saves the Graph's structure as a template for later usage. This code alone will not create the Graph. The BEGINGRAPH and ENDGRAPH statements define the outermost container for the Graph and must contain all the GTL statements. Produce the Graph with the SGRENDER procedure:

```
proc sgrender data = <data>   template = <template-name>;
run;
```

The SGRENDER procedure will associate data with the predefined template and create the Graph[2]. The same template can be used with multiple, compatible data sets to create additional graphs if necessary.

Example of using GTL to create graphs

GTL is a useful tool for programmers to create graphs through SAS, however, it is difficult and even impossible for you to remember all statements and options of SAS GTL. GTL is still an efficient way to enable the user to generate complex, advanced, and customized figures. As figure 1 shows below, three boxplots are created on one page. The SAS codes are also listed below. This approach shows that SAS GTL would require a significant programming effort.

```
**Define the graph’s structure as a template;
proc template;
  define statgraph sgdesignb;
  dynamic _AVISIT _PCHG _TRT01A  _TRT01A2 _BASE  _TRT01A3 _MEAN;
  begingraph/border=true;
  layout lattice/rows=2  rowweights=(0.6 0.4);
    **First row;
    layout lattice/rowdatarange=data columndatarange=data rowgutter=10
                   columngutter=10;
       layout overlay/xaxisopts=(label=('Analysis Visit')
          linearopts=(viewmin=1.0 viewmax=19.0 minorticks=OFF
                      tickvaluesequence=( start=1.0
                      end=19.0 increment=1.0))
                      discreteopts=( tickvaluefitpolicy=splitrotate))
         yaxisopts=(label=("Percent Change(*ESC*){unicode '000a'x} from
                       Baseline(*ESC*){unicode '000a'x}(SOD)"));
         boxplot x=_AVISIT y=_PCHG/group=_TRT01A name='box1'
                 boxwidth=0.75 intervalboxwidth=1.0
         display=(CAPS CONNECT FILL OUTLIERS MEAN MEDIAN )
                 groupdisplay=Cluster;
       endlayout;

       sidebar / align=top spacefill=false;
       discretelegend 'box1' / opaque=true border=true
                 halign=center valign=top
                 displayclipped=true
                 down=1 order=columnmajor
                 titleattrs=(style=NORMAL weight=BOLD );
       endsidebar;
     endlayout;

     *****Second row;
     layout lattice/columns=2 columnweights=(0.52 0.48);
       *left column;
       layout lattice /rowdatarange=data columndatarange=data rowgutter=10
                   columngutter=10;
         layout overlay /xaxisopts=(display=(TICKS TICKVALUES LINE)
               discreteopts=(tickvaluefitpolicy=splitrotate))
               yaxisopts=(label=("Baseline Value(*ESC*){unicode '000a'x}(SOD)")
               offsetmin=0.05 offsetmax=0.05 linearopts=(viewmin=0.0
               viewmax=350.0 tickvaluesequence=(start=0.0 end=350.0
                                              increment=50.0)));
          boxplot x=_TRT01A2 y=_BASE /name='box' boxwidth=0.3
                     groupdisplay=Cluster;
         endlayout;
       endlayout;
       *right column;
       layout lattice/rowdatarange=data columndatarange=data rowgutter=10
                 columngutter=10;
          layout overlay/xaxisopts=(display=(TICKS TICKVALUES LINE )
                   discreteopts=(tickvaluefitpolicy=splitrotate))
                   yaxisopts=( offsetmin=0.05 offsetmax=0.05
                   label=("Percent Change(*ESC*){unicode '000a'x}from
                          Baseline(*ESC*){unicode '000a'x}(SOD)")
                   linearopts=(viewmin=-100.0 viewmax=100.0 tickvaluesequence=(
                         start=-100.0 end=100.0 increment=25.0)));
           boxplot x=_TRT01A3 y=_PCHG / name='box' groupdisplay=Cluster;
           Referenceline y=-30/lineattrs=(thickness=1px pattern=34 color=cx000000);
           Referenceline y=20/lineattrs=(thickness=1px pattern=34 color=cx000000);
          endlayout;
       endlayout;
     endlayout;
   endlayout;
  endgraph;
end;
run;
```

```
**Generate graphs;
proc sgrender data=adtl template=sgdesignb;
  dynamic _PCHG="PCHG" _AVISIT="AVISIT" _TRT01A="TRT01A"
          _TRT01A2="TRT01A"   _BASE="BASE"
          _TRT01A3="TRT01A"   _MEAN="MEANVAL"
            ;
run;
```

Figure 1. Plots for Sum of Diameters of Tumor (SOD) for Baseline Value and Percent Changes from Baseline

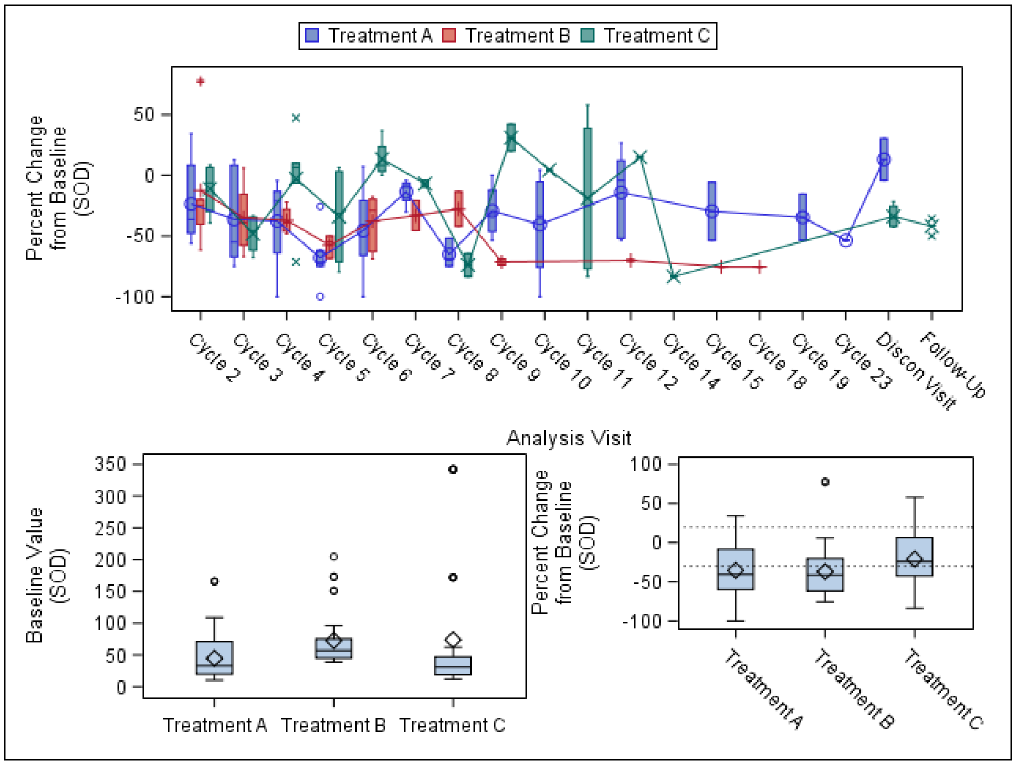


Figure 1: Generated by SAS GLT, combined from three boxplots.

If we want to change the above figure's structure or add some new features, we have to write the complicated TEMPLATE procedure again to define a new Graph's structure template. However, we can add new features and re-arrange the graph's structure by using R functions straightforwardly.

CREATING GRAPHS BY USING GGPLOT2 AND PATCHWORK PACKAGE

Key Syntax of R packages `ggplot2` and `patchwork`

The basic grammar of `ggplot2` package is as the following:

`ggplot2` package:

```
ggplot(data = <DATA>) +
<GEOM_FUNCTION>(mapping = aes(<MAPPINGS>))
```

A layer combines data, aesthetic mapping, a `geom` (geometric object), a stat (statistical transformation), and a position adjustment. An aesthetic is a visual property of the objects which include items such as the size, the shape, or the color. A `geom` is the geometrical object that a plot uses to represent data such as line, point or boxplot. The operator ("+") adds layers together to create graphs you expect. The functions for `ggplot2` package include aesthetic mapping, geometric objects, coordinates of x and y axis, facet, labels, and theme.

The `patchwork` package is a great "composer" of plots, with primary functions listed in table 1 below. The reference document is available online[3].

Table 1. Functions in patchwork package

Function name	Usage
area	Specify a single area in a rectangular grid that should contain a plot. Objects constructed with the area() can be concatenated together with c() to specify multiple areas.
guide_area	Add an area to hold collected guides.
inset_element	Create an inset to be added on top of the previous plot.
multipage_align	Align plots across multiple pages.
plot_annotation	Annotate the final patchwork.
plot_arithmetic	Plot arithmetic. In addition to the + operator known in ggplot2, patchwork defines logic for some of the other operators, such as "/" or "|", which aids in building up your plot composition and reducing code reuse.
plot_layout	Define the grid to compose plots in. In order to control how different plots are laid out, you need to add a layout specification. If you are nesting grids, the layout is scoped to the current nesting level.
plot_spacer	Add a completely blank area.
wrap_elements	Wrap arbitrary graphics in a patchwork-compliant patch.
wrap_ggplot_grob	Make a gtable created from a ggplot object patchwork compliant. This function converts a gtable, as produced by ggplot2::ggplotGrob(), and makes it ready to be added to a patchwork. In contrast to passing the gtable to wrap_elements(), wrap_ggplot_grob() ensures proper alignment as expected. On the other hand, major restructuring of the gtable will result in an object that doesn't work properly with wrap_ggplot_grob().
wrap_plots	Wrap plots into a patchwork. While the use of + is a natural way to add plots together, it can be difficult to string together multiple plots programmatically if the number of plots is not known beforehand. wrap_plots makes it easy to take a list of plots and add them into one composition, along with layout specifications.

The `ggplot2` and `patchwork` package in the R library use a layered approach. The programmer can view the work in progress without completing the final Graph. The code can be developed and executed one layer at a time and adjusted as needed. On the other hand, the `patchwork` package makes it extremely simple to combine separate plots from `ggplot2` into the same graphic. As such, it does the same job as `gridExtra::grid.arrange()` and `cowplot::plot_grid`. It helps you easily assemble plots, define layout, add an annotation on one page and align graphs across pages.

Here are the examples of the basic syntax of the `ggplot2` and `patchwork` packages:

`patchwork` package:
The basic operators to assembly graphs or text content includes "+", "- ", "|" and (). The basic functions are used including plot_layout, plot_annotation, plot_spacer, inset_element, and theme. It can combine different pieces of ggplot or/with text content together in one page or across page.

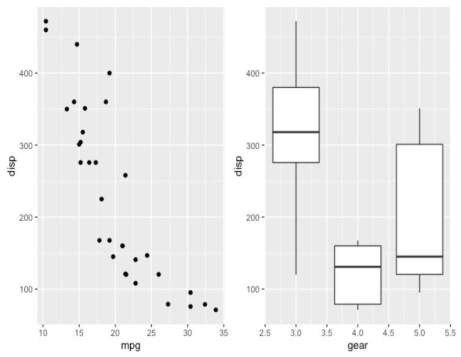
The basic syntax as listed below:

Example 1:

```
library(ggplot2)
library(patchwork)

p11 <- ggplot(mtcars) + geom_point(aes(mpg, disp))
p22 <- ggplot(mtcars) + geom_boxplot(aes(gear, disp, group = gear))
```

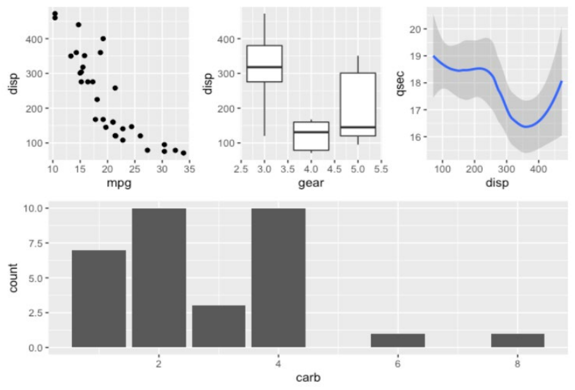
`Figure 2.1: p11 + p22`



Example 2:

```
p33 <- ggplot(mtcars) + geom_smooth(aes(disp, qsec))
p44 <- ggplot(mtcars) + geom_bar(aes(carb))
```

`Figure 2.2: (p11 | p22 | p33) / p44`



Example of using `ggplot2` and `patchwork` to create graphs

Below are the R libraries and functions used in this paper.

```
      library(tidyverse)
      library(haven)
      library(ggplot2)
      library(gridExtra)
      library(devtools)
      library(patchwork)
```

We will use dummy SAS data sets `adtl` as an input data frame in `ggplot2` for the following examples.

First, we need to read SAS data into RStudio.

```
           adtl <- read_sas("adtl.sas7bdat")
```

Second, we will create the single box plot one by one using the `ggplot2` package:

```
p1 <- adtl %>%
       mutate(AVISIT=factor(AVISIT, levels=c("Cycle 2", "Cycle 3", "Cycle 4",
         "Cycle 5", "Cycle 6", "Cycle 7", "Cycle 8", "Cycle 9", "Cycle 10",
         "Cycle 11", "Cycle 12", "Cycle 14", "Cycle 15", "Cycle 18", "Cycle 19",
         "Cycle 23", "Discon Visit", "Follow-Up"))) %>%

      ggplot(aes(x=AVISIT, y=PCHG, fill=TRT01A))  +
        geom_boxplot(shape=3, outlier.size=3, outlier.shape=18,
          outlier.color="red")  +
        stat_summary(fun = mean, geom = "point", col = "green")    +
        labs(#x ="Analysis Visit"  x="Analysis Visit \n (Red points indicate
          outliers, Green points indicate means)" ,
          y ="Percent Change From Baseline",  axis.text.x = element_text(angle =
          45, hjust = 1))   +
        theme(legend.position ="top", legend.title=element_blank(), axis.text.x =
          element_text(angle = 45, hjust = 1))    +
      ylim(-100, 100)

p2 <- ggplot(adtl, aes(TRT01A, BASE)) + geom_boxplot() +
         labs(x = " ", y = "Baseline Value \n (SOD)") +
         coord_cartesian(ylim=c(0, 400))

p3 <- ggplot(adtl, aes(TRT01A, PCHG)) + geom_boxplot() +
         labs(x = " ", y = "Percent Change from Baseline \n (SOD)") +
         coord_cartesian(ylim=c(-100, 100))
```

Finally, we can use the `patchwork` R package to combine three separate plots into one page. The `patchwork` package combines plots by:

- Using a `ggplot2` syntax for the grammar of plot-layout operations.
- Extending the amazing `ggplot2` package.

`Patchwork` has a straightforward syntax where we can create layouts super easily. Here is the general syntax that combines:

- Two-Column layout using the Plus Sign `+`
- Parenthesis `()` to create a subplot group.
- Two-Row layout using the Division Sign `\`

The code is listed below, which `patchwork` through an operator such as "/" and "|" does the same as the `grid.arrange` and `grid.grob` do. Now we use the very simple code to create the same effect output (figure 3) as the above plot (figure 1) which is created by GTL:

```
p1/(p2|p3)
```

Figure 3. Plots for Sum of Diameters of Tumor (SOD) for Baseline Value and Percent Changes from Baseline

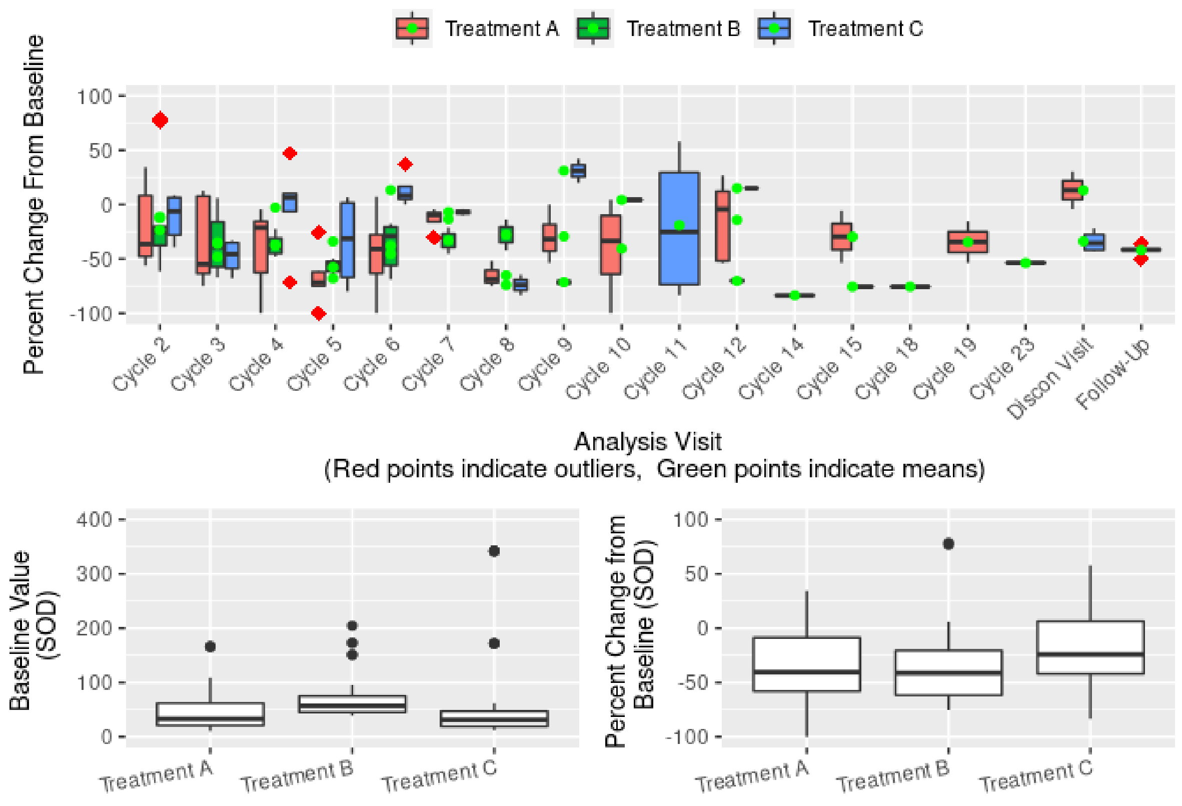


Figure 3: Generated by R packages, combined from three boxplots

The composer of `ggplot2` and `patchwork` can combine separate plots into the same graphic and re-arrange plots into a grid and add figures, labels, and annotations very simply.

More examples of flexible features for `ggplot2` and `patchwork` packages

You can change the position for the Graph on the top in figure 3 to the bottom and assign a tag to it as "Fig c", and put the two figures on the bottom in figure 3 to the top and place the tag "Fig a and Fig b" to them respectively. Also, `patchwork` can add space among graphs on the top and bottom. It has the options such as plot_spacer, plot_layout, and plot_annotation. The code listed below generates the output of figure 4.

```
p01 <- ((p2|plot_spacer () |p3) + plot_layout(widths = c(4.5, 0.2 ,4.5)))/
        (p1 + plot_spacer()+ plot_layout(heights = unit(c(3, 0.2), c('cm',
        'cm'))))
```

```
Pb  <- p01 + plot_annotation(tag_levels = list(c('Fig a', 'Fig b', "Fig c")))
```

Figure 4. Plots for Sum of Diameters of Tumor (SOD) for Baseline Value and Percent Changes from Baseline

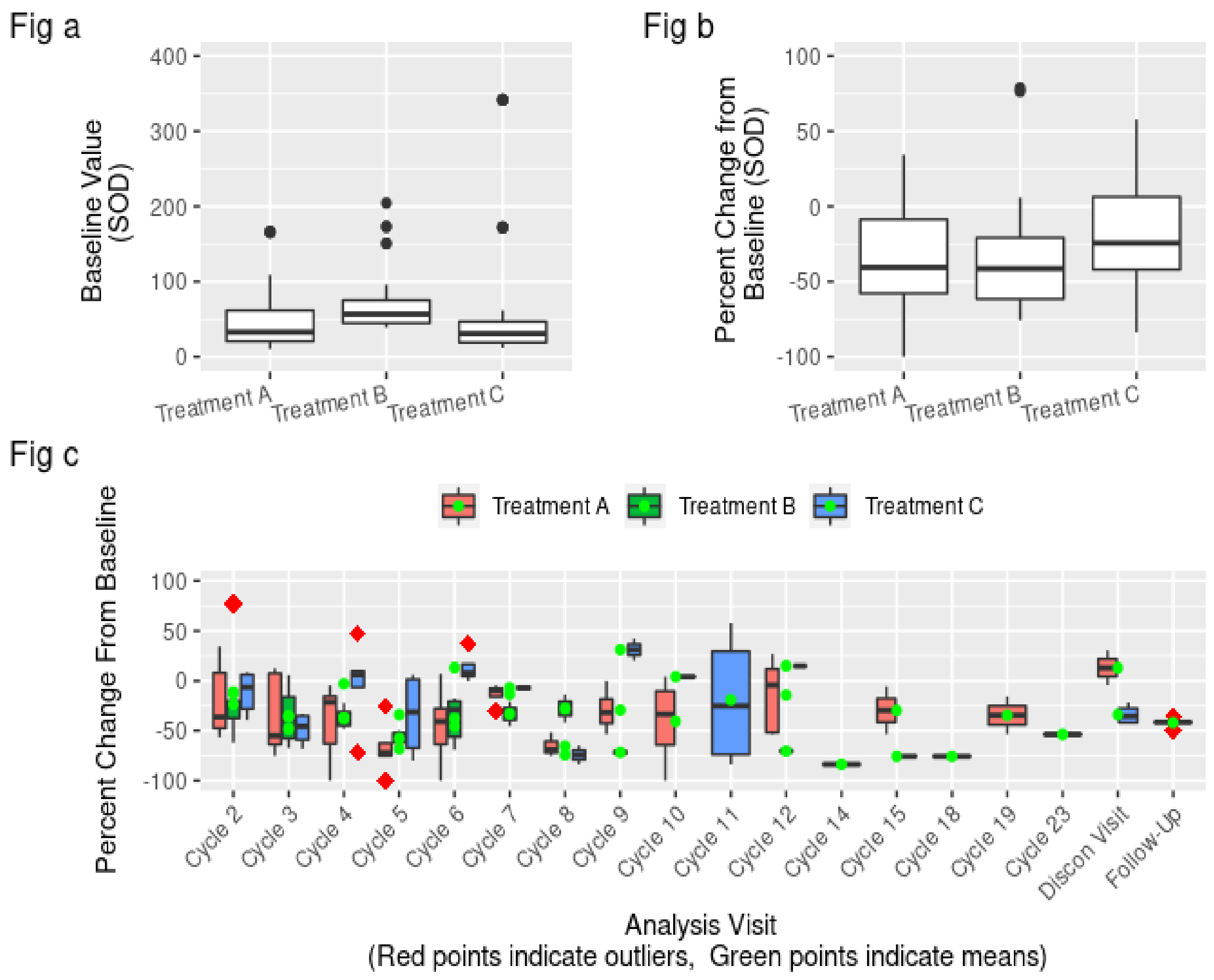


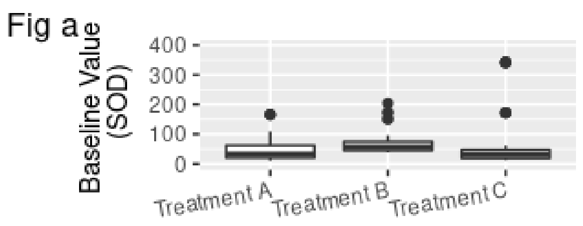
Figure 4: Based on Fig 3, changed position, added space individual graphs, and added tags

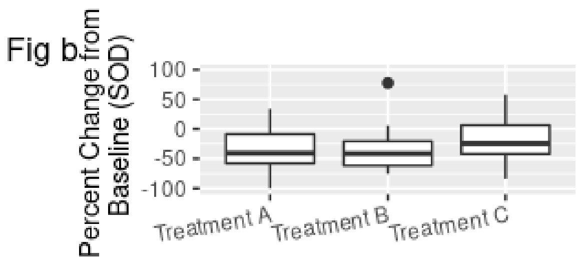
We can also add the non-plot content by combining the `ggplot2` and the `patchwork` package. Below is the example code, and figure 5 is the output.

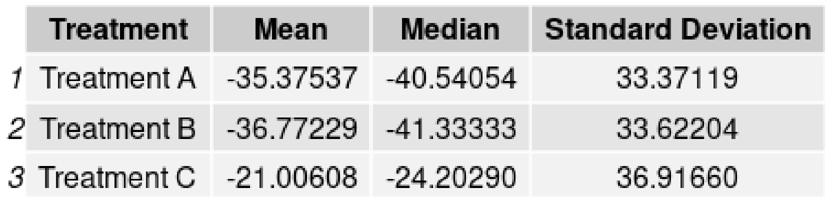
```
adtla <- adtl %>% select(TRT01A, mean, median, std) %>% distinct()
colnames(adtla) <-  c("Treatment", "Mean", 'Median', 'Standard Deviation')

pc <- gridExtra::tableGrob(adtla)
p03 <- ((p2|plot_spacer()|p3) + plot_layout(widths = c(4.5, 0.2 ,4.5)))/pc/p1
+
plot_annotation(tag_levels = list(c('Fig a', 'Fig b', " ", "Fig c")))
```

Figure 5. Plots for Sum of Diameters of Tumor (SOD) for Baseline Value and Percent Changes







	Treatment	Mean	Median	Standard Deviation
1	Treatment A	-35.37537	-40.54054	33.37119
2	Treatment B	-36.77229	-41.33333	33.62204
3	Treatment C	-21.00608	-24.20290	36.91660

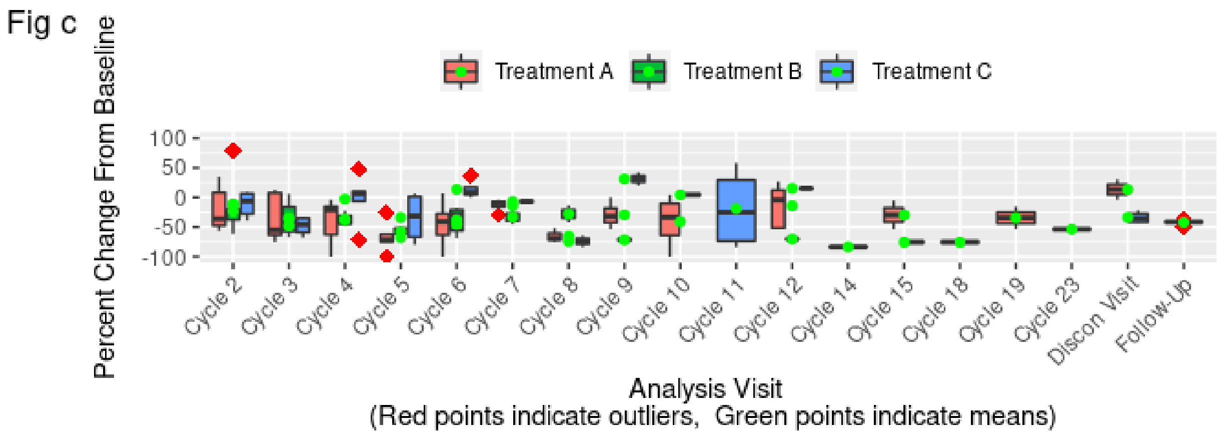


Figure 5: Based on figure4, the non-plot content is added

CONCLUSION

SAS is a powerful statistical analysis software and has made many significant improvements to match industry needs. Still, the programming for graphs is complicated, hard to learn, and difficult to debug, especially since SAS GTL requires a significant amount of programming effort. On the other hand, R is an open-source software environment, and the latest functionality is updated more frequently. R also provides a wide variety of statistical and graphical techniques to create complex graphs with less programming effort, such as `ggplot2` and `patchwork` package. The syntax of R is simple and easy to debug. Since R has already built many statistics and output programming packages for your use, the amount of R coding skills you need to learn is fairly minimal. The return on this investment of effort can be great - access to thousands of additional analytical and graphical methods.

In conclusion, SAS GTL provides data visualization features and awesome technical support, however, R offers tons of packages to create graphs more efficiently and straightforwardly. Both technologies can improve data analysis and visualization capabilities in clinical trials. It is the winning solution to start combining both technics of SAS and R and get maximum benefits.

REFERENCES

1. Wickham, Hadley and Garret Grolemund. (2017). R for Data Science. Sepastopol, CA: O'Reilly Media.
2. MATANGE, S. (2019). GETTING STARTED WITH THE GRAPH TEMPLATE LANGUAGE IN SAS: Examples, tips, and techniques for creating custom graphs. SAS Institute.
3. Patckwork. Thomas Lin Pedersen. (2022). Available at https://cran.r-project.org/web/packages/patchwork/patchwork.pdf.

ACKNOWLEDGMENTS

The authors would like to thank the management team of Merck & Co., Inc., Kenilworth, NJ, USA, for their advice on this paper/presentation.

CONTACT INFORMATION

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

Lily Zhang
Merck & Co., Inc., Kenilworth, NJ, USA
e-mail: lily_zhang2@merck.com

Kaijun Zhang
Merck & Co., Inc., Kenilworth, NJ, USA
e-mail: kaijun.zhang@merck.com

TRADEMARK