PharmaSUG 2024 - Paper PO- 196

Comparing SAS® and R Approaches in Creating Multicell Dot Plots in Statistical Programming

Yi Guo, Pfizer Inc.

ABSTRACT

In clinical studies, comparisons between groups are very common, and researchers are typically interested in significant results or differences. Instead of laboriously examining statistical tables row by row, multicell dot plots that share the same y-axis can provide a more intuitive and efficient way for researchers to pinpoint the results of interest.

Both SAS® and R can create high-quality dot plots. In this paper, we will explain and compare the risk difference (RD) within the system organ class (SOC) for two treatment groups using SAS® GTL versus R ggplot2. Furthermore, we will discuss the challenges when using these different programming languages, compare the syntax and functionality of SAS® and R, and summarize which method is more efficient in different aspects. In addition, the paper will explore the SAS® and R code in an application of dot plots in a real-world evidence (RWE) study.

INTRODUCTION

Dot plots are a useful tool for visualizing and summarizing data. They are often organized and structured into panels to show data points for quick and intuitive visual comparison between different groups. For example, multicell dot plots that share the same y-axis are a good substitute to a table summarizing each event along with its corresponding frequency of occurrence in different subgroups. As another example, dot plots can help assess the balance achieved in covariates between treated and control groups before and after propensity score matching.

Both SAS® Graph Template Language (GTL) and R ggplot2 are powerful tools for data visualizations, each with its own set of advantages. Customizing plots in SAS® may require using options and statements within the relevant procedure. While SAS® provides customization options through GTL, some users may find the syntax less accessible for complex plots. R offers a highly flexible system for creating publication-quality plots. The ggplot2 approach to graphics makes it easier to construct complex plots with layered components.

In the following sections, we will use both SAS® GTL and R's ggplot2 to create a paired dot plot displaying risk rate together with risk difference (RD) in different system organ classes (SOC), explain key components in the syntax of SAS® and R, and make a comparison between them. We use adverse events data in this figure example because safety issues are often reported in tables but rarely in figures in clinical trials. Multicell dot plots that share the same y-axis can be a good substitute for the AE frequency tables, as proposed by Amit, Heiberger, and Lane (2008). These allow the complex information to be presented in a more effective and intuitive way, helping us identify what we are interested in within a large set of adverse events. In the last section, as an extension, we will explore both SAS® and R code in plotting absolute standardized differences before and after propensity score matching comparing covariate values between two treatment groups.

SOURCE DATA

The source data is a dummy data set including system organ classes (SOC) from a dummy ADaM-like analysis data set of adverse events (ADAE), risk rate (i.e., subject per year of follow-up), and the risk difference between treatment group B and A with confidence interval within SOC. Of note, the data used in SAS® is wide-formatted (horizontal) while the data used in R is long-formatted (vertical). The SAS® and R code for dummy data generation are attached in APPENDIX I and II, respectively.

OUTPUT FIGURE

We generated multicell dot plots using both approaches. Figure 1a is the output generated by SAS® GTL and Figure 1b is the figure generated by R's ggplot2. These two figures are a two-panel display that compares the occurrence of AEs in SOCs between two treatment groups. The SOCs are ordered by risk difference and presented in descending order instead of alphabetical order, as it offers a more efficient way for researchers to identify significant risk differences and the importance of the AE SOCs.

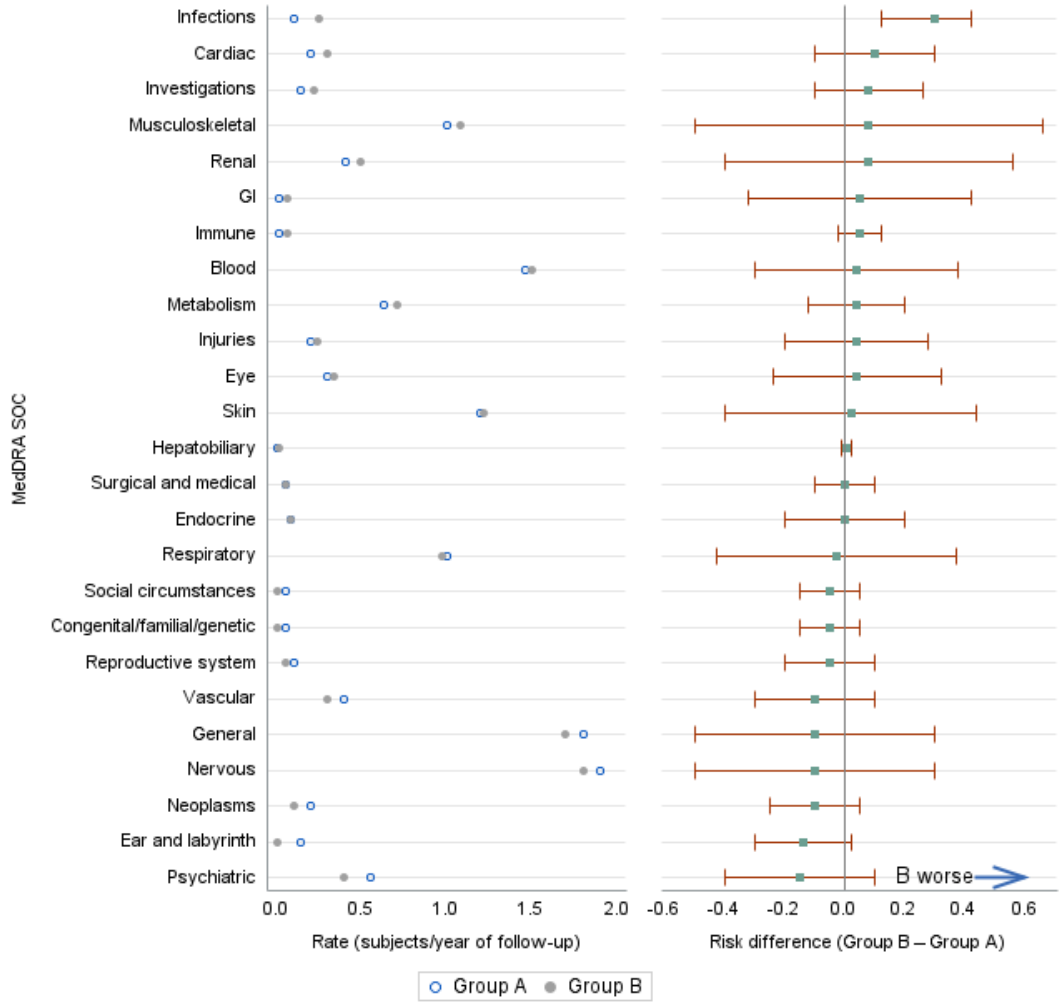

Figure 1a. Multicell dot plots using SAS® GTL.

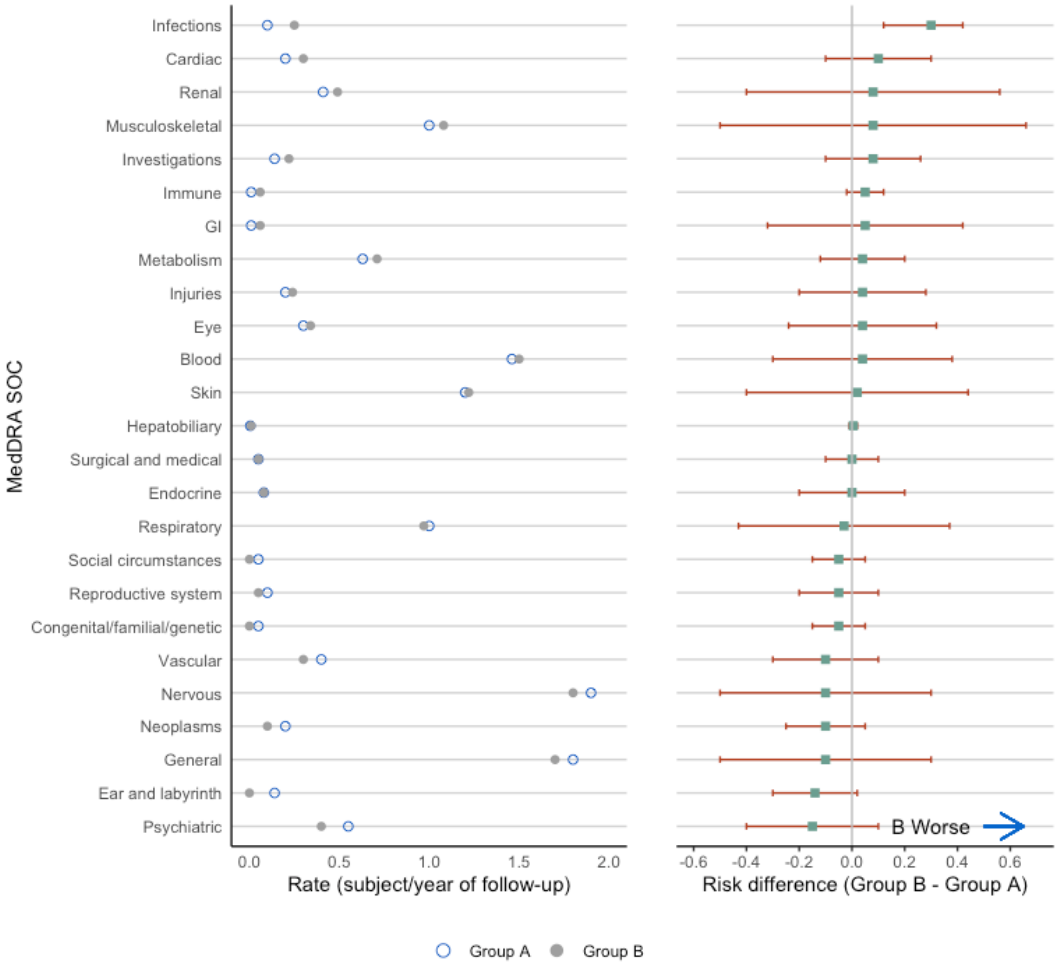

Figure 1b. Multicell dot plots using R's ggplot2.

COMPARISON OF SAS® GTL AND R'S GGPLOT2 IN CREATING MULTICELL DOT PLOTS

Both SAS® and R are popular data visualization tools. Below is a summary table that compares SAS® and R in a general way.

Aspect	Base SAS®	R
Syntax in General	Data step and procedure-based	Expression-based and functional
Data Structure	Tabular: rows and columns	Vectors, lists, dataframes, matrices, arrays, factors
Data Manipulation	Data steps, PROC SQL	Packages such as 'dplyr', 'tidyr', 'data.table', etc.
Data Analysis	Statistical procedures such as PROC FREQ, PROC LIFETEST, PROC GLM, etc.	Packages such as 'stats', 'survival', 'glm', etc.

Data Visualization/ Graphics	Graphical procedure such as PROC SGPLOT, PROC TEMPLATE	Packages such as 'ggplot2','ploty', 'ggvis', 'lattice', etc.
Libraries or Packages in Graphics	Base SAS® Graphics, ODS Graphics	Packages such as 'ggplot2', 'plotly', 'ggvis', 'lattice', etc.
Interactivity in Graphics	Limited support	Packages such as 'shiny', 'plotly', etc.
Cost	Proprietary with licensing costs	Open-source and free

Table 1. General summary based on common features and differences between SAS® and R.

SAS® GTL stands out as a robust and flexible tool within the SAS® system, designed for the creation of highly customizable graphs and visualizations.

'ggplot2' is a powerful and flexible data visualization package in R that is part of the 'tidyverse' ecosystem. It is widely used in the R community and has a large user base, resulting in extensive online resources and documentation. With R's 'ggplot2', users have the capability to intricately define and personalize the visual attributes of graphs using a declarative approach.

The following table makes a comparison between these two methods for creating specific multicell dot plots (Figure 1a and 1b) from different perspectives. The complete SAS® and R codes are attached in Appendix I and II.

#	Aspect	SAS® GTL	R's ggplot2
1	**Syntax**	Proprietary syntax (GTL)	Grammar of Graphics in ggplot2
2	**Language Style**	Procedure-based (e.g., PROC TEMPLATE)	Function-based
3	**Input Data Format**	Tabular data in wide format	Data frames in long format
4	**Code Structure**	proc template; define statgraph <your-graph>; begingraph </options> ; layout lattice </options>; /*Subplot 1*/ layout overlay; <... more-statements ...> endlayout; /*Subplot 2*/ layout overlay; <... more-statements ...> endlayout;	###Step 1: Generate two subplots separately then combine them subplot1 <- ggplot(...) + other functions subplot2 <- ggplot(...) + other functions combined_plot <- grid.arrange(subplot1, subplot2, …) ###Step 2: Create the shared legend plot_legend <-ggplot(...) + other functions get_only_legend <- function(plot) {...} legend <- get_only_legend(plot_legend) ###Step 3: Combined the legend with multicell dot plots grid.arrange(combined_plot, legend, ...)

		/*Shared legend*/ sidebar </options> ; <... more-statements ...> endsidebar; endlayout; endgraph; run; proc sgrender data = <input-data-set> template = <statgraph-template>; run;	
5	**Color**	"CX" prefix followed by six hexadecimal characters, for example, CXFF0000 (Red)	"#" followed by six hexadecimal characters, for example, #FF0000 (Red)
6	**Symbol**	Specific symbol name, such as Circle, SquareFilled, etc.	Numerical codes, for example, 1 is Circle and 15 is SquareFilled.
7	**Plotting Order of AE SOC**	Using PROC SORT before the PROC TEMPLATE procedure	Use fct_reorder() function within ggplot() for each subplot
8	**Shared Y-Axis**	layout lattice / columns = 2 rowdatarange = union columnweights=(.6 .4) columngutter=10px; 1) The argument columns = 2 specifies the number of columns in the lattice. 2) The argument rowdatarange = union scales the Y-axis data ranges separately for each row in the lattice. The two subplots down the row share the same data range and axis type.	Similar as SAS® GTL: grid.arrange(supplot1, subplot2, ncol = 2, widths = c(0.6, 0.4))
9	**Shared Legend**	Directly use SIDEBAR statement within PROC TEMPLATE	Step 1: Create a legend for a subplot using ggplot() function Step 2: Create a function to extract legend from ggplot object Step 3: Combine plot with shared legend using grid.arrange() function

Table 2. Comparison of SAS® GTL and R's ggplot2 in generating multicell dot plots.

Below are the comments for each comparison from Table 2:

1. SAS® code is concise, but the syntax may be verbose; R has more declarative syntax, making it easy for beginners to understand.

2. SAS® GTL is based on a template-driven approach. Users define templates that describe the structure and appearance of the graph. R is vectorized. It uses functions, such as ggplot(), to visualize the data.

3. When plotting the multicell dot plots, SAS® only needs one wide-formatted set of data. In R, multicell dot plot visualizations require additional effort or manipulation in long format. We need to provide two long-formatted data frames for each subplot individually.

4. The SAS® code structures are different from R's. In SAS®, the PROC TEMPLATE defines the figure template, then the SGRENDER procedure generates the output graph using that template. R produces the two subplots individually then combines them together.

5. Both SAS® and R can use hexadecimal color code, but their prefixes are different.

6. In SAS® GTL, the specific name of a marker symbol can be found at SAS® Help Center. In R, plotting character (pch) values can be found by installing the ggpubr package (install.packages("ggpubr")) and then type this line of code: ggpubr::show_point_shapes().

7. In this scenario, SAS® GTL is more efficient in plotting order of categories. We do not need to sort the order of categories (i.e., SOCs) by risk difference again when creating the plots. The plotting order of SOCs is the same as the sort order in data. In contrast, in R's ggplot2, we have to specify the order using fct_reorder() function within ggplot() for each subplot.

8. In SAS® GLT, using the LAYOUT LATTICE statement and LATTICE and ROW options can arrange the two subplots and build the shared y-axis. In R's ggplot2, the function grid.arrange() can do the same task.

9. SAS® GTL creates the user-defined legend in a more efficient way, especially when the legend is shared by multiple subplots. In R's ggplot2, creating a shared legend needs more steps.

EXPLORE SAS® GTL AND R'S GGPLOT2 IN PROPENSITY SCORE MATCHING

Dot plot visualizations can be employed for presenting the results of standardized mean difference before and after propensity score matching within Real-World Evidence (RWE) studies. Figure 3a is generated by SAS® GTL and Figure 3b is generated by R's ggplot2. Source data is a dummy data set. The complete SAS® and R codes are attached in Appendix III and IV.

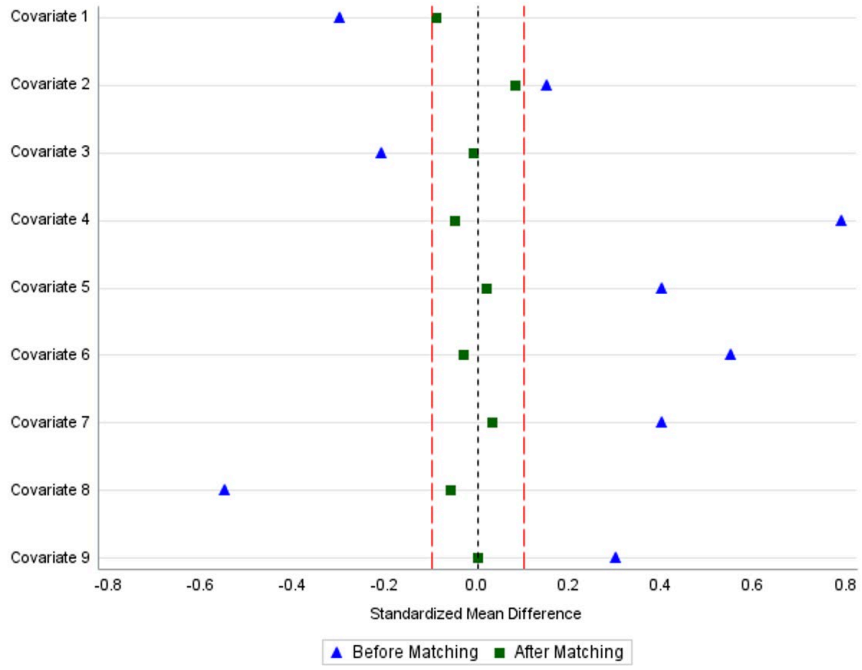

Figure 3a. Plot of absolute standardized mean differences before and after propensity score matching using SAS® GTL.

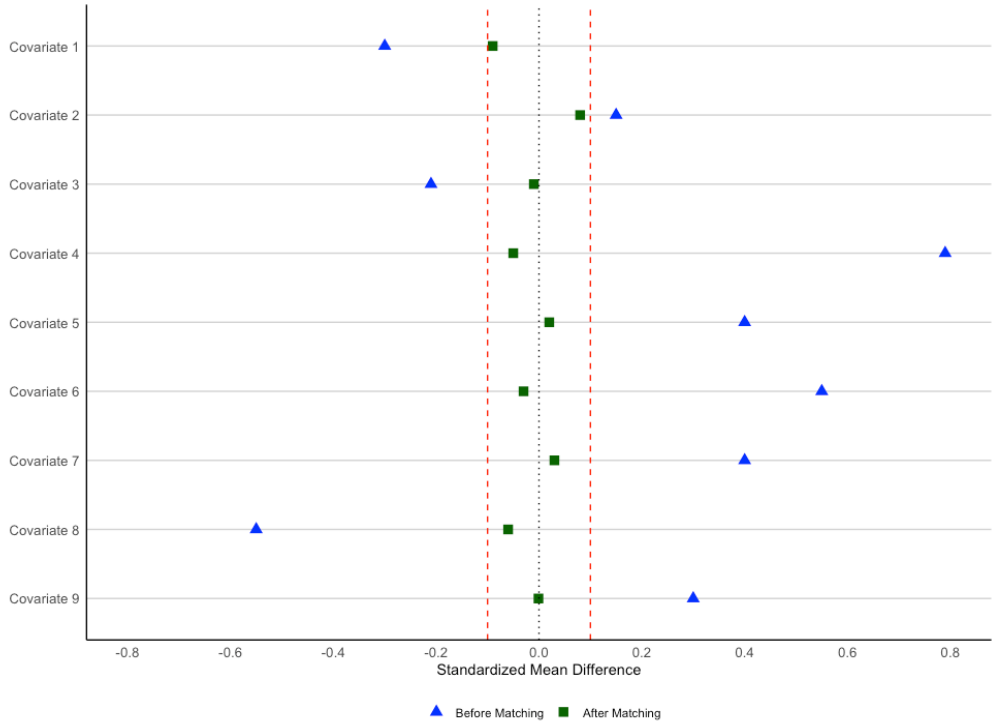

Figure 3b. Plot of absolute standardized mean differences before and after propensity score matching using R's ggplot2.

CONCLUSION

Both SAS® GTL and R ggplot2 are powerful tools and have their own advantages. The choice between them often depends on factors such as the graphic design, existing infrastructure, user familiarity, and the specific requirements of the project or organization. In this paper, we created the same dot plots using both SAS® GTL and R ggplot2 approaches and compared them from various angles. We aimed to provide readers with more references when choosing between SAS® and R for creating visualizations.

APPENDIX I: SAS® CODE FOR MULTICELL DOT PLOTS

```
* Generate sample data;
data mock_data;
  input Param $3-35 RateA RateB Risk_Difference Low  High ;
  datalines;
  Infections                          0.10  0.25 0.30 0.12 0.42
  Investigations                      0.14  0.22 0.08 -0.10 0.26
  Blood                               1.46  1.50 0.04 -0.30 0.38
  GI                                  0.01  0.06 0.05 -0.32 0.42
  Metabolism                          0.63  0.71 0.04 -0.12 0.20
  Hepatobiliary                       0.005 0.01 0.005 -0.01 0.02
  Cardiac                             0.20  0.30 0.10 -0.1 0.3
  Musculoskeletal                     1.00  1.08 0.08 -0.5 0.66
  Renal                               0.41  0.49 0.08 -0.4 0.56
  Immune                              0.01  0.06 0.05 -0.02 0.12
  Skin                                1.20  1.22 0.02 -0.4 0.44
  Injuries                            0.20  0.24 0.04 -0.2 0.28
  Eye                                 0.30  0.34 0.04 -0.24 0.32
  Surgical and medical                0.05  0.05 0.00 -0.10 0.10
  Endocrine                           0.08  0.08 0.00 -0.20 0.20
  Social circumstances                0.05  0.00 -0.05 -0.15 0.05
  Congenital/familial/genetic         0.05  0.00 -0.05 -0.15 0.05
  Respiratory                         1.00  0.97 -0.03 -0.43 0.37
  Reproductive system                 0.10  0.05 -0.05 -0.20 0.10
  Vascular                            0.40  0.30 -0.10 -0.30 0.10
  General                             1.80  1.70 -0.10 -0.50 0.30
  Ear and labyrinth                   0.14  0.00 -0.14 -0.30 0.02
  Nervous                             1.90  1.80 -0.10 -0.50 0.30
  Neoplasms                           0.20  0.10 -0.10 -0.25 0.05
  Psychiatric                         0.55  0.40 -0.15 -0.40 0.10
;
run;

proc sort data=mock_data;
       by descending Risk_Difference;
run;

* X-axis scales for two subplots;
%let xlist1 = 0.0 0.5 1.0 1.5 2.0;
%let xlist2 = -0.6 -0.4 -0.2 0.0 0.2 0.4 0.6;

proc template;
     define statgraph dotplot1;
        * Adjust width and height of image;
        begingraph / designwidth = 6.80in designheight = 6.45in border = FALSE;
            * Make 2 plots sharing y axes;
            layout lattice / columns = 2 rowdatarange = union columnweights=(.6 .4)
                             columngutter=10px;

     /*************************** Rate Subplot ****************************/
            * Define x axis;
            * Define y axis;
            * Remove frame feature;
            layout overlay / xaxisopts = (label = "Rate (subjects/year of follow-up)"
                                          offsetmin=0.03 offsetmax=0.03
                                          labelattrs=(size=8)
                                          tickvalueattrs=(size=8)
                                          display=(tickvalues line label)
                                          linearopts=(tickvaluelist=(&xlist1.)
                                                      viewmin = 0 viewmax = 2.0))

                             yaxisopts = (label = "MedDRA SOC"
```

```
                                    labelattrs=(size=8)
                                    tickvalueattrs=(size=8)
                                    display=(tickvalues line label)
                                    gridDisplay=on
                                    reverse=true)

                        walldisplay=(fill);
            * Define figure type;
            scatterplot y=param  x=RateA / markerattrs =(symbol=circle
                                                         color=cx1961C3)
                                           name='RA'
                                           legendlabel = 'Group A';
            scatterplot y=param  x=RateB / markerattrs =(symbol=circlefilled
                                                         color=cx9F9F9F)
                                           name='RB'
                                           legendlabel = 'Group B';
        endlayout;


 /************************** Risk Difference Subplot **************************/
        * Define x axis;
        * Define y axis;
        * Remove frame feature;
        layout overlay / xaxisopts = (label= "Risk difference (Group B - Group A)"
                                      offsetmin=0.01 offsetmax=0.01
                                      labelattrs=(size=8)
                                      tickvalueattrs=(size=8)
                                      display=(tickvalues line label)
                                      linearopts=(tickvaluelist=(&xlist2.)
                                                  viewmin = -0.6 viewmax = 0.7))

                        yaxisopts=(display=none gridDisplay=on reverse=true)

                        walldisplay=(fill);
            * Define figure type;
            scatterplot y=Param x=Risk_Difference / xerrorlower=low
                                                   xerrorupper=high
                                                   errorbarcapshape=serif
                                                 markerattrs=(symbol=squarefilled
                                                              color = cx6A9E91);

            * Add reference line;
            referenceline x=0 / lineattrs=(color=grey);
        endlayout;

/************** Display the legend at the bottom of the layout grid *************/
        sidebar / align = bottom spacefill=false;
            discretelegend 'RA' 'RB';
        endsidebar;

/****************************** Text and Arrow ******************************/
        * Define x axis;
        * Add text;
        drawtext "B worse" / x=540 y=85
                             drawspace=graphpixel
                             width=120
                             widthunit=pixel
                             anchor=left
                             border=false;
```

```
            * Draw arrow;
            drawarrow x1=590 y1=85 x2=620 y2=85 / drawspace=graphpixel
                                                  lineattrs=(color=cx3768B3
                                                  thickness=2px)
                                                  arrowheadshape=open
                                                  arrowheadscale=1
                                                  arrowheaddirection=out;
            endlayout;
        endgraph;
    end;
run;

* Render the template and create figure;
proc sgrender data = mock_data template = dotplot1;
run;
```

APPENDIX II: R CODE FOR MULTICELL DOT PLOTS

```
library(ggplot2)
library(tidyr)
library(dplyr)
library(gridExtra)
library(forcats)
### Generate the mock data
mock_data <- data.frame(
Param = c("Infections", "Investigations", "Blood", "GI", "Metabolism", "Hepatobiliary", "Cardiac", "Musculoskeletal", "Renal",
"Immune", "Skin", "Injuries", "Eye", "Surgical and medical", "Endocrine", "Social circumstances", "Congenital/familial/genetic",
"Respiratory", "Reproductive system", "Vascular", "General", "Ear and labyrinth", "Nervous", "Neoplasms", "Psychiatric"),
RateA = c(0.10, 0.14, 1.46, 0.01, 0.63, 0.005, 0.20, 1.00, 0.41, 0.01, 1.20, 0.20, 0.30, 0.05, 0.08, 0.05, 0.05, 1.00, 0.10, 0.40, 1.80,
0.14, 1.90, 0.20, 0.55),
RateB = c(0.25, 0.22, 1.50, 0.06, 0.71, 0.01, 0.30, 1.08, 0.49, 0.06, 1.22, 0.24, 0.34, 0.05, 0.08, 0.00, 0.00, 0.97, 0.05, 0.30, 1.70,
0.00, 1.80, 0.10, 0.40),
Risk_Difference = c(0.30, 0.08, 0.04, 0.05, 0.04, 0.005, 0.10, 0.08, 0.08, 0.05, 0.02, 0.04, 0.04, 0.00, 0.00, -0.05, -0.05, -0.03, -
0.05, -0.10, -0.10, -0.14, -0.10, -0.10, -0.15),
Low = c(0.12, -0.10, -0.30, -0.32, -0.12, -0.01, -0.1, -0.5, -0.4, -0.02, -0.4, -0.2, -0.24, -0.1, -0.2, -0.15, -0.15, -0.43, -0.20, -0.30, -
0.50, -0.30, -0.50, -0.25, -0.40),
High = c(0.42, 0.26, 0.38, 0.42, 0.20, 0.02, 0.3, 0.66, 0.56, 0.12, 0.44, 0.28, 0.32, 0.1, 0.2, 0.05, 0.05, 0.37, 0.1, 0.1, 0.3, 0.02, 0.3,
0.05, 0.1)
)
### Generate the mock data 1 for Rate Subplot
mock_data1 <- mock_data %>% select(Param, RateA, RateB, Risk_Difference) %>% pivot_longer(., cols = c(RateA,RateB),
names_to = "Var") %>% mutate(Var= replace(Var, Var == 'RateA', 'Group A'))  %>% mutate(Var= replace(Var, Var == 'RateB',
'Group B'))
### Create Rate Subplot
p1 <- ggplot(mock_data1, aes(x = fct_reorder(Param, Risk_Difference), y = value, group = Var)) +
      scale_y_continuous(breaks = seq(0.0, 2.0, by = 0.5), limits = c(0, 2)) +
      geom_point(aes(shape = Var, color = Var), size = 2) +
      scale_shape_manual(values = c(1, 16)) +
      scale_color_manual(values=c('#1961C3','#9F9F9F'))+
      coord_flip()  +
      xlab("MedDRA SOC")+
      ylab("Rate (subject/year of follow-up)") +
      theme_minimal() +
      theme(panel.border = element_blank(),
            panel.background = element_blank(),
            axis.line.x = element_line(colour = "black"),
            axis.line.y = element_line(colour = "black"),
            panel.grid.major.y = element_line(color = "lightgrey", size = 0.5, linetype = 1),
            panel.grid.major.x = element_blank(),
            panel.grid.minor.x = element_blank(),
```

```
            legend.position = "none"
 )
### Generate the mock data 2 for Risk Difference Subplot
mock_data2 <- mock_data %>% select(Param, Risk_Difference, Low, High)
### Create Risk Difference Subplot
p2 <- ggplot(data = mock_data2, aes(x = fct_reorder(Param, Risk_Difference), y = Risk_Difference)) +
      scale_y_continuous(breaks = seq(-0.6, 0.7, by = 0.2), limits = c(-0.6, 0.7), labels = scales::number_format(accuracy = 0.1)) +
      geom_errorbar(aes(ymin = Low, ymax = High), width = 0.2, color = "#B43E1D")+
      geom_point(color = "#6A9E91", shape = 15, size = 2) +
      geom_hline(yintercept = 0, lty = 1, color = "grey") +
      coord_flip() +
      xlab("") +
      ylab("Risk difference (Group B - Group A)") +
      theme_bw() +
      annotate("text", x = 1.0, y = 0.3, label = "B Worse") +
      geom_segment(aes(x = 1.0, y = 0.5, xend = 1.0, yend = 0.65), arrow = arrow(length = unit(0.5, "cm")),
                   colour = "#0066CC")+
     theme(panel.border = element_blank(),
            panel.background = element_blank(),
            panel.grid = element_blank(),
            axis.line.x = element_line(colour = "black"),
            axis.ticks.y = element_blank(),
            axis.text.y = element_blank(),
            panel.grid.major.y = element_line(color = "lightgrey", size = 0.5, linetype = 1)
 )
 ### Combine the two Plots
combined_plot <- grid.arrange(p1, p2, ncol = 2, widths = c(0.6, 0.4))
### Extract legend from ggplot object
plot1_legend <- ggplot(mock_data1, aes(x = Param, y = value, group = Var)) +
                  geom_point(aes(shape = Var, color = Var), size = 3) +
                  scale_shape_manual(values = c(1, 16)) +
                  scale_color_manual(values=c('#1961C3','#9F9F9F'))+
                  theme(legend.position = "bottom",
                  legend.title=element_blank(),
                  legend.key = element_blank(),
                  legend.direction = "horizontal") +
                  scale_fill_discrete(labels = c("Group A", "Group B"))

get_only_legend <- function(plot) {
          plot_table <- ggplot_gtable(ggplot_build(plot))
          legend_plot <- which(sapply(plot_table$grobs, function(x) x$name) == "guide-box")
          legend <- plot_table$grobs[[legend_plot]]
          return(legend)
}
### Extract legend from plot1 using above function
legend <- get_only_legend(plot1_legend)
### Combine plot with shared legend
grid.arrange(combined_plot, legend, nrow = 2, heights = c(10, 1))
```

APPENDIX III: SAS® CODE FOR PLOTTING STANDARDIZED MEAN DIFFERENCE

```
/*generate dummy data*/
data mock_data;
input cov $15. sd1 sd2;
datalines;
    Covariate 1  -0.3  -0.09
    Covariate 2  0.15  0.08
    Covariate 3  -0.21 -0.01
    Covariate 4  0.79  -0.05
    Covariate 5  0.4   0.02
    Covariate 6  0.55  -0.03
    Covariate 7  0.4   0.03
    Covariate 8  -0.55 -0.06
    Covariate 9  0.3   -0.001
;
run;

/*create plot*/
%let xlist1 = -0.8 -0.6 -0.4 -0.2 0.0 0.2 0.4 0.6 0.8;

proc template;
    define statgraph dotplot1;
        begingraph;
            layout overlay / xaxisopts = (label = "Standardized Mean Difference"
                               labelattrs=(size=8) tickvalueattrs=(size=8)
                               display=(tickvalues line label)
                               linearopts=(tickvaluelist=(&xlist1.)
                                           viewmin = -0.8 viewmax = 0.8))

                             yaxisopts=(label = " " tickvalueattrs=(size=8)
                               display=(tickvalues line label)
                               gridDisplay=on reverse=true)

                             walldisplay=(fill);

              * Define figure type;
              scatterplot y=cov  x=sd1 / markerattrs = (symbol=trianglefilled
                           color=blue) name='BM' legendlabel = 'Before Matching';
              scatterplot y=cov  x=sd2 / markerattrs =(symbol=squarefilled
                           color=darkgreen) name='AM' legendlabel = 'After Matching';

              * Define legend;
              discretelegend 'BM' 'AM' / title="";

              * Add reference lines;
              referenceline x=0 / lineattrs=(color=black pattern=shortdash);
              referenceline x=0.1 / lineattrs=(color=red pattern=mediumdash);
              referenceline x=-0.1 / lineattrs=(color=red pattern=mediumdash);

            endlayout;
        endgraph;
    end;
run;

/* Use the template to create the graph */
proc sgrender data = mock_data template = dotplot1;
run;
```

APPENDIX IV: R CODE FOR PLOTTING STANDARDIZED MEAN DIFFERENCE

```
library(ggplot2)
library(dplyr)
### Create Dummy data
cov <- paste0("Covariate ", 1:9)
sd1  <- c(-0.3,0.15,-0.21, 0.79, 0.4, 0.55, 0.4, -0.55, 0.3) #before matching
sd2  <- c(-0.09,0.08,-0.01,-0.05,0.02, -0.03, 0.03, -0.06, -0.001) #after matching
mock_data <- data.frame(cov, sd1, sd2)
### Create Dummy data
mock_data1 <- mock_data %>% select(cov, sd1, sd2)%>% pivot_longer(., cols = c(sd1,sd2), names_to = "sd") %>% mutate(sd=
replace(sd, sd == 'sd1', 'Before Matching'))  %>% mutate(sd= replace(sd, sd == 'sd2', 'After Matching'))
mock_data1$sd<- factor(mock_data1$sd, levels=c('Before Matching', 'After Matching'))
###  Generate the plot
ggplot(mock_data1, aes(x = cov, y = value, group = sd)) +
  scale_y_continuous(breaks = seq(-0.8, 0.8, by = 0.2), limits = c(-0.8, 0.8)) +
  scale_x_discrete(limits=rev)+
  geom_point(aes(shape = sd, color = sd), size = 3) +
  geom_hline(yintercept = 0, lty = 3, color = "black") +
  geom_hline(yintercept = -0.1, lty = 2, color = "red") +
  geom_hline(yintercept = 0.1, lty = 2, color = "red") +
  scale_shape_manual(values = c(17, 15)) +
  scale_color_manual(values=c('blue','darkgreen'))+
  coord_flip()  +
  xlab("")+
  ylab("Standardized Mean Difference") +
  theme_minimal() +
  theme(panel.border = element_blank(),
        panel.background = element_blank(),
        axis.line.x = element_line(colour = "black"),
        axis.text.x = element_text(size=10),
        axis.line.y = element_line(colour = "black"),
        axis.text.y = element_text(size=10),
        panel.grid.major.y = element_line(color = "lightgrey", size = 0.5, linetype = 1),
        panel.grid.major.x = element_blank(),
        panel.grid.minor.x = element_blank(),
        legend.position = "bottom",
        legend.title=element_blank() #remove legend title
  )
```

REFERENCES

Amit O, Heiberger RM, Lane PW. Graphical approaches to the analysis of safety data from clinical trials. Pharm Stat. 2008 Jan-Mar;7(1):20-35. doi: 10.1002/pst.254. PMID: 17323410.

Cornelius V, Cro S, Phillips R. Advantages of visualisations to evaluate and communicate adverse event information in randomised controlled trials. Trials. 2020 Dec 22;21(1):1028. doi: 10.1186/s13063-020-04903-0. PMID: 33353566; PMCID: PMC7754702.

SAS Institute Inc. 2016. SAS® 9.4 Graph Template Language: User's Guide, Fifth Edition. Cary, NC: SAS Institute Inc.

ACKNOWLEDGEMENT

The author would like to thank Kuldeep Sen and Michiel Hagendoorn for reviewing the paper and providing valuable feedback.

CONTACT INFORMATION

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

Yi Guo
Pfizer Inc.
yi.guo@pfizer.com