PharmaSUG China 2024 - Paper PO-10069

Elegant R: How to Use the Grid Package to Effortlessly Create Customized Graphics

Zhonger Zhu, Mochao Gao, and Yi Yang, Jiangsu Hengrui Pharmaceuticals Co.,Ltd

ABSTRACT

With its convenience and extensive applications in drawing, the R language emerges as the tool of choice for visualization tasks. Emphasis is placed on leveraging R to create customized graphics promptly, catering to the diverse needs of various functions. This paper will include the following sections:(1)A concise overview of installation packages (e.g., grid,ggplot2, etc.) utilized in R.(2)Step-by-step guidance on crafting graphs in R, illustrates the creation of combine histograms by using grid package to stitch together multiple plots. (3)Strategies for seamlessly integrating generated graphs into other functions.

INTRODUCTION

When faced with many complex and customized graphs, it can take a long time to use sas to plot, and many requirements may not be met by sas, such as the composite bar charts(***Figure 1***) drawn in this article. At this time, using R language grid package combined with ggplot2 package to draw can meet the above situation quickly and conveniently.

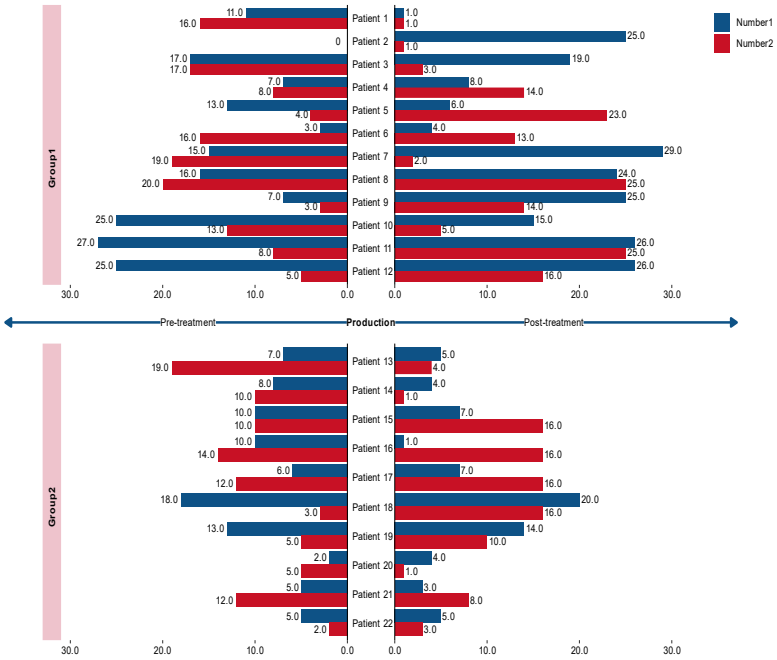

Figure 1. combine histograms

The grid package in R creates and manages viewports for drawing graphics, allowing fine control over layout and appearance. It's a foundation for advanced plotting packages like ggplot2, but doesn't offer statistical graphics directly.ggplot2 is an R package that simplifies statistical graphics creation by combining data, aesthetics, and geometry. It's easy to learn, flexible, and generates visually appealing graphs with rich information. It supports customization, layering, and statistical transformations like linear fits.

Hope that through the brief introduction of this article, you can get a simple idea in dealing with customized complex pictures, that is, reasonably determine the type of data set, decompose the pictures to understand the requirements, process the details of customization to draw beautiful pictures, and finally achieve the purpose of quickly completing the analysis of requirements.

COMBINE HISTOGRAMS

IMAGE ANALYSIS AND DISASSEMBLY

The main part of the ***Figure 1*** is a horizontal histogram, which needs to be differentiated between treatment groups. In addition, each subject had two histograms in different directions to distinguish different time points, i.e., pre-dose and post-dose. In addition, you can observe that each subject has two histogram values at each point in time, representing the specific values of the two continuous indicators, number1 and number2. This can be the value of one of the two specific detection items in the LB domain, which can be changed according to the specific requirements of your project. Therefore sum up, this graph can be broken down into four small plots, which are the first dosing time point of the Group1 group (hereinafter abbreviated: pc1r1), the second dosing time point of the Group1 group (abbreviated: PC2R1), the first dosing time point of the Group2 group (abbreviated: PC1R2), and the second dosing time point of the Group2 group (abbreviated: PC2R2).

PREPARATION FOR DRAWING

Firstly, prepare the dataset required for the image as shown in ***Figure 2***, including sbj, param, avalpre, avalpost, textpre, and textpost variables. Here you can configure any data format, such as sas dataset format, csv data format, etc., just need to import it through the relevant library of R language, but it should be noted that since ***Figure 1*** distinguishes between Group 1 and Group 2, you need to configure two datasets for drawing. The packages required for the relevant drawing are then configured, as well as the variables for the drawing details.

观测	sbj	param	avalpre	avalpost	textpre	textpost
1	Patient 1	Number1	11	1	.	
2	Patient 1	Number2	16	1	.	
3	Patient 2	Number1	0	25	0	
4	Patient 2	Number2	0	1	.	
5	Patient 3	Number1	17	19	.	
6	Patient 3	Number2	17	3	.	
7	Patient 4	Number1	7	8	.	
8	Patient 4	Number2	8	14	.	

Figure 2. Partial data graph

```
library(tidyverse)
library(gridExtra)
library(scales)
library(gtable)
library(grid)
data$sbj <- factor(data$sbj, levels = rev(unique(data$sbj)))
data$param <- factor(data$param, levels = rev(c("Number1", "Number2")))
data$avalpost <- as.numeric(data$avalpost)
data2$sbj <- factor(data2$sbj, levels = rev(unique(data2$sbj)))
data2$param <- factor(data2$param, levels = rev(c("Number1", "Number2")))
data2$avalpost <- as.numeric(data2$avalpost)
dodge_width <- 0.9
colors <- rev(c("#0E5286", "#C81125"))
color2 <- c("#EEC3CC")
tsize <- 3
tvjust1 <- 0.4
tvjust2 <- 0.5
thjust1 <- 1.2
```

```
asize <- 3
wsize <- 3
bminus <- -2.7
tminus <- -2.7
tplus <- 0.5
ymin <- 31
ymax <- 33
ahjust1 <- 20
ahjust2 <- 2.3
```

MAIN PART DRAWING

Partial drawing of PC1R1

This article only shows the code of PC1R1 for the four main parts of the code, and will not repeat the same part in the future.

```
pc1r1 <- ggplot(data, aes(x = sbj, fill=param), na.rm=FALSE) +
  geom_bar(aes(y = avalpre), position = position_dodge(width =
dodge_width), stat="identity", width = dodge_width) +
  geom_text(aes(y = ifelse(is.na(avalpre), 0, avalpre), label =
ifelse(is.na(avalpre), "NA*", ifelse(avalpre==0, "", sprintf("%.1f",
avalpre)))), position = position_dodge(width = dodge_width), hjust=thjust1,
vjust = tvjust1, size=tsize) +
  geom_text(aes(y = 1, label = textpre), vjust = tvjust2, size=tsize) +
  coord_flip() +
  theme(
    panel.background = element_rect(fill = "white", colour = NA),
    panel.grid.major = element_blank(),
    panel.grid.minor = element_blank(),
    axis.line.y = element_line(),
    axis.ticks.y = element_blank(),
    axis.text.y = element_text(color="black"),
    axis.text.x = element_text(color="black"),
    axis.ticks.x = element_line(),
    legend.position = "none",
    plot.margin = margin(t = tplus, r = 0, b = bminus, l =0, unit = "cm")
  ) +
  scale_fill_manual(values=colors) +
  scale_x_discrete(position = "top") +
  scale_y_reverse(limits=c(40, 0), breaks=c(0, 10, 20, 30),
expand=expansion(mult=c(0.05, 0)), labels = label_number(scale = 1,
accuracy = 0.1)) +
  labs(x=NULL, y=NULL) +
  annotate("rect", xmin = -Inf, xmax = Inf, ymin = ymin, ymax = ymax, fill
= color2, colour = NA) +
  annotate("text", y = ymin, x=Inf, label = "Group1", color = "black",
angle = 90, vjust=-1,hjust=5, size = 3, fontface = "bold")
```

The first four lines of code first determine that the drawn dataset is data, the independent variable is sbj, the dependent variable is avalpre, and the graph is drawn as a histogram, and the value of avalpre is displayed. Then, the range of the x-axis and y-axis is set according to the range of avalpre, and the label of Group1 is drawn by setting the text box method through the annotate function in order to display the information of group Group1.Then you get the ***Figure 3.***

You can notice that the bold part of the code is for patient 2. The values of number1 and number2 are both 0, so they will not be displayed in the image, and in order to display the 0, the textpre variable will be

created in the dataset, and when the above situation occurs, it will be assigned a value of 0 to display the "0", as shown in ***Figure 3***.

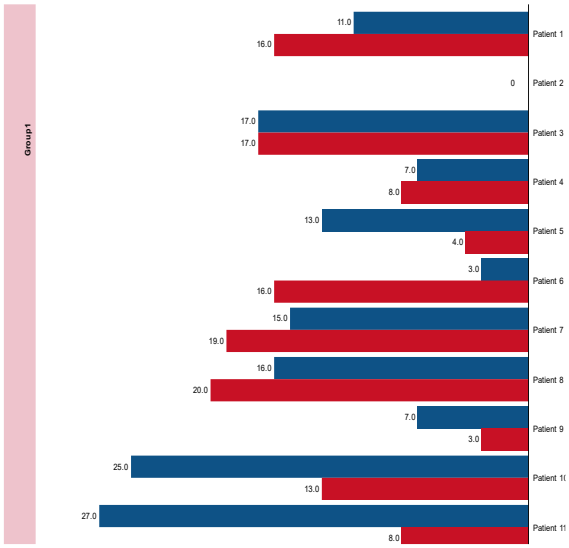

Figure 3.Picture of PC1R1

Partial drawing of PC2R1

```
PC2R1<- ggplot(data, aes(x = sbj, fill=param), na.rm=TRUE) +…+
  theme(
 legend.position = c(0.9, 0.9),
  ) +…
```

The above code is the same as the code in the PC1R1 part which can draw the ***Figure 4***, but it is worth noting that since the general legend position is in the upper right corner of the whole picture, when drawing the graph of PC2R1, you need to set the legend statement in the theme statement, by setting the position coordinates to (0.9, 0.9), where (1, 1) is the part in the upper right corner.

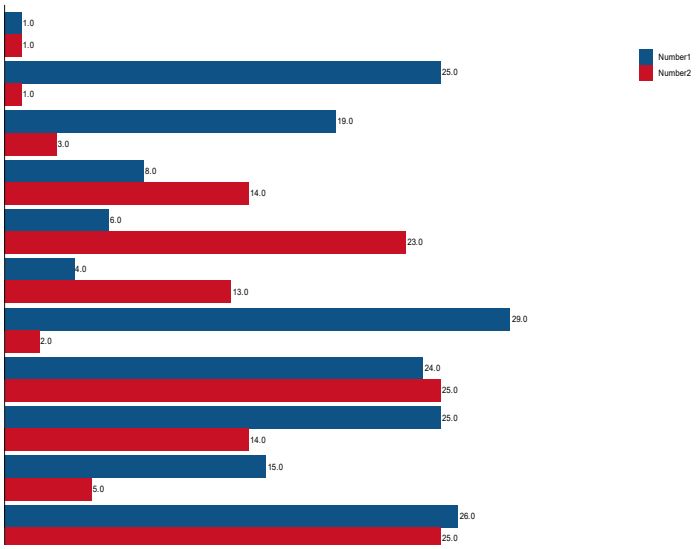

Figure 4.Picture of PC2R1

Partial drawing of PC1R2/PC2R2

```
PC1R2 <- ggplot(data2, aes(x = sbj, fill=param), na.rm=FALSE) +…+
  theme(
    plot.margin = margin(t = tminus, r = 0, b = 0, l = 0, unit = "cm")
  ) +…
```

The above code is the same as the code of the PC1R2 part, but the PC1R2 and PC2R2 parts need to rewrite the label of group1 to group2 to distinguish the treatment group, and replace the dataset data with data2. On top of that, since the Group2 part is below the whole image, you will need to resize the Group2 canvas as well. After that, you can get two images of the Group section at different points in time (***Figure 5***), PC1R2 on the left and PC2R2 on the right.

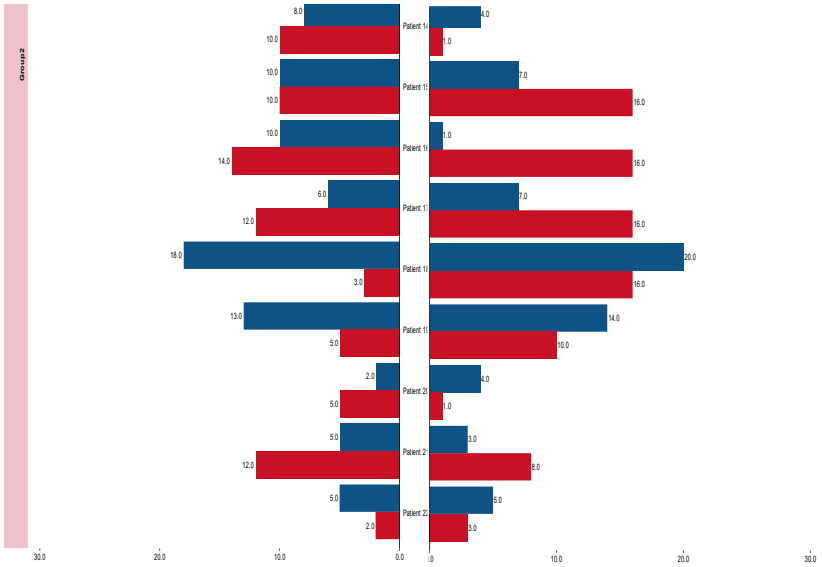

Figure 5 Picture of PC1R2 and PC2R1

Overall drawing

Through the above code, we get four main images, and use the **ggplotGrob** function to convert the ggplot object to a grid graph object (grob). This allows the graphics to be combined subsequently.

```
gc1r1 <- ggplotGrob(pc1r1)
gc2r1 <- ggplotGrob(PC2R1)
gc1r2 <- ggplotGrob(PC1R2)
gc2r2 <- ggplotGrob(PC2R2)
```

The pictures of Group1 group and Group2 group are combined to form the upper and lower pictures.

```
gr1 <- cbind(gc1r1, gc2r1)
gr2 <- cbind(gc1r2, gc2r2)
```

Later, considering that there is still a dividing line (arrow) in Figure 1, this paper also uses ggplot function to draw, but considering that it needs to fit Group1 and Group2 group diagrams more closely, it sets a relatively compact ylim, and then converts it into Grid object.

```
arrow <- ggplot() +
  geom_segment(aes(x=0.96, xend = 1, y=1, yend = 1), arrow = arrow(length =
unit(0.1, "inches"), type = "closed", ends = "both"), size = 1,
color="#0E5286") +
  geom_label(aes(x = 0.98, y = 1), label = "Production", label.size=0,
color = "black", size = wsize, fill = "white", fontface = "bold") +
  geom_label(aes(x = 0.97, y = 1), label = "Pre-treatment", label.size=0,
color = "black", size = wsize, fill = "white") +
  geom_label(aes(x = 0.99, y = 1), label = "Post-treatment", label.size=0,
color = "black", size = wsize, fill = "white") +
  theme_void() +
  theme(plot.margin = margin(t = -4, r = 0, b = -4, l = 0, "cm")) +
  coord_fixed(ratio = 10) +
  ylim(0.999, 1.001)
garrow <- ggplotGrob(arrow)
```

Finally, Group1 group diagram, Group2 group diagram and dividing line are combined by **grid.arrange**, and Figure 1 is finally obtained.

```
grid.arrange(gr1, garrow, gr2, newpage=TRUE, nrow=3.5)
```

COMMUNICATE WITH OTHER FUNCTIONS

When you're done drawing a graph, you need to give it to other departments, such as statisticians, to review it, and you may get some feedback on the changes, which is where R comes in. They may have doubts about the layout of the image, in this case you can adjust the position of the legend, the color of the column, etc., or they may have problems with the text content, then you can modify the content of the text statement, etc., which can be adjusted with R.

CONCLUSION

When faced with some more complex graphics, SAS may not be able to quickly help with drawing in the first place, so you can choose R, an open source tool, as the first choice for drawing, and you need to do the following:

1. Determine the required dataset format for the graph

2. Determine the drawing category and disassemble the drawing

3. Combine graphics, adjust details such as footnotes and legends

4. Submit it to statisticians and other relevant functions for review and communicate

As an open-source tool, R is gradually being used in daily work, so as a sas programmer, you need to constantly learn more than SAS to cope with these complex drawing needs.

REFERENCES

- R Graphics. *Paul Murrell*

CONTACT INFORMATION

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

- Name:Zhonger Zhu
- Enterprise: Jiangsu Hengrui Pharmaceuticals Co.,Ltd
- E-mail: zhonger.zhu.zz6@hengrui.com