PharmaSUG 2026 - Paper DV-304

Path to Consistent Clinical Graphics: An R Shiny Gallery

Michelle Harwood and Austin Taylor, Quantitative Sciences, Alexion, AstraZeneca Rare Disease

ABSTRACT

Consistency in clinical reporting is critical, yet figure styles often vary across studies and teams. Limited awareness of modern plotting capabilities often leads teams to revert to outdated or simple visualizations. We developed an R Shiny application that serves as a centralized gallery of commonly used clinical figures. Each figure is paired with interactive customization and replicable R code. Our goal is to standardize figures and coding practices to improve cohesion, reproducibility, and efficiency across projects.

The application provides a gallery of commonly used figures, such as the Kaplan-Meier, forest, and waterfall plots, while allowing users to tune statistical layers (confidence intervals, censoring marks) and layout options (faceting, annotations) through an intuitive interface. Every selection dynamically updates both the plot and the underlying R code snippet, enabling all users, including new R users, to adopt best practice templates, learn by example, and rapidly produce fit-for-purpose figures.

A key design principle is that the code operates directly on CDISC Pilot ADaM data, increasing reproducibility and seamless integration into existing analysis workflows. This paper will describe the application architecture, standardization guidelines, and the customization options available to users. We will also share lessons learned on building the application and onboarding new R users. This approach encourages best practices and cohesion, enabling teams and new R users to produce consistent, reproducible clinical graphics with confidence.

INTRODUCTION

Consistency in clinical reporting is essential for clear communication, efficient review, reproducibility, and decision-making; however, figure styles and coding approaches often differ across studies and teams, leading to inconsistent visuals. This variability is particularly common due to limited awareness of modern plotting capabilities, which often leads teams to simplify visualizations.

To address these challenges, our objective is twofold: i) to establish standardized visual conventions that bring cohesion to clinical graphics, such as typography, color, axes, and common statistical attributes; and ii) to provide reusable code templates that accelerate figure production while preserving cohesion and reproducibility. By coding best practices and making them accessible through an R Shiny application, we aim to reduce variability and improve consistency and clarity of output across studies.

Here, we present an R Shiny application that serves as a centralized gallery of common clinical figures, including Kaplan-Meier, forest, waterfall, and spaghetti plots. The app allows interactive customization, such as tuning confidence intervals, faceting, and annotations, while also providing reproducible R code based on the customizations selected. Operating on CDISC pilot ADaM inputs enables direct integration with existing analysis workflows and increases usability, particularly for new R users. This paper details the standardization guidelines collated, the R Shiny app's design and implementation, and the pathway taken for onboarding new R users. The goal of this approach is to foster cohesion, enabling teams and new R users to produce consistent, reproducible clinical graphics with confidence.

FIGURE STANDARDIZATION FRAMEWORK

The first step of our approach was to establish a practical list of visualization standards for teams to adopt. We reviewed internal study submissions, style guides from Nature [1], New England Journal of Medicine [2], and publications [3] and assembled a standardized framework that balances flexibility and consistency.

We implemented enforceable standards for routine use, including:

- *Typography:* Axis labels in Helvetica at 12pt; tick labels in Helvetica at 10pt
- *Spacing:* X and Y label offsets equal to one letter height
- *Decoration:* No extra boxes or lines
- *Label Orientation:* Y axis centered at midpoint, oriented bottom-to-top
- *Aspect ratio:* 3 : 2 or 4 : 3 as default; for vertical stacks the total height scales with number of panels
- *Color palette:* company-branded palette while considering accessible contrast
- *Units:* Units in axis labels are enclosed in parentheses

The initial figure library focuses on high-impact, commonly requested plots: Kaplan-Mier, forest plot, mean over time, boxplot, spaghetti plot, bar chart, and waterfall plot (Table 1). Each template encodes the visual standards above, along with figure-specific customizations which will be demonstrated in a later section.

Plot Type	**Customizations Available**
Kaplan-Meier Plot	*Confidence bands* *Hazard ratio estimates* *P-value estimates* *Risk table*
Forest Plot	*Select data type: odds ratio, mean change, hazard ratio* *Show estimates* *Add x-axes arrows*
Mean over Time	*Select data type: change from baseline, mean and SE* *Number of observations* *Different line types* *Baseline on x-axis* *Nudge points*
Boxplot	*Add mean line* *Show all points*
Spaghetti Plot	*Add mean line* *Add median line*
Waterfall	*Single arm facet* *Add annotation* *Add horizontal lines* *Hide patient IDs* *Remove space between bars*

Table 1. Customization options available for each plot type

R SHINY APPLICATION – FUNCTIONALITY

There are several unique design attributes we have applied to achieve a user-friendly interface, which can be divided into three categories: interactivity, technology stack and design, and data handling.

INTERACTIVITY

In the app, users can select the figure type in a series of tabs (Figure 1). For each figure, the user can select from figure-specific customization options, which, when selected, the figure preview will update in real time (Figure 2). Parameters can be explored and compared, and when satisfied, the user can select “Show code” to reveal the corresponding R code (Figure 3). If further changes are made after the code is displayed, the code snippet is automatically hidden until new selections are finalized and “Show code” is clicked again.

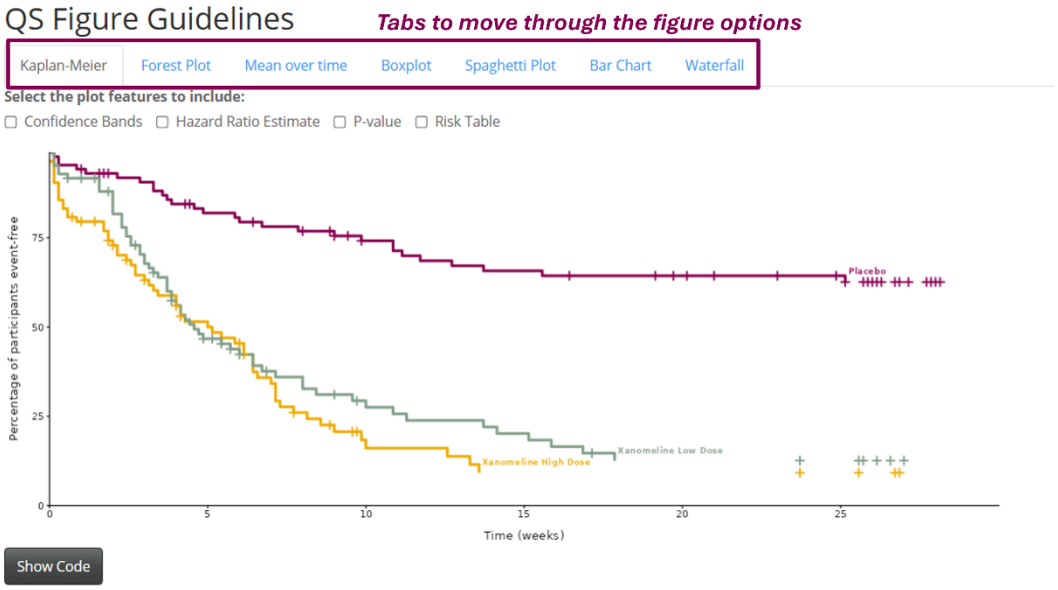

Figure 1. Application layout demonstrating the ability to select figure types

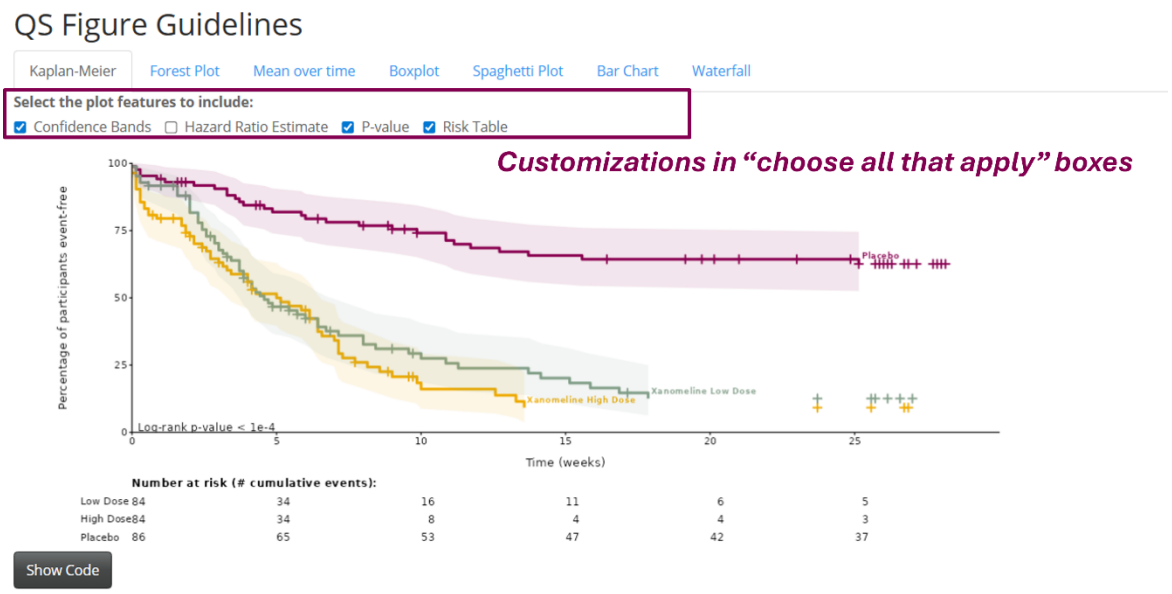

Figure 2. Application layout demonstrating the ability to select customizations unique to the figure type

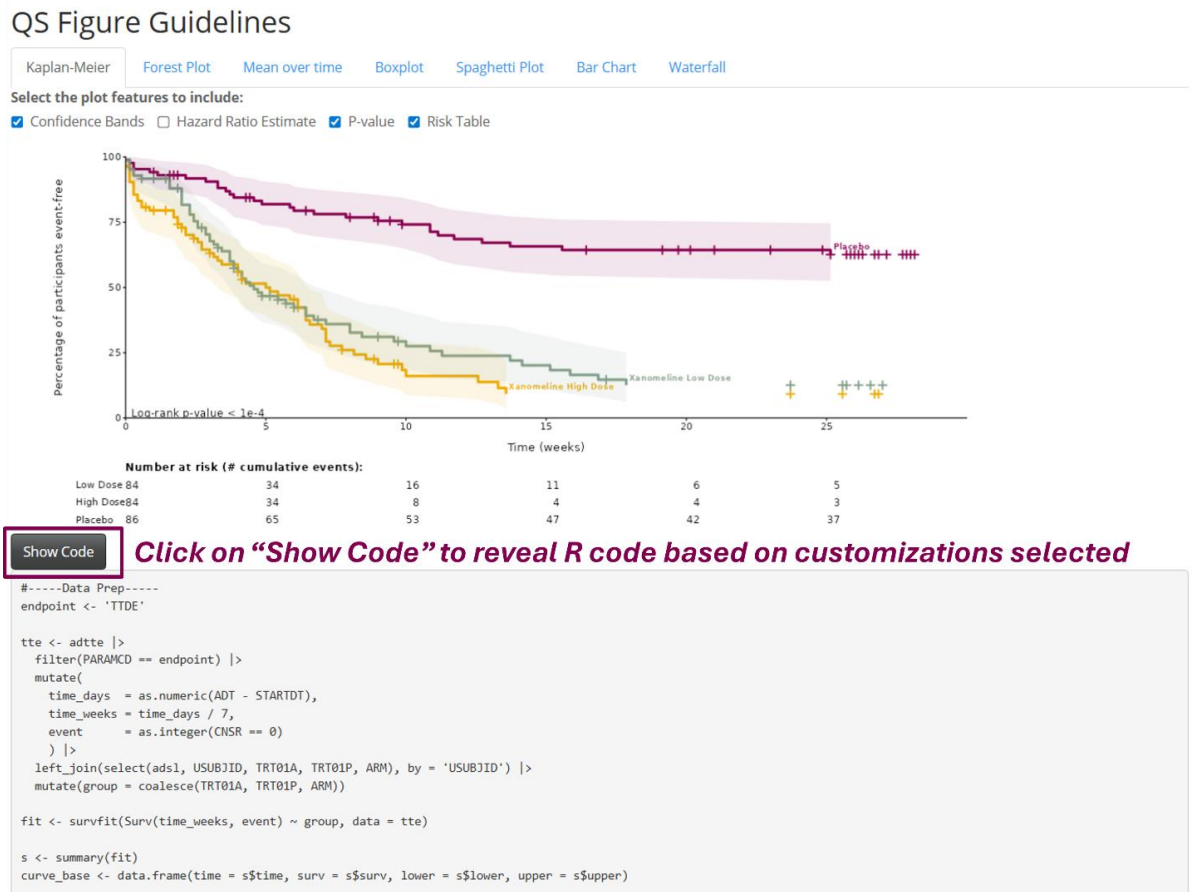

Figure 3. Application layout demonstrating the ability to reveal R code once selections are made

TECHNOLOGY STACK AND DESIGN

Built with R and Shiny, the app primarily uses the ggplot2 package for visualizations. The ggplot2 package allows us to specify a theme (*theme_classic()*) while adding code adjustments based on our standardized criteria.

```
   theme_classic() +
   theme(
     axis.title = element_text(size = 12),
     axis.text  = element_text(size = 10),
     axis.title.x = element_text(margin = margin(t = 12)),
     axis.title.y = element_text(margin = margin(r = 12)),
     legend.title = element_blank())
```

Program 1. Example code for theme within ggplot plotting function

The ***ggplot2*** package also allows additional layers to be added to the plot in separate statements. This feature is beneficial when coding the customizations, as additional layers can be added in *if()* statements.

```
  if ("points" %in% input$plot_features_box){
    code <- paste0(code, "\n
#-----Plotting-----
p<- ggplot(alt_df, aes(x = TRTA, y = AVAL)) +
  geom_boxplot(aes(color=TRTA),show.legend = FALSE, outlier.shape = NA)+
  geom_point(aes(color=TRTA), position=position_jitter(width=0.2),
  alpha=0.4, size=1, show.legend = FALSE)
            ")
  }else{
    code <- paste0(code, "\n
#-----Plotting-----
p<- ggplot(alt_df, aes(x = TRTA, y = AVAL)) +
  geom_boxplot(aes(color=TRTA),show.legend = FALSE)
      ")
  }

  if ("mean_box" %in% input$plot_features_box){
    code <- paste0(code, "\n
p<-  p+
  #Median
  stat_summary(fun = median, geom = 'crossbar',aes(color = TRTA),
    width = 0.7,fatten = 0.5,size = 1, show.legend = FALSE) +
  geom_line(data = alt_df,aes(x = as.numeric(TRTA), y=min(AVAL), linetype = 'Median'),
    size = 0, show.legend = TRUE, inherit.aes = FALSE
  ) +
  #Mean
  stat_summary(fun = mean, geom = 'crossbar', aes(color = TRTA),
    width = 0.7, fatten = 0.5,linetype='dotted', size = 1, show.legend = FALSE)+
  geom_line(data = alt_df, aes(x = as.numeric(TRTA),y=min(AVAL), linetype = 'Mean'),
    size = 1, show.legend = TRUE, inherit.aes = FALSE
  )
```

Program 2. Example code for adding plot layers based on the input features selected

Each plot is implemented as a quotable function, so it can be invoked via *renderPlot()* for the active display, and via *renderUI()* for displaying the code used. This reactive programming ensures that the code and plot updates are synchronized.

```
show_code_reactive_bar <- reactiveVal(FALSE)

observeEvent(input$show_code_bar, {
   show_code_reactive_bar(TRUE)
 })

observeEvent(input$plot_features_bar, {
   show_code_reactive_bar(FALSE)
 })

output$bar_plot <- renderPlot({
   bar_plot_code <- create_bar_plot(input)
   eval(parse(text = bar_plot_code))
 }, height = 400, width = 1000)

output$bar_code_text <- renderUI({
   if (show_code_reactive_bar()) {
     bar_plot_code <- create_bar_plot(input)
     tags$pre(bar_plot_code)
   }
 })
```

Program 3. Example code from server.R demonstrating the use of renderPlot() and renderUI() calling on the same plotting function.

DATA HANDLING

Figures operate directly on ADaM inputs, leveraging the CDISC Pilot Data [4]. Using pilot data allows us to preserve realistic structures and variable naming without exposing real patient information in the app, permitting the app to showcase end-to-end behavior safely. By using pilot datasets, users can clearly see how inputs flow into the rendered plots and generated R code. The R code is separated by “Data Prep” and “Plotting” to help the user understand and break down the code usage. Starting with pilot data lowers barriers to real-world adoption, where users can swap inputs for study datasets with minimal changes.

R SHINY APPLICATION – CUSTOMIZATION

Kaplan-Meier plot (Figure 4) - users can optionally display confidence bands, hazard ratio and p-value estimates, and a risk table positioned below the plot. These elements can be toggled independently to tailor the level of statistical detail.

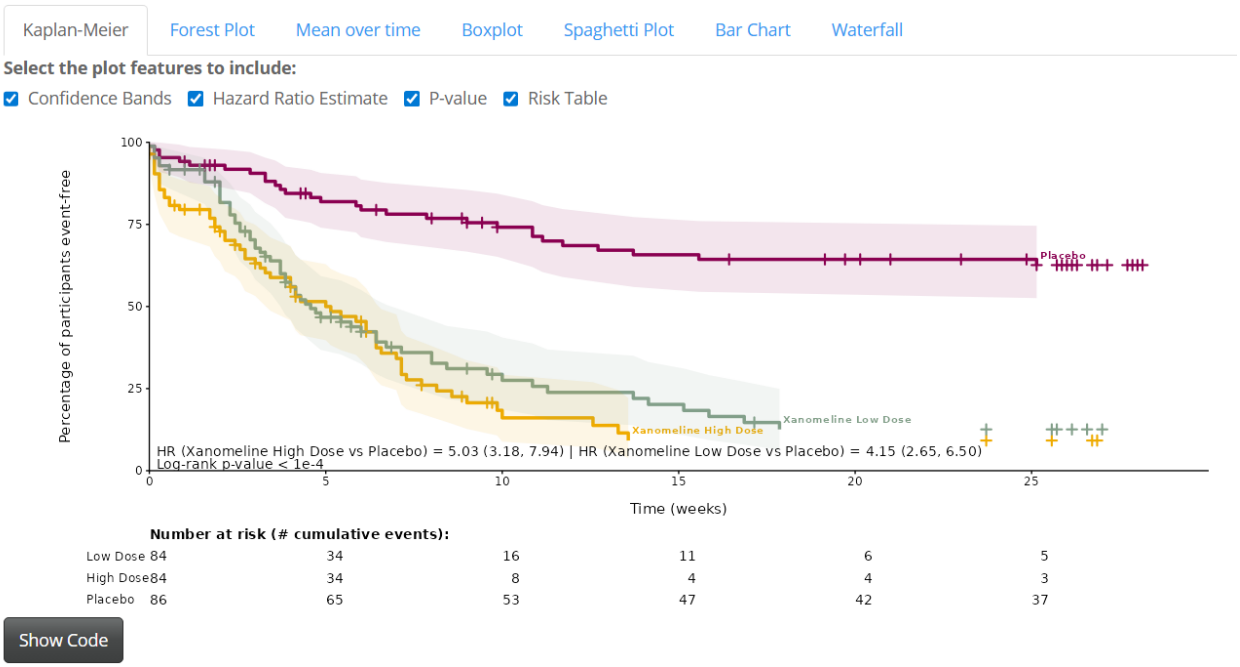

Figure 4. Example figure for Kaplan-Meier plot with all customization options selected

Forest plot (Figure 5) - users first select the summary measure: odds ratio, mean change, or hazard ratio. After selecting the data type, users can i) add columns to display arm-specific estimates (Treatment and Placebo), and/or ii) add directional arrows to the x-axis label to reinforce the interpretation of effect direction.

Mean over time plot (Figure 6) - users can plot either change from baseline or mean with standard error (SE). Additional options include i) adjusting line types by treatment arm, ii) displaying the baseline measurement at time zero on the y-axis, iii) nudging point positions to reduce overlap when values coincide, and iv) adding a table below the plot that reports the number of observations per treatment at each time point.

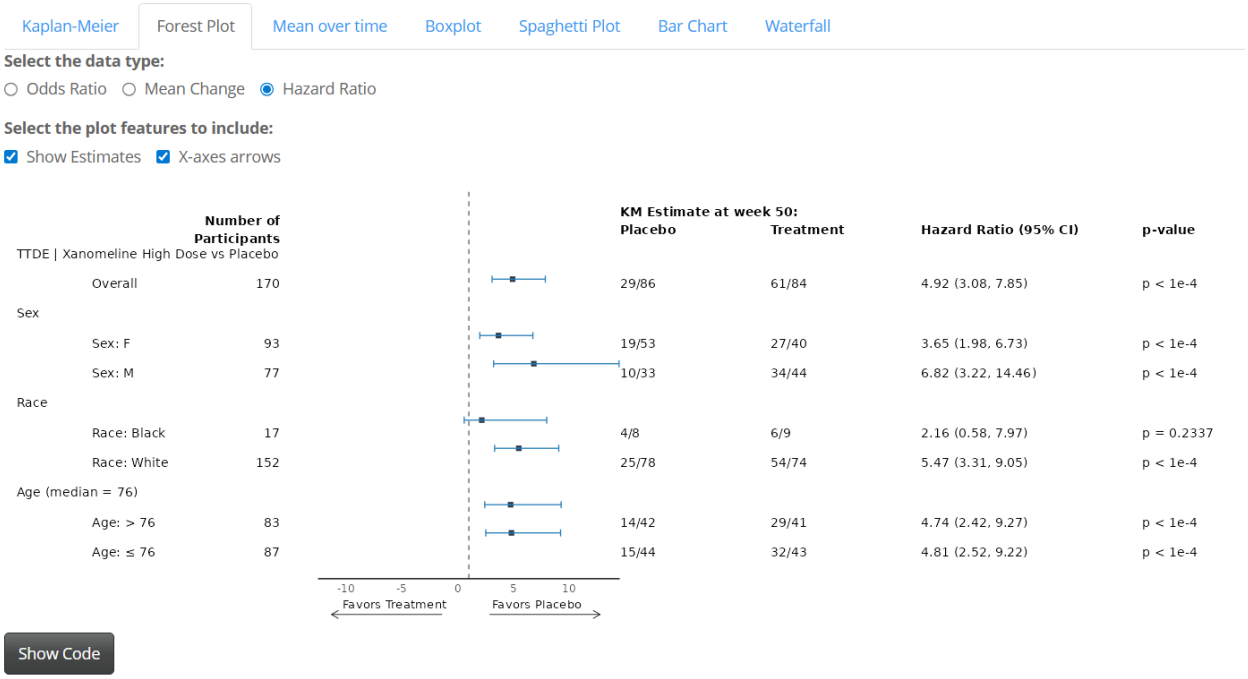

Figure 5. Example figure for forest plot with hazard ratio and all customization options selected

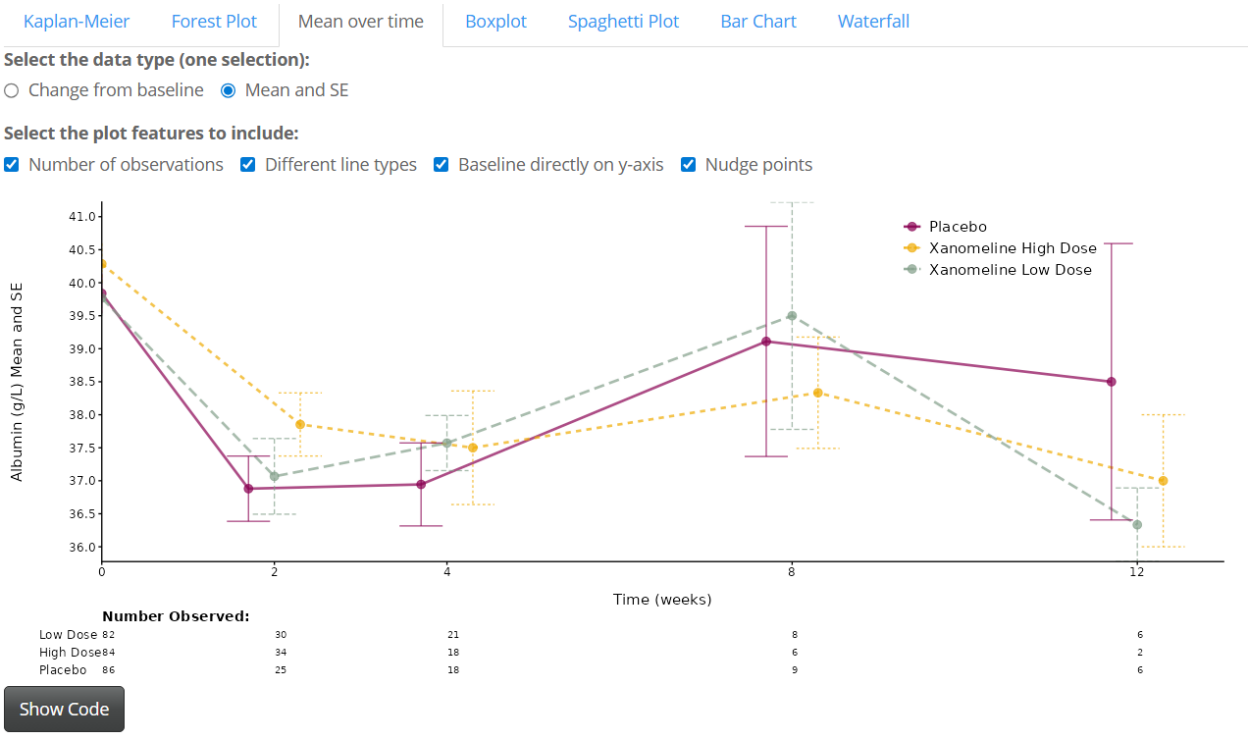

Figure 6. Example figure for mean over time plot, plotting mean and standard error (SE) values and all customization options

Box Plot (Figure 7) - Users can enhance the default box plot by overlaying mean estimates as a dotted line, to complement the default solid median line, and display all individual points (beyond outliers) to show the full distribution.

Spaghetti plot (Figure 8) - By default, all individual patient trajectories are shown with partial transparency. Users may add summary lines for the mean and/or median, which will use distinct line types.

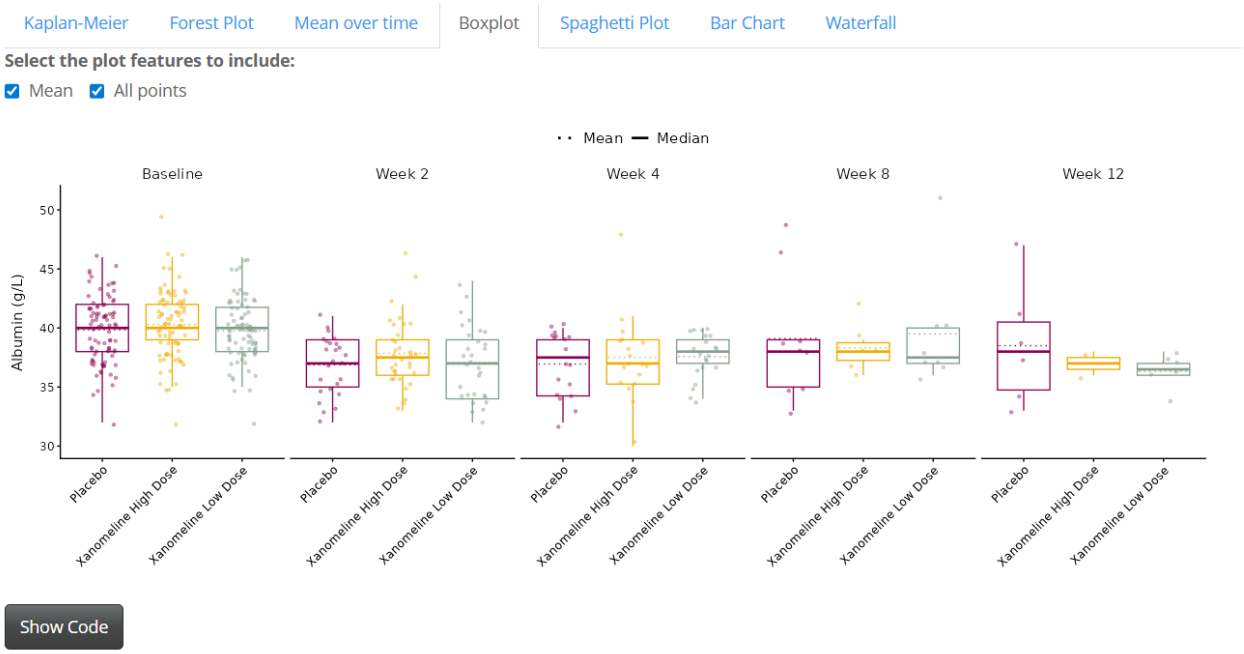

Figure 7. Example figure for boxplot with both customization options selected

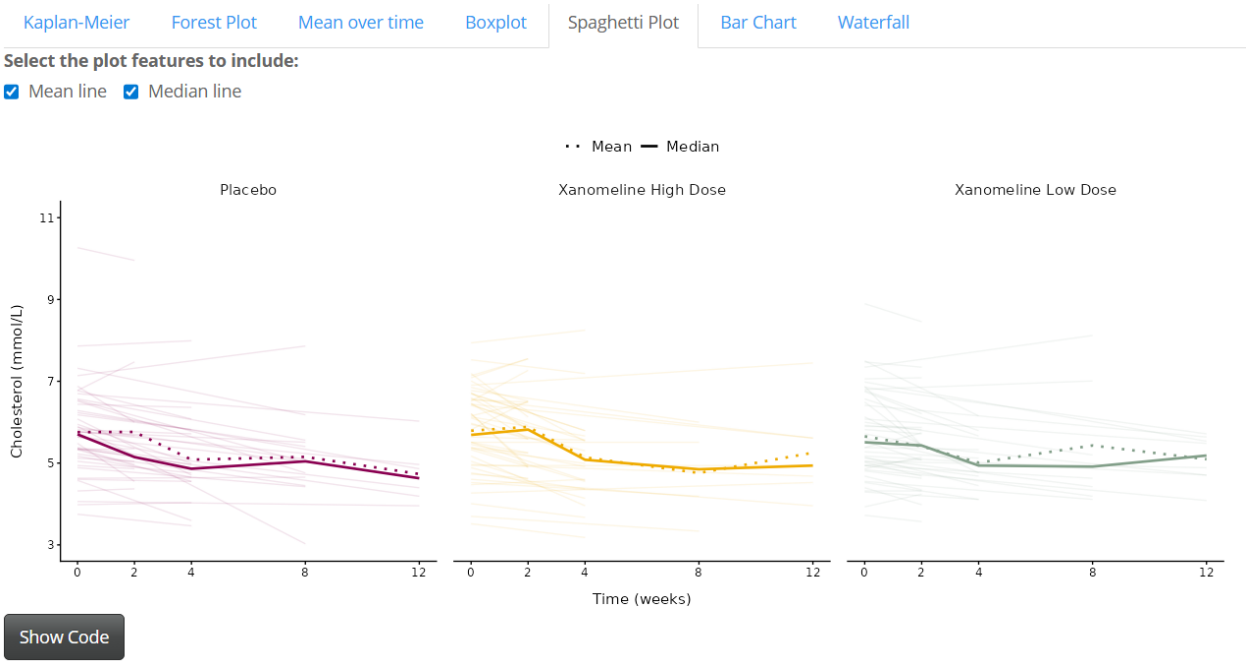

Figure 8. Example figure for spaghetti plot with both mean and median line displayed

Bar Plot (Figure 9) - no customizations are currently available, given the straightforward nature of the visualization. Planned enhancements may include options for stacked bars or horizontal orientation.

Waterfall Plot (Figure 10) - users can i) add annotations to classify outcomes based on a pre-specified criterion, ii) draw horizontal reference lines to delineate criterion thresholds, iii) hide patient IDs, iv) remove spaces between bars, or v) facet the plot by treatment arm to compare distributions across groups.

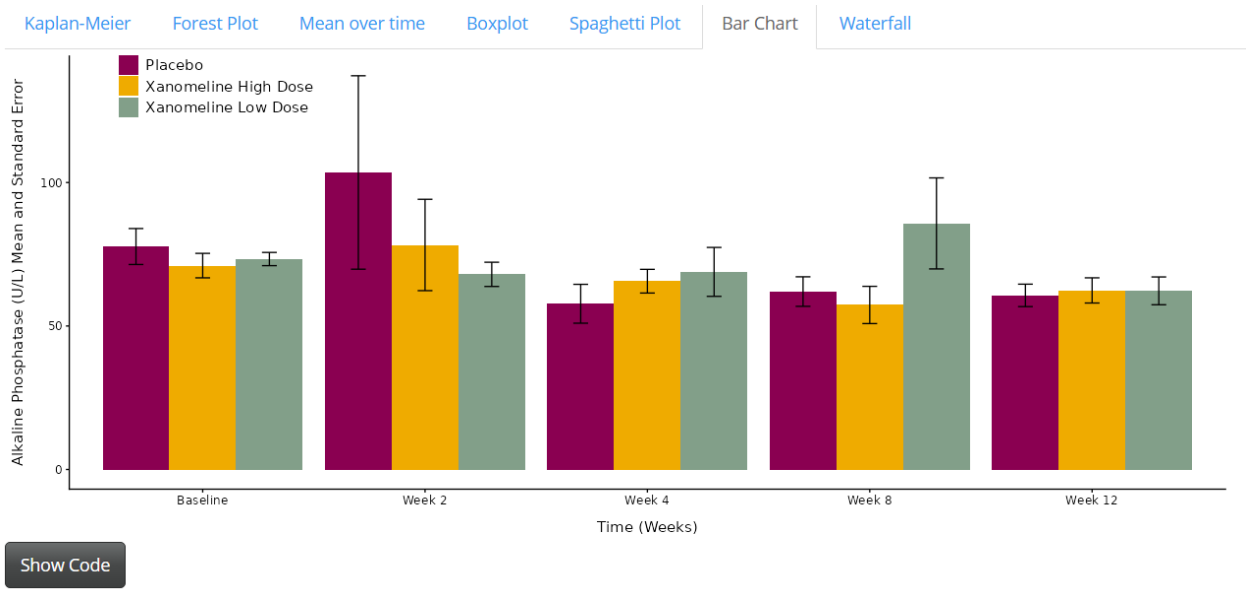

Figure 9. Example figure for bar chart, which currently does not have customization options available

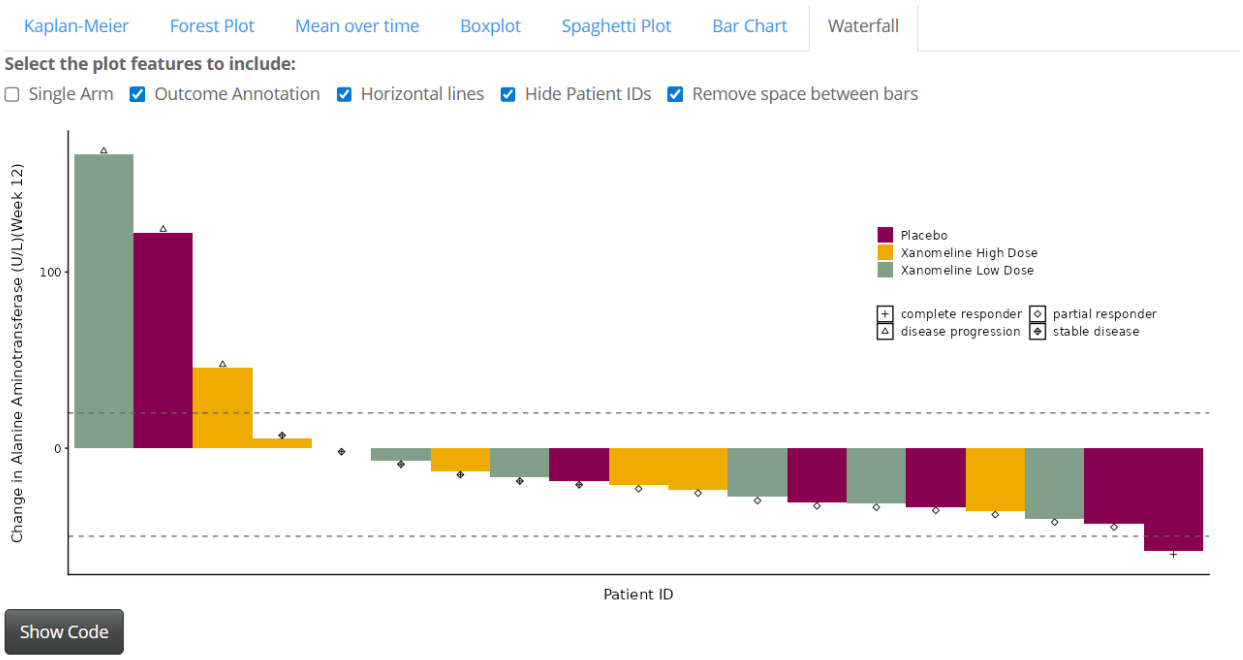

Figure 10. Example figure for waterfall plot with all customizations and displaying all treatment arms together

ONBOARDING AND WORKFLOW INTEGRATION

To get users onboarded, we designed an onboarding plan that builds foundational R skills to users first, followed by the introduction to the application once users are comfortable with working with data in R. The pathway begins with a series of structured R training sessions, covering core syntax, data manipulation with dplyr, and dedicated modules on statistical programming and CDISC concepts.

After this baseline is established, users transition to the app. The app also incorporates inline code explanations to clarify why specific parameters are chosen and point to areas for further customization. This staged approach equips new R users with core competencies needed to leverage the app effectively in real workflows.

CONCLUSION

This work presents a standardized approach to clinical graphics, operationalized through an R shiny application that couples interactive customization with reproducible code. Future enhancements will focus on expanding the figure library (such as volcano plots and heatmaps), and customization features (such as axis breaks). Overall, this approach implements best practices and fosters cohesion, enabling both teams and new R users to produce consistent, reproducible clinical graphics with confidence.

REFERENCES

[1] Nature Research. “Preparing figures – our specifications.” Accessed January 2026. Available at https://research-figure-guide.nature.com/figures/preparing-figures-our-specifications/

[2] The New England Journal of Medicine. “New manuscripts preparation instructions.” Accessed January 2026. Available at https://www.nejm.org/author-center/new-manuscripts

[3] Ou, F., Le-Rademacher, J. G., Ballman, K. V., Adjei, A. A., Mandrekar, S. J. 2020. “Guidelines for statistical reporting in medical journals”. Journal of thoracic oncology, 15: 1722-1726

[4] CDISC organization, “sdtm-adam-pilot-project” GitHub repository. Accessed January 2026. Available at https://github.com/cdisc-org/sdtm-adam-pilot-project

ACKNOWLEDGMENTS

We would like to acknowledge and thank other members of Alexion Quantitative Sciences for their support and valuable feedback throughout this initiative: Peipei Shi, Hui Liu, Ying Tang, Julien Laffaire, and Prafulla Girase.

CONTACT INFORMATION

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

Michelle Harwood
Alexion, AstraZeneca Rare Disease
Michelle.Harwood@alexion.com

Austin Taylor
Alexion, AstraZeneca Rare Disease
Austin.Taylor@alexion.com