Paper PP04

Interactive Patient Profiles in R Shiny: Enhancing Narrative Review in Clinical Studies

Tzu-Chien (Bruce) Chuang, Pfizer Inc., New York, NY, USA
Danny Hsu, Pfizer Inc., New York, NY, USA

ABSTRACT

Traditional patient profile reports often use static tables with limited interactivity, making it difficult for medical writers and clinical reviewers to explore subject-level data efficiently. To address this, we developed an R Shiny application that enables dynamic, code-free exploration of clinical data from oncology and vaccine studies. The tool integrates CDISC-compliant datasets (DM, AE, CM, MH, FA, EX, LB) and provides interactive visualizations of adverse events, lab data, concomitant medications, medical history, and reactogenicity. Built with R packages-plotly, ggplot2, and DT, it allows filtering by treatment arm and subject ID, with hover-enabled pop-ups for added context. Sidebar panels summarize demographics and diagnosis, while tabbed views organize key safety data. This modular, user-friendly interface improves data clarity, reduces manual effort, and supports consistent, high-quality patient profile generation. The application showcases how R Shiny can bridge clinical data and medical writing, aligning with PHUSE's mission to advance analytics and automation in clinical research.

INTRODUCTION

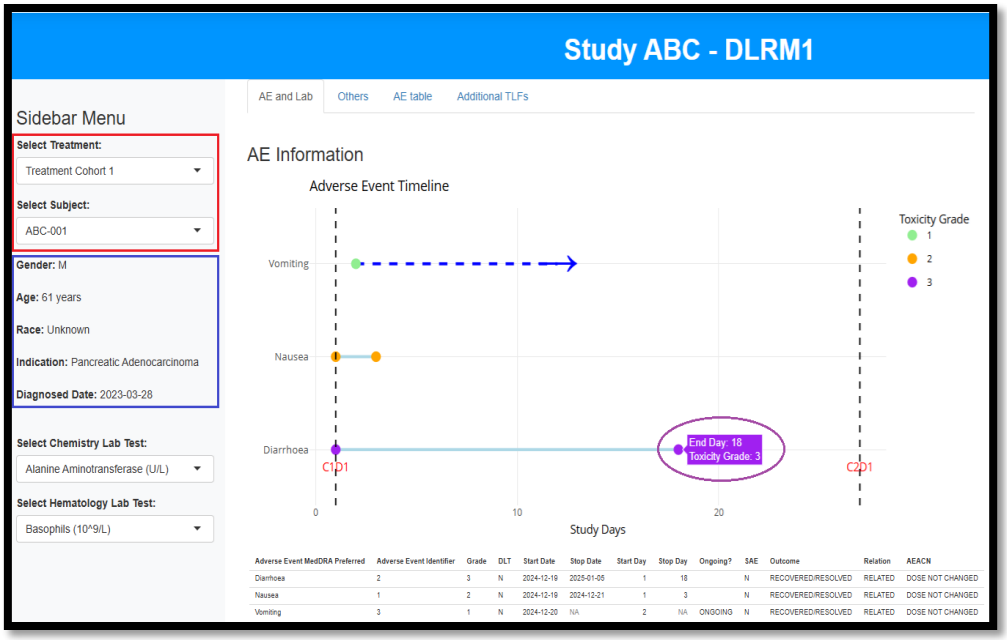
Patient profiles are essential for summarizing individual clinical experiences in regulatory submissions and internal reviews. However, traditional static formats can be time-consuming to generate and difficult to navigate, especially when dealing with large volumes of subject-level data. Some commercial tools already exist for interactive patient profiling, especially in the clinical trial space. However, many of them are limited in terms of customization and tight integration within specific ecosystems. With R Shiny, we're able to build a flexible and cost-effective platform that aligns directly with our specific needs. This is particularly valuable when working with SDTM data, as we can tailor the visualization, control logic, and filters to our team's exact specifications—and deploy it securely on our own server or in the cloud. To meet these demands, we developed an interactive patient profile application using R Shiny, tailored for oncology and vaccine studies.

The application integrates CDISC-compliant datasets—including DM, AE, CM, MH, EX, and FA—and presents them through a modular, user-friendly interface. Users can filter by treatment arm and subject ID to explore adverse event timelines, lab trends, concomitant medications, medical history, and reactogenicity data. The app uses plotly for interactive charts, ggplot2 for visualizations, and DT for dynamic tables. Hover-over popups provide immediate access to key clinical details, improving clarity and reducing the need to cross-reference static listings.

EXAMPLES

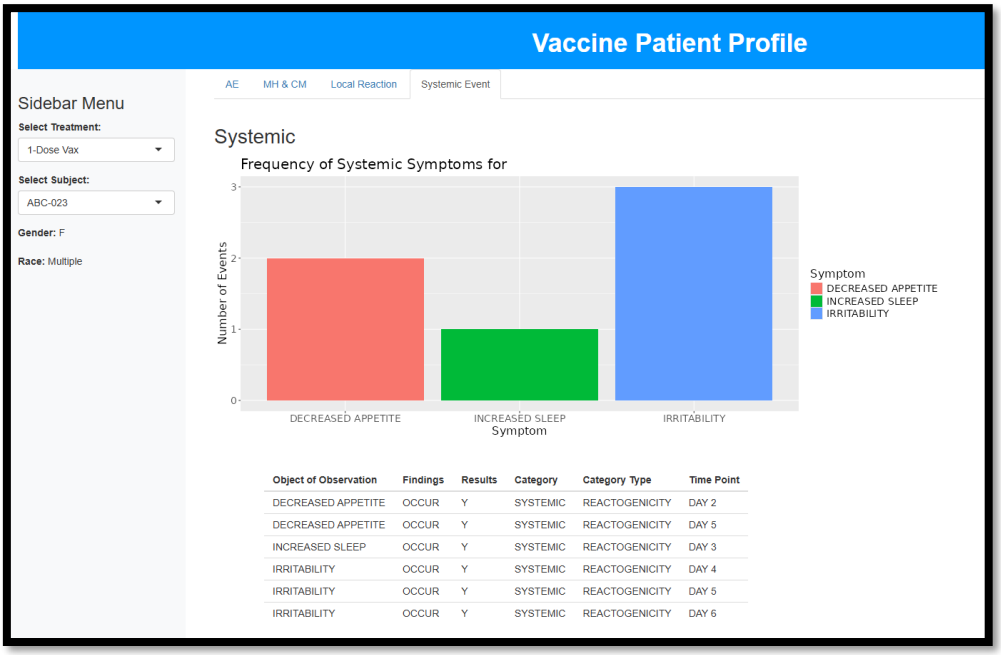
1. Example of AE Information

The Sidebar Menu on the left (highlighted in red) allows users to select a treatment arm and subject ID. Once a subject is selected, their information is automatically populated in the blue-highlighted section. At the top of the main panel, a dynamic figure visualizes the selected subject's adverse event (AE) profile, with vertical lines indicating dosing events (e.g., Cycle 1 Day 1 and Cycle 2 Day 1 in this example). Interactive tooltips (highlighted in purple) appear when hovering over the figure, providing additional context. Below the figure, a detailed AE table presents structured information for in-depth review.



2. Example of Systemic Event

Once users select a treatment arm and subject ID, the application automatically generates a bar plot and displays relevant systemic event details. This visualization enhances clarity and helps the team quickly identify patterns, trends, or anomalies in the data.

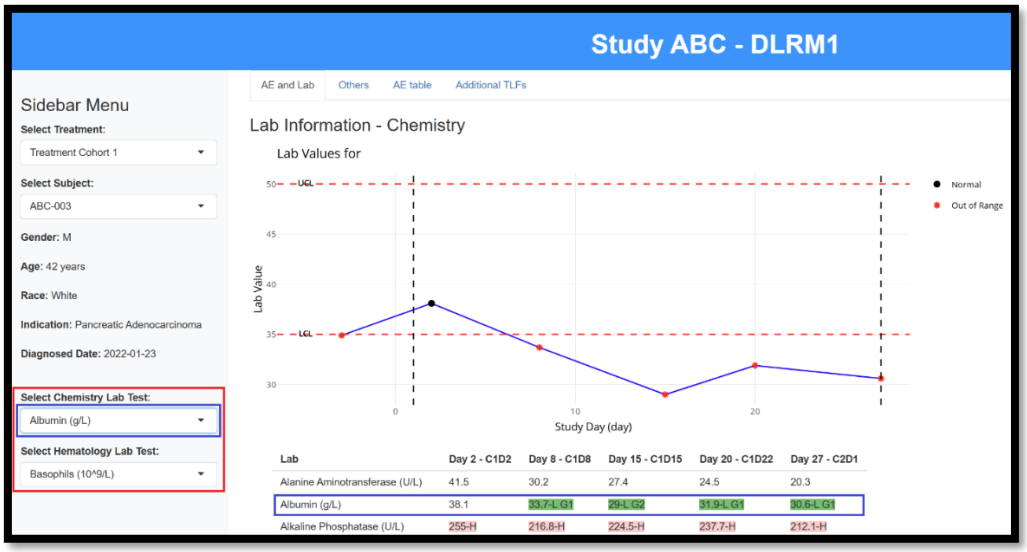


3. Example of Lab Information

A different subject (Subject ABC-003 in this example) was selected, and the subject's information was automatically updated in the sidebar panel as well. The lab parameter of interest can also be selected from the sidebar (highlighted in red). Once selected, the corresponding lab values are displayed in the figure, which includes vertical lines to indicate dosing events (e.g., Cycle 1 Day 1 and Cycle 2 Day 1), along with reference ranges (UCL and LCL). Out-of-

range values are visually highlighted in the figure and also flagged in the detailed lab table below. Hover functionality is available only in the figure, providing interactive tooltips for additional context.

In addition to the AE, systemic events and lab examples described above, other subject-level data—such as concomitant medications, medical history, and reactogenicity—are available in separate tabs within the application. An AE summary table and additional supplemental information can also be included based on study-specific requirements, offering flexibility to support diverse clinical review needs.



.

CONCLUSION

R Shiny-based Interactive Patient Profile is a powerful tool for CROs and biopharmaceutical companies, transforming static SDTM data into dynamic dashboards. It enables medical monitors, data managers, and safety reviewers to quickly assess patient-level trends, adverse events, and protocol compliance—enhancing data quality and accelerating decision-making. For instance, during dose-level review meetings, clinical scientists and site vendors can easily detect safety signals using this application.

Fully compatible with CDISC-compliant studies, this application also supports medical writers and clinical reviewers by improving transparency, enabling consistent, high-quality patient profile generation, speeds up issue detection, and helps teams prepare for regulatory submissions more efficiently—all without requiring any coding expertise, thanks to intuitive controls built into the R Shiny sidebar panel.

CONTACT INFORMATION

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

Author Name: Tzu-Chien Chuang; Danny Hsu
Company: Pfizer Inc.
Address: 66 Hudson Blvd E, New York, NY 10001
Email: Tzu-Chien.Chuang@pfizer.com; danny.hsu@pfizer.com
Website: https://www.pfizer.com

APPENDIX 1: SAMPLE PROGRAM

```
library(shiny)
library(plotly)
library(ggplot2)

# --- UI: Layout and Inputs ---
ui <- fluidPage(
  titlePanel("Vaccine Patient Profile"),
  sidebarLayout(
    sidebarPanel(
      selectInput("treatment", "Select Treatment:", choices = unique(dm$ARM)),
      selectInput("subject", "Select Subject:", choices = NULL),
      uiOutput("subject_info")
    ),
    mainPanel(
      tabsetPanel(
        tabPanel("AE", plotlyOutput("ae_plot"), tableOutput("ae_table"), textOutput("no_records_msg_ae")),
        tabPanel("MH & CM", tableOutput("mh_table"), textOutput("no_records_msg_mh"), tableOutput("cm_table"),
textOutput("no_records_msg_cm")),
        tabPanel("Local Reaction", plotOutput("reactoPlot"), tableOutput("reacto_table"), textOutput("no_records_msg_reacto")),
        tabPanel("Systemic Event", plotOutput("sysPlot"), tableOutput("sys_table"), textOutput("no_records_msg_sys"))
      ))))

# --- Server: Logic and Outputs ---
server <- function(input, output, session) {
  # Update subject choices when treatment changes
  observeEvent(input$treatment, {
    updateSelectInput(session, "subject", choices = dm$SUBJID[dm$ARM == input$treatment])
  })

  # Show selected subject info
  output$subject_info <- renderUI({
    req(input$subject)
    subj <- dm[dm$SUBJID == input$subject, ]
    div(strong("Gender:"), subj$SEX, br(), strong("Race:"), subj$RACE)
  })

  # AE plot and table
  output$ae_plot <- renderPlotly({ /* ...plot code... */ })
  output$ae_table <- renderTable({ /* ...table code... */ })
  output$no_records_msg_ae <- renderText({ /* ...message code... */ })

  # MH & CM tables
  output$mh_table <- renderTable({ /* ...table code... */ })
  output$no_records_msg_mh <- renderText({ /* ...message code... */ })
  output$cm_table <- renderTable({ /* ...table code... */ })
  output$no_records_msg_cm <- renderText({ /* ...message code... */ })

  # Local Reaction plot and table
  output$reactoPlot <- renderPlot({ /* ...plot code... */ })
  output$reacto_table <- renderTable({ /* ...table code... */ })
  output$no_records_msg_reacto <- renderText({ /* ...message code... */ })

  # Systemic Event plot and table
  output$sysPlot <- renderPlot({ /* ...plot code... */ })
  output$sys_table <- renderTable({ /* ...table code... */ })
  output$no_records_msg_sys <- renderText({ /* ...message code... */ })
}
shinyApp(ui, server)
```