```
library(shiny)
library(shinyjs)
library(ggplot2)

ui <- fluidPage(
  useShinyjs(),
  titlePanel("Simulating Kelley's Paradox and Test Bias With Two Groups"),

  sidebarLayout(
    sidebarPanel(
      actionButton("plotButton", "Generate Plot"),

      # Checkbox inputs to toggle display elements
      checkboxInput("showPoints", "Show Points", TRUE),
      checkboxInput("showLines", "Show Regression Lines", TRUE),
      checkboxInput("showEllipses", "Show Confidence Ellipses", FALSE),

      # Optional Seed Input
      numericInput("seed", "Seed for Random Number Generation (optional):", value = NA, min = NA, step
= 1),

      # Slider inputs (default)
      div(id = "sliderInputs",
        sliderInput("slope_A_slider", "Slope for Group A:", min = -1, max = 1, value = 1, step = 0.05),
        sliderInput("intercept_A_slider", "Intercept for Group A:", min = -5, max = 5, value = 0, step = 0.05),
        sliderInput("mean_A_slider", "Mean True Score for Group A:", min = -5, max = 5, value = 0.1, step =
0.05),
        sliderInput("sd_A_slider", "True Score SD for Group A:", min = 0.1, max = 3, value = 1),
        sliderInput("obs_sd_A_slider", "Observed Score SD for Group A:", min = 0.1, max = 3, value = 1),
        sliderInput("n_A_slider", "Number of Observations for Group A:", min = 0, max = 1000, value = 1000)
,

        sliderInput("slope_B_slider", "Slope for Group B:", min = -1, max = 1, value = 1, step = 0.05),
        sliderInput("intercept_B_slider", "Intercept for Group B:", min = -5, max = 5, value = 0, step = 0.05),
        sliderInput("mean_B_slider", "Mean True Score for Group B:", min = -5, max = 5, value = -1, step = 0
.05),
        sliderInput("sd_B_slider", "True Score SD for Group B:", min = 0.1, max = 3, value = 1),
        sliderInput("obs_sd_B_slider", "Observed Score SD for Group B:", min = 0.1, max = 3, value = 1),
        sliderInput("n_B_slider", "Number of Observations for Group B:", min = 0, max = 1000, value = 1000)

      ),

      # Numeric inputs (hidden by default)
      div(id = "numericInputs", style = "display: none;",
        numericInput("slope_A", "Slope for Group A:", 1),
        numericInput("intercept_A", "Intercept for Group A:", 0),
        numericInput("mean_A", "Mean True Score for Group A:", 0.1),
        numericInput("sd_A", "True Score SD for Group A:", 1),
        numericInput("obs_sd_A", "Observed Score SD for Group A:", 1),
        numericInput("n_A", "Number of Observations for Group A:", 1000),

        numericInput("slope_B", "Slope for Group B:", 1),
        numericInput("intercept_B", "Intercept for Group B:", 0),
        numericInput("mean_B", "Mean True Score for Group B:", -1),
        numericInput("sd_B", "True Score SD for Group B:", 1),
```

```
      numericInput("obs_sd_B", "Observed Score SD for Group B:", 1),
      numericInput("n_B", "Number of Observations for Group B:", 1000)
    ),

    actionButton("toggleButton", "Swap True/Observed Scores"),
    actionButton("inputMode", "Switch to Numeric Input", icon = icon("exchange-alt")),

    # Preset configurations dropdown
    selectInput("presets", "Load Preset Configurations:",
                choices = c("Select a Preset", "Preset 1", "Preset 2", "Preset 3", "Preset 4", "Preset 5"))
  ),

  mainPanel(
    plotOutput("plot")
  )
 )
)

server <- function(input, output, session) {
  inputMode <- reactiveVal(FALSE)
  toggleState <- reactiveVal(FALSE)

  observeEvent(input$inputMode, {
    inputMode(!inputMode())
    if (inputMode()) {
      shinyjs::hide("sliderInputs")
      shinyjs::show("numericInputs")
      updateActionButton(session, "inputMode", label = "Switch to Slider Input", icon = icon("exchange-
alt"))
    } else {
      shinyjs::show("sliderInputs")
      shinyjs::hide("numericInputs")
      updateActionButton(session, "inputMode", label = "Switch to Numeric Input", icon = icon("exchang
e-alt"))
    }
  })

  observeEvent(input$toggleButton, {
    toggleState(!toggleState())
  })

  # Define preset configurations
  presets <- list(
    "Preset 1" = list(slope_A = 1, intercept_A = 0, mean_A = 0, sd_A = 1, obs_sd_A = 1, n_A = 1000,
                      slope_B = 1, intercept_B = 0, mean_B = 0, sd_B = 1, obs_sd_B = 1, n_B = 1000),
    "Preset 2" = list(slope_A = 1, intercept_A = 0, mean_A = 1, sd_A = 1, obs_sd_A = 1, n_A = 1000,
                      slope_B = 1, intercept_B = 0, mean_B = 0, sd_B = 1, obs_sd_B = 1, n_B = 1000),
    "Preset 3" = list(slope_A = 1, intercept_A = 1, mean_A = 0, sd_A = 1, obs_sd_A = 1, n_A = 1000,
                      slope_B = 1, intercept_B = 0, mean_B = 0, sd_B = 1, obs_sd_B = 1, n_B = 1000),
    "Preset 4" = list(slope_A = 1, intercept_A = 1, mean_A = 0, sd_A = 1, obs_sd_A = 1, n_A = 1000,
                      slope_B = 0.6, intercept_B = 0, mean_B = 0, sd_B = 1, obs_sd_B = 1, n_B = 1000),
    "Preset 5" = list(slope_A = 1, intercept_A = 0, mean_A = 1, sd_A = 1, obs_sd_A = 1, n_A = 1000,
                      slope_B = 1, intercept_B = 1, mean_B = 0, sd_B = 1, obs_sd_B = 1, n_B = 1000)
  )
```

```
observeEvent(input$presets, {
    if (input$presets != "Select a Preset") {
        vals <- presets[[input$presets]]
        updateSliderInput(session, "slope_A_slider", value = vals$slope_A)
        updateSliderInput(session, "intercept_A_slider", value = vals$intercept_A)
        updateSliderInput(session, "mean_A_slider", value = vals$mean_A)
        updateSliderInput(session, "sd_A_slider", value = vals$sd_A)
        updateSliderInput(session, "obs_sd_A_slider", value = vals$obs_sd_A)
        updateSliderInput(session, "n_A_slider", value = vals$n_A)
        updateSliderInput(session, "slope_B_slider", value = vals$slope_B)
        updateSliderInput(session, "intercept_B_slider", value = vals$intercept_B)
        updateSliderInput(session, "mean_B_slider", value = vals$mean_B)
        updateSliderInput(session, "sd_B_slider", value = vals$sd_B)
        updateSliderInput(session, "obs_sd_B_slider", value = vals$obs_sd_B)
        updateSliderInput(session, "n_B_slider", value = vals$n_B)
    }
})

generate_data <- eventReactive(input$plotButton, {
    if (!is.na(input$seed) && input$seed != 0) {
        set.seed(input$seed)
    } else {
        set.seed(NULL)
    }

    if (inputMode()) {
        slope_A <- input$slope_A
        intercept_A <- input$intercept_A
        mean_A <- input$mean_A
        sd_A <- input$sd_A
        obs_sd_A <- input$obs_sd_A
        n_A <- input$n_A

        slope_B <- input$slope_B
        intercept_B <- input$intercept_B
        mean_B <- input$mean_B
        sd_B <- input$sd_B
        obs_sd_B <- input$obs_sd_B
        n_B <- input$n_B
    } else {
        slope_A <- input$slope_A_slider
        intercept_A <- input$intercept_A_slider
        mean_A <- input$mean_A_slider
        sd_A <- input$sd_A_slider
        obs_sd_A <- input$obs_sd_A_slider
        n_A <- input$n_A_slider

        slope_B <- input$slope_B_slider
        intercept_B <- input$intercept_B_slider
        mean_B <- input$mean_B_slider
        sd_B <- input$sd_B_slider
        obs_sd_B <- input$obs_sd_B_slider
        n_B <- input$n_B_slider
    }
```

```
    X_A <- rnorm(n_A, mean = mean_A, sd = sd_A)
    X_B <- rnorm(n_B, mean = mean_B, sd = sd_B)
    Y_A <- intercept_A + slope_A * X_A + rnorm(n_A, sd = obs_sd_A)
    Y_B <- intercept_B + slope_B * X_B + rnorm(n_B, sd = obs_sd_B)

    data <- data.frame(Group = rep(c("Group A", "Group B"), times = c(n_A, n_B)),
                       TrueScore = c(X_A, X_B),
                       ObservedScore = c(Y_A, Y_B))
    return(data)
  })

  output$plot <- renderPlot({
    data <- generate_data()
    if (toggleState()) {
      x_var <- "ObservedScore"
      y_var <- "TrueScore"
    } else {
      x_var <- "TrueScore"
      y_var <- "ObservedScore"
    }

    p <- ggplot(data, aes_string(x = x_var, y = y_var, color = "Group", fill = "Group"))

    if (input$showPoints) {
      p <- p + geom_point(alpha = 0.6, size = 1.5, alpha = 0.8)
    }
    if (input$showLines) {
      p <- p + geom_smooth(method = "lm", se = FALSE, size = 1, alpha = 0.8)
    }
    if (input$showEllipses) {
      p <- p + stat_ellipse(level = 0.95, linetype = "dashed", linewidth = 1.2, size = 0.5, alpha = 0.8)
    }

    p + scale_color_brewer(palette = "Set1") +
      theme_minimal() +
      theme(
        legend.position = "bottom",
        plot.title = element_text(hjust = 0.5, size = 16, face = "bold"),
        axis.title.x = element_text(size = 14, face = "bold"),
        axis.title.y = element_text(size = 14, face = "bold"),
        axis.text.x = element_text(size = 12),
        axis.text.y = element_text(size = 12),
        legend.title = element_text(size = 14),
        legend.text = element_text(size = 12),
        legend.key.size = unit(1.5, "lines"),
        plot.caption = element_text(size = 12, hjust = 0.5)
      ) +
      labs(title = "True Scores and Observed Scores for Different Groups",
           x = ifelse(toggleState(), "Observed Score", "True Score"),
           y = ifelse(toggleState(), "True Score", "Observed Score"),
           color = NULL,
           fill = NULL,
           caption = "An Applet by Crémieux Recueil (@cremieuxrecueil) for Aporia Magazine")
  })
}
```

```
shinyApp(ui = ui, server = server)
```