```
Data: Data frame input by user
Result: Interactive volcano plot
/* Declare Shiny server
server ← function(input, output, session){
    /* User input options
    observeEvent(input$goButton, values$x ← values$x + 1)
    observeEvent(input$selPair, values$x ← 0)
    observeEvent(input$selMetric, values$x ← 0)
    observeEvent(input$selOrder, values$x ← 0)
    observeEvent(input$binSize, values$x ← 0)
    observeEvent(input$selPair, values$selPair ← input$selPair)
    /* Define largest fold change dynamically based on data
    fcInMax ← max(ldply(dataMetrics, rbind)[["logFC"]])
    /* Construct dynamic input Shiny slider for fold change
    output$slider ← renderUI(sliderInput("logFC", "Log fold change:",
      min=0, max=fcInMax, step=0.1))
    /* Declare shiny output volcano plot
    output$volPlot ← renderPlotly({
        /* Create reactive expression of plotly background volcano plot
        gP ← reactive(p ← ggplot(data); gP ← ggplotly(p))
        /* Create reactive expression of plotly background volcano plot
        plotlyVol ← reactive(gP())
        /* Tailor interactivity of the plotly volcano plot object using custom
          JavaScript
        plotlyVol() %>% onRender("function(el, x, data){
            /* Read handle called 'points' to obtain variables sent from R into
              JavaScript
            Shiny.addCustomMessageHandler('points', function(drawPoints){
                /* Delete any old superimposed plotly geoms (dots)
                if (x.data.length > 0){Plotly.deleteTraces(el.id)}
                /* Create traces for selected gene IDs as points that state gene
                  names upon hovering
                trace = {x: drawPoints.geneX, y: drawPoints.geneY, mode:
                  'markers', color: drawPoints.pointColor, size:
                  drawPoints.pointSize, text: drawPoints.geneID, hoverinfo:
                  'text'}
                /* Superimpose traces onto the plotly litre plot object
                Plotly.addTraces(el.id, trace)
            })
        }")
    })
    /* If the user changes the superimposed gene
    observe({
        /* Save information about superimposed gene selected by user with a
          handle called 'points'. These values can then be sent from R to
          JavaScript
```