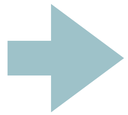
/shiny-modules

The solution to scalability

- Modules are *functions* they help you reuse code;
  anything you can do with a function, you can do with a module
- *Namespacing makes it easier*

```r
your_UI <- function(id, title, ...) {
    ns <- NS(id)
    fluidPage(
        h4(title),

        # shiny UI code here
        # ...

    )
}
```

```r
your_server <- function(id, dataset_location, ...) {
    ns <- NS(id)
    moduleServer(
        id,
        function(input, output, session) {
            data <- shiny::reactive({
                arrow::open_dataset(dataset_location)
            })

            output$pickerUI <- shiny::renderUI({
                pickerInput(
                    inputId = ns("selected_grps"),
                    choices = unique(data()[["group"]])
                )
            })

            output$plot_ts <- shiny::renderPlot({
                data() |>
                    filter(group %in% input$selected_grps) |>
                    make_a_plot()
            })
        }
    )
}
```



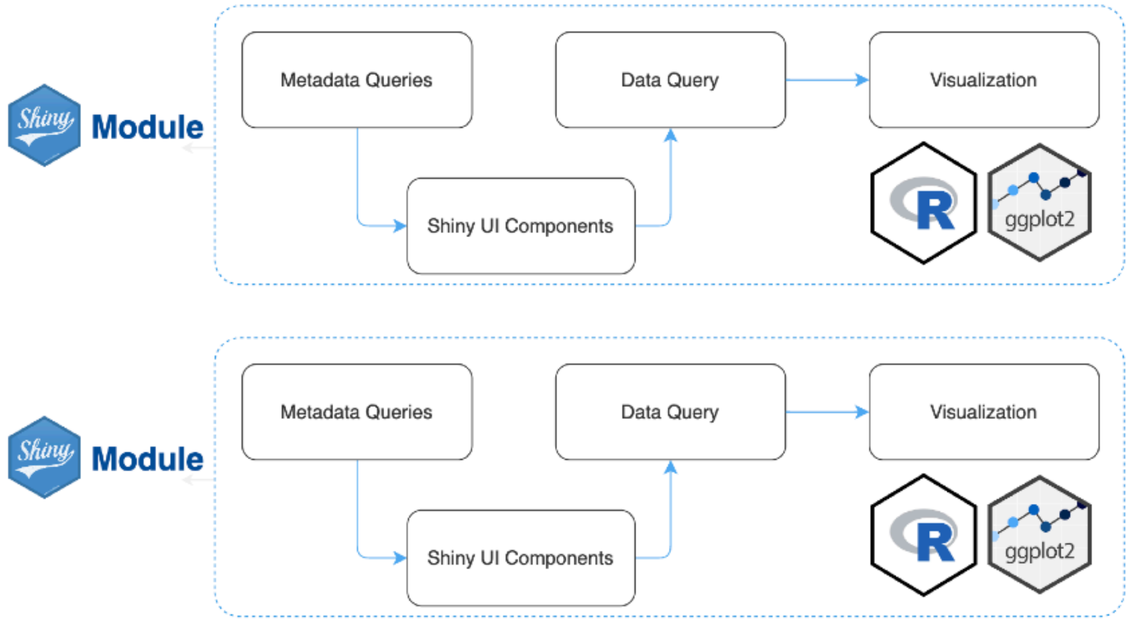
```r
# app.R
ui <- fluidPage(
    your_UI("tab1"),
    your_UI("tab2")
)

# Server ----
server <- function(input, output, session) {
    your_server("tab1")
    your_server("tab2")
}

shinyApp(ui, server)
```

/shiny-modules

The solution to scalability

- Modules are *functions* they help you reuse code;
  anything you can do with a function, you can do with a module

- *Namespacing makes it easier*




```r
# app.R
ui <- fluidPage(
    your_UI("tab1"),
    your_UI("tab2")
)

# Server ----
server <- function(input, output, session) {
    your_server("tab1")
    your_server("tab2")
}

shinyApp(ui, server)
```