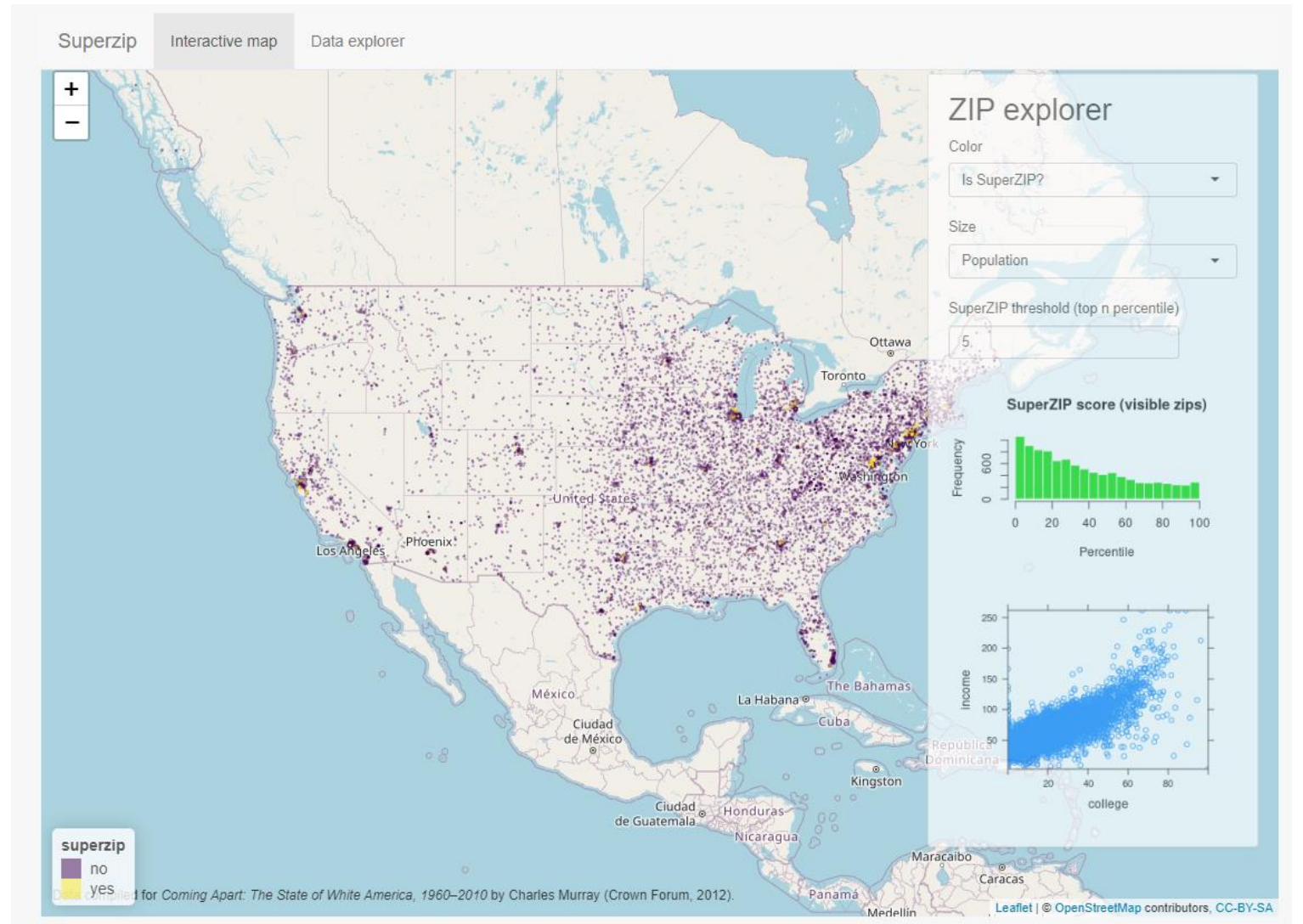# Bigfoot sightings: Learning to create Shiny modules by turning an existing app modular

- Deepsha Menghani

Shiny is a framework for building web applications using R and Python.

Deepsha Menghani
https://deepshamenghani.quarto.pub/dmenghani/

Learning Shiny…. A simple app.R file


```r
# Define UI for application that draws a histogram
ui <- fluidPage(

    # Application title
    titlePanel("Old Faithful Geyser Data"),

    # Sidebar with a slider input for number of bins
    sidebarLayout(
        sidebarPanel(
            sliderInput("bins",
                        "Number of bins:",
                        min = 1,
                        max = 50,
                        value = 30)
        ),

        # Show a plot of the generated distribution
        mainPanel(
            plotOutput("distPlot")
        )
    )
)

# Define server logic required to draw a histogram
server <- function(input, output) {

    output$distPlot <- renderPlot({
        # generate bins based on input$bins from ui.R
        x    <- faithful[, 2]
        bins <- seq(min(x), max(x), length.out = input$bins + 1)

        # draw the histogram with the specified number of bins
        hist(x, breaks = bins, col = 'darkgray', border = 'white',
             xlab = 'Waiting time to next eruption (in mins)',
             main = 'Histogram of waiting times')
    })
}

# Run the application
shinyApp(ui = ui, server = server)
```
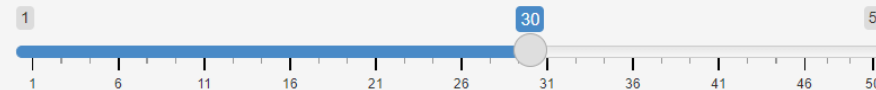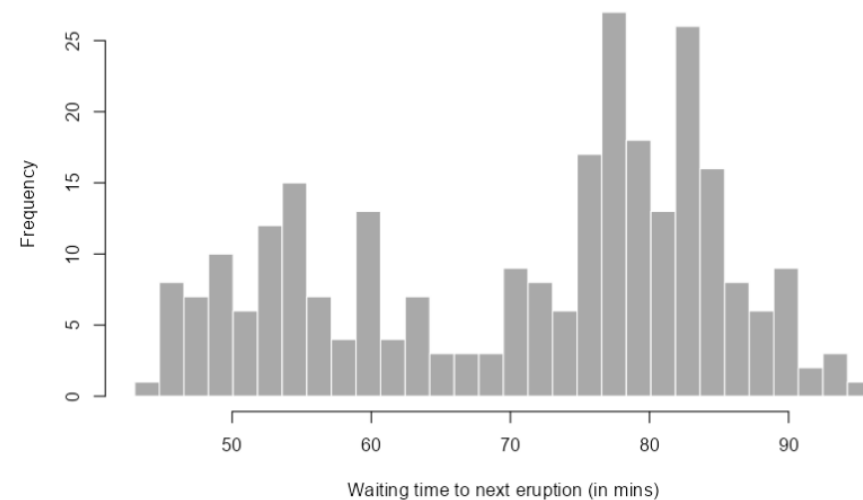

Deepsha Menghani
https://deepshamenghani.quarto.pub/dmenghani/

Building Shiny…. A much larger app.R file

Deepsha Menghani
https://deepshamenghani.quarto.pub/dmenghani/

The challenges with a monolith app.R file

- Difficult to keep track of components

- Challenging to debug if something fails

- Harder for someone new to learn and contribute

- Tedious to reuse its elements within itself and other applications

Enter Shiny modules... A reusable piece of Shiny code written as a function

Functions

♥

Shiny

Deepsha Menghani
https://deepshamenghani.quarto.pub/dmenghani/

Superpower of Shiny and R functions combined

Mastering Shiny book by Hadley Wickham - https://mastering-shiny.org/

Let's learn to write modules with a Shiny application

Step 1 : Decompose your application components

Bigfoot Sightings in the United States

Deepsha Menghani
https://deepshamenghani.quarto.pub/dmenghani/

Step 1 : Decompose your application components

Bigfoot Sightings in the United States

Select State

Washington ▼

Sightings for top 10 counties

County	Sightings
Pierce County	78
Snohomish County	51
Skamania County	49
Lewis County	47
King County	41
Grays Harbor County	37
Okanogan County	25
Chelan County	24
Kittitas County	23
Mason County	22

Sightings over time

Highest yearly sighting: 32
Year: 2004

1993   1999   2005   2011   2017   2023
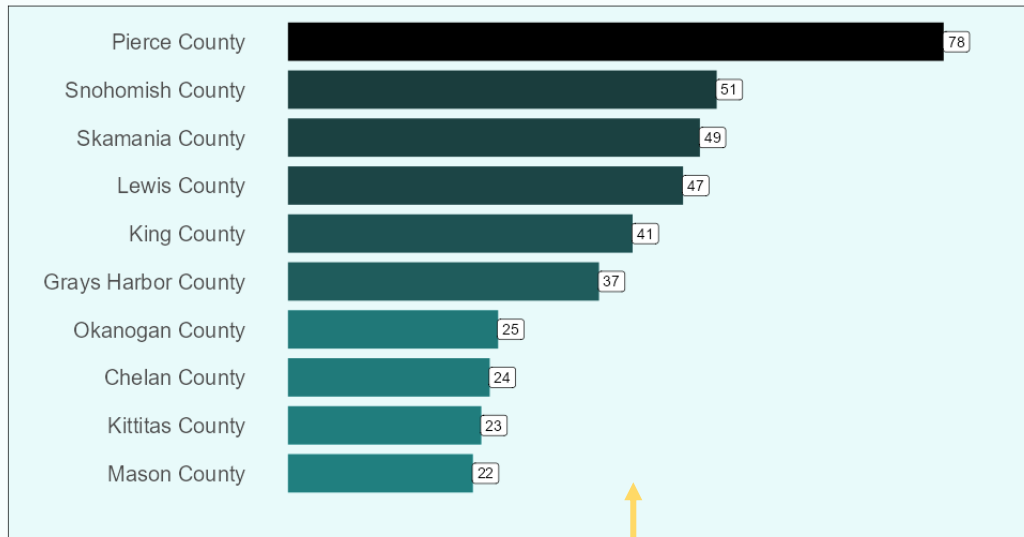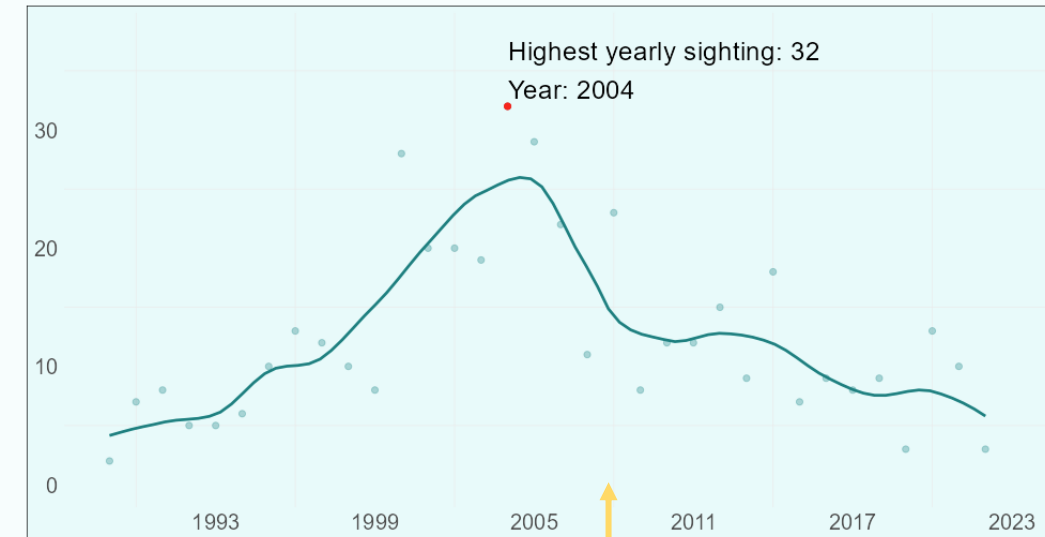
Input for
filtering data

County plot
(on filtered data)

Yearly plot
(on filtered data)

Deepsha Menghani
https://deepshamenghani.quarto.pub/dmenghani/

Main app.R file

```r
dataset <- read.csv("./data/bfro_reports_geocoded.csv")

ui <- fluidPage(h1("Bigfoot Sightings in the United States", align="center"),
                fluidRow(
                    # Input state
                    column(2,selectizeInput(inputId = "state",
                                            label = "Select State",
                                            choices = unique(dataset$state),
                                            selected = "Washington",
                                            multiple = FALSE)),

                    # Show county plot
                    column(5, plotOutput(outputId = "plotcounty")),

                    # Show yearly plot
                    column(5, plotOutput(outputId = "plotyearly"))
                ))

# Define server logic ----
server <- function(input, output, session) {

  # filter data based on user selection
  data_filtered <- reactive({dataset |> filter(state == input$state)})

  # County plot
  output$plotcounty <- renderPlot(
    data_filtered() |>
      count(county) |>
      ggplot() +
      geom_col(aes(county, n, fill = n), colour = NA, width = 0.8)
      # ... rest of the plot code
  )

  # Yearly plot
  output$plotyearly <- renderPlot(
    data_filtered() |>
      count(year) |>
      ggplot(aes(year, n)) +
      geom_point(aes(year, highest_count), alpha=1, size = 2)
      # ... rest of the plot code
  )
}

# Run the application
shinyApp(ui = ui, server = server)
```

3 ui elements ⟵

3 server elements ⟵

Main app.R file

```r
dataset <- read.csv("./data/bfro_reports_geocoded.csv")

ui <- fluidPage(h1("Bigfoot Sightings in the United States", align="center"),
                fluidRow(
                        # Input state
                        column(2,selectizeInput(inputId = "state",
                                                label = "Select State",
                                                choices = unique(dataset$state),
                                                selected = "Washington",
                                                multiple = FALSE)),

                        # Show county plot
                        column(5, plotOutput(outputId = "plotcounty")),

                        # Show yearly plot
                        column(5, plotOutput(outputId = "plotyearly"))
                ))

# Define server logic ----
server <- function(input, output, session) {

    # filter data based on user selection
    data_filtered <- reactive({dataset |> filter(state == input$state)})

    # County plot
    output$plotcounty <- renderPlot(
      data_filtered() |>
        count(county) |>
        ggplot() +
        geom_col(aes(county, n, fill = n), colour = NA, width = 0.8)
        # ... rest of the plot code
    )

    # Yearly plot
    output$plotyearly <- renderPlot(
      data_filtered() |>
        count(year) |>
        ggplot(aes(year, n)) +
        geom_point(aes(year, highest_count), alpha=1, size = 2)
        # ... rest of the plot code
    )
}

# Run the application
shinyApp(ui = ui, server = server)
```

3 ui elements

3 server elements

Main app.R file

```r
dataset <- read.csv("./data/bfro_reports_geocoded.csv")

ui <- fluidPage(h1("Bigfoot Sightings in the United States", align="center"),
                fluidRow(
                    # Input state
                    column(2,selectizeInput(inputId = "state",
                                            label = "Select State",
                                            choices = unique(dataset$state),
                                            selected = "Washington",
                                            multiple = FALSE)),

                    # Show county plot
                    column(5, plotOutput(outputId = "plotcounty")),

                    # Show yearly plot
                    column(5, plotOutput(outputId = "plotyearly"))
                ))

# Define server logic ----
server <- function(input, output, session) {

  # filter data based on user selection
  data_filtered <- reactive({dataset |> filter(state == input$state)})

  # County plot
  output$plotcounty <- renderPlot(
    data_filtered() |>
      count(county) |>
      ggplot() +
      geom_col(aes(county, n, fill = n), colour = NA, width = 0.8)
      # ... rest of the plot code
  )

  # Yearly plot
  output$plotyearly <- renderPlot(
    data_filtered() |>
      count(year) |>
      ggplot(aes(year, n)) +
      geom_point(aes(year, highest_count), alpha=1, size = 2)
      # ... rest of the plot code
  )
}

# Run the application
shinyApp(ui = ui, server = server)
```
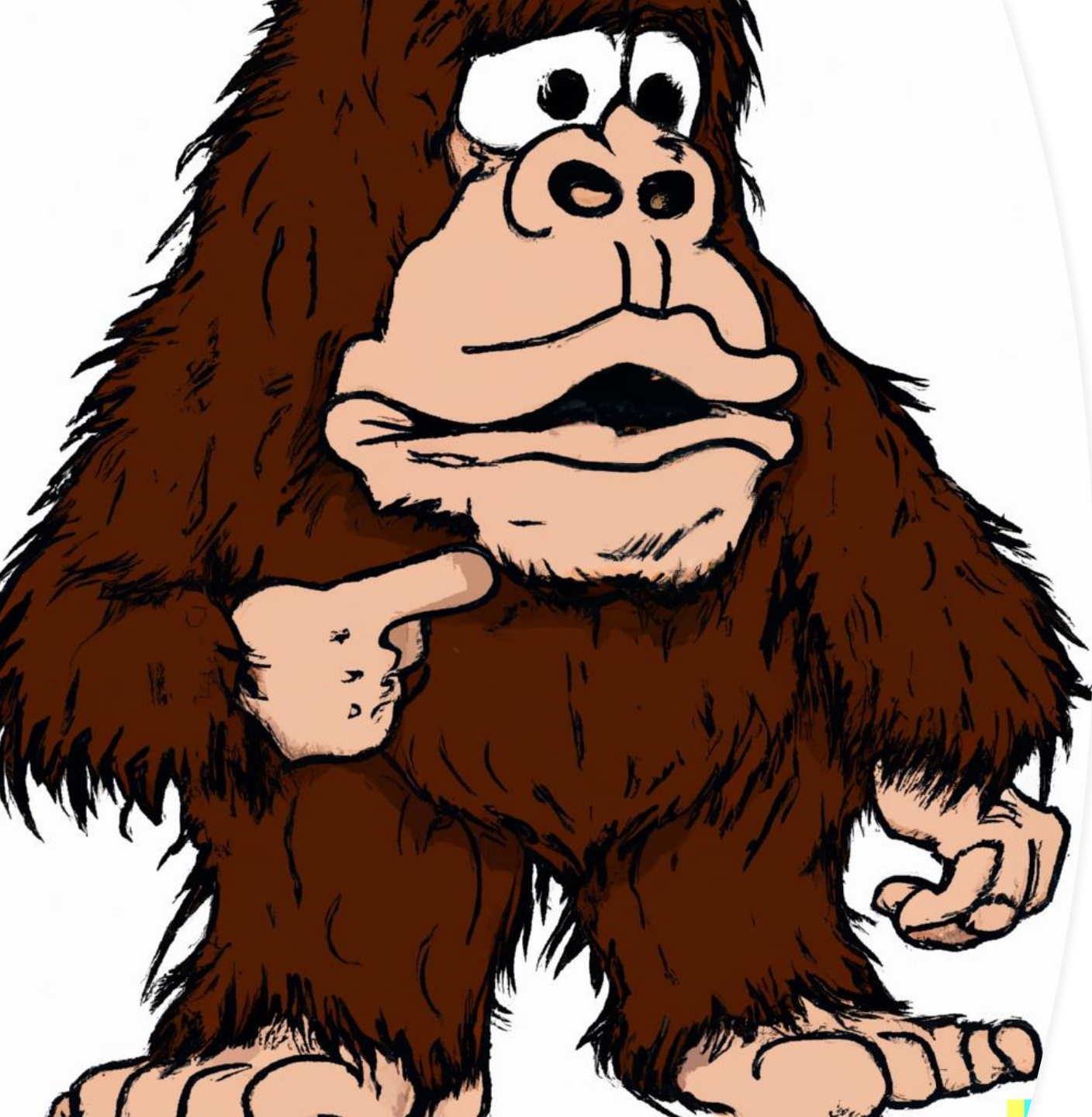
3 ui elements

3 server elements

Main app.R file

```r
dataset <- read.csv("./data/bfro_reports_geocoded.csv")

ui <- fluidPage(h1("Bigfoot Sightings in the United States", align="center"),
                fluidRow(
                    # Input state
                    column(2,selectizeInput(inputId = "state",
                                            label = "Select State",
                                            choices = unique(dataset$state),
                                            selected = "Washington",
                                            multiple = FALSE)),

                    # Show county plot
                    column(5, plotOutput(outputId = "plotcounty")),      # <-- 3 ui elements

                    # Show yearly plot
                    column(5, plotOutput(outputId = "plotyearly"))
                )
)

# Define server logic ----
server <- function(input, output, session) {

  # filter data based on user selection
  data_filtered <- reactive({dataset |> filter(state == input$state)})

  # County plot
  output$plotcounty <- renderPlot(                                      # <-- 3 server elements
    data_filtered() |>
      count(county) |>
      ggplot() +
      geom_col(aes(county, n, fill = n), colour = NA, width = 0.8)
      # ... rest of the plot code
  )

  # Yearly plot
  output$plotyearly <- renderPlot(
    data_filtered() |>
      count(year) |>
      ggplot(aes(year, n)) +
      geom_point(aes(year, highest_count), alpha=1, size = 2)
      # ... rest of the plot code
  )
}

# Run the application
shinyApp(ui = ui, server = server)
```

3 ui elements

3 server elements

Step 2 : Create modules (baby shiny apps) for the distributed components

- Module 1: Module_input.R
  - Ui: Create state input ui
  - Server: Filter data based on state input

- Module 2: Module_countyplot.R
  - Ui: Create county plot ui
  - Server: Generate the county plot with ggplot code

- Module 3: Module_yearlyplot.R
  - Ui: Create yearly plot ui
  - Server: Generate the yearly plot with ggplot code

```r
dataset <- read.csv("./data/bfro_reports_geocoded.csv")

ui <- fluidPage(h1("Bigfoot Sightings in the United States", align="center"),
                fluidRow(
                    # Input state
                    column(2,selectizeInput(inputId = "state",
                                            label = "Select State",
                                            choices = unique(dataset$state),
                                            selected = "Washington",
                                            multiple = FALSE)),

                    # Show county plot
                    column(5, plotOutput(outputId = "plotcounty")),

                    # Show yearly plot
                    column(5, plotOutput(outputId = "plotyearly"))
                ))
# Define server logic ----
server <- function(input, output, session) {

  # filter data based on user selection
  data_filtered <- reactive({dataset |> filter(state == input$state)})

  # County plot
  output$plotcounty <- renderPlot(
    data_filtered() |>
      count(county) |>
      ggplot() +
      geom_col(aes(county, n, fill = n), colour = NA, width = 0.8)
      # ... rest of the plot code
  )

  # Yearly plot
  output$plotyearly <- renderPlot(
    data_filtered() |>
      count(year) |>
      ggplot(aes(year, n)) +
      geom_point(aes(year, highest_count), alpha=1, size = 2)
      # ... rest of the plot code
  )
}

# Run the application
shinyApp(ui = ui, server = server)
```

```r
module_input_ui <- function(id, df, defaultstate = "Washington") {
  # Namespace
  ns <- NS(id)
  # Input UI command
  selectizeInput(inputId = ns("stateinput"), label = "Select state",
                 choices = unique(df$state),
                 selected = defaultstate, multiple = FALSE)
}


module_input_server <- function(id, df) {
  moduleServer(id,
               function(input, output, session) {
                 # Filter data set based on input
                 table <- reactive({df |> filter(state == input$stateinput)})
                 return(table)
               }
  )
}
```

Deepsha Menghani
https://deepshamenghani.quarto.pub/dmenghani/

```r
dataset <- read.csv("./data/bfro_reports_geocoded.csv")

ui <- fluidPage(h1("Bigfoot Sightings in the United States", align="center"),
                fluidRow(
                    # Input state
                    column(2,selectizeInput(inputId = "state",
                                            label = "Select State",
                                            choices = unique(dataset$state),
                                            selected = "Washington",
                                            multiple = FALSE)),

                    # Show county plot
                    column(5, plotOutput(outputId = "plotcounty")),

                    # Show yearly plot
                    column(5, plotOutput(outputId = "plotyearly"))
                ))
# Define server logic ----
server <- function(input, output, session) {

  # filter data based on user selection
  data_filtered <- reactive({dataset |> filter(state == input$state)})

  # County plot
  output$plotcounty <- renderPlot(
    data_filtered() |>
      count(county) |>
      ggplot() +
      geom_col(aes(county, n, fill = n), colour = NA, width = 0.8)
      # ... rest of the plot code
  )

  # Yearly plot
  output$plotyearly <- renderPlot(
    data_filtered() |>
      count(year) |>
      ggplot(aes(year, n)) +
      geom_point(aes(year, highest_count), alpha=1, size = 2)
      # ... rest of the plot code
  )
}

# Run the application
shinyApp(ui = ui, server = server)
```

ui and server components are written as two separate modules, just like a shiny app.

```r
module_input_ui <- function(id, df, defaultstate = "Washington") {
  # Namespace
  ns <- NS(id)
  # Input UI command
  selectizeInput(inputId = ns("stateinput"), label = "Select state",
                 choices = unique(df$state),
                 selected = defaultstate, multiple = FALSE)
}
```

```r
module_input_server <- function(id, df) {
  moduleServer(id,
               function(input, output, session) {
                 # Filter data set based on input
                 table <- reactive({df |> filter(state == input$stateinput)})
                 return(table)
               }
  )
}
```

Original app.R file

```r
dataset <- read.csv("./data/bfro_reports_geocoded.csv")

ui <- fluidPage(h1("Bigfoot Sightings in the United States", align="center"),
                fluidRow(
                    # Input state
                    column(2,selectizeInput(inputId = "state",
                                            label = "Select State",
                                            choices = unique(dataset$state),
                                            selected = "Washington",
                                            multiple = FALSE)),

                    # Show county plot
                    column(5, plotOutput(outputId = "plotcounty")),

                    # Show yearly plot
                    column(5, plotOutput(outputId = "plotyearly"))
                ))

# Define server logic ----
server <- function(input, output, session) {

  # filter data based on user selection
  data_filtered <- reactive({dataset |> filter(state == input$state)})

  # County plot
  output$plotcounty <- renderPlot(
    data_filtered() |>
      count(county) |>
      ggplot() +
      geom_col(aes(county, n, fill = n), colour = NA, width = 0.8)
      # ... rest of the plot code
  )

  # Yearly plot
  output$plotyearly <- renderPlot(
    data_filtered() |>
      count(year) |>
      ggplot(aes(year, n)) +
      geom_point(aes(year, highest_count), alpha=1, size = 2)
      # ... rest of the plot code
  )
}

# Run the application
shinyApp(ui = ui, server = server)
```

Module 1: Module_input.R

In this module we want to ask user to input "state" and then filter data based on the user input

```r
module_input_ui <- function(id, df, defaultstate = "Washington") {
  # Namespace
  ns <- NS(id)
  # Input UI command
  selectizeInput(inputId = ns("stateinput"), label = "Select state",
                 choices = unique(df$state),
                 selected = defaultstate, multiple = FALSE)
}


module_input_server <- function(id, df) {
  moduleServer(id,
               function(input, output, session) {
                 # Filter data set based on input
                 table <- reactive({df |> filter(state == input$stateinput)})
                 return(table)
               }
  )
}
```

Original app.R file

```r
dataset <- read.csv("./data/bfro_reports_geocoded.csv")

ui <- fluidPage(h1("Bigfoot Sightings in the United States", align="center"),
            fluidRow(
                # Input state
                column(2,selectizeInput(inputId = "state",
                                        label = "Select State",
                                        choices = unique(dataset$state),
                                        selected = "Washington",
                                        multiple = FALSE)),

                # Show county plot
                column(5, plotOutput(outputId = "plotcounty")),

                # Show yearly plot
                column(5, plotOutput(outputId = "plotyearly"))
            ))

# Define server logic ----
server <- function(input, output, session) {

    # filter data based on user selection
    data_filtered <- reactive({dataset |> filter(state == input$state)})

    # County plot
    output$plotcounty <- renderPlot(
        data_filtered() |>
            count(county) |>
            ggplot() +
            geom_col(aes(county, n, fill = n), colour = NA, width = 0.8)
            # ... rest of the plot code
    )

    # Yearly plot
    output$plotyearly <- renderPlot(
        data_filtered() |>
            count(year) |>
            ggplot(aes(year, n)) +
            geom_point(aes(year, highest_count), alpha=1, size = 2)
            # ... rest of the plot code
    )
}

# Run the application
shinyApp(ui = ui, server = server)
```

Module 1: Module_input.R

We create the ui and server as functions that can be called from the main app.R file

```r
module_input_ui <- function(id, df, defaultstate = "Washington") {
    # Namespace
    ns <- NS(id)
    # Input UI command
    selectizeInput(inputId = ns("stateinput"), label = "Select state",
                   choices = unique(df$state),
                   selected = defaultstate, multiple = FALSE)
}


module_input_server <- function(id, df) {
    moduleServer(id,
                 function(input, output, session) {
                     # Filter data set based on input
                     table <- reactive({df |> filter(state == input$stateinput)})
                     return(table)
                 }
    )
}
```

Original app.R file

```r
dataset <- read.csv("./data/bfro_reports_geocoded.csv")

ui <- fluidPage(h1("Bigfoot Sightings in the United States", align="center"),
                fluidRow(
                    # Input state
                    column(2,selectizeInput(inputId = "state",
                                            label = "Select State",
                                            choices = unique(dataset$state),
                                            selected = "Washington",
                                            multiple = FALSE)),

                    # Show county plot
                    column(5, plotOutput(outputId = "plotcounty")),

                    # Show yearly plot
                    column(5, plotOutput(outputId = "plotyearly"))
                ))
# Define server logic ----
server <- function(input, output, session) {

  # filter data based on user selection
  data_filtered <- reactive({dataset |> filter(state == input$state)})

  # County plot
  output$plotcounty <- renderPlot(
    data_filtered() |>
      count(county) |>
      ggplot() +
      geom_col(aes(county, n, fill = n), colour = NA, width = 0.8)
      # ... rest of the plot code
  )

  # Yearly plot
  output$plotyearly <- renderPlot(
    data_filtered() |>
      count(year) |>
      ggplot(aes(year, n)) +
      geom_point(aes(year, highest_count), alpha=1, size = 2)
      # ... rest of the plot code
  )
}

# Run the application
shinyApp(ui = ui, server = server)
```

Module 1: Module_input.R

ID serves as a unique identifier for each instance of the module within the application, allowing multiple instances of the same module to coexist and be independently controlled.

```r
module_input_ui <- function(id, df, defaultstate = "Washington") {
  # Namespace
  ns <- NS(id)
  # Input UI command
  selectizeInput(inputId = ns("stateinput"), label = "Select state",
                 choices = unique(df$state),
                 selected = defaultstate, multiple = FALSE)
}


module_input_server <- function(id, df) {
  moduleServer(id,
               function(input, output, session) {
                 # Filter data set based on input
                 table <- reactive({df |> filter(state == input$stateinput)})
                 return(table)
               }
  )
}
```

Deepsha Menghani
https://deepshamenghani.quarto.pub/dmenghani/

Original app.R file

```r
dataset <- read.csv("./data/bfro_reports_geocoded.csv")

ui <- fluidPage(h1("Bigfoot Sightings in the United States", align="center"),
                fluidRow(
                    # Input state
                    column(2,selectizeInput(inputId = "state",
                                            label = "Select State",
                                            choices = unique(dataset$state),
                                            selected = "Washington",
                                            multiple = FALSE)),

                    # Show county plot
                    column(5, plotOutput(outputId = "plotcounty")),

                    # Show yearly plot
                    column(5, plotOutput(outputId = "plotyearly"))
                ))
# Define server logic ----
server <- function(input, output, session) {

  # filter data based on user selection
  data_filtered <- reactive({dataset |> filter(state == input$state)})

  # County plot
  output$plotcounty <- renderPlot(
    data_filtered() |>
      count(county) |>
      ggplot() +
      geom_col(aes(county, n, fill = n), colour = NA, width = 0.8)
      # ... rest of the plot code
  )

  # Yearly plot
  output$plotyearly <- renderPlot(
    data_filtered() |>
      count(year) |>
      ggplot(aes(year, n)) +
      geom_point(aes(year, highest_count), alpha=1, size = 2)
      # ... rest of the plot code
  )
}

# Run the application
shinyApp(ui = ui, server = server)
```

Module 1: Module_input.R

ID serves as a unique identifier for each instance of the module within the application, allowing multiple instances of the same module to coexist and be independently controlled.

```r
module_input_ui <- function(id, df, defaultstate = "Washington") {
  # Namespace
  ns <- NS(id)
  # Input UI command
  selectizeInput(inputId = ns("stateinput"), label = "Select state",
                 choices = unique(df$state),
                 selected = defaultstate, multiple = FALSE)
}


module_input_server <- function(id, df) {
  moduleServer(id,
               function(input, output, session) {
                 # Filter data set based on input
                 table <- reactive({df |> filter(state == input$stateinput)})
                 return(table)
               }
  )
}
```

ID is instantiated when the module function is called from the main app. Each ui and server call have the same unique id connecting the two.

Deepsha Menghani
https://deepshamenghani.quarto.pub/dmenghani/

```r
dataset <- read.csv("./data/bfro_reports_geocoded.csv")

ui <- fluidPage(h1("Bigfoot Sightings in the United States", align="center"),
                fluidRow(
                    # Input state
                    column(2,selectizeInput(inputId = "state",
                                            label = "Select State",
                                            choices = unique(dataset$state),
                                            selected = "Washington",
                                            multiple = FALSE)),

                    # Show county plot
                    column(5, plotOutput(outputId = "plotcounty")),

                    # Show yearly plot
                    column(5, plotOutput(outputId = "plotyearly"))
                ))

# Define server logic ----
server <- function(input, output, session) {

  # filter data based on user selection
  data_filtered <- reactive({dataset |> filter(state == input$state)})

  # County plot
  output$plotcounty <- renderPlot(
    data_filtered() |>
      count(county) |>
      ggplot() +
      geom_col(aes(county, n, fill = n), colour = NA, width = 0.8)
      # ... rest of the plot code
  )

  # Yearly plot
  output$plotyearly <- renderPlot(
    data_filtered() |>
      count(year) |>
      ggplot(aes(year, n)) +
      geom_point(aes(year, highest_count), alpha=1, size = 2)
      # ... rest of the plot code
  )
}

# Run the application
shinyApp(ui = ui, server = server)
```

A namespace is like a separate room that contains all the functions and variables specific to that module, preventing naming conflicts and keeping things organized

```r
module_input_ui <- function(id, df, defaultstate = "Washington") {
  # Namespace
  ns <- NS(id)
  # Input UI command
  selectizeInput(inputId = ns("stateinput"), label = "Select state",
                 choices = unique(df$state),
                 selected = defaultstate, multiple = FALSE)
}


module_input_server <- function(id, df) {
  moduleServer(id,
               function(input, output, session) {
                 # Filter data set based on input
                 table <- reactive({df |> filter(state == input$stateinput)})
                 return(table)
               }
  )
}
```

Original app.R file

```r
dataset <- read.csv("./data/bfro_reports_geocoded.csv")

ui <- fluidPage(h1("Bigfoot Sightings in the United States", align="center"),
                fluidRow(
                    # Input state
                    column(2,selectizeInput(inputId = "state",
                                            label = "Select State",
                                            choices = unique(dataset$state),
                                            selected = "Washington",
                                            multiple = FALSE)),

                    # Show county plot
                    column(5, plotOutput(outputId = "plotcounty")),

                    # Show yearly plot
                    column(5, plotOutput(outputId = "plotyearly"))
                ))

# Define server logic ----
server <- function(input, output, session) {

  # filter data based on user selection
  data_filtered <- reactive({dataset |> filter(state == input$state)})

  # County plot
  output$plotcounty <- renderPlot(
    data_filtered() |>
      count(county) |>
      ggplot() +
      geom_col(aes(county, n, fill = n), colour = NA, width = 0.8)
      # ... rest of the plot code
  )

  # Yearly plot
  output$plotyearly <- renderPlot(
    data_filtered() |>
      count(year) |>
      ggplot(aes(year, n)) +
      geom_point(aes(year, highest_count), alpha=1, size = 2)
      # ... rest of the plot code
  )
}

# Run the application
shinyApp(ui = ui, server = server)
```

Module 1: Module_input.R

A namespace is like a separate room that contains all the functions and variables specific to that module, preventing naming conflicts and keeping things organized

```r
module_input_ui <- function(id, df, defaultstate = "Washington") {
    # Namespace
    ns <- NS(id)
    # Input UI command
    selectizeInput(inputId = ns("stateinput"), label = "Select state",
                   choices = unique(df$state),
                   selected = defaultstate, multiple = FALSE)
}

module_input_server <- function(id, df) {
    moduleServer(id,
                 function(input, output, session) {
                     # Filter data set based on input
                     table <- reactive({df |> filter(state == input$stateinput)})
                     return(table)
                 }
    )
}
```

Each input name is encapsulated in ns() to ensure that it is properly namespaced and avoid conflicts with input names from other module calls in the main application

Step 3: Test your module

Call the module function from the main app.R file

Module 1: Module_input.R

```r
module_input_ui <- function(id, df, defaultstate = "Washington") {
  # Namespace
  ns <- NS(id)
  # Input UI command
  selectizeInput(inputId = ns("stateinput"), label = "Select state",
                 choices = unique(df$state),
                 selected = defaultstate, multiple = FALSE)
}

module_input_server <- function(id, df) {
  moduleServer(id,
              function(input, output, session) {
                # Filter data set based on input
                table <- reactive({df |> filter(state == input$stateinput)})
                return(table)
              }
  )
}
```

Test app.R

```r
source("./modules/module_input.R")

dataset <- read.csv("./data/bfro_reports_geocoded.csv")

ui_test <- fluidPage(
  # Call the ui module as a function
  module_input_ui("input_test", df=dataset))

server_test <- function(input, output, session) {
  # Call the server module as a function
  data_filtered <- module_input_server("input_test", df=dataset)
}

# Call the shiny app
shinyApp(ui=ui_test, server=server_test)
```

Shiny App output

Select state

Washington ▼

Step 3: Test your module

Call the module function from the main app.R file

Test app.R

```r
source("./modules/module_input.R")

dataset <- read.csv("./data/bfro_reports_geocoded.csv")

ui_test <- fluidPage(
  # Call the ui module as a function
  module_input_ui("input_test", df=dataset))

server_test <- function(input, output, session) {
  # Call the server module as a function
  data_filtered <- module_input_server("input_test", df=dataset)
}

# Call the shiny app
shinyApp(ui=ui_test, server=server_test)
```

Source your module_input.R file

Module 1: Module_input.R

```r
module_input_ui <- function(id, df, defaultstate = "Washington") {
  # Namespace
  ns <- NS(id)
  # Input UI command
  selectizeInput(inputId = ns("stateinput"), label = "Select state",
                 choices = unique(df$state),
                 selected = defaultstate, multiple = FALSE)
}

module_input_server <- function(id, df) {
  moduleServer(id,
               function(input, output, session) {
                 # Filter data set based on input
                 table <- reactive({df |> filter(state == input$stateinput)})
                 return(table)
               }
  )
}
```

Shiny App output

Select state

Washington ▼

Step 3: Test your module

Call the module function from the main app.R file

Test app.R

```r
source("./modules/module_input.R")

dataset <- read.csv("./data/bfro_reports_geocoded.csv")

ui_test <- fluidPage(
    # Call the ui module as a function
    module_input_ui("input_test", df=dataset))

server_test <- function(input, output, session) {
    # Call the server module as a function
    data_filtered <- module_input_server("input_test", df=dataset)
}

# Call the shiny app
shinyApp(ui=ui_test, server=server_test)
```

Call the ui module and server module functions
from the main app ui and server

Module 1: Module_input.R

```r
module_input_ui <- function(id, df, defaultstate = "Washington") {
    # Namespace
    ns <- NS(id)
    # Input UI command
    selectizeInput(inputId = ns("stateinput"), label = "Select state",
                   choices = unique(df$state),
                   selected = defaultstate, multiple = FALSE)
}
```

```r
module_input_server <- function(id, df) {
    moduleServer(id,
                 function(input, output, session) {
                     # Filter data set based on input
                     table <- reactive({df |> filter(state == input$stateinput)})
                     return(table)
                 }
    )
}
```

Shiny App output

Select state

Washington ▼

Step 3: Test your module

Call the module function from the main app.R file

Module 1: Module_input.R

```
module_input_ui <- function(id, df, defaultstate = "Washington") {
  # Namespace
  ns <- NS(id)
  # Input UI command
  selectizeInput(inputId = ns("stateinput"), label = "Select state",
                 choices = unique(df$state),
                 selected = defaultstate, multiple = FALSE)
}

module_input_server <- function(id, df) {
  moduleServer(id,
               function(input, output, session) {
                 # Filter data set based on input
                 table <- reactive({df |> filter(state == input$stateinput)})
                 return(table)
               }
  )
}
```

Test app.R

```
source("./modules/module_input.R")

dataset <- read.csv("./data/bfro_reports_geocoded.csv")

ui_test <- fluidPage(
  # Call the ui module as a function
  module_input_ui("input_test", df=dataset))

server_test <- function(input, output, session) {
  # Call the server module as a function
  data_filtered <- module_input_server("input_test", df=dataset)
}

# Call the shiny app
shinyApp(ui=ui_test, server=server_test)
```

Call the shiny app command

Shiny App output

Select state

Washington ▼

Shiny app is created

Handling multiple module calls

Call the module function from the main app.R file
multiple times with separate unique IDs – it is that simple!!

```r
module_input_ui <- function(id, df, defaultstate = "Washington") {
  # Namespace
  ns <- NS(id)
  # Input UI command
  selectizeInput(inputId = ns("stateinput"), label = "Select state",
                 choices = unique(df$state),
                 selected = defaultstate, multiple = FALSE)
}
```

```r
module_input_server <- function(id, df) {
  moduleServer(id,
               function(input, output, session) {
                 # Filter data set based on input
                 table <- reactive({df |> filter(state == input$stateinput)})
                 return(table)
               }
  )
}
```

Test app.R

```r
source("./modules/module_input.R")

dataset <- read.csv("./data/bfro_reports_geocoded.csv")

ui_test_multiple <- fluidPage(
  # Call the ui module as a function
  module_input_ui("input_test1", df=dataset),
  module_input_ui("input_test2", df=dataset, defaultstate = "Ohio"),
  module_input_ui("input_test3", df=dataset, defaultstate = "California"))

server_test_multiple <- function(input, output, session) {
  # Call the server module as a function
  data_filtered1 <- module_input_server("input_test1", df=dataset)
  data_filtered2 <- module_input_server("input_test2", df=dataset)
  data_filtered3 <- module_input_server("input_test3", df=dataset)
}

# Call the shiny app
shinyApp(ui=ui_test_multiple, server=server_test_multiple)
```

Shiny App output

Select state

Washington

Select state

Ohio

Select state

California

Handling multiple module calls

Call the module function from the main app.R file
multiple times with separate unique IDs – it is that simple!!

Test app.R

```r
source("./modules/module_input.R")

dataset <- read.csv("./data/bfro_reports_geocoded.csv")

ui_test_multiple <- fluidPage(
    # Call the ui module as a function
    module_input_ui("input_test1", df=dataset),
    module_input_ui("input_test2", df=dataset, defaultstate = "Ohio"),
    module_input_ui("input_test3", df=dataset, defaultstate = "California"))

server_test_multiple <- function(input, output, session) {
    # Call the server module as a function
    data_filtered1 <- module_input_server("input_test1", df=dataset)
    data_filtered2 <- module_input_server("input_test2", df=dataset)
    data_filtered3 <- module_input_server("input_test3", df=dataset)
}

# Call the shiny app
shinyApp(ui=ui_test_multiple, server=server_test_multiple)
```

Module 1: Module_input.R

```r
module_input_ui <- function(id, df, defaultstate = "Washington") {
    # Namespace
    ns <- NS(id)
    # Input UI command
    selectizeInput(inputId = ns("stateinput"), label = "Select state",
                   choices = unique(df$state),
                   selected = defaultstate, multiple = FALSE)
}

module_input_server <- function(id, df) {
    moduleServer(id,
                 function(input, output, session) {
                     # Filter data set based on input
                     table <- reactive({df |> filter(state == input$stateinput)})
                     return(table)
                 }
    )
}
```

Shiny App output

Select state

| Washington ▼ |

Select state

| Ohio ▼ |

Select state

| California ▼ |

Step 4: Replicate modularization for other decomposed components

- Module 1: Module_input.R
  - Ui: Create state input ui
  - Server: Filter data based on state input

- Module 2: Module_countyplot.R
  - Ui: Create county plot ui
  - Server: Generate the county plot with ggplot code

- Module 3: Module_yearlyplot.R
  - Ui: Create yearly plot ui
  - Server: Generate the yearly plot with ggplot code

```r
dataset <- read.csv("./data/bfro_reports_geocoded.csv")

ui <- fluidPage(h1("Bigfoot Sightings in the United States", align="center"),
                fluidRow(
                    # Input state
                    column(2,selectizeInput(inputId = "state",
                                            label = "Select State",
                                            choices = unique(dataset$state),
                                            selected = "Washington",
                                            multiple = FALSE)),

                    # Show county plot
                    column(5, plotOutput(outputId = "plotcounty")),

                    # Show yearly plot
                    column(5, plotOutput(outputId = "plotyearly"))
                ))

# Define server logic ----
server <- function(input, output, session) {

    # filter data based on user selection
    data_filtered <- reactive({dataset |> filter(state == input$state)})

    # County plot
    output$plotcounty <- renderPlot(
        data_filtered() |>
            count(county) |>
            ggplot() +
            geom_col(aes(county, n, fill = n), colour = NA, width = 0.8)
            # ... rest of the plot code
    )

    # Yearly plot
    output$plotyearly <- renderPlot(
        data_filtered() |>
            count(year) |>
            ggplot(aes(year, n)) +
            geom_point(aes(year, highest_count), alpha=1, size = 2)
            # ... rest of the plot code
    )
}

# Run the application
shinyApp(ui = ui, server = server)
```

```r
module_county_ui <- function(id) {
  # Namespace
  ns <- NS(id)
  plotOutput(outputId = ns("plotcounty"))
}

module_county_server <- function(id, df_filtered) {
  moduleServer(id,
            function(input, output, session) {
                # County plot code
                output$plotcounty <- renderPlot(
                    df_filtered() |>
                        count(county) |>
                        ggplot() +
                        geom_col(aes(county, n, fill = n),
                                colour = NA, width = 0.8)
                        # ... rest of the plot code
                    )
                }
            )
}
```

Deepsha Menghani
https://deepshamenghani.quarto.pub/dmenghani/

```r
dataset <- read.csv("./data/bfro_reports_geocoded.csv")

ui <- fluidPage(h1("Bigfoot Sightings in the United States", align="center"),
                fluidRow(
                    # Input state
                    column(2,selectizeInput(inputId = "state",
                                            label = "Select State",
                                            choices = unique(dataset$state),
                                            selected = "Washington",
                                            multiple = FALSE)),

                    # Show county plot
                    column(5, plotOutput(outputId = "plotcounty")),

                    # Show yearly plot
                    column(5, plotOutput(outputId = "plotyearly"))
                ))

# Define server logic ----
server <- function(input, output, session) {

  # filter data based on user selection
  data_filtered <- reactive({dataset |> filter(state == input$state)})

  # County plot
  output$plotcounty <- renderPlot(
    data_filtered() |>
      count(county) |>
      ggplot() +
      geom_col(aes(county, n, fill = n), colour = NA, width = 0.8)
      # ... rest of the plot code
  )

  # Yearly plot
  output$plotyearly <- renderPlot(
    data_filtered() |>
      count(year) |>
      ggplot(aes(year, n)) +
      geom_point(aes(year, highest_count), alpha=1, size = 2)
      # ... rest of the plot code
  )
}

# Run the application
shinyApp(ui = ui, server = server)
```

```r
module_yearly_ui <- function(id) {
  # Namespace
  ns <- NS(id)
  plotOutput(outputId = ns("plotyearly"))
}

module_yearly_server <- function(id, df_filtered) {
  moduleServer(id,
               function(input, output, session) {
                 # Yearly plot code
                 output$plotyearly <- renderPlot(
                   df_filtered() |>
                     count(year) |>
                     ggplot(aes(year, n)) +
                     geom_point(aes(year, highest_count), alpha=1, size = 2)
                     # ... rest of the plot code
                 )
               }
  )
}
```

Original app.R file

```r
dataset <- read.csv("./data/bfro_reports_geocoded.csv")

ui <- fluidPage(h1("Bigfoot Sightings in the United States", align="center"),
                fluidRow(
                  # Input state
                  column(2,selectizeInput(inputId = "state",
                                          label = "Select State",
                                          choices = unique(dataset$state),
                                          selected = "Washington",
                                          multiple = FALSE)),

                  # Show county plot
                  column(5, plotOutput(outputId = "plotcounty")),

                  # Show yearly plot
                  column(5, plotOutput(outputId = "plotyearly"))
                ))

# Define server logic ----
server <- function(input, output, session) {

  # filter data based on user selection
  data_filtered <- reactive({dataset |> filter(state == input$state)})

  # County plot
  output$plotcounty <- renderPlot(
    data_filtered() |>
      count(county) |>
      ggplot() +
      geom_col(aes(county, n, fill = n), colour = NA, width = 0.8)
      # ... rest of the plot code
  )

  # Yearly plot
  output$plotyearly <- renderPlot(
    data_filtered() |>
      count(year) |>
      ggplot(aes(year, n)) +
      geom_point(aes(year, highest_count), alpha=1, size = 2)
      # ... rest of the plot code
  )
}

# Run the application
shinyApp(ui = ui, server = server)
```

Step 5: Call modules from the main app.R file

Updated app.R file

```r
source("./modules/module_input.R")
source("./modules/module_countyplot.R")
source("./modules/module_yearlyplot.R")

dataset <- read.csv("./data/bfro_reports_geocoded.csv")

# App with single section

ui <- fluidPage(h1("Bigfoot Sightings in the United States", align="center"),
                fluidRow(
                  # Input state
                  column(2,module_input_ui("inputs", dataset)),
                  # Show county plot
                  column(5, module_county_ui("countyplot")),
                  # Show yearly plot
                  column(5, module_yearly_ui("timeplot"))
                ))

# Define server logic ----
server <- function(input, output, session) {

  data_filtered <- module_input_server("inputs", dataset)
  module_county_server("countyplot", df_filtered = data_filtered)
  module_yearly_server("timeplot", df_filtered = data_filtered)

}

# Run the application
shinyApp(ui = ui, server = server)
```

Deepsha Menghani
https://deepshamenghani.quarto.pub/dmenghani/

```r
dataset <- read.csv("./data/bfro_reports_geocoded.csv")

ui <- fluidPage(h1("Bigfoot Sightings in the United States", align="center"),
                fluidRow(
                    # Input state
                    column(2,selectizeInput(inputId = "state",
                                            label = "Select State",
                                            choices = unique(dataset$state),
                                            selected = "Washington",
                                            multiple = FALSE)),

                    # Show county plot
                    column(5, plotOutput(outputId = "plotcounty")),

                    # Show yearly plot
                    column(5, plotOutput(outputId = "plotyearly"))
                ))

# Define server logic ----
server <- function(input, output, session) {

  # filter data based on user selection
  data_filtered <- reactive({dataset |> filter(state == input$state)})

  # County plot
  output$plotcounty <- renderPlot(
    data_filtered() |>
      count(county) |>
      ggplot() +
      geom_col(aes(county, n, fill = n), colour = NA, width = 0.8)
      # ... rest of the plot code
  )

  # Yearly plot
  output$plotyearly <- renderPlot(
    data_filtered() |>
      count(year) |>
      ggplot(aes(year, n)) +
      geom_point(aes(year, highest_count), alpha=1, size = 2)
      # ... rest of the plot code
  )
}

# Run the application
shinyApp(ui = ui, server = server)
```

Updated app.R file

```r
source("./modules/module_input.R")
source("./modules/module_countyplot.R")
source("./modules/module_yearlyplot.R")

dataset <- read.csv("./data/bfro_reports_geocoded.csv")

# App with single section

ui <- fluidPage(h1("Bigfoot Sightings in the United States", align="center"),
                fluidRow(
                    # Input state
                    column(2,module_input_ui("inputs", dataset)),
                    # Show county plot
                    column(5, module_county_ui("countyplot")),
                    # Show yearly plot
                    column(5, module_yearly_ui("timeplot"))
                ))

# Define server logic ----
server <- function(input, output, session) {

  data_filtered <- module_input_server("inputs", dataset)
  module_county_server("countyplot", df_filtered = data_filtered)
  module_yearly_server("timeplot", df_filtered = data_filtered)

}

# Run the application
shinyApp(ui = ui, server = server)
```

Deepsha Menghani
https://deepshamenghani.quarto.pub/dmenghani/

Original app.R file

```r
dataset <- read.csv("./data/bfro_reports_geocoded.csv")

ui <- fluidPage(h1("Bigfoot Sightings in the United States", align="center"),
                fluidRow(
                    # Input state
                    column(2,selectizeInput(inputId = "state",
                                            label = "Select State",
                                            choices = unique(dataset$state),
                                            selected = "Washington",
                                            multiple = FALSE)),

                    # Show county plot
                    column(5, plotOutput(outputId = "plotcounty")),

                    # Show yearly plot
                    column(5, plotOutput(outputId = "plotyearly"))
                ))

# Define server logic ----
server <- function(input, output, session) {

  # filter data based on user selection
  data_filtered <- reactive({dataset |> filter(state == input$state)})

  # County plot
  output$plotcounty <- renderPlot(
    data_filtered() |>
      count(county) |>
      ggplot() +
      geom_col(aes(county, n, fill = n), colour = NA, width = 0.8)
      # ... rest of the plot code
  )

  # Yearly plot
  output$plotyearly <- renderPlot(
    data_filtered() |>
      count(year) |>
      ggplot(aes(year, n)) +
      geom_point(aes(year, highest_count), alpha=1, size = 2)
      # ... rest of the plot code
  )
}

# Run the application
shinyApp(ui = ui, server = server)
```

Call state input ui and server modules with "same" id

Updated app.R file

```r
source("./modules/module_input.R")
source("./modules/module_countyplot.R")
source("./modules/module_yearlyplot.R")

dataset <- read.csv("./data/bfro_reports_geocoded.csv")

# App with single section

ui <- fluidPage(h1("Bigfoot Sightings in the United States", align="center"),
                fluidRow(
                    # Input state
                    column(2,module_input_ui("inputs", dataset)),
                    # Show county plot
                    column(5, module_county_ui("countyplot")),
                    # Show yearly plot
                    column(5, module_yearly_ui("timeplot"))
                ))

# Define server logic ----
server <- function(input, output, session) {

  data_filtered <- module_input_server("inputs", dataset)
  module_county_server("countyplot", df_filtered = data_filtered)
  module_yearly_server("timeplot", df_filtered = data_filtered)

}

# Run the application
shinyApp(ui = ui, server = server)
```

Deepsha Menghani
https://deepshamenghani.quarto.pub/dmenghani/

Updated app.R file

```r
dataset <- read.csv("./data/bfro_reports_geocoded.csv")

ui <- fluidPage(h1("Bigfoot Sightings in the United States", align="center"),
                fluidRow(
                    # Input state
                    column(2,selectizeInput(inputId = "state",
                                            label = "Select State",
                                            choices = unique(dataset$state),
                                            selected = "Washington",
                                            multiple = FALSE)),

                    # Show county plot
                    column(5, plotOutput(outputId = "plotcounty")),

                    # Show yearly plot
                    column(5, plotOutput(outputId = "plotyearly"))
                ))

# Define server logic ----
server <- function(input, output, session) {

  # filter data based on user selection
  data_filtered <- reactive({dataset |> filter(state == input$state)})

  # County plot
  output$plotcounty <- renderPlot(
    data_filtered() |>
      count(county) |>
      ggplot() +
      geom_col(aes(county, n, fill = n), colour = NA, width = 0.8)
      # ... rest of the plot code
  )

  # Yearly plot
  output$plotyearly <- renderPlot(
    data_filtered() |>
      count(year) |>
      ggplot(aes(year, n)) +
      geom_point(aes(year, highest_count), alpha=1, size = 2)
      # ... rest of the plot code
  )
}

# Run the application
shinyApp(ui = ui, server = server)
```

```r
source("./modules/module_input.R")
source("./modules/module_countyplot.R")
source("./modules/module_yearlyplot.R")

dataset <- read.csv("./data/bfro_reports_geocoded.csv")

# App with single section

ui <- fluidPage(h1("Bigfoot Sightings in the United States", align="center"),
                fluidRow(
                    # Input state
                    column(2,module_input_ui("inputs", dataset)),
                    # Show county plot
                    column(5, module_county_ui("countyplot")),
                    # Show yearly plot
                    column(5, module_yearly_ui("timeplot"))
                ))

# Define server logic ----
server <- function(input, output, session) {

  data_filtered <- module_input_server("inputs", dataset)
  module_county_server("countyplot", df_filtered = data_filtered)
  module_yearly_server("timeplot", df_filtered = data_filtered)

}

# Run the application
shinyApp(ui = ui, server = server)
```

Updated app.R file

```r
dataset <- read.csv("./data/bfro_reports_geocoded.csv")

ui <- fluidPage(h1("Bigfoot Sightings in the United States", align="center"),
                fluidRow(
                    # Input state
                    column(2,selectizeInput(inputId = "state",
                                            label = "Select State",
                                            choices = unique(dataset$state),
                                            selected = "Washington",
                                            multiple = FALSE)),

                    # Show county plot
                    column(5, plotOutput(outputId = "plotcounty")),

                    # Show yearly plot
                    column(5, plotOutput(outputId = "plotyearly"))
                ))

# Define server logic ----
server <- function(input, output, session) {

  # filter data based on user selection
  data_filtered <- reactive({dataset |> filter(state == input$state)})

  # County plot
  output$plotcounty <- renderPlot(
    data_filtered() |>
      count(county) |>
      ggplot() +
      geom_col(aes(county, n, fill = n), colour = NA, width = 0.8)
      # ... rest of the plot code
  )

  # Yearly plot
  output$plotyearly <- renderPlot(
    data_filtered() |>
      count(year) |>
      ggplot(aes(year, n)) +
      geom_point(aes(year, highest_count), alpha=1, size = 2)
      # ... rest of the plot code
  )
}

# Run the application
shinyApp(ui = ui, server = server)
```

```r
source("./modules/module_input.R")
source("./modules/module_countyplot.R")
source("./modules/module_yearlyplot.R")

dataset <- read.csv("./data/bfro_reports_geocoded.csv")

# App with single section

ui <- fluidPage(h1("Bigfoot Sightings in the United States", align="center"),
                fluidRow(
                    # Input state
                    column(2,module_input_ui("inputs", dataset)),
                    # Show county plot
                    column(5, module_county_ui("countyplot")),
                    # Show yearly plot
                    column(5, module_yearly_ui("timeplot"))
                ))

# Define server logic ----
server <- function(input, output, session) {

  data_filtered <- module_input_server("inputs", dataset)
  module_county_server("countyplot", df_filtered = data_filtered)
  module_yearly_server("timeplot", df_filtered = data_filtered)

}

# Run the application
shinyApp(ui = ui, server = server)
```

Updated app.R file

```r
dataset <- read.csv("./data/bfro_reports_geocoded.csv")

ui <- fluidPage(h1("Bigfoot Sightings in the United States", align="center"),
                fluidRow(
                    # Input state
                    column(2,selectizeInput(inputId = "state",
                                            label = "Select State",
                                            choices = unique(dataset$state),
                                            selected = "Washington",
                                            multiple = FALSE)),

                    # Show county plot
                    column(5, plotOutput(outputId = "plotcounty")),

                    # Show yearly plot
                    column(5, plotOutput(outputId = "plotyearly"))
                ))

# Define server logic ----
server <- function(input, output, session) {

  # filter data based on user selection
  data_filtered <- reactive({dataset |> filter(state == input$state)})

  # County plot
  output$plotcounty <- renderPlot(
    data_filtered() |>
      count(county) |>
      ggplot() +
      geom_col(aes(county, n, fill = n), colour = NA, width = 0.8)
      # ... rest of the plot code
  )

  # Yearly plot
  output$plotyearly <- renderPlot(
    data_filtered() |>
      count(year) |>
      ggplot(aes(year, n)) +
      geom_point(aes(year, highest_count), alpha=1, size = 2)
      # ... rest of the plot code
  )
}

# Run the application
shinyApp(ui = ui, server = server)
```

```r
source("./modules/module_input.R")
source("./modules/module_countyplot.R")
source("./modules/module_yearlyplot.R")

dataset <- read.csv("./data/bfro_reports_geocoded.csv")

# App with single section

ui <- fluidPage(h1("Bigfoot Sightings in the United States", align="center"),
                fluidRow(
                    # Input state
                    column(2,module_input_ui("inputs", dataset)),
                    # Show county plot
                    column(5, module_county_ui("countyplot")),
                    # Show yearly plot
                    column(5, module_yearly_ui("timeplot"))
                ))

# Define server logic ----
server <- function(input, output, session) {

  data_filtered <- module_input_server("inputs", dataset)
  module_county_server("countyplot", df_filtered = data_filtered)
  module_yearly_server("timeplot", df_filtered = data_filtered)

}

# Run the application
shinyApp(ui = ui, server = server)
```
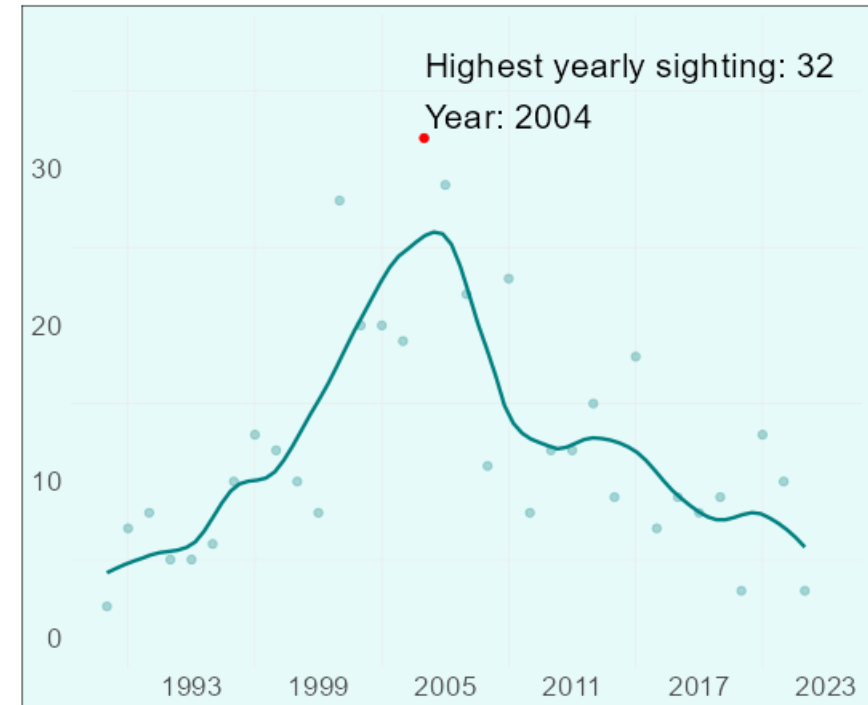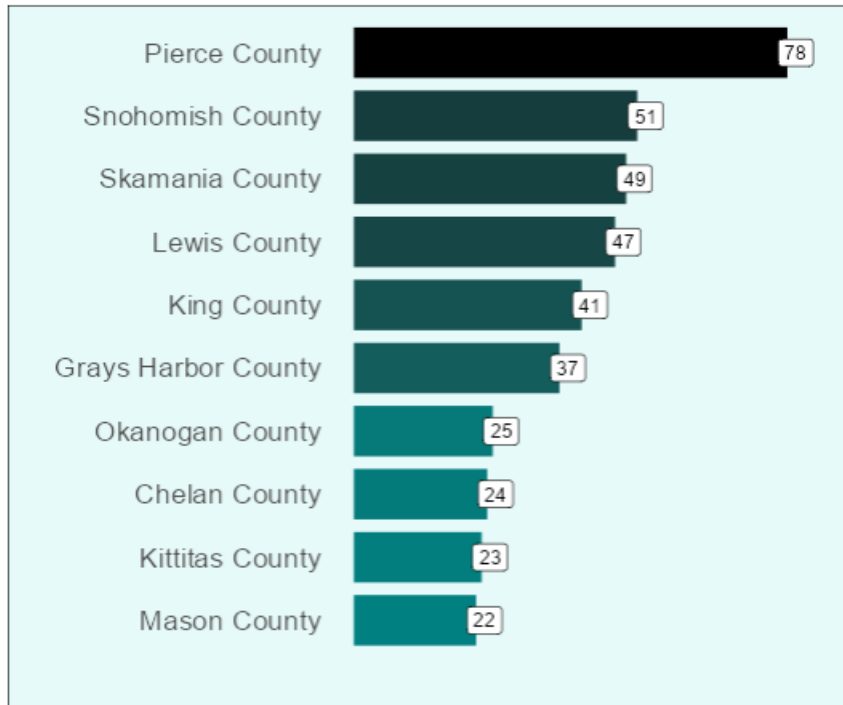
app.R is so clean and simple to read!!!
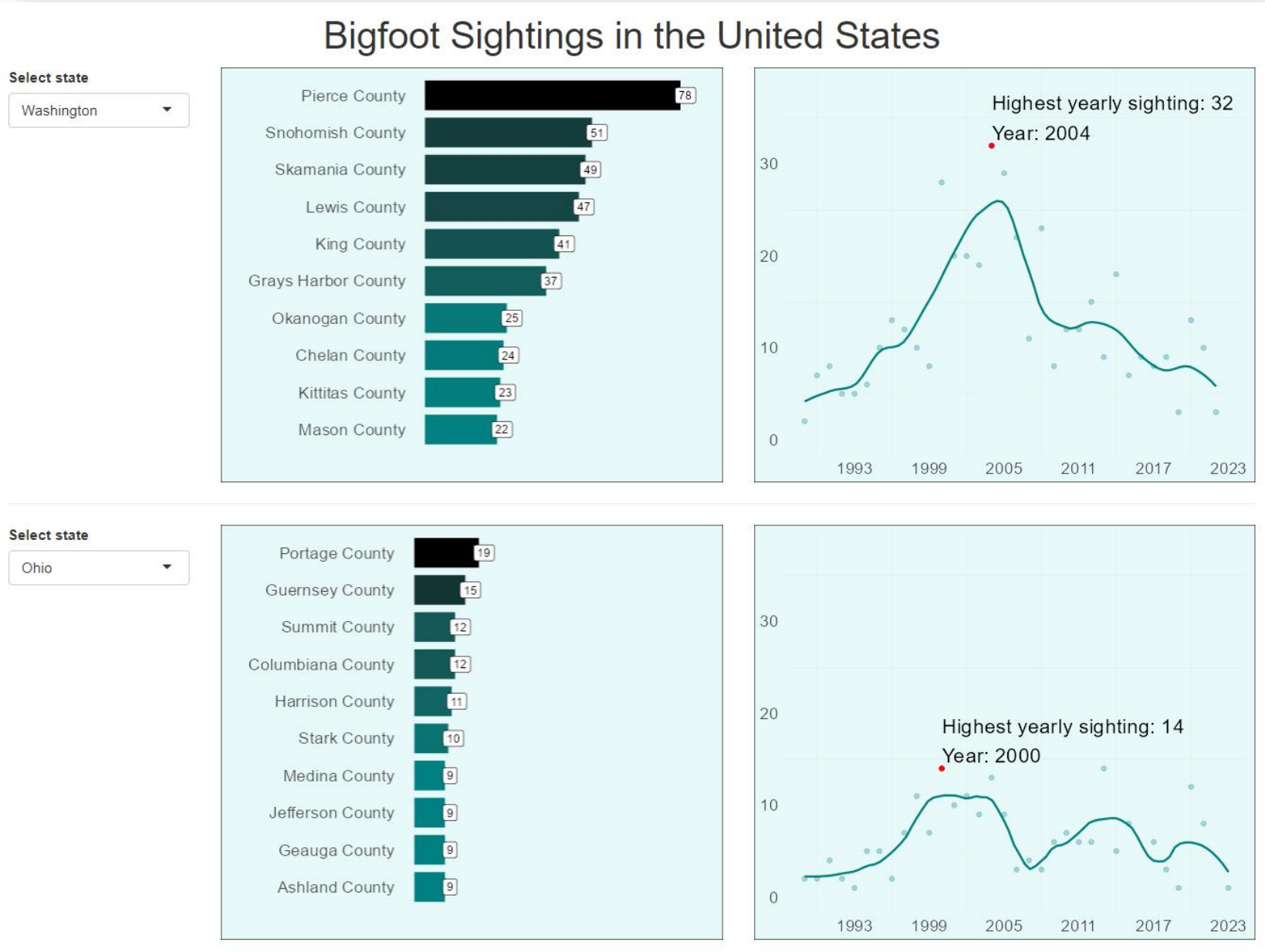
Shiny output

Bigfoot Sightings in the United States

Now, let's replicate input and output plots for two state comparison

Like this ->

Original app.R file

```r
dataset <- read.csv("./data/bfro_reports_geocoded.csv")

ui <- fluidPage(h1("Bigfoot Sightings in the United States", align="center"),
                fluidRow(
                    # Input state
                    column(2,selectizeInput(inputId = "state",
                                            label = "Select State",
                                            choices = unique(dataset$state),
                                            selected = "Washington",
                                            multiple = FALSE)),

                    # Show county plot
                    column(5, plotOutput(outputId = "plotcounty")),

                    # Show yearly plot
                    column(5, plotOutput(outputId = "plotyearly"))
                ))

# Define server logic ----
server <- function(input, output, session) {

  # filter data based on user selection
  data_filtered <- reactive({dataset |> filter(state == input$state)})

  # County plot
  output$plotcounty <- renderPlot(
    data_filtered() |>
      count(county) |>
      ggplot() +
      geom_col(aes(county, n, fill = n), colour = NA, width = 0.8)
      # ... rest of the plot code
  )

  # Yearly plot
  output$plotyearly <- renderPlot(
    data_filtered() |>
      count(year) |>
      ggplot(aes(year, n)) +
      geom_point(aes(year, highest_count), alpha=1, size = 2)
      # ... rest of the plot code
  )
}

# Run the application
shinyApp(ui = ui, server = server)
```

Updated app.R file – single module, multiple calls

```r
source("./modules/module_input.R")
source("./modules/module_countyplot.R")
source("./modules/module_yearlyplot.R")

dataset <- read.csv("./data/bfro_reports_geocoded.csv")

# App with two sections

ui <- fluidPage(h1("Bigfoot Sightings in the United States",
                   align="center"),
                fluidRow(
                    # Input state
                    column(2,module_input_ui("inputs", dataset)),
                    # Show county plot
                    column(5, module_county_ui("countyplot")),
                    # Show yearly plot
                    column(5, module_yearly_ui("timeplot"))
                ),
                fluidRow(
                    column(2,module_input_ui("inputs_2", dataset,
                                             defaultstate = "Ohio")),
                    column(5, module_county_ui("countyplot_2")),
                    column(5, module_yearly_ui("timeplot_2"))
                ))

# Define server logic ----
server <- function(input, output, session) {

  data_filtered <- module_input_server("inputs", dataset)
  module_county_server("countyplot", df_filtered = data_filtered)
  module_yearly_server("timeplot", df_filtered = data_filtered)

  data_filtered_2 <- module_input_server("inputs_2", dataset)
  module_county_server("countyplot_2", df_filtered = data_filtered_2)
  module_yearly_server("timeplot_2", df_filtered = data_filtered_2)

}

# Run the application
shinyApp(ui = ui, server = server)
```

```r
dataset <- read.csv("./data/bfro_reports_geocoded.csv")

ui <- fluidPage(h1("Bigfoot Sightings in the United States", align="center"),
                fluidRow(
                    # Input state
                    column(2,selectizeInput(inputId = "state",
                                            label = "Select State",
                                            choices = unique(dataset$state),
                                            selected = "Washington",
                                            multiple = FALSE)),

                    # Show county plot
                    column(5, plotOutput(outputId = "plotcounty")),

                    # Show yearly plot
                    column(5, plotOutput(outputId = "plotyearly"))
                ))

# Define server logic ----
server <- function(input, output, session) {

  # filter data based on user selection
  data_filtered <- reactive({dataset |> filter(state == input$state)})

  # County plot
  output$plotcounty <- renderPlot(
    data_filtered() |>
      count(county) |>
      ggplot() +
      geom_col(aes(county, n, fill = n), colour = NA, width = 0.8)
      # ... rest of the plot code
  )

  # Yearly plot
  output$plotyearly <- renderPlot(
    data_filtered() |>
      count(year) |>
      ggplot(aes(year, n)) +
      geom_point(aes(year, highest_count), alpha=1, size = 2)
      # ... rest of the plot code
  )
}

# Run the application
shinyApp(ui = ui, server = server)
```

```r
source("./modules/module_input.R")
source("./modules/module_countyplot.R")
source("./modules/module_yearlyplot.R")

dataset <- read.csv("./data/bfro_reports_geocoded.csv")

# App with two sections

ui <- fluidPage(h1("Bigfoot Sightings in the United States",
                   align="center"),
                fluidRow(
                    # Input state
                    column(2,module_input_ui("inputs", dataset)),
                    # Show county plot
                    column(5, module_county_ui("countyplot")),
                    # Show yearly plot
                    column(5, module_yearly_ui("timeplot"))

                fluidRow(
                    column(2,module_input_ui("inputs_2", dataset,
                                             defaultstate = "Ohio")),
                    column(5, module_county_ui("countyplot_2")),
                    column(5, module_yearly_ui("timeplot_2"))
                ))

# Define server logic ----
server <- function(input, output, session) {

  data_filtered <- module_input_server("inputs", dataset)
  module_county_server("countyplot", df_filtered = data_filtered)
  module_yearly_server("timeplot", df_filtered = data_filtered)

  data_filtered_2 <- module_input_server("inputs_2", dataset)
  module_county_server("countyplot_2", df_filtered = data_filtered_2)
  module_yearly_server("timeplot_2", df_filtered = data_filtered_2)

}

# Run the application
shinyApp(ui = ui, server = server)
```
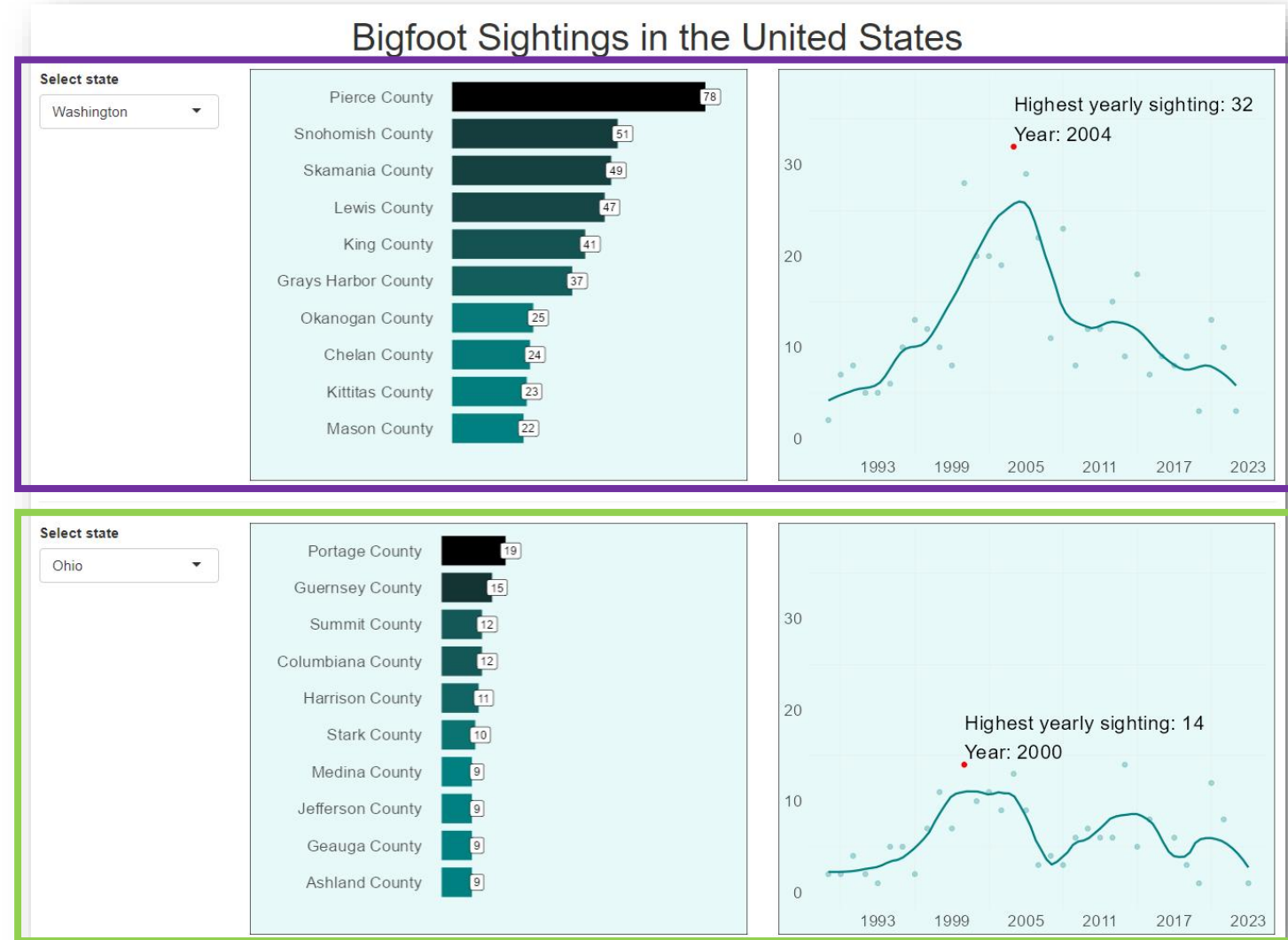
Updated app.R file –
single module, multiple calls

Shiny App output

```r
source("./modules/module_input.R")
source("./modules/module_countyplot.R")
source("./modules/module_yearlyplot.R")

dataset <- read.csv("./data/bfro_reports_geocoded.csv")

# App with two sections

ui <- fluidPage(h1("Bigfoot Sightings in the United States",
                align="center"),
            fluidRow(
                # Input state
                column(2,module_input_ui("inputs", dataset)),
                # Show county plot
                column(5, module_county_ui("countyplot")),
                # Show yearly plot
                column(5, module_yearly_ui("timeplot"))
            ), br(),
            fluidRow(
                column(2,module_input_ui("inputs_2", dataset,
                                    defaultstate = "Ohio")),
                column(5, module_county_ui("countyplot_2")),
                column(5, module_yearly_ui("timeplot_2"))
            ))

# Define server logic ----
server <- function(input, output, session) {

    data_filtered <- module_input_server("inputs", dataset)
    module_county_server("countyplot", df_filtered = data_filtered)
    module_yearly_server("timeplot", df_filtered = data_filtered)

    data_filtered_2 <- module_input_server("inputs_2", dataset)
    module_county_server("countyplot_2", df_filtered = data_filtered_2)
    module_yearly_server("timeplot_2", df_filtered = data_filtered_2)

}

# Run the application
shinyApp(ui = ui, server = server)
```

Deepsha Menghani
https://deepshamenghani.quarto.pub/dmenghani/

Original app.R

ui
- ui Input
- ui Plot display

Server
- Server data filter
- Server Plot
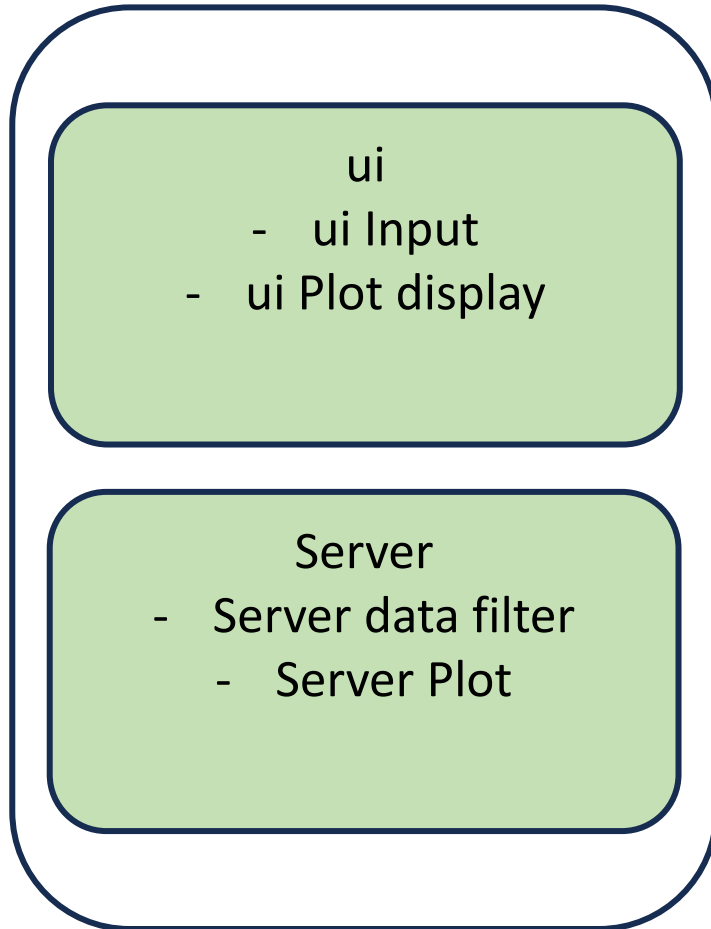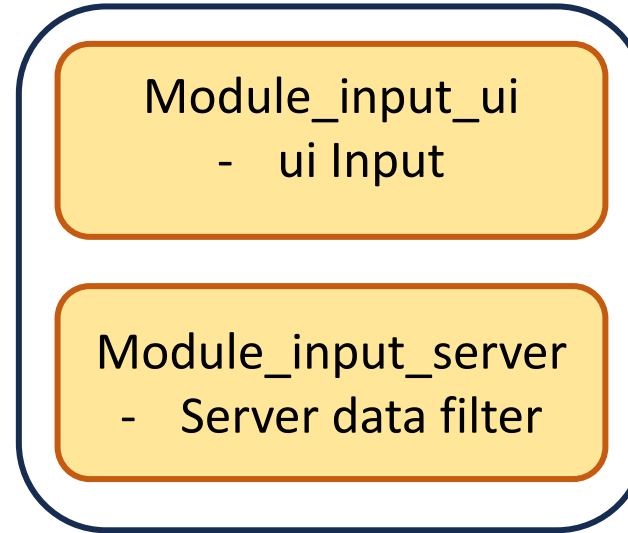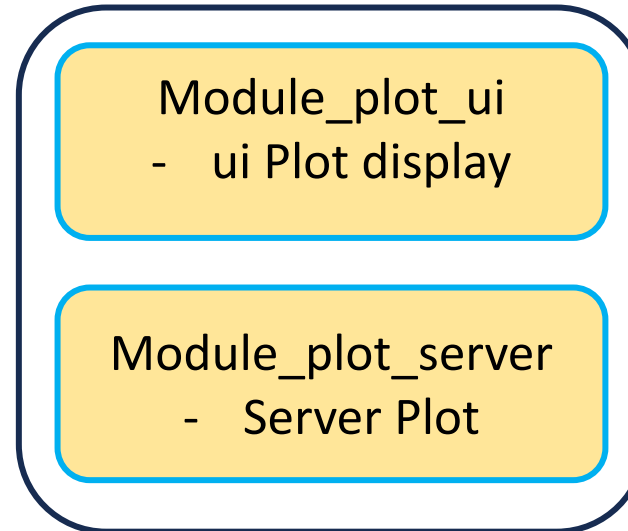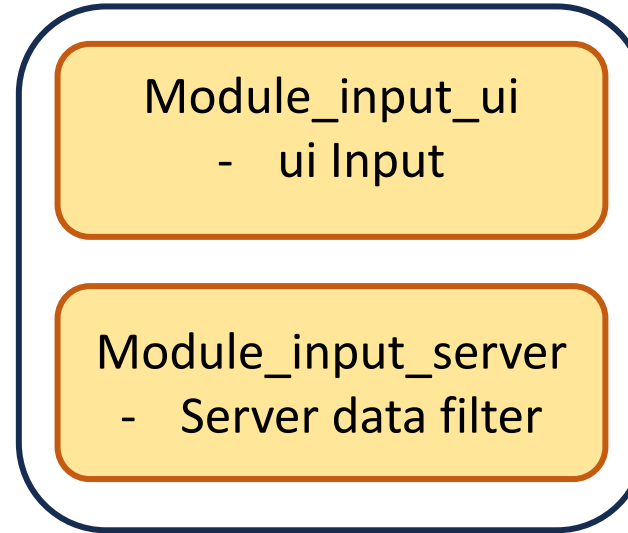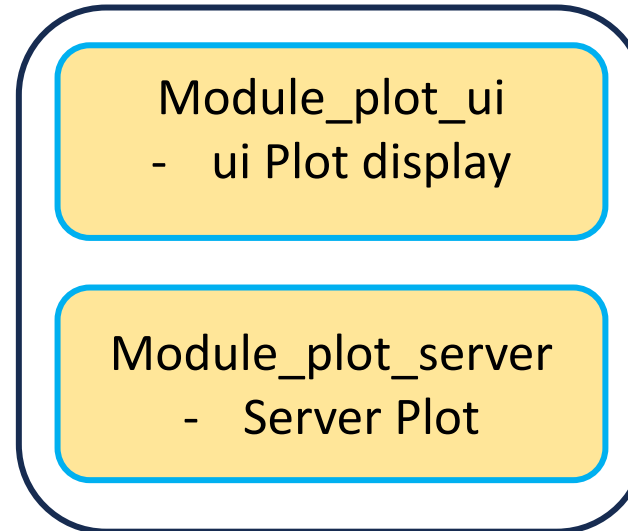
The challenges with a monolith app.R file

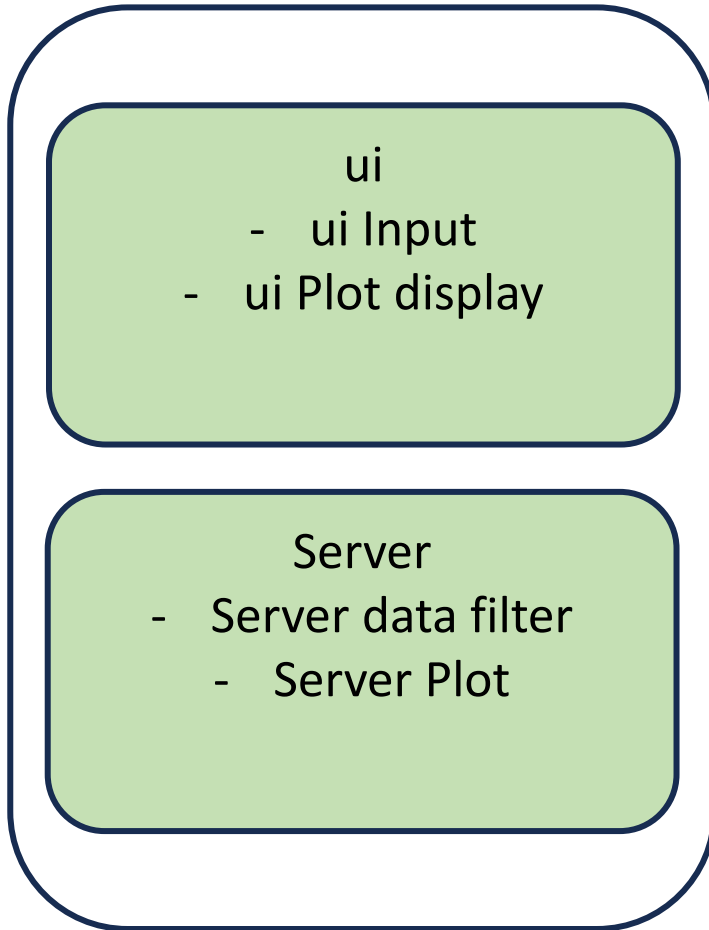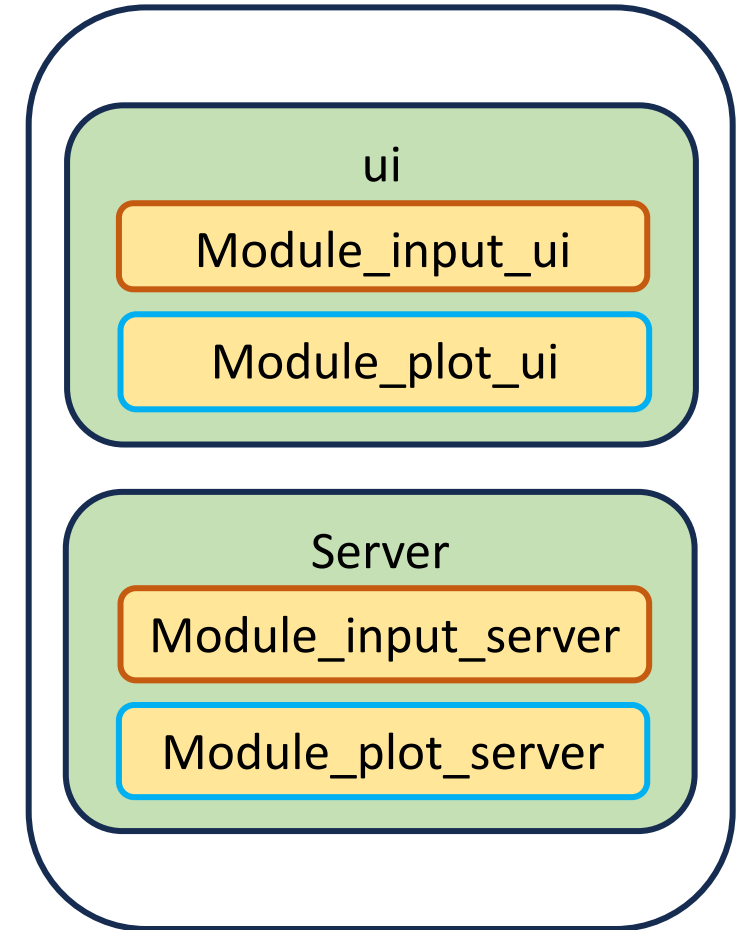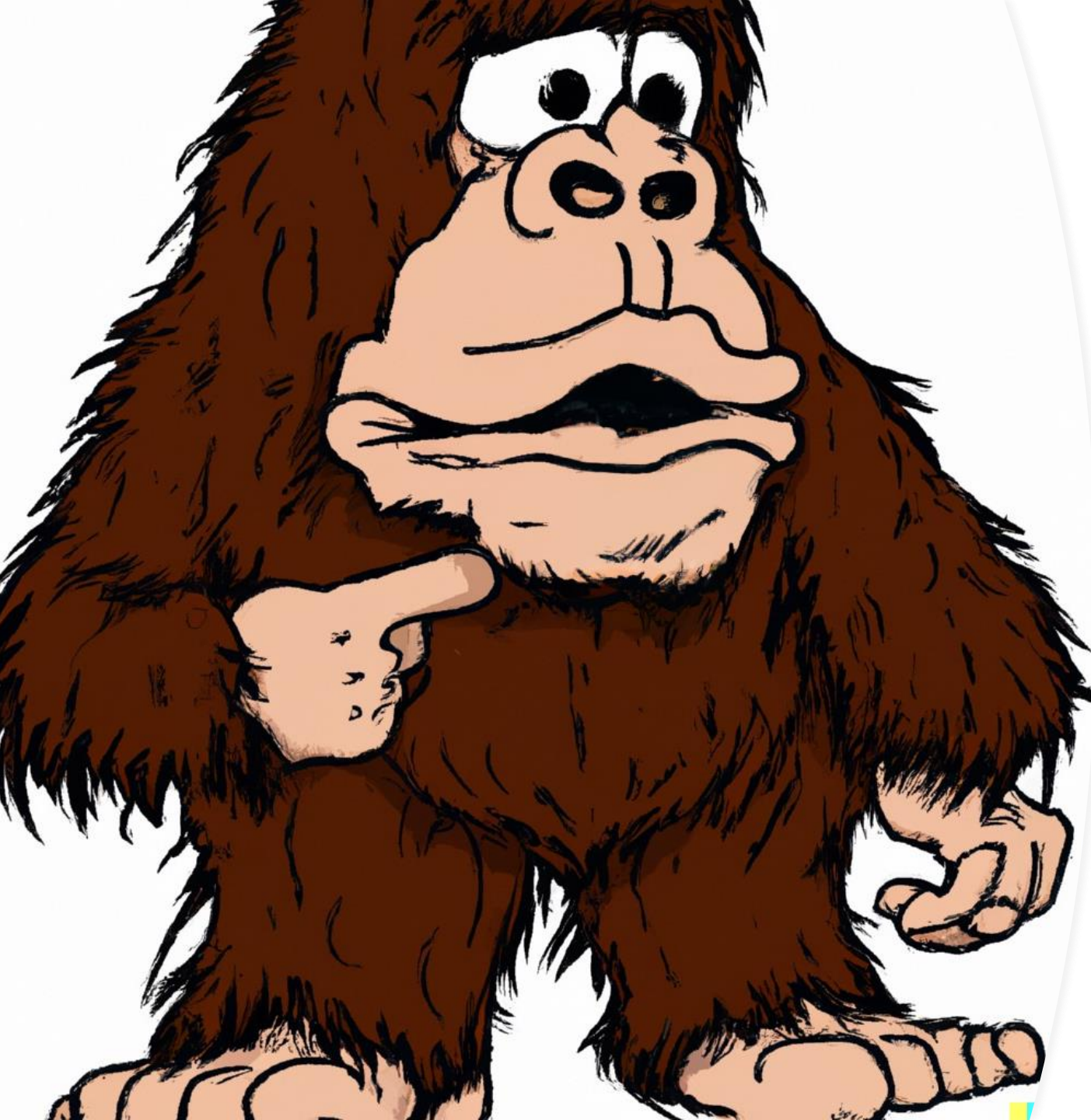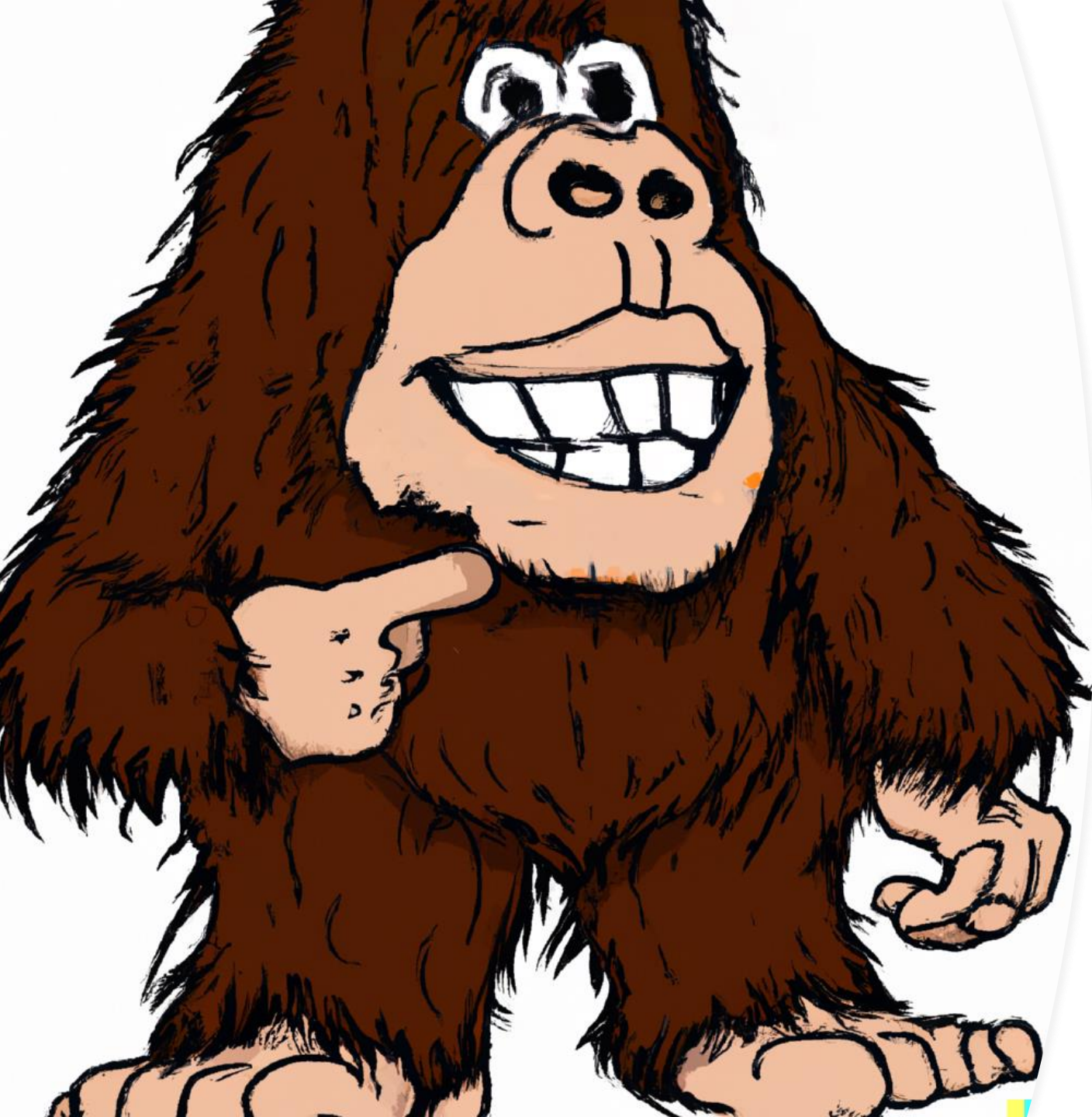- Difficult to keep track of components

- Challenging to debug if something fails

- Harder for someone new to learn and contribute

- Tedious to reuse its elements within itself and other applications

With a modular approach

- Easy to keep track of components

- Single location to debug if something fails

- Structured cleanly for someone new to learn and contribute

- Element usability within itself and other applications

Deepsha Menghani
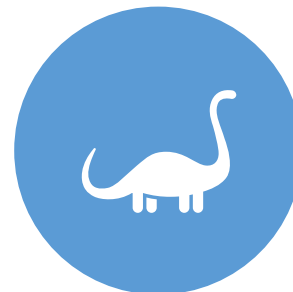https://deepshamenghani.quarto.pub/dmenghani/

Next steps

If you are yet to learn shiny, start with modular approach
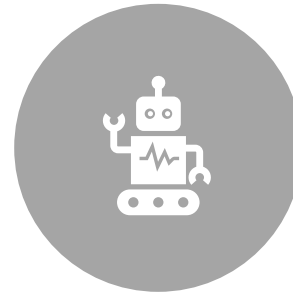
If you learned it the monolith way, start with converting an existing app modular

Check out Mastering Shiny book by Hadley Wickham

Check out the Github repo for modular bigfoot app

Deepsha Menghani: https://deepshamenghani.quarto.pub/dmenghani/