

# Introduction to R Shiny

---

Workshop on Data Visualization in R

**Lokesh Mano** • 18-Jan-2022

NBIS, SciLifeLab

# Contents

- Introduction to RShiny
- Code Structure
- App execution
- UI
- Server

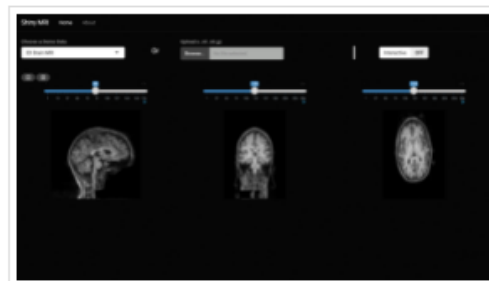
## Life sciences



COVID-19 tracker



Exploring large hospital data for better use of antimicrobials



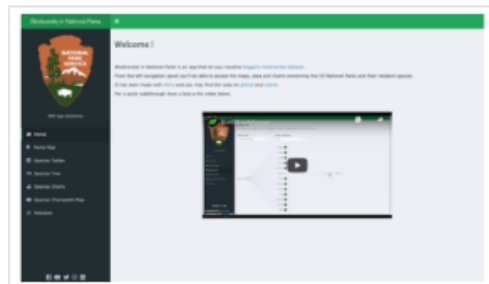
ShinyMRI - View MRI images in Shiny



A/B Testing Sample Size Calculator



ctmweb, a web app to analysis Animal tracking data



Visualizing Biodiversity in National Parks data



iSEE



MOTE: An Effect Size Calculator



Interactively view and subset phylogenetic trees

# What is shiny?

- Interactive documents & web applications
- Completely created using R
- Needs a live environment

## Usage

- [Standalone web applications](#)
- [Dashboard/Flexboard](#)
- Interactive RMarkdown
- Gadgets/RStudio extensions

## App structure

- UI Layout
- UI Inputs (Widgets)
- UI Outputs
- Renderer
- Builder

# Code structure

## One file format

*app.R*

```
ui <- fluidPage()  
server <- function(input,output) {}  
shinyApp(ui=ui,server=server)
```

## Two file format

*ui.R*

```
ui <- fluidPage()
```

*server.R*

```
server <- function(input,output) {}
```

# App execution

- Change to app directory, then run `runApp()`
- Use `shinyApp()`

```
shinyApp(  
  ui=fluidPage(),  
  server=function(input,output) {}  
)
```

- From Rmd file using `rmarkdown::run()`
- Running as a separate process from terminal

```
R -e "shiny::runApp('~/.shinyapp')"
```

# UI • Layout

```
shinyApp(  
  ui=fluidPage(  
    titlePanel("Title Panel"),  
    sidebarLayout(  
      sidebarPanel(  
        helpText("Sidebar Panel")  
      ),  
      mainPanel(tabsetPanel(  
        tabPanel("tab1",  
          fluidRow(  
            column(6,helpText("Col1")),  
            column(6,  
              helpText("Col2"),  
              fluidRow(  

```

## Title Panel



# UI • Widgets • Input

```
shinyApp(  
  ui=fluidPage(  
    fluidRow(  
      column(4,  
        fileInput("file-input", "fileInput:"),  
        selectInput("select-input", label="selectI  
        numericInput("numeric-input", label="numer  
        sliderInput("slider-input", label="sliderI  
        textInput("text-input", label="textInput")  
        textAreaInput("text-area-input", label="te  
        dateInput("date-input", label="dateInput")  
        dateRangeInput("date-range-input", label=""  
        radioButtons("radio-button", label="radioB  
        checkboxInput("checkbox", "checkboxInput"),  
        actionButton("action-button", "Action"),  
        hr(),  
        submitButton()  
      )  
    )  
  ),  
  server=function(input, output) {  
  })  
})
```

fileInput:

selectInput

numericInput

sliderInput

textInput

textAreaInput

dateInput

dateRangeInput

to

radioButtons

☒ A ☐ B ☐ C

☐ checkboxInput

Widgets gallery



# UI • Widgets • Outputs

```
shinyApp(  
  ui=fluidPage(fluidRow(column(5,  
    textInput("text_input",label="textInput",  
    hr(),  
    htmlOutput("html_output"),  
    textOutput("text_output"),  
    verbatimTextOutput("verbatim_text_output"),  
    tableOutput("table_output"),  
    plotOutput("plot_output",width="300px",height="300px"),  
  )),  
  server=function(input, output) {  
    output$html_output <- renderText({input$text_input})  
    output$text_output <- renderText({input$text_input})  
    output$verbatim_text_output <- renderText({input$text_input})  
    output$table_output <- renderTable({iris[1:3,1:3]})  
    output$plot_output <- renderPlot({  
      plot(iris[,1],iris[,2])  
    })  
  })  
})
```

textInput

<h3 style='color:red'>Red text</h3>

Red text

<h3 style='color:red'>Red text</h3>

<h3 style='color:red'>Red text</h3>

Sepal.Length	Sepal.Width	Petal.Length
5.10	3.50	1.40
4.90	3.00	1.40
4.70	3.20	1.30



# Dynamic UI

- UI elements are created conditionally using `uiOutput()` / `renderUI()`

```
shinyApp(  
  ui=fluidPage(  
    selectInput("data",label="Select data",  
               choices=c("mtcars","faithful","iris")),  
    tableOutput("table"),  
    uiOutput("ui")  
  ),  
  server=function(input, output) {  
  
    data <- reactive({ get(input$data, 'package:datasets') })  
  
    output$ui <- renderUI({  
      if(input$data=="iris") plotOutput("plot",width="400px")  
    })  
  
    output$plot <- renderPlot({hist(data()[, 1])})  
    output$table <- renderTable({head(data())})  
  })  
)
```

- Other options include `conditionalPanel()`, `insertUI()` and `removeUI()`

# Server

- `Server` is a function that assembles your `input` into `output` using R based code.
- Three rules to be followed to write a server function:

Rule 1: Save objects to display to `output$`



Rule 2: Build objects to display with `render*()`

- R-Code block (can even be an entire R script) between the braces `{}` inside the `render*()` function.

```
output$hist <- renderPlot({  
  title <- "histogram of 100 random numbers"  
  hist(rnorm(100), main = title)  
})
```

Different Render functions



# Thank you. Questions?

Slide courtesy: Roy Francis (NBIS, RaukR2021)

R version 4.1.2 (2021-11-01)

Platform: x86\_64-pc-linux-gnu (64-bit)

OS: Ubuntu 18.04.6 LTS

---

Built on : 📅 18-Jan-2022 at 🕒 14:20:51

2022 • SciLifeLab • NBIS