

Shiny_HW

Jie Fei

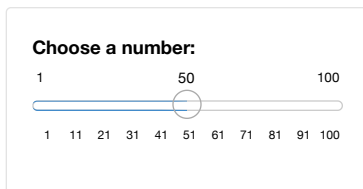
Library(shiny)

```
# shiny app
ui <- fluidPage(
  titlePanel("Hello Shiny!"),
  sidebarLayout(
    sidebarPanel(
      sliderInput("num", "Choose a number:", min = 1, max = 100, value = 50)
    ),
    mainPanel(
      textOutput("result")
    )
  )
)

# define server logic
server <- function(input, output) {
  output$result <- renderText({
    paste("You chose", input$num)
  })
}

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

Hello Shiny!



You chose 50

```
# shiny app
ui <- fluidPage(
  titlePanel("Interactive Plot"),
  sidebarLayout(
    sidebarPanel(
      numericInput("n", "Number of points:", 100),
      selectInput("color", "Color:", choices = c("Red", "Blue", "Green"))
    ),
    mainPanel(
      plotOutput("scatterPlot")
    )
  )
)

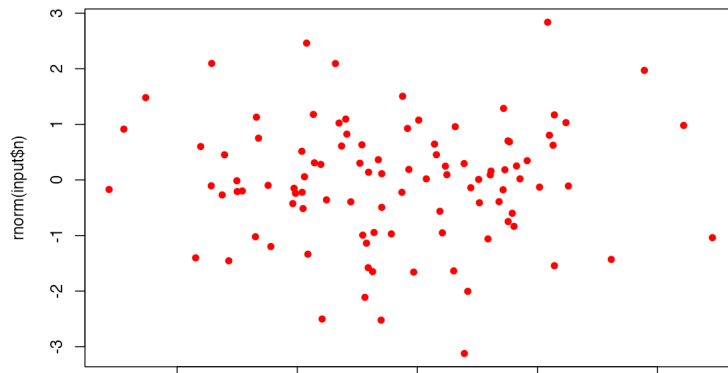
server <- function(input, output) {
  output$scatterPlot <- renderPlot({
    plot(rnorm(input$n), rnorm(input$n), col = input$color, pch = 16)
  })
}

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

Interactive Plot

Number of points:

Color:



2.3.5 Exercises

Q1

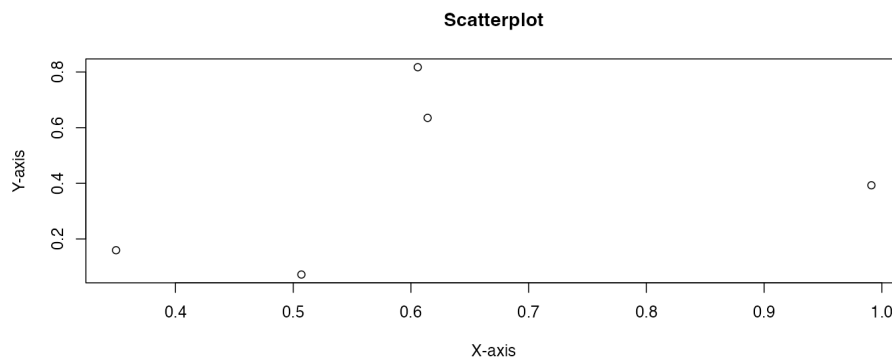
1a. `renderPrint(summary(mtcars))` Paired with: `verbatimTextOutput()` 1b. `renderText("Good morning!")` Paired with: `textOutput()` 1c. `renderPrint(t.test(1:5, 2:6))` Paired with: `verbatimTextOutput()` 1d. `renderText(str(lm(mpg ~ wt, data = mtcars)))` Paired with: `textOutput()`

Q2

```
ui <- fluidPage(
  tags$div(
    plotOutput("scatterplot", height = "300px", width = "700px"),
    tags$p("Scatterplot of five random numbers", class = "sr-only")
    # The "sr-only" class is commonly used for screen readers
  )
)

server <- function(input, output, session) {
  output$scatterplot <- renderPlot({
    plot(runif(5), runif(5), xlab = "X-axis", ylab = "Y-axis", main = "Scatterplot")
  })
}

shinyApp(ui, server)
```






Q3

```
library(DT)

ui <- fluidPage(
  DTOutput("table")
)

server <- function(input, output, session) {
  output$table <- renderDT(
    mtcars,
    options = list(
      dom = 't',
      paging = FALSE
    )
  )
}

shinyApp(ui, server)
```

	mpg 	cyl 	disp 	hp 	drat 	wt 	qsec 	vs 	am 	gear 	carb 
Mazda RX4	21	6	160	110	3.9	2.62	16.46	0	1	4	4
Mazda RX4 Wag	21	6	160	110	3.9	2.875	17.02	0	1	4	4
Datsun 710	22.8	4	108	93	3.85	2.32	18.61	1	1	4	1
Hornet 4 Drive	21.4	6	258	110	3.08	3.215	19.44	1	0	3	1
Hornet Sportabout	18.7	8	360	175	3.15	3.44	17.02	0	0	3	2
Valiant	18.1	6	225	105	2.76	3.46	20.22	1	0	3	1
Duster 360	14.3	8	360	245	3.21	3.57	15.84	0	0	3	4
Merc 240D	24.4	4	146.7	62	3.69	3.19	20	1	0	4	2
Merc 230	22.8	4	140.8	95	3.92	3.15	22.9	1	0	4	2
Merc 280	19.2	6	167.6	123	3.92	3.44	18.3	1	0	4	4

Q4

```
library(reactable)

ui <- fluidPage(
  reactableOutput("table")
)

server <- function(input, output, session) {
  output$table <- renderReactable({
    reactable(
      mtcars,
      searchable = FALSE,
      sortable = FALSE,
      pagination = FALSE
    )
  })
}

shinyApp(ui, server)
```

	mpg	cyl	disp	hp	drat	wt	qsec	\
Mazda RX4	21	6	160	110	3.9	2.62	16.46	
Mazda RX4 Wag	21	6	160	110	3.9	2.875	17.02	
Datsun 710	22.8	4	108	93	3.85	2.32	18.61	
Hornet 4 Drive	21.4	6	258	110	3.08	3.215	19.44	
Hornet Sportabout	18.7	8	360	175	3.15	3.44	17.02	
Valiant	18.1	6	225	105	2.76	3.46	20.22	
Duster 360	14.3	8	360	245	3.21	3.57	15.84	
Merc 240D	24.4	4	146.7	62	3.69	3.19	20	
Merc 230	22.8	4	140.8	95	3.92	3.15	22.9	

3.3.6 Exercises

Q1

```
# server 1
server1 <- function(input, output, session) {
  output$greeting <- renderText({
    paste0("Hello ", input$name)
  })
}

# server 2
server2 <- function(input, output, session) {
  output$greeting <- renderText({
    paste0("Hello ", input$name)
  })
}

# server 3
server3 <- function(input, output, session) {
  output$greeting <- renderText({
    paste0("Hello ", input$name)
  })
}
```

Q2

server 1

[inputa] → c() → [inputb] → e() → outputf[inputd] →

server 2

[inputx1] → [inputx2] → x() → [inputx3] → out putz [inputy1] → [inputy2] → y() →

server 3

[inputa] → a() → [inputb] → b() → [inputc] → c() → [inputd]
 → d()

Q3

1. range() is a base R function for computing the range of values. By using the same name for a reactive, you overwrite this function in the current environment. This can cause conflicts and errors when Shiny tries to evaluate range().
2. Similarly, var() is a base R function to compute variance. Overwriting this function leads to confusion and unexpected behavior.
3. Reactives with the same name as base R functions make it harder to debug and trace dependencies because they mask the original functions.

4.8 Exercises

Q1

app 1

```

[input
code] ──▶ selected() ──▶ summary() | _____
diag  ◀── count_top(selected(), diag) output age, ex(plot) output body_part ◀── count_top(selected(), body_part)
output$location ◀── count_top(selected(), location)

```

app 2

```

[inputcode] ──▶ selected() ──▶ summary()[inputy] _____ | output
diag  ◀── count_top(selected(), diag) output body_part ◀── count_top(selected(), body_part) output
location ◀── count_top(selected(), location) output age_sex ◀── ggplot(summary(), aes(age, input$y))

```

app 3

```

[inputcode] ──▶ selected() ──▶ summary()[inputstory] ──▶ narrative_sample ◀── selected() | output$narrative

```

Q2

If `fct_lump()` comes first: Infrequent levels are grouped into "Other" before ordering by frequency. This means the "Other" category will not reflect its frequency compared to other levels.

If `fct_infreq()` comes first: Levels are ordered by frequency before collapsing infrequent levels into "Other." This ensures that "Other" represents the sum of the least frequent levels and is placed appropriately in the frequency order.

Q3

```

ui <- fluidPage( fluidRow( column(6, selectInput("code", "Product", choices = prod_codes)), column(6, numericInput("nrows", "Number of rows:",
value = 5, min = 1, max = 20)) ), fluidRow( column(4, tableOutput("diag")), column(4, tableOutput("body_part")), column(4, tableOutput("location"))
))

```

```

server <- function(input, output, session) { selected <- reactive(injuries %>% filter(prod_code == input$code))

```

```

outputdiag <- renderTable(count_top(selected(), diag, n = inputnrows), width = "100%") output
body_part <- renderTable(count_top(selected(), body_part, n = inputnrows), width = "100%") output
location <- renderTable(count_top(selected(), location, n = inputnrows), width = "100%") }

```

```

shinyApp(ui, server)

```

Q4

```

ui <- fluidPage(
  fluidRow(
    column(2, actionButton("prev", "Previous")),
    column(2, actionButton("next_btn", "Next")),
    column(8, textOutput("narrative"))
  )
)

server <- function(input, output, session) {
  selected <- reactive(injuries %>% filter(prod_code == input$code))
  narratives <- reactive(selected() %>% pull(narrative))
  index <- reactiveVal(1)

  observeEvent(input$next_btn, {
    new_index <- index() + 1
    if (new_index > length(narratives())) {
      new_index <- 1
    }
    index(new_index)
  })

  observeEvent(input$prev, {
    new_index <- index() - 1
    if (new_index < 1) {
      new_index <- length(narratives())
    }
    index(new_index)
  })

  output$narrative <- renderText({
    narratives()[index()]
  })
}

shinyApp(ui, server)

```

Previous

Next

compare Hadley1 and Hadley2

compare UI code

```
diffobj::diffChr(readLines("Hadley_1/ui.R"), readLines("Hadley_2/ui.R"))
```

compare server logic

```
diffobj::diffChr(readLines("Hadley_1/server.R"), readLines("Hadley_2/server.R"))
```

run both apps

```
shiny::runApp("Hadley_1") shiny::runApp("Hadley_2")
```