615 Shiny

Ziheng Li

2024-11-15

```
#install.packages("shiny")
library(shiny)
```

```
#library(shiny)
ui <- fluidPage(
  "Hello, world!"
)
server <- function(input, output, session) {
}
shinyApp(ui, server)
```

Hello, world!

```
#Hadley_1
ui <- fluidPage(
  selectInput("dataset", label = "Dataset", choices = ls("package:datasets")),
  verbatimTextOutput("summary"),
  tableOutput("table")
)
server <- function(input, output, session) {
  output$summary <- renderPrint({
    dataset <- get(input$dataset, "package:datasets")
    summary(dataset)
  })

  output$table <- renderTable({
    dataset <- get(input$dataset, "package:datasets")
    dataset
  })
}
shinyApp(ui, server)
```

Dataset

ability.cov

```
       Length Class  Mode
cov    36     -none- numeric
center  6     -none- numeric
n.obs   1     -none- numeric
```

cov.general	cov.picture	cov.blocks	cov.maze	cov.reading	cov.vocab	center	n.c
24.64	5.99	33.52	6.02	20.75	29.70	0.00	112
5.99	6.70	18.14	1.78	4.94	7.20	0.00	112
33.52	18.14	149.83	19.42	31.43	50.75	0.00	112
6.02	1.78	19.42	12.71	4.76	9.07	0.00	112
20.75	4.94	31.43	4.76	52.60	66.76	0.00	112
29.70	7.20	50.75	9.07	66.76	135.29	0.00	112

```
#Hadley_2

ui <- fluidPage(
  selectInput("dataset", label = "Dataset", choices = ls("package:datasets")),
  verbatimTextOutput("summary"),
  tableOutput("table")
)
server <- function(input, output, session) {
  # Create a reactive expression
  dataset <- reactive({
    get(input$dataset, "package:datasets")
  })

  output$summary <- renderPrint({
    # Use a reactive expression by calling it like a function
    summary(dataset())
  })

  output$table <- renderTable({
    dataset()
  })
}
shinyApp(ui, server)
```

Dataset

ability.cov

```
       Length Class  Mode
cov    36     -none- numeric
center  6     -none- numeric
n.obs   1     -none- numeric
```

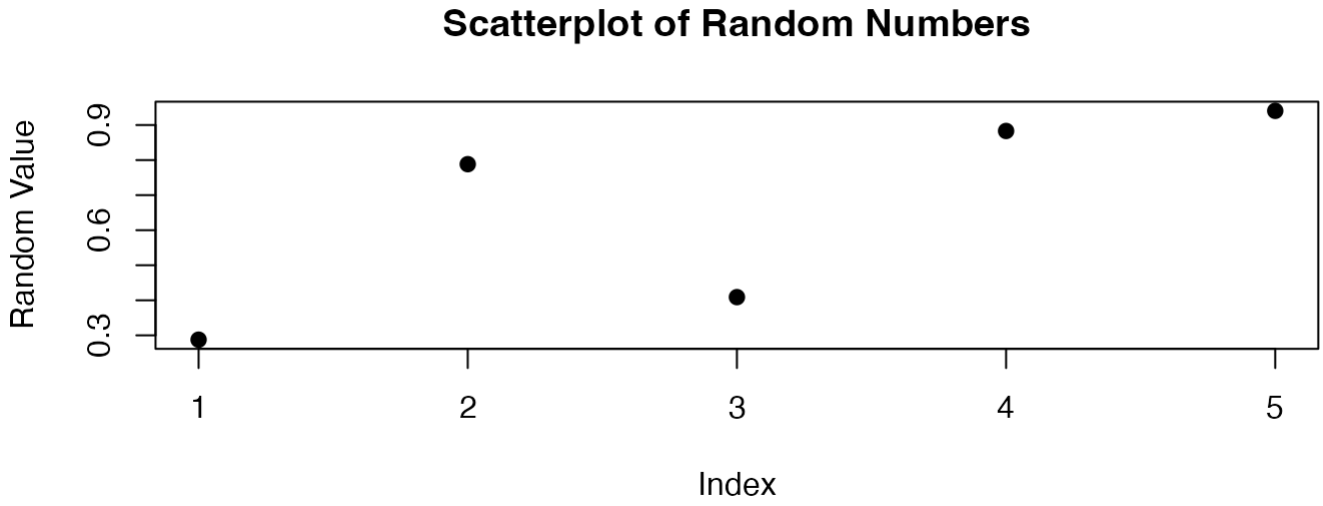
cov.general	cov.picture	cov.blocks	cov.maze	cov.reading	cov.vocab	center	n.c
24.64	5.99	33.52	6.02	20.75	29.70	0.00	112
5.99	6.70	18.14	1.78	4.94	7.20	0.00	112
33.52	18.14	149.83	19.42	31.43	50.75	0.00	112
6.02	1.78	19.42	12.71	4.76	9.07	0.00	112
20.75	4.94	31.43	4.76	52.60	66.76	0.00	112
29.70	7.20	50.75	9.07	66.76	135.29	0.00	112

The difference between the two is that we use a new mechanism: reaction expressions. Duplicating code is bad practice in any type of programming; it wastes computational resources and, more importantly, makes it harder to maintain or debug the code.

2.3.5 #1 renderText(str(lm(mpg ~ wt, data = mtcars)))

#2

```
ui <- fluidPage(

  plotOutput("plot", width = "700px", height = "300px")
)

server <- function(input, output, session) {
  output$plot <- renderPlot({

    set.seed(123)
    random_numbers <- runif(5)


    plot(1:5, random_numbers, main = "Scatterplot of Random Numbers",
         xlab = "Index", ylab = "Random Value", pch = 19)
  }, res = 96)

  # Provide a textual description of what the plot shows
  output$plotDesc <- renderText({
    "This scatterplot displays five randomly generated numbers plotted against their
respective indices from 1 to 5. Each point represents the value of a random number at
that position."
  })
}

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



#3

```
library(DT)
```

```
##
## Attaching package: 'DT'
```

```
## The following objects are masked from 'package:shiny':
##
##     dataTableOutput, renderDataTable
```

```
ui <- fluidPage(
  DTOutput("table")  # Use DTOutput from the DT package
)

server <- function(input, output, session) {
  output$table <- renderDT({
    mtcars
  }, options = list(
    pageLength = 5,
    searching = FALSE,
    ordering = FALSE,
    lengthChange = FALSE,
    info = FALSE,
    paging = FALSE
  ))
}


shinyApp(ui = ui, server = 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

#4

```
#install.packages("reactable")
library(reactable)

reactable(iris)
```

Sepal.Length	Sepal.Width	Petal.Length	Petal.Width	Species
5.1	3.5	1.4	0.2	setosa
4.9	3	1.4	0.2	setosa
4.7	3.2	1.3	0.2	setosa
4.6	3.1	1.5	0.2	setosa
5	3.6	1.4	0.2	setosa
5.4	3.9	1.7	0.4	setosa
4.6	3.4	1.4	0.3	setosa
5	3.4	1.5	0.2	setosa
4.4	2.9	1.4	0.2	setosa
4.9	3.1	1.5	0.1	setosa

1–10 of 150 rows Previous **1** 2 3 4 5 ... 15 Next

```
library(shiny)
library(reactable)

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

server <- function(input, output) {
  output$table <- renderReactable({
    reactable(iris)
  })
}

shinyApp(ui, server)
```

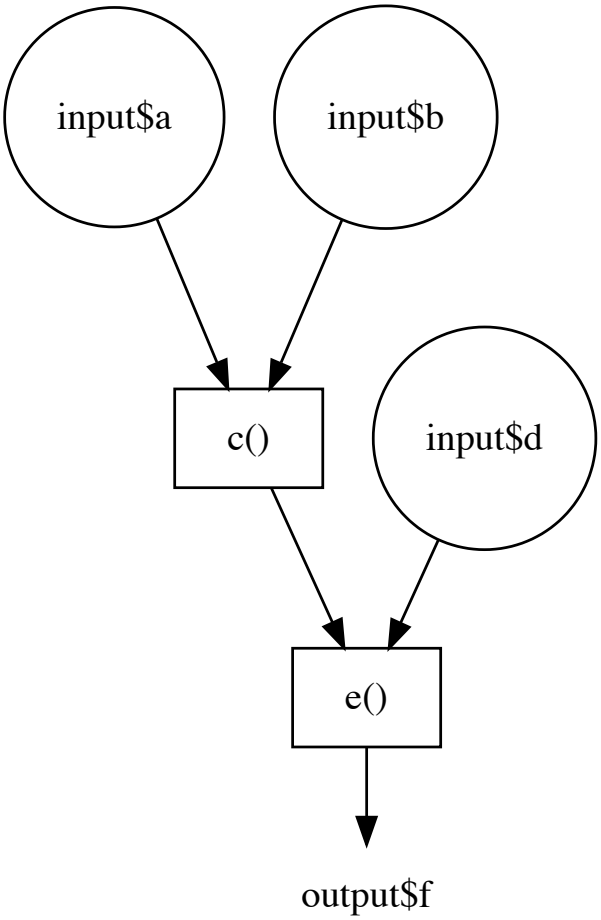
Sepal.Length	Sepal.Width	Petal.Length	Petal.Width	Species
5.1	3.5	1.4	0.2	setosa
4.9	3	1.4	0.2	setosa
4.7	3.2	1.3	0.2	setosa
4.6	3.1	1.5	0.2	setosa
5	3.6	1.4	0.2	setosa
5.4	3.9	1.7	0.4	setosa
4.6	3.4	1.4	0.3	setosa
5	3.4	1.5	0.2	setosa
4.4	2.9	1.4	0.2	setosa
4.9	3.1	1.5	0.1	setosa

3.3.6

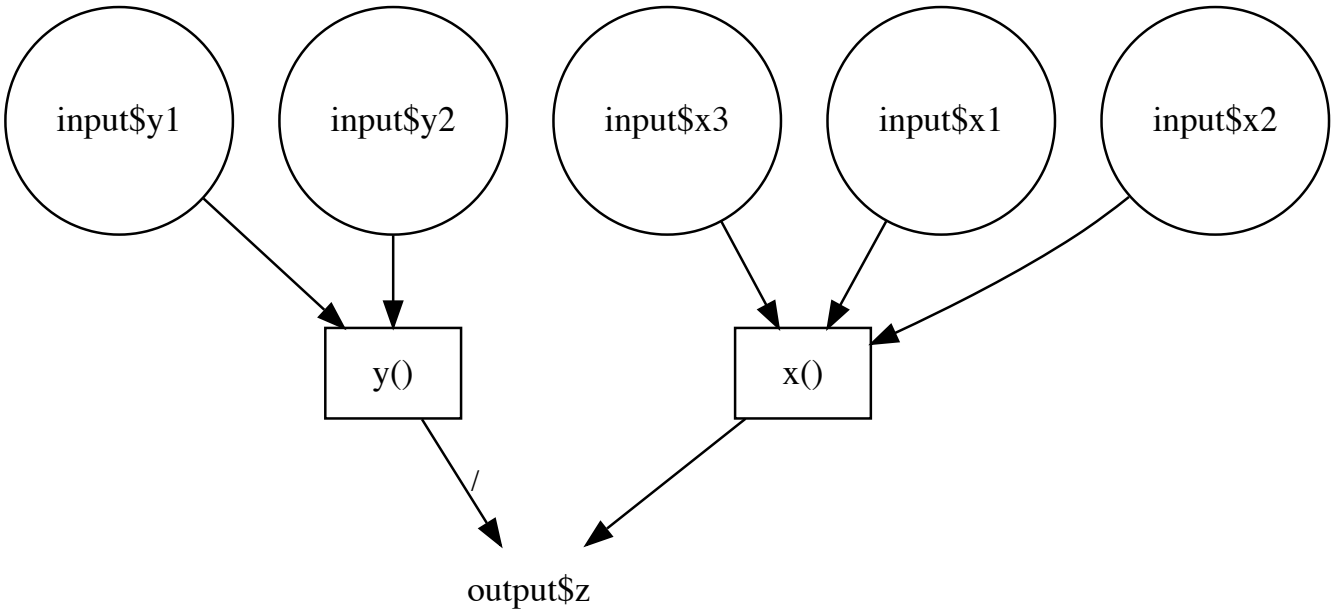
```
ui <- fluidPage(
  textInput("name", "What's your name?"),
  textOutput("greeting")
)
```

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

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

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

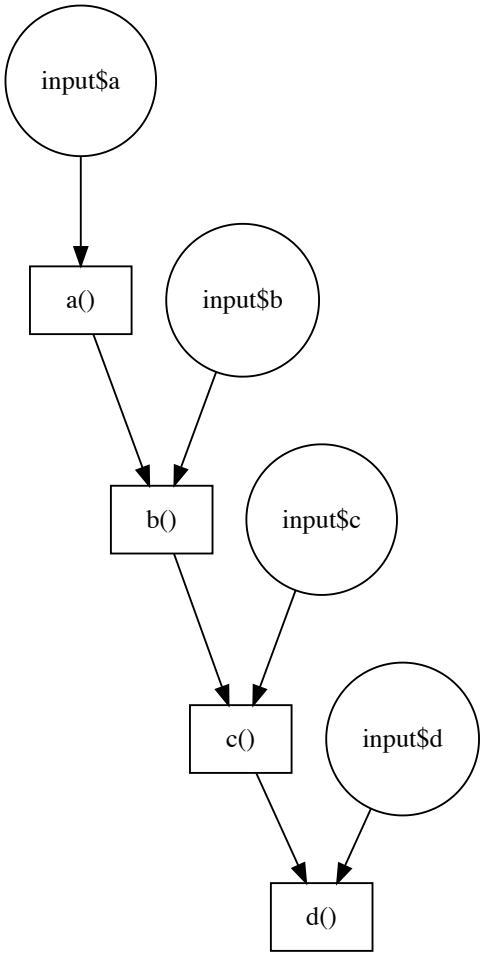
```
#install.packages("DiagrammeR")
library(DiagrammeR)


graph1 <- grViz("
digraph server1 {
  graph [layout = dot]

  node [shape = circle]
  input_a [label = 'input$a']
  input_b [label = 'input$b']
  input_d [label = 'input$d']

  node [shape = box]
  c [label = 'c()']
  e [label = 'e()']

  node [shape = plaintext]
  output_f [label = 'output$f']

  input_a -> c
  input_b -> c
  c -> e
  input_d -> e
  e -> output_f
}
")

graph1
```


input$a
input$b
c()
input$d
e()
output$f

```
graph2 <- grViz("
digraph server2 {
  graph [layout = dot]

  node [shape = circle]
  input_x1 [label = 'input$x1']
  input_x2 [label = 'input$x2']
  input_x3 [label = 'input$x3']
  input_y1 [label = 'input$y1']
  input_y2 [label = 'input$y2']

  node [shape = box]
  x [label = 'x()']
  y [label = 'y()']

  node [shape = plaintext]
  output_z [label = 'output$z']

  input_x1 -> x
  input_x2 -> x
  input_x3 -> x
  input_y1 -> y
  input_y2 -> y
  x -> output_z
  y -> output_z [label = '/', fontsize = 12]
}
")

graph2
```




```
graph3 <- grViz("
digraph server3 {
  graph [layout = dot]

  node [shape = circle]
  input_a [label = 'input$a']
  input_b [label = 'input$b']
  input_c [label = 'input$c']
  input_d [label = 'input$d']

  node [shape = box]
  a [label = 'a()']
  b [label = 'b()']
  c [label = 'c()']
  d [label = 'd()']

  input_a -> a
  a -> b
  input_b -> b
  b -> c
  input_c -> c
  c -> d
  input_d -> d
}
")

graph3
```



#3.3.6 3 The reason the code may fail or cause confusion is due to the use of var and range as names for reactive expressions in Shiny, and using these names for reactive expressions can lead to name conflicts and unexpected behavior

#4.8

```
ui <- fluidPage(
    numericInput("num1", "Enter number 1:", 1),
    numericInput("num2", "Enter number 2:", 1),
    textOutput("sum"),
    textOutput("product")
)

server <- function(input, output, session) {
    sum <- reactive({
        input$num1 + input$num2
    })

    product <- reactive({
        input$num1 * input$num2
    })

    output$sum <- renderText({
        paste("Sum:", sum())
    })

    output$product <- renderText({
        paste("Product:", product())
    })
}
shinyApp(ui, server)
```

Enter number 1:

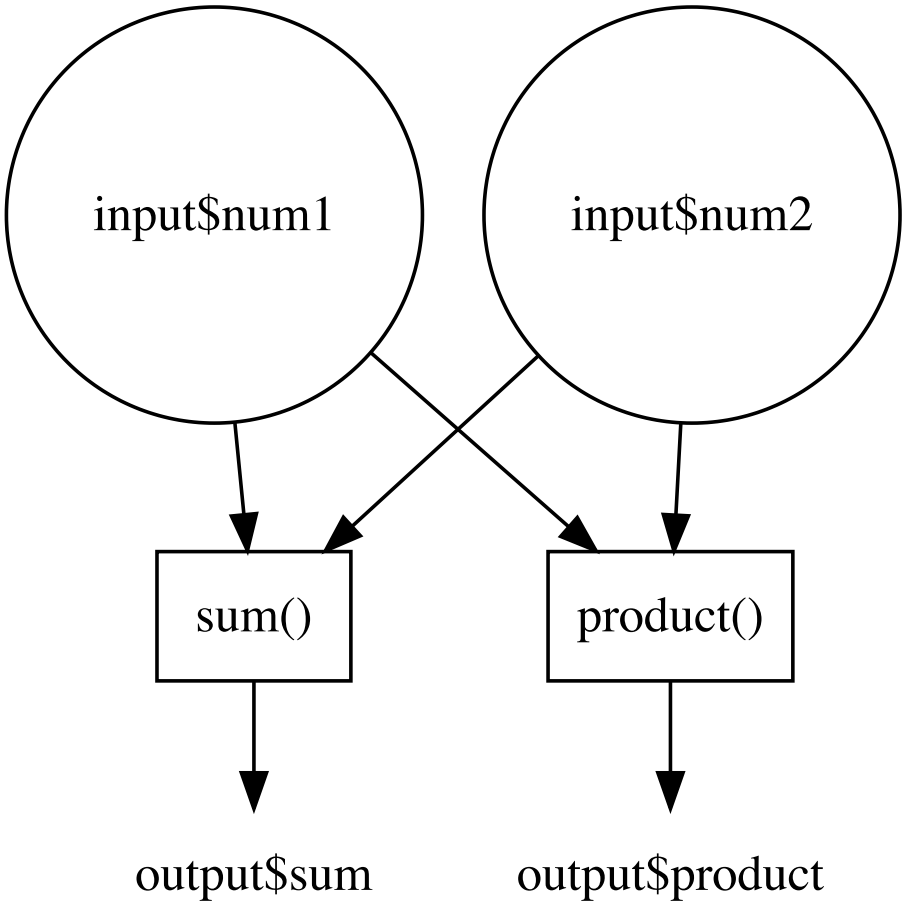
1

Enter number 2:

1

Sum: 2
Product: 1

```
library(DiagrammeR)

grViz("
digraph shiny_app {
  graph [layout = dot]

  node [shape = circle]
  input_num1 [label = 'input$num1']
  input_num2 [label = 'input$num2']

  node [shape = box]
  sum [label = 'sum()']
  product [label = 'product()']

  node [shape = plaintext]
  output_sum [label = 'output$sum']
  output_product [label = 'output$product']

  input_num1 -> sum
  input_num2 -> sum
  input_num1 -> product
  input_num2 -> product
  sum -> output_sum
  product -> output_product
}
")
```



#4.8

```
#library(shiny)
#library(dplyr)

ui <- fluidPage(
  titlePanel("Dynamic Row Number in Data Tables"),

  # Select input to choose number of rows to display
  selectInput("nRows", "Number of Rows:",
              choices = c(5, 10, 15, 20, 25, 50, 100), selected = 10),

  # Table output
  DTOutput("dataTable")
)

server <- function(input, output) {
  # Render the data table
  output$dataTable <- renderDT({
    datatable(
      iris,  # Using the built-in iris dataset for demonstration
      options = list(pageLength = as.numeric(input$nRows))
    )
  })
}

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

Dynamic Row Number in Data Tables

Number of Rows:

10

Show 10 **entries** **Search:**

	Sepal.Length	Sepal.Width	Petal.Length	Petal.Width	Species
1	5.1	3.5	1.4	0.2	setosa
2	4.9	3	1.4	0.2	setosa
3	4.7	3.2	1.3	0.2	setosa
4	4.6	3.1	1.5	0.2	setosa
5	5	3.6	1.4	0.2	setosa

```
ui <- fluidPage(
    titlePanel("Navigate Narratives"),
    actionButton("prev", "Previous", icon = icon("arrow-left")),
    actionButton("next", "Next", icon = icon("arrow-right")),
    textOutput("narrative")
)

server <- function(input, output, session) {
    narratives <- c(
        "Narrative One: The beginning of our journey.",
        "Narrative Two: The challenge emerges.",
        "Narrative Three: Overcoming obstacles.",
        "Narrative Four: The triumphant conclusion."
    )

    narrative_index <- reactiveVal(1)

    observeEvent(input$Next, {
        current_index <- narrative_index()
        if (current_index >= length(narratives)) {
            narrative_index(1)  # Wrap around to the first narrative
        } else {
            narrative_index(current_index + 1)
        }
    })

    observeEvent(input$prev, {
        current_index <- narrative_index()
        if (current_index <= 1) {
            narrative_index(length(narratives))  # Wrap to the last narrative
        } else {
            narrative_index(current_index - 1)
        }
    })

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

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

Navigate Narratives

← Previous → Next

Narrative One: The beginning of our journey.