Building Web Applications with Shiny in R

for Exercises

Eszter Katalin Bognar

09-06-2020

Get Started with Shiny

App input (UI)

```
# use the shiny package
library(shiny)
#Add a text input to the UI called "name" along with an informative label for the user.
ui <- fluidPage(
    textInput("name","Enter a name:")
)
server <- function(input, output) {
}
shinyApp(ui = ui, server = server)
```

App output (UI/Server)

```
ui <- fluidPage(
    textInput("name", "What is your name?"),
    # Display the text output, greeting
  textOutput("greeting")
)
server <- function(input, output) {
    # Render a text output, greeting
    output$greeting <- renderText({paste("Hello ",input$name)

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

Using plots

```
library("ggplot2")
ui <- fluidPage(
```

```
  textInput('name', 'Enter Name', 'David'),
  # Display the plot output named 'trend'
  plotOutput("trend")
)
server <- function(input, output, session) {
  # Render an empty plot and assign to output named 'trend'
  output$trend <- renderPlot({ggplot()})
}
shinyApp(ui = ui, server = server)
```

Using layout functions

```
library("babynames")
ui <- fluidPage(
  titlePanel("Baby Name Explorer"),
  sidebarLayout(
    sidebarPanel(textInput('name', 'Enter Name', 'David')),
    mainPanel(plotOutput('trend'))
  )
)
server <- function(input, output, session) {
  output$trend <- renderPlot({
    # CODE BELOW: Update to display a line plot of the input name

    ggplot(subset(babynames, input$foo)) +
    geom_line(aes(x = year, y = prop, color = sex))

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

Inputs, Outputs, and Layouts

selectInput

```
ui <- fluidPage(
  titlePanel("What's in a Name?"),
  # Add select input named "sex" to choose between "M" and "F"
  selectInput("sex", "Male or female?", selected = "F", choices = c("F", "M")),
  # Add plot output to display top 10 most popular names
  plotOutput('plot_top_10_names')
)

server <- function(input, output, session){
  # Render plot of top 10 most popular names
  output$plot_top_10_names <- renderPlot({
    # Get top 10 names by sex and year
    top_10_names <- babynames %>%
```

```
      # Filter for the selected sex
      filter(sex == input$sex) %>%
      filter(year == 1900) %>%
      top_n(10, prop)
    # Plot top 10 names by sex and year
    ggplot(top_10_names, aes(x = name, y = prop)) +
      geom_col(fill = "#263e63")
  })
}

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

sliderInput

```
ui <- fluidPage(
  titlePanel("What's in a Name?"),
  # Add select input named "sex" to choose between "M" and "F"
  selectInput('sex', 'Select Sex', choices = c("F", "M")),
  # Add slider input named 'year' to select years  (1900 - 2010)
  sliderInput("year","Select year",value=1900,min=1900,max=2010),
  # Add plot output to display top 10 most popular names
  plotOutput('plot_top_10_names')
)

server <- function(input, output, session){
  # Render plot of top 10 most popular names
  output$plot_top_10_names <- renderPlot({
    # Get top 10 names by sex and year
    top_10_names <- babynames %>%
      filter(sex == input$sex) %>%
    # Filter for the selected year
      filter(year == input$year) %>%
      top_n(10, prop)
    # Plot top 10 names by sex and year
      ggplot(top_10_names, aes(x = name, y = prop)) +
        geom_col(fill = "#263e63")
  })
}

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

tableOutput

```
ui <- fluidPage(
  titlePanel("What's in a Name?"),
  # Add select input named "sex" to choose between "M" and "F"
  selectInput('sex', 'Select Sex', choices = c("F", "M")),
  # Add slider input named "year" to select year between 1900 and 2010
  sliderInput('year', 'Select Year', min = 1900, max = 2010, value = 1900),
```

```
  # Add table output named "table_top_10_names"
  tableOutput("table_top_10_names")
)
server <- function(input, output, session){
  # Function to create a data frame of top 10 names by sex and year
  top_10_names <- function(){
    top_10_names <- babynames %>%
      filter(sex == input$sex) %>%
      filter(year == input$year) %>%
      top_n(10, prop)
  }
  # Render a table output named "table_top_10_names"
  output$table_top_10_names <- renderTable({top_10_names()})

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

interactive tableOutput

```
library("DT")
ui <- fluidPage(
  titlePanel("What's in a Name?"),
  # Add select input named "sex" to choose between "M" and "F"
  selectInput('sex', 'Select Sex', choices = c("M", "F")),
  # Add slider input named "year" to select year between 1900 and 2010
  sliderInput('year', 'Select Year', min = 1900, max = 2010, value = 1900),
  # Add a DT output named "table_top_10_names"
  DT::DTOutput('table_top_10_names')
)
server <- function(input, output, session){
  top_10_names <- function(){
    babynames %>%
      filter(sex == input$sex) %>%
      filter(year == input$year) %>%
      top_n(10, prop)
  }
  # Render a DT output named "table_top_10_names"
  output$table_top_10_names <- DT::renderDT({
    top_10_names()
  })
}
shinyApp(ui = ui, server = server)
```

interactive plot output

```
library("plotly")
ui <- fluidPage(
  selectInput('name', 'Select Name', top_trendy_names$name),
  # Add a plotly output named 'plot_trendy_names'
```

```
  plotly::plotlyOutput("plot_trendy_names")
)
server <- function(input, output, session){
  # Function to plot trends in a name
  plot_trends <- function(){
     babynames %>%
      filter(name == input$name) %>%
      ggplot(aes(x = year, y = n)) +
      geom_col()
  }
  # Render a plotly output named 'plot_trendy_names'
  output$plot_trendy_names <- plotly::renderPlotly({plot_trends()})

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

Sidebar layouts

```
ui <- fluidPage(
  # Wrap in a sidebarLayout
  sidebarLayout(
    # Wrap in a sidebarPanel
    sidebarPanel(
    selectInput('name', 'Select Name', top_trendy_names$name)),
    # Wrap in a mainPanel
    mainPanel(
    plotly::plotlyOutput('plot_trendy_names'),
    DT::DTOutput('table_trendy_names'))
))
```

Tab layouts

```
ui <- fluidPage(
  sidebarLayout(
    sidebarPanel(
      selectInput('name', 'Select Name', top_trendy_names$name)
    ),
    mainPanel(
      # Wrap in a tabsetPanel
      tabsetPanel(
        # Wrap in a tabPanel providing an appropriate label
        tabPanel("Plot",
        plotly::plotlyOutput('plot_trendy_names')),
        # Wrap in a tabPanel providing an appropriate label
        tabPanel("Table",
        DT::DTOutput('table_trendy_names'))
    ))
  )
)
```

Themes

```
ui <- fluidPage(
  # CODE BELOW: Add a titlePanel with an appropriate title
  titlePanel("Title"),
  # REPLACE CODE BELOW: with theme = shinythemes::shinytheme("<your theme>")
  theme <- shinythemes::shinytheme('superhero'),
  sidebarLayout(
    sidebarPanel(
      selectInput('name', 'Select Name', top_trendy_names$name)
    ),
    mainPanel(
      tabsetPanel(
        tabPanel('Plot', plotly::plotlyOutput('plot_trendy_names')),
        tabPanel('Table', DT::DTOutput('table_trendy_names'))
      )
    )
  )
)
```

Building apps

App 1

```
ui <- fluidPage(
  selectInput("language","Select language",selected='Hello',choices=c("Hello","Bonjour")),
  textInput('name', 'Enter Name', 'David'),
  textOutput("greeting")
)

server <- function(input, output, session) {
  output$greeting <- renderText({paste(input$language,input$name)
})}

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

App 2

```
ui <- fluidPage(
  titlePanel("Baby Name Explorer"),
  sidebarLayout(
    sidebarPanel(
      selectInput("sex","Select sex",selected='F',choices=c("M","F")),
      sliderInput("year","Select year",value=1880,min=1880,max=2017)
    ),
    mainPanel(
      tabsetPanel(
        tabPanel("Plot",plotOutput('plot')),
```

```
        tabPanel("Table",tableOutput("table"))
      )
    )
  )
)

server <- function(input, output, session) {
  output$plot <- renderPlot({
    # Update to display a line plot of the input name
    data = babynames %>%
    filter(year=input$year) %>%
    filter(sex=input$sex)
    ggplot(data,aes(x = year, y = sex)) +
    geom_bar()
  })
  output$table <- renderTable({
    get_top_names(input=input$year, sex=input$sex)
  })
}
shinyApp(ui = ui, server = server)
```

Reactive Programming

Reactive expression

```
server <- function(input, output, session) {
  rval_bmi <- reactive({
    input$weight/(input$height^2)
  })
  # Add a reactive expression rval_bmi_status to
  rval_bmi_status <- reactive({
    cut(rval_bmi(),
      breaks = c(0, 18.5, 24.9, 29.9, 40),
      labels = c('underweight', 'healthy', 'overweight', 'obese')
    )
  })
  # return health status as underweight etc. based on inputs
  output$bmi <- renderText({
    bmi <- rval_bmi()
    paste("Your BMI is", round(bmi, 1))
  })
  output$bmi_status <- renderText({
    # Replace right-hand-side with
    # reactive expression rval_bmi_status
    bmi_status <- rval_bmi_status()
    paste("You are", bmi_status)
  })
}
```

Add an observer to display notifications

```
ui <- fluidPage(
  textInput('name', 'Enter your name')
)

server <- function(input, output, session) {
  # Add an observer to display a notification
  # 'You have entered the name xxxx' where xxxx is the name
  observe({
    showNotification(
      paste("You have entered the name",input$name)
    )
  })
}

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

Stop reactions with isolate()

```
server <- function(input, output, session) {
  rval_bmi <- reactive({
    input$weight/(input$height^2)
  })
  output$bmi <- renderText({
    bmi <- rval_bmi()
    # Use isolate to stop output from updating when name changes.
    paste("Hi", isolate({input$name}), ". Your BMI is", round(bmi, 1))
  })
}
```

Delay reactions with eventReactive()

```
server <- function(input, output, session) {
  # Use eventReactive to delay the execution of the
  # calculation until the user clicks on the show_bmi button (Show BMI)
  rval_bmi <- eventReactive(input$show_bmi,{
    input$weight/(input$height^2)
  })
  output$bmi <- renderText({
    bmi <- rval_bmi()
    paste("Hi", input$name, ". Your BMI is", round(bmi, 1))
  })
}
```

Trigger reactions with observeEvent()

```
server <- function(input, output, session) {
  # Wrap in observeEvent() so the help text
  # is displayed when a user clicks on the Help button.
  observeEvent(input$show_help, {
     # Display a modal dialog with bmi_help_text
     showModal(modalDialog(bmi_help_text))
  })
  rv_bmi <- eventReactive(input$show_bmi, {
    input$weight/(input$height^2)
  })
  output$bmi <- renderText({
    bmi <- rv_bmi()
    paste("Hi", input$name, ". Your BMI is", round(bmi, 1))
  })
}
```

Sample Dashboards

Explore the Mental Health in Tech 2014 Survey

```
ui <- fluidPage(
  # Add an appropriate title
  titlePanel("2014 Mental Health in Tech Survey"),
  sidebarPanel(
    # Add a checkboxGroupInput
    checkboxGroupInput(
      inputId = "mental_health_consequence",
      label = "Do you think that discussing a mental health issue with your employer would have negativ
      choices = c("Maybe", "Yes", "No"),
      selected = "Maybe"
    ),
    # Add a pickerInput
    pickerInput(
      inputId = "mental_vs_physical",
      label = "Do you feel that your employer takes mental health as seriously as physical health?",
      choices = c("Don't Know", "No", "Yes"),
      multiple = TRUE
    )
  ),
  mainPanel(
    # Display the output
    plotOutput("age")
  )
)

server <- function(input, output, session) {
  # Build a histogram of the age of respondents
  # Filtered by the two inputs
```

```
  output$age <- renderPlot({
    mental_health_survey %>%
      filter(
        mental_health_consequence %in% input$mental_health_consequence,
        mental_vs_physical %in% input$mental_vs_physical
      ) %>%
      ggplot(aes(Age)) +
      geom_histogram()
  })
}

shinyApp(ui, server)
```

Explore cuisines: wordclouds

```
ui <- fluidPage(
  titlePanel('Explore Cuisines'),
  sidebarLayout(
    sidebarPanel(
      selectInput('cuisine', 'Select Cuisine', unique(recipes$cuisine)),
      sliderInput('nb_ingredients', 'Select No. of Ingredients', 5, 100, 20),
    ),
    mainPanel(
      tabsetPanel(
        # Add `d3wordcloudOutput` named `wc_ingredients` in a `tabPanel`
        tabPanel('Word Cloud', d3wordcloud::d3wordcloudOutput('wc_ingredients', height = '400')),
        tabPanel('Plot', plotly::plotlyOutput('plot_top_ingredients')),
        tabPanel('Table', DT::DTOutput('dt_top_ingredients'))
      )
    )
  )
)
server <- function(input, output, session){
  # Render an interactive wordcloud of top distinctive ingredients
  # and the number of recipes they get used in, using
  # `d3wordcloud::renderD3wordcloud`, and assign it to an output named
  # `wc_ingredients`.
  output$wc_ingredients <- d3wordcloud::renderD3wordcloud({
     ingredients_df <- rval_top_ingredients()
     d3wordcloud(ingredients_df$ingredient, ingredients_df$nb_recipes, tooltip = TRUE)
  })
  rval_top_ingredients <- reactive({
    recipes_enriched %>%
      filter(cuisine == input$cuisine) %>%
      arrange(desc(tf_idf)) %>%
      head(input$nb_ingredients) %>%
      mutate(ingredient = forcats::fct_reorder(ingredient, tf_idf))
  })
  output$plot_top_ingredients <- plotly::renderPlotly({
    rval_top_ingredients() %>%
      ggplot(aes(x = ingredient, y = tf_idf)) +
```

```
      geom_col() +
      coord_flip()
  })
  output$dt_top_ingredients <- DT::renderDT({
    recipes %>%
      filter(cuisine == input$cuisine) %>%
      count(ingredient, name = 'nb_recipes') %>%
      arrange(desc(nb_recipes)) %>%
      head(input$nb_ingredients)
  })
}
shinyApp(ui = ui, server= server)
```

Mass shootings

```
ui <- bootstrapPage(
  theme = shinythemes::shinytheme('simplex'),
  leaflet::leafletOutput('map', width = '100%', height = '100%'),
  absolutePanel(top = 10, right = 10, id = 'controls',
    sliderInput('nb_fatalities', 'Minimum Fatalities', 1, 40, 10),
    dateRangeInput(
      'date_range', 'Select Date', "2010-01-01", "2019-12-01"
    ),
    # Add an action button named show_about
    actionButton('show_about', 'About')
  ),
  tags$style(type = "text/css", "
    html, body {width:100%;height:100%}
    #controls{background-color:white;padding:20px;}
  ")
)
server <- function(input, output, session) {
  # Use observeEvent to display a modal dialog
  # with the help text stored in text_about.
  observeEvent(input$show_about, {
    showModal(modalDialog(text_about, title = 'About'))
  })
  output$map <- leaflet::renderLeaflet({
    mass_shootings %>%
      filter(
        date >= input$date_range[1],
        date <= input$date_range[2],
        fatalities >= input$nb_fatalities
      ) %>%
      leaflet() %>%
      setView( -98.58, 39.82, zoom = 5) %>%
      addTiles() %>%
      addCircleMarkers(
        popup = ~ summary, radius = ~ sqrt(fatalities)*3,
        fillColor = 'red', color = 'red', weight = 1
      )
```

```
  })
}

shinyApp(ui, server)
```