Lab 09 Intoduction to Shiny

Lusine Zilfimian

April 15 (Wednesday), 2020

Building web applications with shiny

- https://www.rstudio.org/links/shiny_cheat_sheet
- www.shiny.rstudio.com
- Watch out for commas!:)

First install the shiny library

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

- ui.R **ui** <- **fluidPage()**. User interface: input defined + outputs laid out: Titles, text input, radio buttons, drop-down menus, graphs, etc.
- server.R **server** <- **function(input, output){}** Output calculated and any other calculations needed to run the app. Actually, set of instructions that produce the required output which further displayed by ui.r script.
- **shinyApp (ui = ui, server = server)** = Run the app

Look at the default shinyApp and first run it.

```r
ui <- fluidPage(
    titlePanel("Old Faithful Geyser Data"),
    sidebarLayout(
        sidebarPanel( sliderInput("bins","Number of bins:",
          min = 1,max = 50,value = 30) ),
        mainPanel(plotOutput("distPlot"))
                )
)
server <- function(input, output) {
  output$distPlot <- renderPlot({
   x    <- faithful[, 2]
   bins <- seq(min(x), max(x), length.out = input$bins + 1)
   hist(x, breaks = bins, col = 'darkgray', border='black')})
}

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

```
## PhantomJS not found. You can install it with webshot::install_phantomjs(
```

Shiny applications not supported in static R Markdown documents

Separately:

```
ui <- fluidPage(
    titlePanel("Old Faithful Geyser Data"),
    sidebarLayout(
        sidebarPanel(),
        mainPanel()
                )
)
server <- function(input, output) {}

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

Shiny applications not supported in static R Markdown documents

User interface:

fluidPage(

sidebarLayout(

- sidebarPanel()
- mainPanel()

)

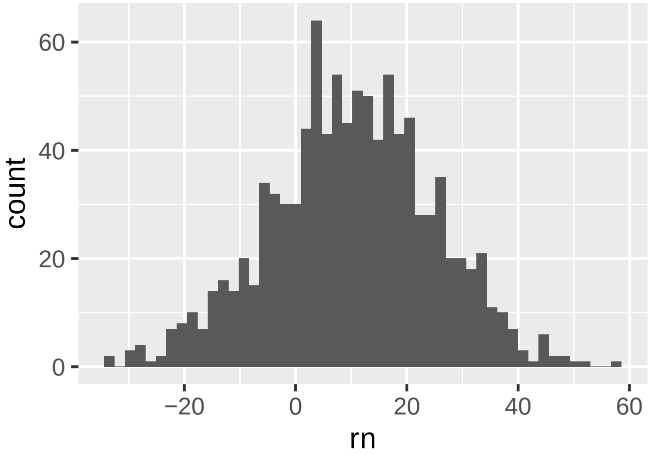
)

sliderInput(inputId = “mu”, label = “Mean of distribution:”, min = 1, max = 100, value = 50)

- inputId – you will use this to refer to the input in server
- label – the label of the slider
- min, max – the minimum and maximum values that the mean can have
- value – the default value

Suppose we are going to create histogram by changing the parameter for mean.

```
set.seed(2708)
rn <- rnorm(n = 1000, mean = 10, sd = 15)
ggplot() + geom_histogram(aes(x = rn), bins = 50)
```



Mean

sliderInput + plotOutput (renderPlot)

```
# ui
ui1 <- fluidPage(
    titlePanel("Normally distributed data"),
    sidebarLayout(
        sidebarPanel(
            sliderInput(inputId = "mu",
                        label = "Mean of distribution:",
                        min = 1,
                        max = 100,
                        value = 50)
        ),
        mainPanel( plotOutput("myhistogram") )
    )
)
```

```r
# Server
server2 <- function(input, output) {

    output$myhistogram <- renderPlot({
      rn <- rnorm(n = 1000,
        # -------------
        mean = input$mu,
        # -------------
        sd = 15)
      ggplot() + geom_histogram(aes(x = rn), bins = 50)
    })
}
shinyApp(ui=ui1, server = server2)
```

Shiny applications not supported in static R Markdown documents

- server takes the value for mean from ui
- to specify input from ui use input$'name of the input'

SD

Add another sliderInput for standard deviation

```r
# ui
ui2 <- fluidPage(
    titlePanel("Normally distributed data"),
    sidebarLayout(
        sidebarPanel(
  sliderInput("mu", "Mean for distribution:",
      min = 1, max = 100, value = 50),
  #------------------------------------------
  sliderInput("mysd", "Standard deviation for distribution:",
      min = 5, max = 50, value = 40)),
  #------------------------------------------
        mainPanel( plotOutput("myhistogram") )
    )
)
```

```r
# Server
server3 <- function(input, output) {

    output$myhistogram <- renderPlot({
      set.seed(2708)
      rn <- rnorm(n = 1000, mean = input$mu,
        # ------------
        sd = input$mysd
        # ------------
        )
      ggplot() + geom_histogram(aes(x = rn), bins = 50)
    })
}

shinyApp(ui=ui2, server = server3)
```

Shiny applications not supported in static R Markdown documents

Size

Add a numeric input for the sample size with min = 100, max = 10000

- numericInput(inputId, label, value, min, max, step)

```r
ui3 <- fluidPage(
    titlePanel("Normally distributed data"),
    sidebarLayout(
        sidebarPanel(
    sliderInput("mu", "Mean for distribution:",
      min = 1, max = 100, value = 50),
    sliderInput("mysd", "Standard deviation for distribution:",
          min = 5, max = 50, value = 40),
    # ------------------------------------------------
    numericInput(inputId = "sizeMy",label="The size of data",
         value = 1000, min = 100, max = 10000, step = 0.5)
    # ------------------------------------------------
          ), mainPanel( plotOutput("myhistogram") )
    ))
```

```r
# Server
server4 <- function(input, output) {

    output$myhistogram <- renderPlot({
      set.seed(2708)
      rn <- rnorm(
        # --------------
        n = input$sizeMy,
        # --------------
        mean = input$mu, sd = input$mysd)
      ggplot() + geom_histogram(aes(x = rn), bins = 50)
    })
}

shinyApp(ui=ui3, server = server4)
```

Shiny applications not supported in static R Markdown documents

Step 1: Add an R object to the UI

Here are the shiny Widgets:

- Action button
- Single checkbox / Checkbox group
- Date input / Date range
- File input
- Numeric input / Text input
- Radio buttons / Select box
- Slider / Slider range
- Help text

Step 2: Create an action in server

Step 3: Return the output to Main Panel

Shiny provides a family of functions that turn R objects into output for your user interface. Each function creates a specific type of output.

- imageOutput - image
- plotOutput - plot
- verbatimTextOutput - text
- tableOutput - table
- etc.

Median

Add median as a text on the Main Panel

```r
uim <- fluidPage(
    titlePanel("Normally distributed data"),
    sidebarLayout(
        sidebarPanel(
    sliderInput("mu", "Mean for distribution:",
      min = 1, max = 100, value = 50),
    sliderInput("mysd","Standard dev for distribution:",
          min = 5, max = 50, value = 40),
    numericInput(inputId = "sizeMy",label="The size of data",
         value = 1000, min = 100, max = 10000, step = 0.5)
          ), mainPanel( plotOutput("myhistogram"),
           #----------------------------------
                textOutput("median_of_data")
           #----------------------------------
          )))
```

```r
# Server
serverm <- function(input, output) {

    output$myhistogram <- renderPlot({
  set.seed(2708)
  rn <- rnorm(n=input$sizeMy,mean=input$mu,sd=input$mysd)
  ggplot() + geom_histogram(aes(x = rn), bins = 50)
    })
    # --------------------------------------------------
    output$median_of_data <- renderText({
      set.seed(2708)
      median(rn <- rnorm(n = input$sizeMy,
        mean = input$mu, sd = input$mysd))
    })
    # --------------------------------------------------
}
shinyApp(ui=uim, server = serverm)
```

Shiny applications not supported in static R Markdown documents

Reactive

- In order to not repeat the code twice (or more) use reactive to make a reactive object, thus an object that reacts to user input
- If you use reactive object in later code, use rn()
- Thus this value will be recalculated only when inputs would change

```
# Server
serverm2 <- function(input, output) { set.seed(2708)
  # ------------------------------------------
    rn <- reactive(rnorm(n = input$sizeMy,
      mean = input$mu, sd = input$mysd))
  # ------------------------------------------
    output$myhistogram <- renderPlot({
      ggplot() + geom_histogram(aes(x = rn()), bins = 50)})
    output$median_of_data <- renderText(median(rn()))
    }
shinyApp(ui=uim, server = serverm2)
```

Shiny applications not supported in static R Markdown documents

Text formating

- paste() to add other text element

```
# Server
serverm3 <- function(input, output) {

    set.seed(2708)
    rn <- reactive(rnorm(n = input$sizeMy, mean = input$mu,
      sd = input$mysd))

    output$myhistogram <- renderPlot(
      ggplot() + geom_histogram(aes(x = rn()), bins = 50)
    )
    output$median_of_data <- renderText(
      paste("Can you see me?",
      round(median(rn())),"this is the median"))}
shinyApp(ui=uim, server = serverm3)
```

Shiny applications not supported in static R Markdown documents

Text formating

Use html headers (h1, h2, h3, etc) to change the size of the text (in UI!)

```r
uim2 <- fluidPage(
    titlePanel("Normally distributed data"),
    sidebarLayout(
        sidebarPanel(
    sliderInput("mu", "Mean for distribution:",
      min = 1, max = 100, value = 50),
    sliderInput("mysd","Standard dev. for distribution:",
          min = 5, max = 50, value = 40),
    numericInput(inputId="sizeMy",label="The size of data",
         value = 1000, min = 100, max = 10000, step = 0.5)
          ),
         mainPanel( plotOutput("myhistogram"),
                h3(textOutput("median_of_data"))
          )))
shinyApp(ui=uim2, server = serverm3)
```

Shiny applications not supported in static R Markdown documents

Thus **render.()** function from server and **.Output()** from UI work together to add R output to the UI

Output function	Creates	Render Function
verbatimTextOutput	any printed output	renderPrint
plotOutput	`plots`	renderPlot
textOutput	character strings	renderText

Full list here

Task 1

```r
uia <- fluidPage(
  sidebarLayout(
    sidebarPanel(
      selectInput(inputId = "first", label="X axis",
                  choices = colnames(anscombe),
                  selected = "x1"),
      selectInput(inputId = "second", label="Y axis",
                  choices = colnames(anscombe),
                  selected = "y1")
                   ),
    mainPanel(
        plotOutput("scatterplot"),
        textOutput("corr"),
        tableOutput("summary")
                         )))
```

- aes_string is used when you have quoted names (“x1” and not x1)
- the input$ format from selectInput is a quoted tex

```
servera <- function(input, output){
  output$scatterplot <- renderPlot({
    ggplot(data=anscombe, aes_string(x=input$first,
      y=input$second))+
      geom_point(size = 2, alpha=0.5, col = "coral")+
      geom_smooth(method = "lm", se = F)
  })
  output$summary <- renderTable({
    anscombe[,c(input$first, input$second)]
    })
  output$corr <- renderText(paste("Cor is",
    cor(anscombe[,c(input$first, input$second)])[1,2]))
 }
shinyApp(ui=uia, server = servera)
```

Shiny applications not supported in static R Markdown documents

Brush

Select points on the plot via brushing, and report the selected points in a data table underneath the plot.

```r
uiab <- fluidPage(
  sidebarLayout(
    sidebarPanel(
      selectInput(inputId = "first", label="X axis",
                  choices = colnames(anscombe),
                  selected = "x1"),
      selectInput(inputId = "second", label="Y axis",
                  choices = colnames(anscombe),
                  selected = "y1")),
    mainPanel(
        plotOutput("scatterplot", brush = "nameofbrush"),
        dataTableOutput("summary"),
      uiOutput("link"))))
```

```r
serverab <- function(input, output){
  output$scatterplot <- renderPlot({
    ggplot(data=anscombe, aes_string(x=input$first,
      y=input$second))+
      geom_point(size = 2, alpha=0.5, col = "coral")+
      geom_smooth(method = "lm", se = F)
  })
  output$summary <- renderDataTable({
    brushedPoints(anscombe, input$nameofbrush)
    })
  output$link <- renderUI({
  tagList(h3(a("Click on me",
    href="https://github.com/zilfimian", target="_blank")))
    })

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

Shiny applications not supported in static R Markdown documents

Grid Layout

- Rows are created by the fluidRow() function and include columns defined by the column() function.
- Column widths are based on the Bootstrap 12-wide grid system, so should add up to 12 within a fluidRow()
- The sum of widths of the columns inside every row needs to be no more than 12

```r
uia <- fluidPage(
#  sidebarLayout( #  sidebarPanel(
    fluidRow(
      column(width = 1,  selectInput(inputId = "first",
        label="X axis",
       choices = colnames(anscombe), selected = "x1")),
     column(width = 1, selectInput(inputId = "second",
       label="Y axis",
        choices = colnames(anscombe), selected = "y1"))
      ), #  <- Do not forget this comma    # mainPanel(
      fluidRow(
        column(4, plotOutput("scatterplot")),
        column(4, textOutput("corr")),
        column(4, dataTableOutput("summary")))
)
shinyApp(ui=uia, server = servera)
```

Shiny applications not supported in static R Markdown documents

Another inputs

```
uia1 <- fluidPage(
  sidebarLayout(
    sidebarPanel(
      selectInput(inputId = "first", label="X axis",
         choices = colnames(anscombe), selected = "x1"),
      selectInput(inputId = "second", label="Y axis",
         choices = colnames(anscombe), selected = "y1"),
      checkboxInput(inputId = "plots", label = "Show plot",
        value = T)
                ),
    mainPanel(plotOutput("scatterplot"))))
shinyApp(ui=uia1, server = servera)
```

Shiny applications not supported in static R Markdown documents

```r
servera1 <- function(input, output){

  output$scatterplot <- renderPlot({
    if(input$plots){
    ggplot(data=anscombe, aes_string(x=input$first,
      y=input$second))+
      geom_point(size = 2, alpha=0.5, col = "coral")+
      geom_smooth(method = "lm", se = F)}
  })
 }
shinyApp(ui=uia1, server = servera1)
```

Shiny applications not supported in static R Markdown documents

Shiny Dashboard

```r
# install.packages("shinydashboard")
library(shinydashboard)
```

- A dashboard has three parts: a **header**, a **sidebar**, and a **body**. Here's the most minimal possible UI for a dashboard page.

```r
ui <- dashboardPage(
  dashboardHeader(),
  dashboardSidebar(),
  dashboardBody()
)
server <- function(input, output) { }
shinyApp(ui, server)
```

Shiny applications not supported in static R Markdown documents

```r
#runExample("01_hello")
```

```r
ui <- dashboardPage(
  dashboardHeader(
    title = h4("Here should be the best dash ever")
    ),
  dashboardSidebar(),
    dashboardBody()
)

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

Shiny applications not supported in static R Markdown documents

- App user inputs are located in dashboardSidebar
- Output is in dashboardBody

```r
ui <- dashboardPage(
  dashboardHeader(
    title = h4("Here should be the best dash ever")
    ),
  dashboardSidebar(sliderInput(inputId = "mu",
                   label = "Mean for distribution:",
                   min = 1,max = 100, value = 50),
   sliderInput(inputId = "sd",
                  label = "SD for distribution:",
                   min = 5,max = 50, value = 40),
  numericInput(inputId="sample", label = "Sample size",
                    min=100, max=10000, value=1000)),
    dashboardBody(plotOutput("histogram"))
)
```

```r
server <- function(input, output){
  output$histogram <- renderPlot({
    rn <- rnorm(n=input$sample, mean=input$mu,
      sd=input$sd)
    ggplot()+geom_histogram(aes(x=rn), bins=50)
  })}

shinyApp(ui, server)
```

Default examples

```
#runExample("01_hello")
```

Using shinyapps.io

When your app is deployed, the computer serving your app is a web server.

Is free, but limited to 5 active applications

1. Register with shinyapps.io
2. Install the following package

```r
#install.packages('rsconnect')
```

3. Login into your page on shinyapps.io
4. Copy token and secret If planning to use an external file, read it Don't forget to upload the data file into server too No need to install libraries, just call them

Other important/fun stuff

- IGV plotly
- IGV like ggplot ggViz
- shiny gallery
- https://adminlte.io/themes/AdminLTE/pages/calendar.html