Web App With Shiny

(Shiny)

2113 Prachi Gore

M.Sc.(Statistics) Department of Statistics
School of Mathematical Sciences
Kavayitri Bahinabai Chaudhari North Maharashtra University, Jalgaon

May 31, 2023

Contents

Shiny

What is Shiny ?

Shiny is an R package that makes it easy to build interactive web applications (apps) straight from R.

How to install Shiny Package

install.packages("shiny")

How to use Shiny Package

library("shiny")

Structure of a Shiny App

Shiny apps are contained in a single script called app.R

app.R has three components:

a user interface object

a server function

a shinyApp function

The user interface ui object controls the layout and appearance of our app. The server function contains the instructions that our computer needs to build app. Finally the shinyApp function

User Interface

Let's Build a User Interface

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

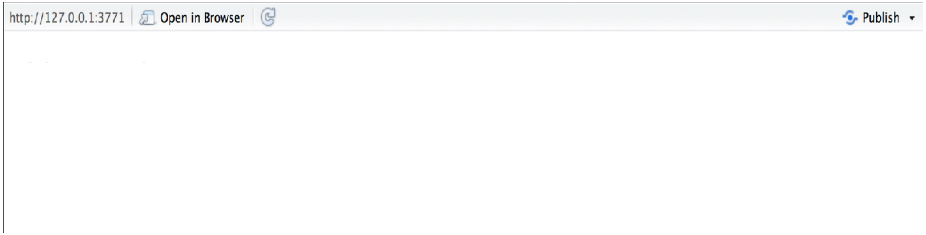


Figure: blank fluidpage

User Interface

Layout

Shiny uses the function fluidPage to create a display that automatically adjusts according to the dimensions of user's browser window. we layout the user interface of our app by placing elements in the fluidPage function.

For example, the ui function below creates a user interface
that has a title panel and a sidebar layout , which includes a sidebar panel and a main panel.
Note that these elements are placed within the fluidPage function.

```
ui = fluidPage(
    titlePanel("title panel"),
    sidebarLayout(
      sidebarPanel("sidebar panel"),
      mainPanel("main panel")
    )
  )
```

User Interface

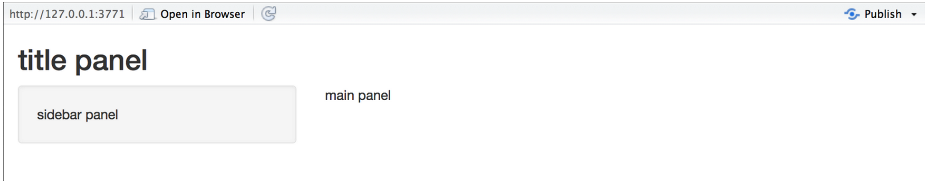


Figure: fluidpage

Server

Display output

```
library(shiny)
ui = fluidPage(
            titlePanel("title panel"),
            sidebarLayout(
                sidebarPanel("sidebar panel"),
                mainPanel(plotOutput('graph'))
                         )
                )
server=function(input,output){
                output$graph=renderPlot({hist(rnorm(100))})
               }
shinyApp(ui, server)
```

Server

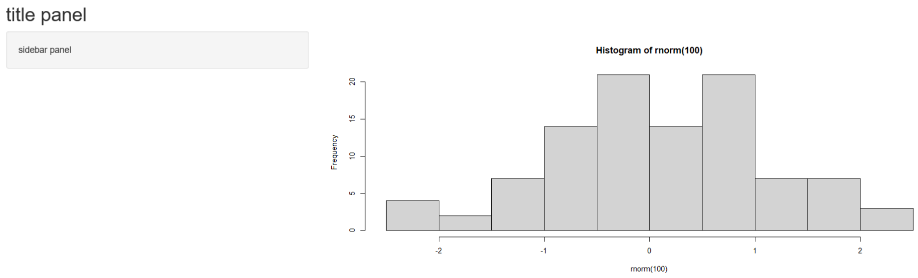


Figure: histogram output

Server

Display reactive output

```
library(shiny)
ui = fluidPage(
     titlePanel("title panel"),
     sidebarLayout(
      sidebarPanel("sidebar panel",
                   numericInput(inputId = "n",label = "Enter Sample
      mainPanel(plotOutput('graph')))
             )
server=function(input,output){
    size= reactive({input$n})
    output$graph=renderPlot({hist(rnorm(size()))})
  }
shinyApp(ui, server)
```

Server

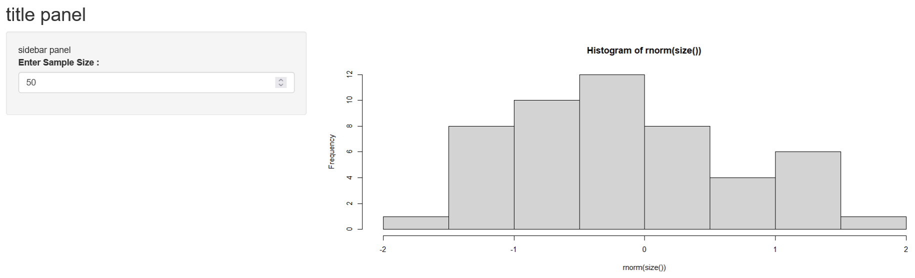


Figure: reactive output

here histogram is depend on sample size and sample size is in user's hand it could be anything 10,30,100,500,...so we will write it in reactive() function Whenever User update the sample size plot will be re render. Now this time entire file will not run again Whenever changes happens only that part will be run again. this is the power of reactive function and it help to improve speed of app.

Shiny Dashboard

Installation

install.packages("shinydashboard")

Layout

A dashboard has three parts. a dashboardHeader( ) , a dashboardSidebar( ) and a dashboard-Body( ) .

```
library(shiny)
library(shinydashboard)
ui = dashboardPage(
dashboardHeader(),
dashboardSidebar(),
dashboardBody()
)
server = function(input, output) { }
shinyApp(ui, server)
```

Shiny Dashboard

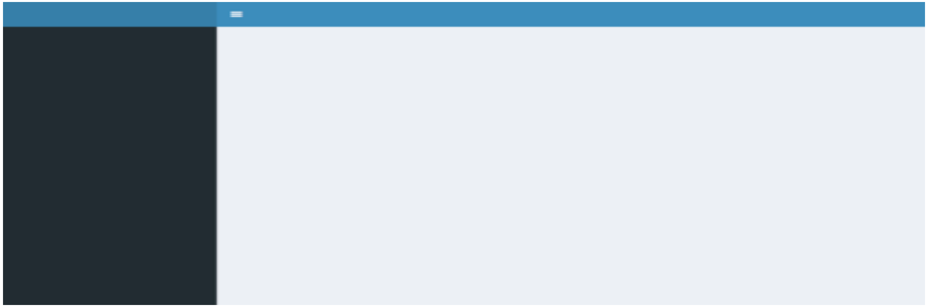


Figure: blank dashboard

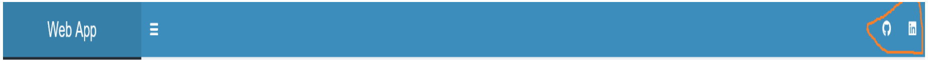


Figure: header

Shiny Dashboard

Header

```
library(shiny)
 library(shinydashboard)
 ui = dashboardPage(
 dashboardHeader(title = "Web App",
 tags$li(class="dropdown",tags$a(href="https://github.com/Prachi-Gor
 icon("github"),target="_blank")),
 tags$li(class="dropdown",tags$a(href="https://www.linkedin.com/in/p
 icon("linkedin"),target="_blank"))),
 dashboardSidebar(),
 dashboardBody()
 )
 server = function(input, output) { }
 shinyApp(ui, server)
```

Shiny Dashboard

Sidebar

```
library(shiny)
library(shinydashboard)
ui = dashboardPage(
dashboardHeader(),
dashboardSidebar(
sidebarMenu(
id = "tabs",menuItem("Graph", tabName = "graph",menuSubItem("Scatter
)),
dashboardBody()
)
server = function(input, output) { }
shinyApp(ui, server)
```

Shiny Dashboard

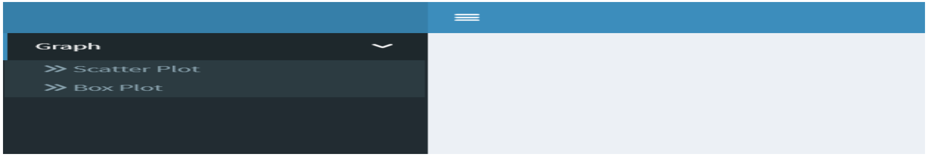


Figure: blank dashboard

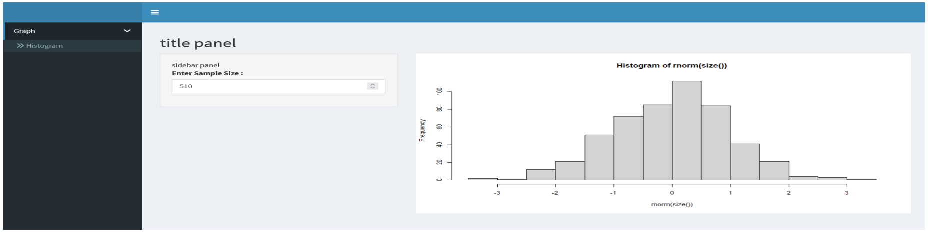


Figure: header

Shiny Dashboard

Body

```
library(shiny)
library(shinydashboard)
ui_hist = fluidPage(
titlePanel("title panel"),
sidebarLayout(
sidebarPanel("sidebar panel",
numericInput(inputId = "n",label = "Enter Sample Size :",
value = 50)),
mainPanel(plotOutput("histogram"))
)
)
```

Shiny Dashboard

```
ui = dashboardPage(
title="dashboard page",
dashboardHeader(),
dashboardSidebar(
sidebarMenu(
id = "tabs",
menuItem("Graph", tabName = "graph",
menuSubItem("Histogram", tabName = "histogram"))
          )
          ),
dashboardBody(tabItems(tabItem(tabName = "histogram",ui_hist)))
)
```

Shiny Dashboard

```
server=function(input,output){
size= reactive({input$n})
output$histogram=renderPlot({hist(rnorm(size()))})
}
shinyApp(ui, server)
```

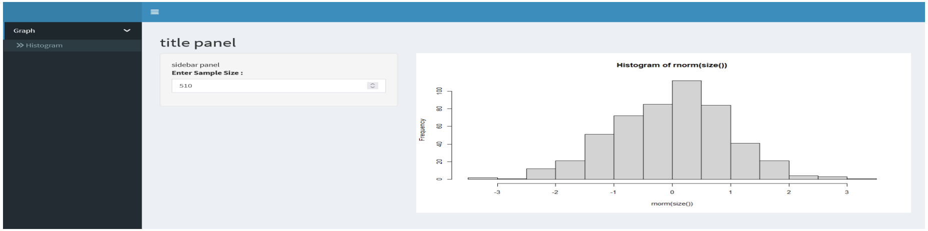


Figure: dashboard body

How to render Scatter Plot

```
library(shiny)
library(shinydashboard)
library(tools) #to check file extension
library(dplyr) #select_if()
library(readxl)
scatter_ui=fluidPage(title="scatter",sidebarLayout(sidebarPanel(
fileInput(inputId = "file_scatter", label = "Select Dataset",
accept = c(".csv",".xlsx"),
buttonLabel = "Browse...",placeholder = "No file selected"),
selectInput(inputId = "scatter_var1_id",
label = "Select x variable",choices=""),
selectInput(inputId = "scatter_var2_id",
label = "Select y variable",choices="")),
mainPanel (plotOutput("scatter")) ))
```

How to render Scatter Plot

```
header = dashboardHeader()
sidebar=dashboardSidebar(sidebarMenu(id = "tabs",
               menuItem("Graph", tabName = "graph",
               menuSubItem("Scatter Plot", tabName = "Scatter-Plot"
body=dashboardBody(tabItems(tabItem("Scatter-Plot",scatter_ui)))
ui = dashboardPage(title ="Web App With Shiny",header,sidebar,body)
update_input= function(input_id,label,data){return(
updateSelectInput(
session = getDefaultReactiveDomain(),
inputId = input_id,
label = label,
choices = names(data()),
selected = NULL))
}
```

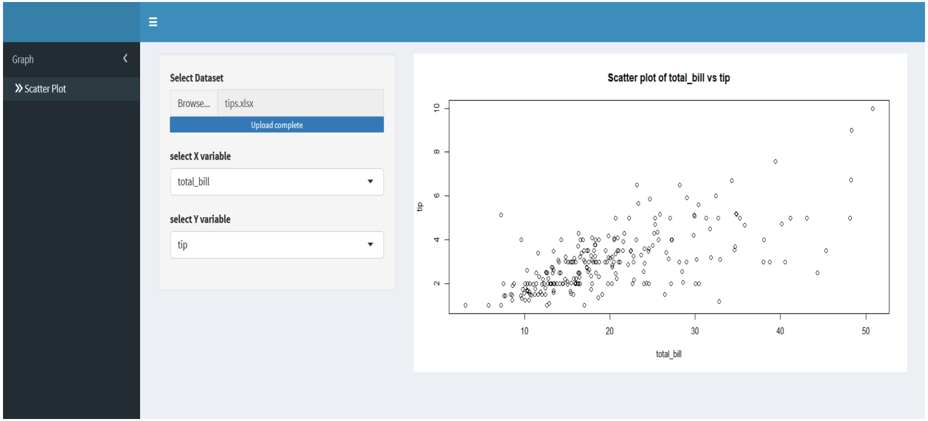
How to render Scatter Plot

```
server= function(input,output){
data_scatter=reactive({
req(input$file_scatter)
file_ext= file_ext(input$file_scatter$datapath)
if(file_ext=="xlsx"|file_ext=="xls"){
df=as.data.frame(read_excel(input$file_scatter$datapath))}
else{df = read.csv(input$file_scatter$datapath )}
return(select_if(df, is.numeric))
})
```

How to render Scatter Plot

```
observe(update_input("scatter_var1_id",label="select X variable",
                     data_scatter))
observe(update_input("scatter_var2_id",label="select Y variable",
                      data_scatter))
output$scatter = renderPlot({
x = data_scatter()[,input$scatter_var1_id]
y=data_scatter()[,input$scatter_var2_id]
plot(x,y,xlab=input$scatter_var1_id,ylab=input$scatter_var2_id,
main = paste("Scatter plot of" ,input$scatter_var1_id,"vs",
input$scatter_var2_id))
})
}
shinyApp(ui,server)
```

How to render Scatter Plot



Select Dataset: tips.xlsx

select X variable: total_bill

select Y variable: tip

Figure: scatter plot

References

- Official Documentation
- YouTube Channel
- StackOverflow