R Shiny Tutorial

Shiny is an R package which helps in making interactive web applications. It can be installed directly using the '*install.packages("shiny")*' in R.

Structure of Shiny App:

Shiny App is contained in a single script called *app.R* which has three components:

i) A user Interface object
ii) A server object
iii) A call to ShinyApp function

There are two options for executing the R Shiny App:

- The UI and server object can be contained in a single R file
- We can create a two-file app where the two objects can be distributed across the files ui.R, server.R which must be in the same directory. Additionally, we can also have an optional file called global.R which can contain the processing of data objects which are common to both ui.R and server.R

Step 1: Create the UI object

The UI object contains the user interface definitions and provides with a range of R functions which help in controlling the layout and appearance of the app.

Structure of a basic UI object:

Shiny provides various layout features for customizing the UI components. The simplest default layout is the one with the sidebar for inputs and a mainPanel for generating outputs.

```
# Define UI for app that draws a histogram ----
ui <- fluidPage(
  # App title ----
  titlePanel("Hello Shiny!"),
  # Sidebar layout with input and output definitions ----
  sidebarLayout(
    # Sidebar panel for inputs ----
    sidebarPanel(
      # Input: Slider for the number of bins ----
      sliderInput(inputId = "bins",
                  label = "Number of bins:",
                  min = 1,
                  max = 50,
                  value = 30)

    ),

    # Main panel for displaying outputs ----
```

```
        mainPanel(

       # Output: Histogram ----
       plotOutput(outputId = "distPlot")
     )))
```

Step 2: Create the Server object

The server object contains instructions for building the UI components of the app. It uses various functions to perform calculations and processing on the available data from the user defined input to renders the output accordingly.

Structure of a basic server object:

```
# Define server logic required to draw a histogram ----
server <- function(input, output) {

  # Histogram of the Old Faithful Geyser Data ----
  # with requested number of bins
  # This expression that generates a histogram is wrapped in a call
  # to renderPlot to indicate that:
  #
  # 1. It is "reactive" and therefore should be automatically
  #    re-executed when inputs (input$bins) change
  # 2. Its output type is a plot

  output$distPlot <- renderPlot({
    x    <- faithful$waiting
    bins <- seq(min(x), max(x), length.out = input$bins + 1)
    hist(x, breaks = bins, col = "#75AADB", border = "white",
         xlab = "Waiting time to next eruption (in mins)",
         main = "Histogram of waiting times")
    })

}
```

Step 3: Making a call to ShinyApp()

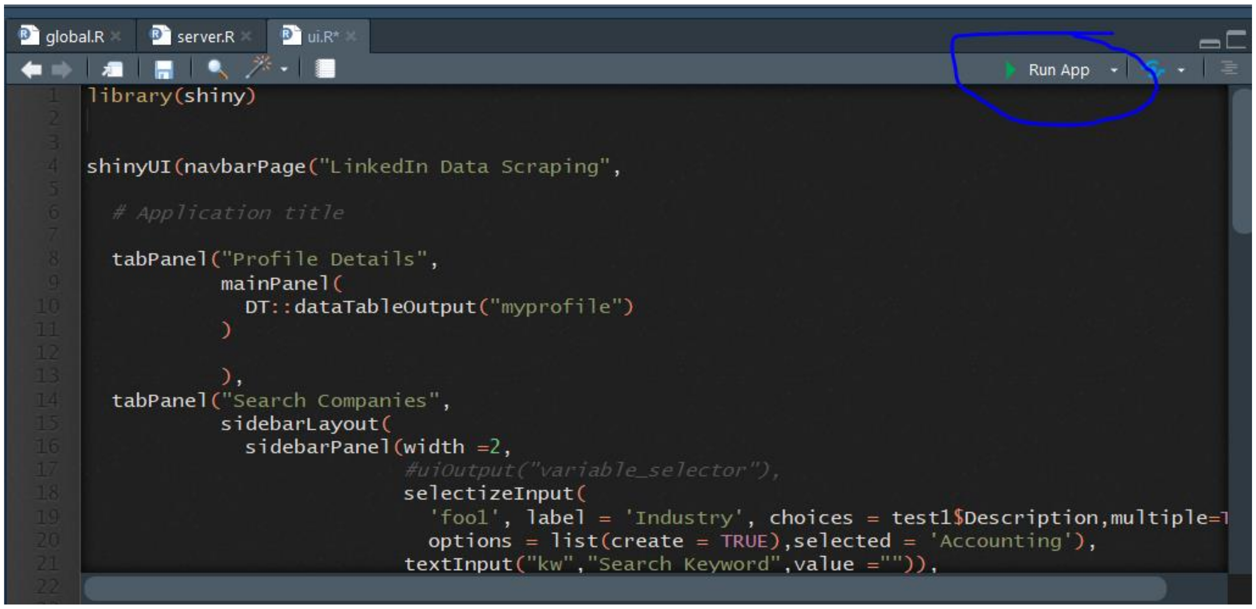
Once the ui and server objects have been defined, the the last step is to make a call to shinyApp() function using the respective UI and server objects.

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

Step 4: Running an App

It is recommended that every shiny app is maintained in a new directory for a modular structure. There are two ways of launching the R Shiny App

1) Using the RunApp button on the top right corner of the screen. This would be provided by RStudio once any of the Shiny files(app.R,ui,R,server.R) are opened.



```
library(shiny)


shinyUI(navbarPage("LinkedIn Data Scraping",

  # Application title

  tabPanel("Profile Details",
           mainPanel(
             DT::dataTableOutput("myprofile")
           )

           ),
  tabPanel("Search Companies",
           sidebarLayout(
             sidebarPanel(width =2,
                          #uiOutput("variable_selector"),
                          selectizeInput(
                            'foo1', label = 'Industry', choices = test1$Description,multiple=T
                            options = list(create = TRUE),selected = 'Accounting'),
                          textInput("kw","Search Keyword",value ="")),
```

2) Using the R console we can explicitly call the runApp() function depending on the structure of the Shiny App.

i) **For a single file app**

If a single file contains both ui and server components use the below command for launching the app:

runApp(app.R) - - *app.R is the name of the file*

ii) **For a two-file app**

If the structure of the shiny app files is as below:

```
~/Example
|-- ui.R
|-- server.R
```

Use **runApp("Example")** for launching the app.

Step 4: Publishing the R Shiny App

The easiest way to share the R Shiny App is to host it on shinyapps.io which is platform as a service (PaaS) for hosting Shiny web apps (applications). The app can be hosted using any of the below two options:

1) **Using the *rsconnect* R package:**

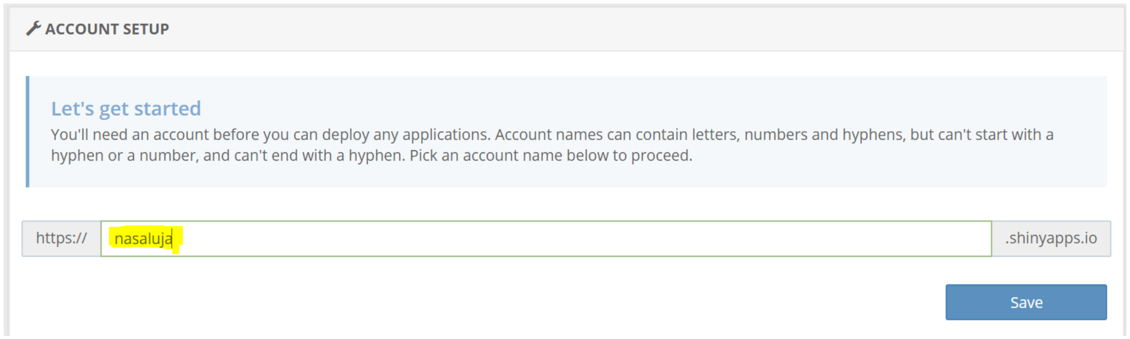
i) Install the rsconnect package using the below commands:

```
install.packages("rsconnect")

library(rsconnect)
```

ii) Configure the shinyapps.io account by clicking on the below link: https://www.shinyapps.io/ → Click Dashboard(upper right corner of the screen) →Log in with your Google/Github credentials.

iii) Enter a Account name on the below screen:



ACCOUNT SETUP

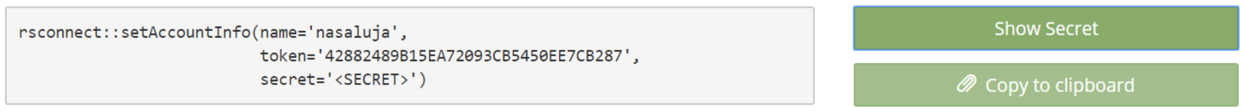
Let's get started

You'll need an account before you can deploy any applications. Account names can contain letters, numbers and hyphens, but can't start with a hyphen or a number, and can't end with a hyphen. Pick an account name below to proceed.

https:// nasaluja .shinyapps.io

Save

iv) To authorize rsconnect to your account, click the 'Copy to Clickboard' button in the below screen



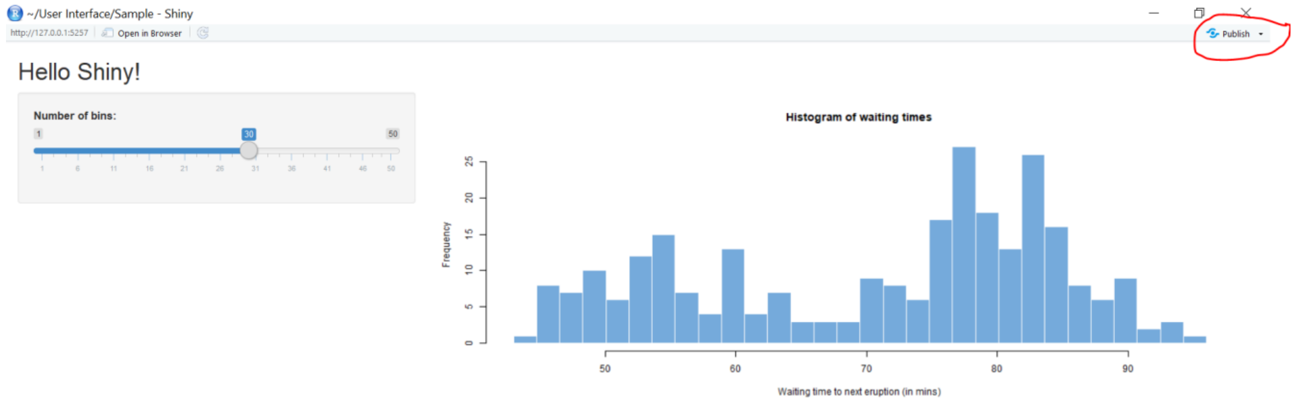
```
rsconnect::setAccountInfo(name='nasaluja',
                          token='42882489B15EA72093CB5450EE7CB287',
                          secret='<SECRET>')
```

v) Deploy the app using the below commands on the console:

```
library(rsconnect)
rsconnect::deployApp('path/to/your/app')
```

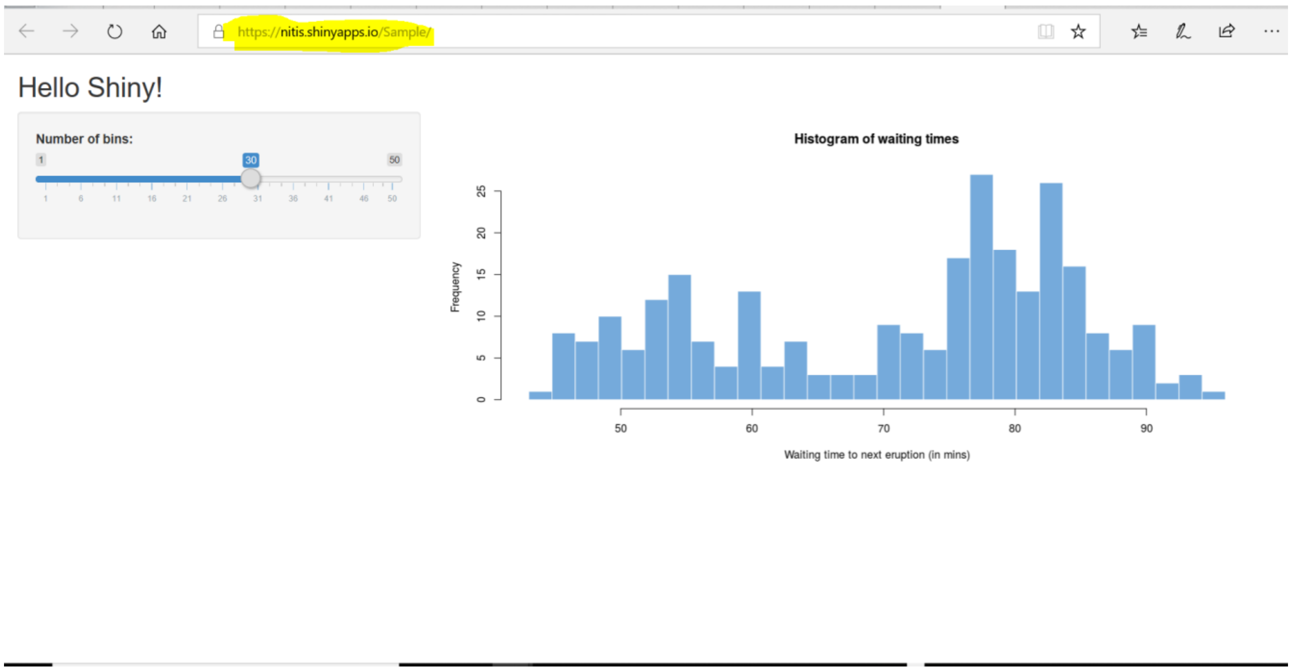
2) **Using RStudio IDE:**

Click the Publish button which is present on the top right corner of the App for hosting it.



~/User Interface/Sample - Shiny
http://127.0.0.1:5257
Open in Browser
Publish
Hello Shiny!
Number of bins:
1
30
50
Histogram of waiting times
Frequency
Waiting time to next eruption (in mins)
50
60
70
80
90

Once the deployment is complete, your browser will automatically open to the deployed R Shiny app and the link can be easily shared with others.


https://nitis.shinyapps.io/Sample/
Hello Shiny!
Number of bins:
1
30
50
1
6
11
16
21
26
31
36
41
46
50
Histogram of waiting times
Frequency
0
5
10
15
20
25
50
60
70
80
90
Waiting time to next eruption (in mins)