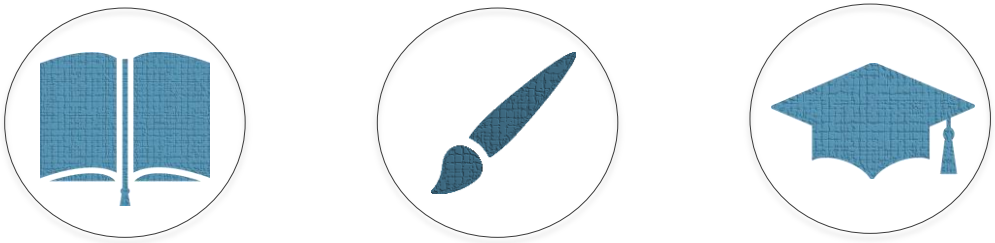


Data Science Using R

Lesson07–Introduction to Shiny R

Objective

After completing this lesson you will be able to:

- Explain the importance of Shiny R
- Describe the structure of Shiny application development using R
- Run the Shiny app from R Studio
- Deploy Shiny app on the web

Shiny R–Web Development Interface

- Shiny package provides a web development interface to R which can help build and run interactive web applications.
- The first step to build applications is by installing Shiny

***On R Studio Console**:*
>`Install.packages("Shiny")`

Or install using the Rstudio Install packages options

Running the Shiny Examples

- There are eleven inbuilt examples in Shiny package each of which is a shiny app. An example can be accessed by the following command

Example 1: On R Studio Console:

```
>library("Shiny")
>runExample("01_Hello") #opens an interactive histogram
```

Example 2: On R Studio Console:

```
>library("Shiny")
>runExample("02_text")     #shows descriptive stats for
the selected datasets
```

- All shiny examples can be accessed by navigating to the location where R is installed.
- If R version 3.2.2 is installed in C drive then the path will be 'C:\Program Files\R\R-3.2.2\library\shiny\examples'.
- In Mac: 'Macintosh HD/library/Frameworks/Versions/Current/Resources/library/shiny/examples'

Shiny App Structure

- Shiny app has two basic components.

User Interface Script

- Controls the layout and appearance of the shiny app.
- The source script is named ‘ui.R’.

Server Script

- Responsible for the calculations which is to be performed to show the result on the UI.
- The source script is named as ‘server.R’.

The default working directory for shiny is the place where ui.R and server.R is saved for a specific shiny app.

Demo of the 01_Hello example

Making a Shiny App

- Every shiny app has the same structure of ui.R and server.R saved in one directory. To build your first shiny app, follow the below steps

1. *Copy '01_Hello' example and paste in your working directory.*
2. *Rename the 01_Helllo to any other folder name say, myapp.*
3. *Open the ui.R and server.R scripts in R Studio. Edit the server.R to change the color of the histogram from 'darkgrey' to 'blue'. Save the script.*
4. *On the console type:*
   >runApp("myapp")

Your first shiny app will be launched.

While giving name to the folder/directory ensure that the name is more than 3 characters long. If it is less than 3 character long, the ShinyApps throws error in deployment. More on deployment later.

Shiny App–UI Layout

- The ui.R has a basic structure as explained below:

```
# ui.R
shinyUI(fluidPage(              #creates a page which adjusts to browser
dimensions
  titlePanel("title panel"),          #title panel at the top

  sidebarLayout(                      #has two components as below
    sidebarPanel( "sidebar panel"),   #used to place the controls to
give interactions
    mainPanel("main panel")           #used to display the results
  )
))
```

More on how to write formatted paragraphs inside the sidebarPanel and mainPanel: http://shiny.rstudio.com/tutorial/lesson2/

Shiny App–UI Control Widgets

- Control widgets are used to send inputs by the user to shiny apps. The standard shiny widgets are:

Widgets	Functions
Action button	actionButton, submitButton
Checkboxes	checkboxInput, checkboxGroupInput
Date input	dateInput, dateRangeInput
File upload	fileInput
Field to enter input	numericInput, textInput
Radio buttons	radioButton
Slider bar	sliderInput
Box with choices to select from	selectInput

More on how to place control widgets inside the sidebarPanel and mainPanel:
http://shiny.rstudio.com/tutorial/lesson3/

Shiny App–UI Reactive Output

- First step to make a reactive app is to add R objects using the output functions and control widgets through ui.R. Some of the output functions

Output Functions	Creates
htmlOutput	Raw HTML
imageOutput	Image
plotOutput	Plot
tableOutput	Table
textOutput	Text
uiOutput	Raw HTML
verbatimTextOutput	Text

In 01_Hello example, plotOutput("text1") was the output function used inside the main panel. More on this at : http://shiny.rstudio.com/tutorial/lesson4/

Shiny App–UI Reactive Output

- Second step is to build the object using the render function and pass the widget value to the code. This step is carried out in server.R. Some of the render function are:

Render Functions	Creates
renderImage	Images
renderPlot	Plot
renderPrint	Any printed output
renderTable	Data frame, Matrix, Other table like structure
renderText	Character strings
renderUI	Shiny tag object or HTML

Render function should correspond to the type of reactive object being made. In 01_Hello example, renderPlot was the inside the server.R to build the histogram. More on this at : http://shiny.rstudio.com/tutorial/lesson4/

Shiny App–Execution Flow

- The placement of code inside the server.R determines the efficiency at which the app will execute.

```
#server.R

#place to put a code

shinyServer(
  function(input,output) {

  #another place to put a code

    output$text1 <-renderText({
    #third place to put the code
    })
  }
)
```

Block 3 which is a render function runs on every change of widget value

Block 2 which is an unnamed function runs each time a user visits the app.

Block 1 which is server.R script runs once when the app is launched.

Load libraries, read datasets outside the shinyServer function, put user specific objects inside the unnamed function and control widget specific code in render function. More on this at : http://shiny.rstudio.com/tutorial/lesson5/

Shiny App–UI Reactive Expressions

- Reactive expressions are used to improve efficiency of code in server.R in cases where user controls the data upload through certain widgets.

```
#server.R
#place to put a code
shinyServer(

  function(input,output) {
  #another place to put a code

    reactiveInput <-reactive({
    #another place to put the code
    })

    output$text1 <-renderText({
    #reactiveInput used within
    })

  }

)
```

Block 4 which is a reactive function runs on certain widget value.

Block 3 which is a render function runs on every change of widget value.

Block 2 which is an unnamed function runs each time a user visits the app.

Block 1 which is server.R script runs once when the app is launched.

Shiny App–UI Reactive Expressions

- Code snippet to highlight the specific case of reading data from yahoo finance.

#without reactive function

```
output$plot <- renderPlot({
data <- getSymbols(input$symb,
src = "yahoo",
from = input$dates[1],
to = input$dates[2],
auto.assign = FALSE)

chartSeries(data, theme =
chartTheme("white"),
    type = "line", log.scale =
input$log, TA = NULL)
})
```

#with reactive function

```
dataInput <- reactive({
  getSymbols(input$symb, src =
"yahoo",
    from = input$dates[1],
    to = input$dates[2],
    auto.assign = FALSE)
})
output$plot <- renderPlot({
  chartSeries(dataInput(),
theme = chartTheme("white"),
    type = "line", log.scale =
input$log, TA = NULL)
})
```

More on reactive expressions at:
http://shiny.rstudio.com/tutorial/lesson6/

Shiny App Deployment

- The easiest way to make shiny app online is through shinyapps.io. Follow the below steps to create account at shinyapps and configure your system to host apps at shinyserver.

1. *Install devtools version 1.4 or later by running the command at the R studio console : install.packages('devtools')*
2. *Restart the R studio session and then install rsconnect through R studio console: devtools::install_github('rstudio/rsconnect')*
3. *Load rsconnect into the session through R studio console: library(rsconnect)*
4. *Create an account at http://www.shinyapps.io/*
5. *Configure rsconnect following the dashboard which appears after creating the account at step 4*
6. *Open a shiny app in your machine and from the console run the command to deploy the app*

   library(rsconnect)

   deployApp(<your app name>)

Or

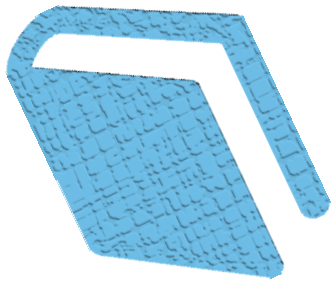
shinyapps::deployApp(<your app name>)

More on the deployment at: http://shiny.rstudio.com/articles/shinyapps.html

Demo of the deployment on a shiny app

Summary

Summary of the topics covered in this lesson:



- Shiny is a web development interface to R which can be used to build and host applications online.
- All the applications developed using Shiny R will have two basic script: ui.R and server.R.
- ui.R helps build interactive UI using widgets whereas server.R helps in calculations whose results are shown on UI.
- The code within the render function is used to give interactivity to the applications.
- Reactive expressions are used to save on computing power of applications and make the app more efficient.
- Shiny apps can be deployed on the web by creating an account at shinyapps.io and configuring your system to deploy the apps.

QUIZ TIME

Quiz Question 1

Quiz 1

What is the command line syntax to install shiny?

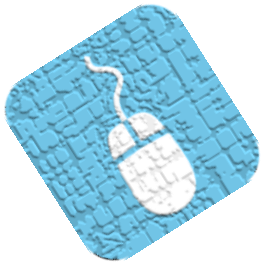
Select all that apply.

a. install.packages(“Shiny”)

b. *install.package(“shiny”)*

c. *install.packages(‘Shiny’)*

d. *install.packages(“shiny”)*



Quiz Question 1

Quiz 1 What is the command line syntax to install shiny?

Select all that apply.

a. install.packages("Shiny")

b. *install.package("shiny")*

c. *install.packages('Shiny')*

d. *install.packages("shiny")*

Correct answer is: *a & c*

Both a and c has the correct syntax.

End of Lesson07–Introduction to Shiny R