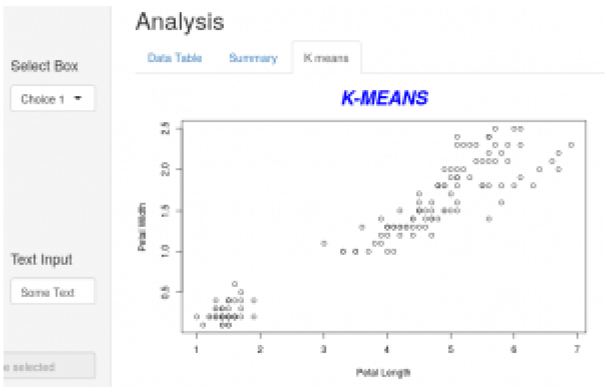
Building Shiny App exercises part 6



RENDER FUNCTIONS

In the sixth part of our series we will talk about the renderPlot and the renderUI function and then we will be ready to create our first visualization. (Find part 1-5 here).
We are going to create a simple interactive scatterplot that will help us see the clusters that are created when we run the k-means algorithm on our dataset. Read the examples below to understand how to activate a renderPlot function and the test yous skills with the exercise set we prepared for you. Lets begin!

Answers to the exercises are available here.

If you obtained a different (correct) answer than those listed on the solutions page, please feel free to post your answer as a comment on that page.

DESCRIPTIVE STATISTICS

As in every statistical application it is wise to apply descriptive statistics on your dataset and also provide this information to user in an easy-readable way. So, first of all we will place a Data Table inside the “SUMMARY” tabPanel. The example below can be your guide.

```
#ui.R
library(shiny)
```

```
shinyUI(fluidPage(
sidebarLayout(
sidebarPanel(
),
mainPanel(
dataTableOutput("Table")
)
)))
#server.R
shinyServer(function(input, output, session) {
sum<-as.data.frame.array(summary(iris))
output$Table <- renderDataTable(sum)
})
```

Learn more about Shiny in the online course R Shiny Interactive Web Apps – Next Level Data Visualization. In this course you will learn how to create advanced Shiny web apps; embed video, pdfs and images; add focus and zooming tools; and many other functionalities (30 lectures, 3hrs.).

Exercise 1

Create a Data Table(“Table2”) with the descriptive statistics of your dataset. HINT: Use summary, as.data.frame.array and renderDataTable.

renderPlot

The renderPlot function enders a reactive plot that is suitable for assigning to an output slot. The general form of the function that generates the plot is below:

```
renderPlot(expr, width = "auto", height = "auto", res = 72,
...,
env = parent.frame(), quoted = FALSE, execOnResize = FALSE,
outputArgs = list())
```

The example below shows you how to create a simple scatterplot between two variables of the iris dataset(“Sepal Length” and

“Sepal Width”).

```
# ui.R
library(shiny)
shinyUI(fluidPage(
sidebarLayout(
sidebarPanel(
),
mainPanel(
plotOutput("plot1")
)
)))
#server.R
shinyServer(function(input, output, session) {
output$plot1 <- renderPlot({
plot(iris$Sepal.Length,iris$Sepal.Width)
})
})
```

Initially remove renderImage and radioButtons from the tabPanel “K means”.

Exercise 2

Add a scatterplot inside the tabPanel “K Means” between two variables of the iris dataset.

INTERACTIVE PLOTS

Shiny has built-in support for interacting with static plots generated by R’s base graphics functions,this makes it easy to add features like selecting points and regions, as well as zooming in and out of images.
To get the position of the mouse when a plot is clicked, you simply need to use the click option with the plotOutput. For example, this app will print out the x and y coordinate position of the mouse cursor when a click occurs.

```
#ui.R
```

```
library(shiny)
shinyUI(fluidPage(
sidebarLayout(
sidebarPanel(),
mainPanel(
plotOutput("plot1", click = "plot_click"),
verbatimTextOutput("info")
)
)))
#server.R
shinyServer(function(input, output, session) {
output$plot1 <- renderPlot({
plot(iris$Sepal.Length,iris$Sepal.Width)
})
output$info <- renderText({
paste0("x=", input$plot_click$x, "\ny=", input$plot_click$y)
})
})
```

Exercise 3

Add click inside the plotOutput you just created. Name it “mouse”.

Exercise 4

Add a verbatimTextOutput inside the “K Means” tabPanel,under the plotOutput you created before. Name it “coord”.

Exercise 5

Make “x” and “y” coordinates appear in the pre-tag you just created. HINT : Use renderText and paste0 and do not forget to activate it with the submitButton.

Exercise 6

Set height = “auto” and width = “auto”.

PLOT ANNOTATION

This function can be used to add labels to a plot. Its first four principal arguments can also be used as arguments in most high-level plotting functions. They must be of type character or expression. In the latter case, quite a bit of mathematical notation is available such as sub- and superscripts, greek letters, fraction, etc.

```
title(main = NULL, sub = NULL, xlab = NULL, ylab = NULL,
line = NA, outer = FALSE, ...)
```

Look at the example below:

```
# ui.R
library(shiny)
shinyUI(fluidPage(
sidebarLayout(
sidebarPanel(),
mainPanel(
plotOutput("plot1", click = "plot_click"),
verbatimTextOutput("info")
)
)))
#server.R
shinyServer(function(input, output, session) {
output$plot1 <- renderPlot({
plot(iris$Sepal.Length,iris$Sepal.Width,main = "SCATTER PLOT",sub = "K Means",xlab="Sepal Length",ylab = "Sepal Width")
})
output$info <- renderText({
paste0("x=", input$plot_click$x, "\ny=", input$plot_click$y)
})
})
```

Exercise 7

Set scatterplot title to “K-Means”, the X-axis label to “Petal Length” and the Y-axis label to “Petal Width”. HINT: Use main,xlab,ylab.

You can also modify and set other graphical parameters related to the title and subtitle like the example below:

```
# ui.R
library(shiny)
shinyUI(fluidPage(
sidebarLayout(
sidebarPanel(),
mainPanel(
plotOutput("plot1", click = "plot_click"),
verbatimTextOutput("info")
)
)))
#server.R
shinyServer(function(input, output, session) {
output$plot1 <- renderPlot({
plot(iris$Sepal.Length,iris$Sepal.Width,main = "SCATTER
PLOT",sub = "K Means",xlab="Sepal Length",ylab = "Sepal
Width",
cex.main = 3, font.main= 5, col.main= "green",
cex.sub = 0.65, font.sub = 4, col.sub = "orange")
})
output$info <- renderText({
paste0("x=", input$plot_click$x, "\ny=", input$plot_click$y)
})
})
```

Exercise 8

Give values to the rest of the graphical parameters of the title like the example above and get used to them. HINT: Use cex.main, font.main and col.main.

renderUI

```
renderUI(expr, env = parent.frame(), quoted = FALSE,
outputArgs = list())
```

Makes a reactive version of a function that generates HTML using the Shiny UI library. As you can see in the example below this expression returns a tag object.

```
# ui.R
library(shiny)
shinyUI(fluidPage(
sidebarLayout(
sidebarPanel( uiOutput("Controls")),
mainPanel(
plotOutput("plot1", click = "plot_click"),
verbatimTextOutput("info")
)
)))
#server.R
shinyServer(function(input, output, session) {
output$plot1 <- renderPlot({
plot(iris$Sepal.Length,iris$Sepal.Width,main = "SCATTER
PLOT",sub = "K Means",xlab="Sepal Length",ylab = "Sepal
Width",
cex.main = 2, font.main= 4, col.main= "blue",
cex.sub = 0.75, font.sub = 3, col.sub = "red")
})
output$info <- renderText({
paste0("x=", input$plot_click$x, "\ny=", input$plot_click$y)
})
output$Controls <- renderUI({
tagList(
sliderInput("n", "N", 1, 1000, 500),
textInput("label", "Label")
)
})
})
```

Exercise 9

Put a uiOutput inside tabPanel “K-Means” and name it “All”. Then create its output in server.R with a tagList into it. HINT: Use uiOutput, renderUI and tagList.

Exercise 10

Remove the submitButton and move the sliderInput and the textOutput from the ui.R into the tagList.