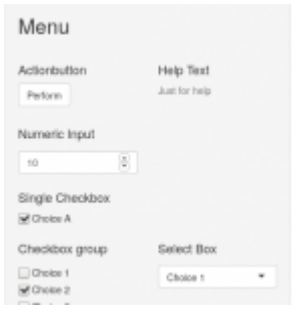
Building Shiny App exercises part 7



Connect widgets & plots

In the seventh part of our journey we are ready to connect more of the widgets we created before with our k-means plot in order to totally control its output. Of cousre we will also reform the plot itself properly in order to make it a real k-means plot.
Read the examples below to understand the logic of what we are going to do and then 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.

First of all let's move the widgets we are going to use from the sidebarPanel into the mainPanel and specifically under our plot.

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

Remove the textInput from your server.R file. Then place the checkboxGroupInput and the selectInput in the same row with the sliderInput. Name them “Variable X” and “Variable Y” respectively. HINT: Use fluidrow and column.

Create a reactive expression

Reactive expressions are expressions that can read reactive values and call other reactive expressions. Whenever a reactive value changes, any reactive expressions that depended on it are marked as “invalidated” and will automatically re-execute if necessary. If a reactive expression is marked as invalidated, any other reactive expressions that recently called it are also marked as invalidated. In this way, invalidations ripple through the expressions that depend on each other.
The reactive expression is activated like this: example <- reactive({ })

Exercise 2

Place a reactive expression in server.R, at any spot except inside output$All and name it “Data”. HINT: Use reactive

Connect your dataset’s variables with your widgets.

Now let’s connect your selectInput with the variables of your dataset as in the example below.

```
#ui.R
library(shiny)
shinyUI(fluidPage(
titlePanel("Shiny App"),

sidebarLayout(
sidebarPanel(h2(“Menu”),
selectInput(‘ycol’, ‘Y Variable’, names(iris)) ),
mainPanel(h1(“Main”)
```

```
)
)
))
#server.R
shinyServer(function(input, output) {
example <- reactive({
iris[, c(input$ycol)]
})
})
```

Exercise 3

Put the variables of the iris dataset as inputs in your selectInput as “Variable Y” . HINT: Use names.

Exercise 4

Do the same for checkboxGroupInput and “Variable X”. HINT: Use names.

Select the fourth variabale as default like the example below.

```
#ui.R
library(shiny)
shinyUI(fluidPage(
titlePanel("Shiny App"),

sidebarLayout(
sidebarPanel(h2(“Menu”),
checkboxGroupInput(“xcol”, “Variable X”,names(iris),
selected=names(iris)[[4]]),
selectInput(“ycol”, “Y Variable”, names(iris),
selected=names(iris)[[4]])
),
mainPanel(h1(“Main”)
)
)
))
#server.R
```

```
shinyServer(function(input, output) {
example <- reactive({
iris[, c(input$xcol,input$ycol)
]
})
})
```

Exercise 5

Make the second variable the default choise for both widgets. HINT: Use selected.

Now follow the example below to create a new function and place there the automated function for k means calculation.

```
#ui.R
library(shiny)
shinyUI(fluidPage(
titlePanel("Shiny App"),

sidebarLayout(
sidebarPanel(h2(“Menu”),
checkboxGroupInput(“xcol”, “Variable X”,names(iris),
selected=names(iris)[[4]]),
selectInput(“ycol”, “Y Variable”, names(iris),
selected=names(iris)[[4]])
),
mainPanel(h1(“Main”)
)
)
))
#server.R
shinyServer(function(input, output) {
example <- reactive({
iris[, c(input$xcol,input$ycol)
]
})
example2 <- reactive({
```

```
kmeans(example())
})
})
```

Exercise 6

Create the reactive function Clusters and put in there the function kmeans which will be applied on the function Data. HINT: Use reactive.

Connect your plot with the widgets.

It is time to connect your plot with the widgets.

Exercise 7

Put Data inside renderPlot as first argument replacing the data that you have chosen to be plotted until now. Moreover delete xlab and ylab.

Improve your k-means visualiztion.

You gan change automatically the colours of your clusters by copying and pasting this part of code as first argument of renderPlot before the plot function:

```
palette(c("#E41A1C", "#377EB8", "#4DAF4A", "#984EA3",
"#FF7F00", "#FFFF33", "#A65628", "#F781BF", "#999999"))
```

We will choose to have up to nine clusters so we choose nine colours.

Exercise 8

Set min of your sliderInput to 1, max to 9 and value to 4 and use the palette function to give colours.

This is how you can give different colors to your clusters. To activate these colors put this part of code into your plot function.

col = Clusters()$cluster,

Exercise 9

Activate the palette function.

To make your clusters easily foundable you can fully color them by adding into plot function this:
pch = 20, cex = 3

Exercise 10

Fully color the points of your plot.