Seminario-Practica RShiny

Elena Villalobos

1ro de Septiembre 2017

1. Lo que necesitas...

1. Paquetes: ('shiny') y ('rsconnect')
2. Dos archivos R en un sólo folder: server.R y ui.R
3. Cuenta en shinyapps.io

2. Archivos en R

Los archivos en R deben de tener exactamente el mismo nombre de server.R y ui.R (convención de RShiny, no mía).

- El archivo de ui.R debe de contener las siguientes líneas para comenzar:

```
library('shiny')
library('rsconnect')

# Input
fluidPage()
```

- Y el archivo de server.R debe de contener las siguientes líneas:

```
library('shiny')
library('rsconnect')


# Output
server <- function(input,output){}
```

El archivo de ui.R es donde se colocan los objetos estáticos y en el archivo de server.R los objetos que se actualizaran.

2.1. Probemos ¡Hola Mundo!

Al archivo de ui.R agregaremos 'Hola mundo'.

```
library('shiny')
library('rsconnect')

# Input
fluidPage('Hola mundo')
```

En este punto, al momento de correr su app, se tuvo que haber abierto una ventana donde aparece ¡Hola mundo!

2.2. Slider!

A continuación pondremos una slider. Es importante mencionar que existen diferentes tipos de inputs, además del slider, como texto, botones, etc. Estos se pueden checar en la página de RShiny.

```
library('shiny')
library('rsconnect')

# Input ui.R
fluidPage(sliderInput(inputId= 'num', #identificacion de input
                      label='Pon el numero que quieras', #etiqueta de slider
                      value=25, #valor inicial
                      min=1, #valor minimo
                      max=100)) #valor maximo
```

Aquí deben de aparecer ya la slider en la ventana.

2.3. Graficos!

Existen diferentes tipos de output como graficos, tablas, texto, etc. Todos se modifican dentro del archivo de server.R pero también se debe de mencionar en el archivo de ui.R que habrá un plot.

```
library('shiny')
library('rsconnect')

# Input ui.R
fluidPage(sliderInput(inputId= 'num',
```

```
                        label='Pon el numero que quieras',
                        value=25,
                        min=1,
                        max=100),
          plotOutput('grafico')) #mencion de que habra un grafico
```

```
library('shiny')
library('rsconnect')


# Output server.R
server <- function(input,output){
 output$grafico <- renderPlot({
  x <- c(0:10)
  y <- x^2
  plot(x,y)
 })
}
```

Importante notar que 'grafico' es igual en output$grafico es igual a plotOutput('grafico'). Y render es la funcion que crea el tipo de output que en este caso es un plot. Todo lo que se ponde dentro de la función de render es el código para crear los plots.

```
server <- function(input,output){
 output$grafico <- renderPlot({
  x <- c(0:10)
  y <- x^2
  plot(x,y)
 })
}
```

En el código anterior tenemos un plot fijo, para hacer que este se mueva en función de los valores del slider se debe de agregar input$num (notando que num es igual a inputId= "num") en el parámetro fijo que queremos modificar, y en este caso es el dos.

```
server <- function(input,output){
 output$grafico <- renderPlot({
  x <- c(0:10)
  y <- x^input$num #aqui sustituimos
```

```
  plot(x,y)
 })
}
```

Ahora juega con los valores del slider para que se vean mejor los cambios. A mínimo = 0.01, máximo = 2, valor inicial = 2. Y fija los axes de tu plot para que se vea mejor el efecto.

3. A compartir la app!!

R es lo que corre el código en tu computadora, shinyapps.io lo corre en internet. Para que se publica sólo corre la app y busca el botón de publicar y R lo hará por ti. Para esto ya tendrás que tener lista tu cuenta de shinyapps.

4. La función de Reactive

Reactive es la función que hace que una misma variable se pueda usar en más de un plot. Ahora cambiaremos la variable x una variable reactive llamada datos. Nota: checa bien que se utilizan paréntesis y llaves para generar la variable.

```
server <- function(input,output){

  datos <- reactive({c(0:100)})

 output$grafico <- renderPlot({
  y <- datos()^input$num
  plot(datos(),y)
 })
}
```

Ahora hagamos la variable 'y', también reactive. Y observa que para llamar esa variable se debe de poner el nombre y abrir y cerrar paréntesis.

```
server <- function(input,output){

  datos <- reactive({c(0:100)})
  y <- reactive({datos()^input$num})

 output$grafico <- renderPlot({
  plot(datos(),y())
```

```
 })
}
```

Para ver la función de las variables reactive, hagamos un segundo plot con una segunda variable.

```
server <- function(input,output){

  datos <- reactive({c(0:100)})
  y <- reactive({datos()^input$num})
  y_2 <- reactive({y()*4}) #segunda variable

 output$grafico <- renderPlot({
  layout(matrix(c(1,2),ncol=2))

  plot(datos(),y(),xlim=c(0,100),ylim=c(0,200))
  plot(datos(),y_2(),xlim=c(0,100),ylim=c(0,200))

 })
}
```

Ahora, colocamos una segunda slider con diferente inputId, que será nuestro segundo parámetro.

```
fluidPage(sliderInput(inputId= 'num', label='Parametro 1',
                      value=0.5, min=0.01, max=2),
          sliderInput(inputId= 'num2', label='Parametro 2',
                      value=0.5, min=0.01,  max=2),
           plotOutput('grafico')
          )
```

Sustituimos el número 4 por input$num2 y tendremos dos plots que se modifican uno en función del otro, y el otro en función de sí mismo.

```
server <- function(input,output){

  datos <- reactive({c(0:100)})
  y <- reactive({datos()^input$num})
  y_2 <- reactive({y()*input$num2})
```

```
 output$grafico <- renderPlot({
  layout(matrix(c(1,2),ncol=2))

  plot(datos(),y(),xlim=c(0,100),ylim=c(0,200))
  plot(datos(),y_2(),xlim=c(0,100),ylim=c(0,200))

 })
}
```

5. Pa' que se vea bonita su app un poco de HTML

HTML es un documento en el cual sólo se pone el texto que se verá en una página web. En HTML las principales funciones son:

```
# <h1> </h1> ----------  h1() h2() hasta h6() donde se colocan encabezados

# <p> </p> ----------  p() parrafos

# <a> </a> ----------  a() a(href='http://www.git.com',Git) referencias (links)

#
# strong(bold) em(empha) code(computer code) br(line break) hr(horizontal rule)

#
```

Ahora modifiquemos de la siguiente manera:

```
fluidPage(h1('Mi primer Shiny app'),
          p('by Elena'),
          a(href='https://shiny.rstudio.com/','Shiny'),
          hr(),
          sliderInput(inputId= 'num', label='Parametro 1',
                      value=0.5, min=0.01, max=2),
          sliderInput(inputId= 'num2', label='Parametro 2',
                      value=0.5, min=0.01,  max=2),
          plotOutput('grafico')
)
```

6. Posición de elementos

Para la posición de elementos se hace una fila que tiene longitud de 12 columnas.Intenta el siguiente código...

```
fluidPage(h1('Mi primer Shiny app'),
          p('by Elena'),
          a(href='https://shiny.rstudio.com/','Shiny'),
          hr(),
          fluidRow(column(2),
                   column(4,sliderInput(inputId= 'num', label='Parametro 1',
```

```
                    value=0.5, min=0.01, max=2)),
                column(4,sliderInput(inputId= 'num2', label='Parametro 2',
                    value=0.5, min=0.01,  max=2))),
          plotOutput('grafico')
)
```

Con esto hay que checar siempre los paréntesis y las comas, el tipo de error común siempre tiene que ver con la mala colocación de esto. Y no olvidar ser ordenados en sus códigos!

Con este código, las sliders se centraron.

```
fluidPage(h1('Mi primer Shiny app'),
          p('by Elena'),
          a(href='https://shiny.rstudio.com/','Shiny'),
          hr(),
          fluidRow(column(4,offset=2,
                          sliderInput(inputId= 'num',label='Parametro 1',
                                      value=0.5, min=0.01, max=2)),
                   column(4,sliderInput(inputId= 'num2', label='Parametro 2',
                                        value=0.5, min=0.01,  max=2))),
          plotOutput('grafico')
)
```

7. Páneles

Para poner el fonde gris se utiliza wellPanel()

```
fluidPage(h1('Mi primer Shiny app'),
          p('by Elena'),
          a(href='https://shiny.rstudio.com/','Shiny'),
          hr(),
          wellPanel(
          fluidRow(column(4,offset=2,
                          sliderInput(inputId= 'num',label='Parametro 1',
                                      value=0.5, min=0.01, max=2)),
                   column(4,sliderInput(inputId= 'num2', label='Parametro 2',
                                        value=0.5, min=0.01,  max=2)))),
          plotOutput('grafico')
)
```

8. Pestañas

Se pueden hacer diferentes tipos de pestañas, este es uno de ellos (tabsetPanel)

```
fluidPage(tabsetPanel(
          tabPanel('simulador',
          h1('Mi primer Shiny app'),
          p('by Elena'),
          a(href='https://shiny.rstudio.com/','Shiny'),
          hr(),
          wellPanel(
          fluidRow(
           column(4,offset=2,sliderInput(inputId= 'num', label='Parametro 1',
                      value=0.5, min=0.01, max=2)),
           column(4,sliderInput(inputId= 'num2', label='Parametro 2',
                      value=0.5, min=0.01,  max=2)))),
          plotOutput('grafico')),
          tabPanel('instucciones',
                   br(),
                   h1('Instrucciones de como usar el simulador'),
                   hr(),
                   p('Aqui se pueden poner las instucciones de todo'))
))
```

Y así terminaron de hacer su primer shiny app.
Saludos!