

# CURSO SOBRE BIOHERRAMIENTAS EN BIOESTADISTICA Y BIOINFORMATICA (1ªEdición)

Barcelona, 16, 17 y 18 de Mayo 2017

## Cómo crear aplicaciones con Shiny

### Parte IV: maneras de mejorar la aplicación

## Inserir código HTML y CSS

- Usando la función **HTML** se pueden escribir código HTML para cambiar el estilo (color, tamaño, poner links, etc.) en un texto.
- En la cabecera se pueden especificar los estilos de objetos (**#elemento**) que son aplicados a los elementos de entrada a partir de su 'id'.
- También se puede especificar la etiqueta de cualquier elemento de entrada con la función **HTML** para inserir textos con enlaces, etc.

### HTML examples



Figure 1:

Fíjate qué pasa al posicionar el ratón encima de las etiquetas de los inputs (media ó sd).

```

ui <- fluidPage(
  headerPanel("HTML examples"),

  HTML("<style type='text/css'> #inputpanel{background-color: rgb(230,230,230);
    border: 2px solid grey; box-shadow: 2px 2px 1px #888888;} </style>"),
  HTML("<style type='text/css'> #outpanel { background-color: rgb(250,250,250);
    border: 2px solid grey; box-shadow: 2px 2px 1px #888888; overflow:scroll; height:500px} </style>"),
  HTML("<style type='text/css'> #OK {color:white; background-color:rgb(10,101,212);
    border: solid 1px rgb(10,101,212)} </style>"),
  HTML("<style type='text/css'> #inputpanel .wellPanel {background-color:rgb(215,215,215); </style>"),

  sidebarLayout(
    sidebarPanel(id="inputpanel",
      wellPanel(
        numericInput("media", HTML("<p title='Introduce la media'>Media</p>"), 0),
        numericInput("sd", HTML("<p title='Introduce la desviación estándar'>SD</p>"),3),
        actionButton("OK", "OK")
      )
    ),
    mainPanel(id="outpanel",
      plotOutput("results")
    )
  )
)

server <- function(input,output){
  output$results <- renderPlot({
    if (input$OK == 0) return(invisible(NULL))
    isolate({
      hist(rnorm(100, input$media, input$sd), xlab="var", ylab = "frec", main = "")
    })
  }, 500, 500)
}

shinyApp(ui=ui,server=server)

```

## Creación “Pop-ups”

```
library(shinyBS)

ui <- fluidPage(
  wellPanel(id="person",
    textInput("nombre", "nombre", ""),
    numericInput("edad", "edad", 30),
    radioButtons("sexo", "sexo", c("Hombre", "Mujer")),
    textInput("dni", "NIF", "")
  ),
  wellPanel(id="product",
    radioButtons("recom", "Recom.", c("Sí", "No")),
    sliderInput("valor", "Valoración", 0, 10, 5)
  ),
  bsTooltip("nombre", "Escribe tu nombre y apellidos"),
  bsTooltip("edad", "Introduce tu edad"),
  bsTooltip("dni", "Escribe tu DNI con la letra"),
  bsTooltip("recom", "¿Recomendaría el producto?"),
  bsTooltip("valor", "Valore el producto de 0 a 10", "bottom"),
  bsTooltip("person", "Formulario sobre los datos personas", "click"),
  bsTooltip("product", "Formulario de sobre satisfacción", "click")
)

server <- function(input, output, session) {}

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

nombre

edad

sexo

☒ Hombre

☐ Mujer

NIF

Recom.

☒ Si

## Creación de Modales



Figure 2:

UI:

```
library(shinyBS)

ui <- fluidPage(

  sidebarLayout(

    sidebarPanel(
      checkboxInput("grupos",
                    "Distinguir especies")
    ),

    mainPanel(
      plotOutput("grafico"),

      bsModal("modal",
              "Descargar gráfico",
              "grafico",
              radioButtons("tipo", "Formato",
                           c("pdf", "png", "tiff")),
              downloadButton("down", "OK")
            )
    )
  )
)
```

Server:

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

  output$grafico <- renderPlot({
```

```

    if (input$grupos)
      pairs(iris, col = iris[,5])
    else
      pairs(iris)
  }, width = 500)

output$down <- downloadHandler(
  filename = function(){
    paste("grafico", input$tipo, sep = ".")
  },
  content = function(ff){
    if (input$tipo == "pdf") pdf(ff)
    if (input$tipo == "png") png(ff)
    if (input$tipo == "tiff") tiff(ff)
    if (input$grupos)
      pairs(iris, col = iris[,5])
    else
      pairs(iris)
    dev.off()
  }
)
}

```

**Ejercicio:** Haz que aparezca un “popup” al colocar el puntero encima del gráfico, para guardarlo.

# Colapsables

Tamaño del gráfico

**Anchura**

100 500 1,000

100 200 300 400 500 600 700 800 900 1,000

**Altura**

100 500 1,000

100 200 300 400 500 600 700 800 900 1,000

Opciones

Tamaño del gráfico

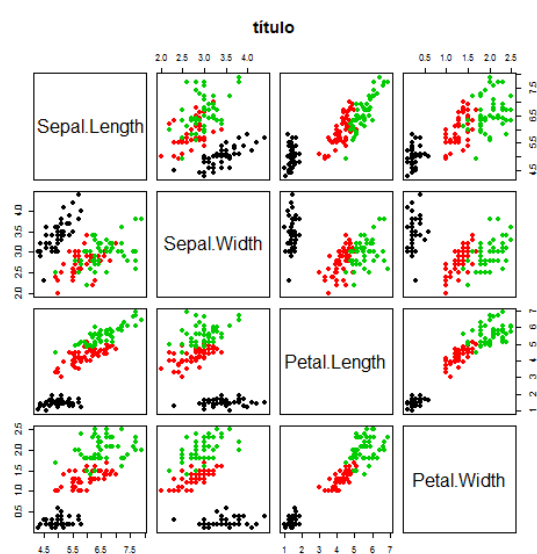
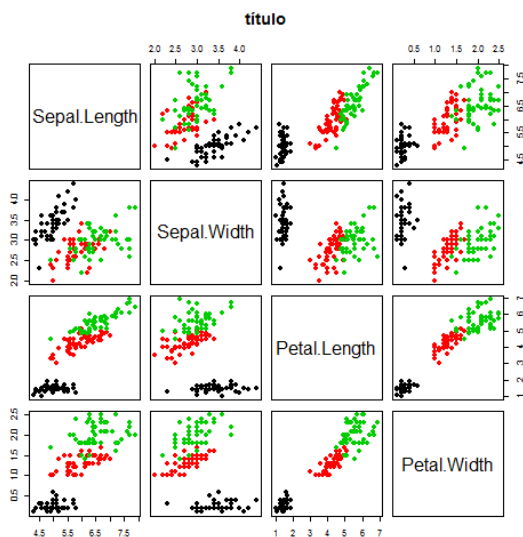
Opciones

**Título**

título

**Tipo de punto**

punto



```
library(shinyBS)
ui <- fluidPage(

  bsCollapse(id = "collapseExample", open = "Tamaño del gráfico",
    bsCollapsePanel("Tamaño del gráfico",
      sliderInput("width", "Anchura", 100, 1000, 500, 50),
      sliderInput("height", "Altura", 100, 1000, 500, 50)
    , style = "info"),
    bsCollapsePanel("Opciones",
      textInput("main", "Título", "título"),
      selectInput("pch", "Tipo de punto",
        c("punto"=19, "cuadrado"=22, "diamante"=23))
    , style = "primary")
  ),

  uiOutput("result")
)

server <- function(input, output) {
  output$plot <- renderPlot({
```

```

    pairs(iris[,-5], col=iris[,5],
          main = input$main,
          pch = as.double(input$pch))
  })
  output$result <- renderUI({
    plotOutput("plot", width=input$width, height=input$height)
  })
}

runApp(list(ui=ui,server=server))

```

**Ejercicio** cambia el argumento `multiple` de la función `bsCollapse` a `TRUE`

## Toogle, show, hide

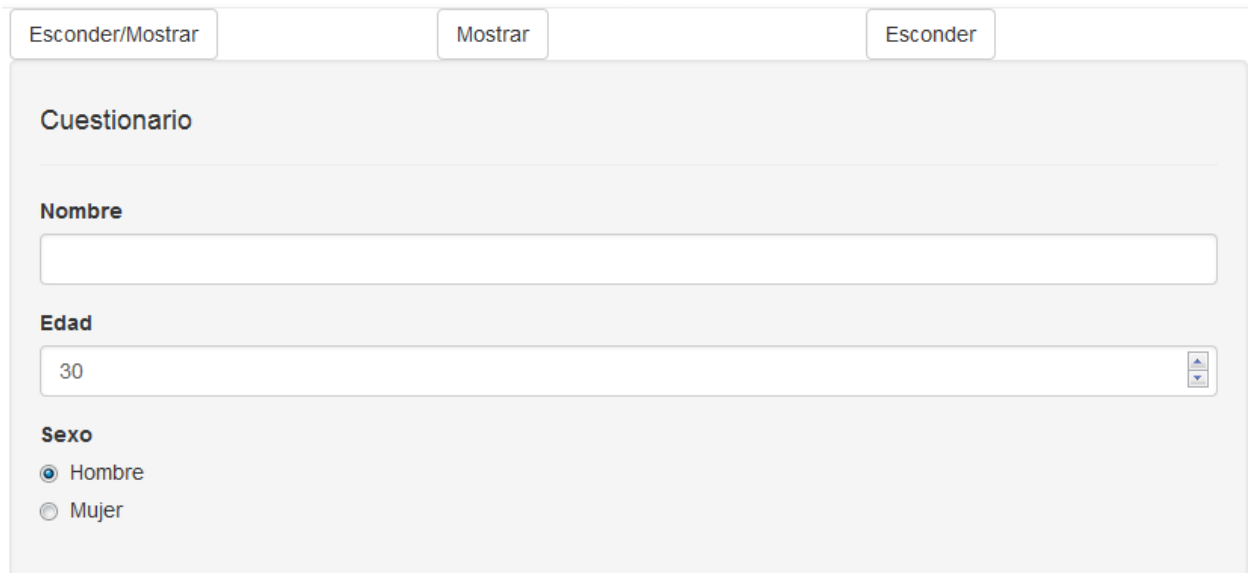
- El package **shinyjs** permite distintos efectos:
  - Esconder/mostrar elementos/textos (*toggle*)
  - Habilitar/deshabilitar botones
  - ....
- Está disponible en CRAN:

```
install.packages(shinyjs)
```

- Para ver las posibilidades, visita <http://daattali.com/shiny/shinyjs-demo/>



## Ejemplo 1. Botones



Esconder/Mostrar      Mostrar      Esconder

**Cuestionario**

---

**Nombre**

**Edad**

**Sexo**

☒ Hombre

☐ Mujer

Figure 3:

### UI

```
ui <- fluidPage(  
  
  useShinyjs(), # Set up shinyjs  
  
  fluidRow(  
    column(4, actionButton("btntoggle",  
                          "Esconder/Mostrar")),  
    column(4, actionButton("btnshow", "Mostrar")),  
    column(4, actionButton("btnhide", "Esconder"))  
  ),  
  
  hidden(  
    wellPanel(id="elemento",  
      h4("Cuestionario"),  
      hr(),  
      textInput("nombre", "Nombre", ""),  
      numericInput("edad", "Edad", 30),  
      radioButtons("sexo", "Sexo", c("Hombre", "Mujer"))  
    )  
  )  
)
```

### Server

```
server <- function(input, output, session) {  
  
  observeEvent(input$btntoggle, {  
    shinyjs::toggle("elemento", TRUE, "fade")  
  })  
}
```

```
observeEvent(input$btnshow, {  
  shinyjs::show("elemento", TRUE, "slide")  
})  
  
observeEvent(input$btnhide, {  
  shinyjs::hide("elemento", FALSE)  
})  
}
```

## Ejemplo 2. Password

- Hacer una aplicación visible cuando se introduce el password correcto.
- Una vez introducido el password correcto **hay que hacer desaparecer las ventanas de password y verificación de password**

```
ui <- fluidPage(  
  
  useShinyjs(),  
  
  div(id="passScreen",  
    passwordInput("pass", "Password", ""),  
    actionButton("valida", "Valida")  
  ),  
  
  shinyjs::hidden(  
    div(id="miapp",  
      titlePanel("Hello Shiny!"),  
      sidebarLayout(  
        sidebarPanel(  
          sliderInput("obs", "Number obs.",  
            min=1, max=1000, value=500)  
        ),  
        mainPanel(  
          plotOutput("distPlot")  
        )  
      )  
    )  
  )  
)  
)  
)  
)
```

```
server <- function(input, output) {  
  
  observeEvent(input$valida, {  
    if (input$pass=="123"){  
      shinyjs::show("miapp", FALSE)  
      shinyjs::hide("passScreen", FALSE)  
    } else {  
      shinyjs::hide("miapp", FALSE)  
      shinyjs::show("passScreen", FALSE)  
    }  
  })  
  
  output$distPlot <- renderPlot({  
    hist(rnorm(input$obs))  
  })  
  
}
```

Pruébalo aquí: <http://apps.datarus.eu/cursShiny/Bloque4Password>

## Plantillas (“Themes”)

- Con el package **shinythemes** se puede cambiar fácilmente el aspecto de la app.
- **shinythemes** contiene una colección de plantillas CSS
- La plantilla se elige mediante la función **shinythemes** y se especifica en el argumento **themes** de la función **fluidPage**, **bootstrapPage**, ...
- Las plantillas que contiene **shinythemes** son: “cerulean”, “cosmo”, “flatly”, “journal”, “readable”, “spacelab”, “united”
- Alternativamente a **shinythemes**, se puede personalizar el aspecto de la app mediante la inclusión de un archivo CSS en la carpeta **www** y especificando el nombre del archivo CSS en el argumento **themes**.
- Mira más plantillas en <http://bootswatch.com/>

### UI

```
library(shinythemes)

ui <- fluidPage(

  theme = shinytheme("united"), # especifica el CSS

  titlePanel("Example Shiny web"),
  sidebarLayout(
    sidebarPanel(
      selectInput("dataset", "Choose a dataset:",
                  c("rock", "pressure", "cars")),
      numericInput("obs",
                   "Number of observations to view:", 10),
      helpText("Note: while the data view will
                show only the specified,
                number of observations,
                the summary will still be based,
                on the full dataset."),
      submitButton("Update View")
    ),
    mainPanel(
      tabsetPanel(type="pills",
                  tabPanel("Summary",
                           verbatimTextOutput("summary"))
                ),
      tabPanel("Observations",
               tableOutput("view"))
    )
  )
)
```

### Server

```
server <- function(input, output) {
  datasetInput <- reactive({
    switch(input$dataset,
           "rock"=rock, "pressure"=pressure, "cars"=cars)
  })
}
```

```

})
output$summary <- renderPrint({
  dataset <- datasetInput()
  summary(dataset)
})
output$view <- renderTable({
  head(datasetInput(), n = input$obs)
})
}

```

## Example Shiny web

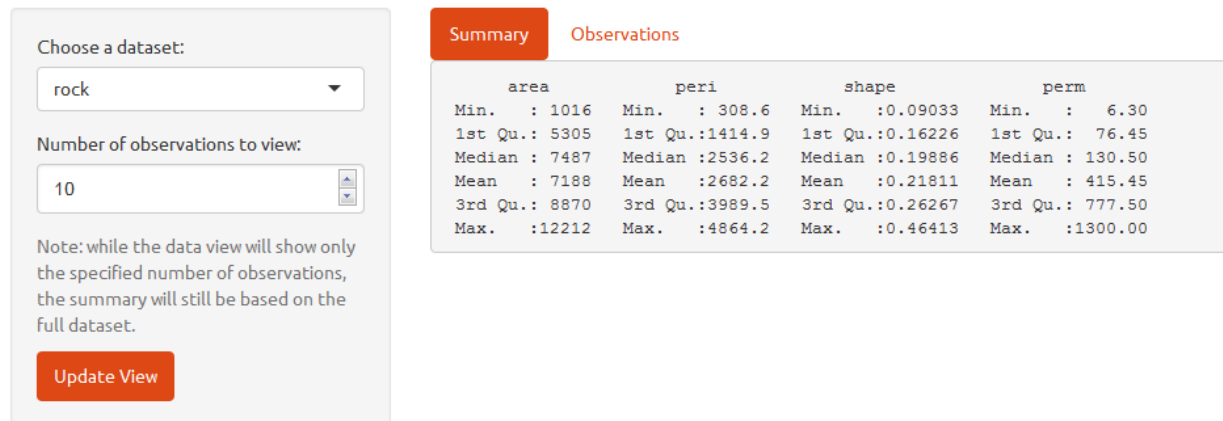


Figure 4:

### EJERCICIOS:

- Cambia el argumento `theme` a otra plantilla.
- Especifica un archivo CSS de la página <http://bootswatch.com/>

## Ejercicio

A partir de la aplicación de la parte III ó II:

- Añade pop-ups.
- Modifica el aspecto de algún botón ó *frame* (`wellPanel` p.e.).
- crea colapsables.
- Incorpora una password.
- Cambia el aspecto mediante `shinythemes`.

### Ejemplo Distribución

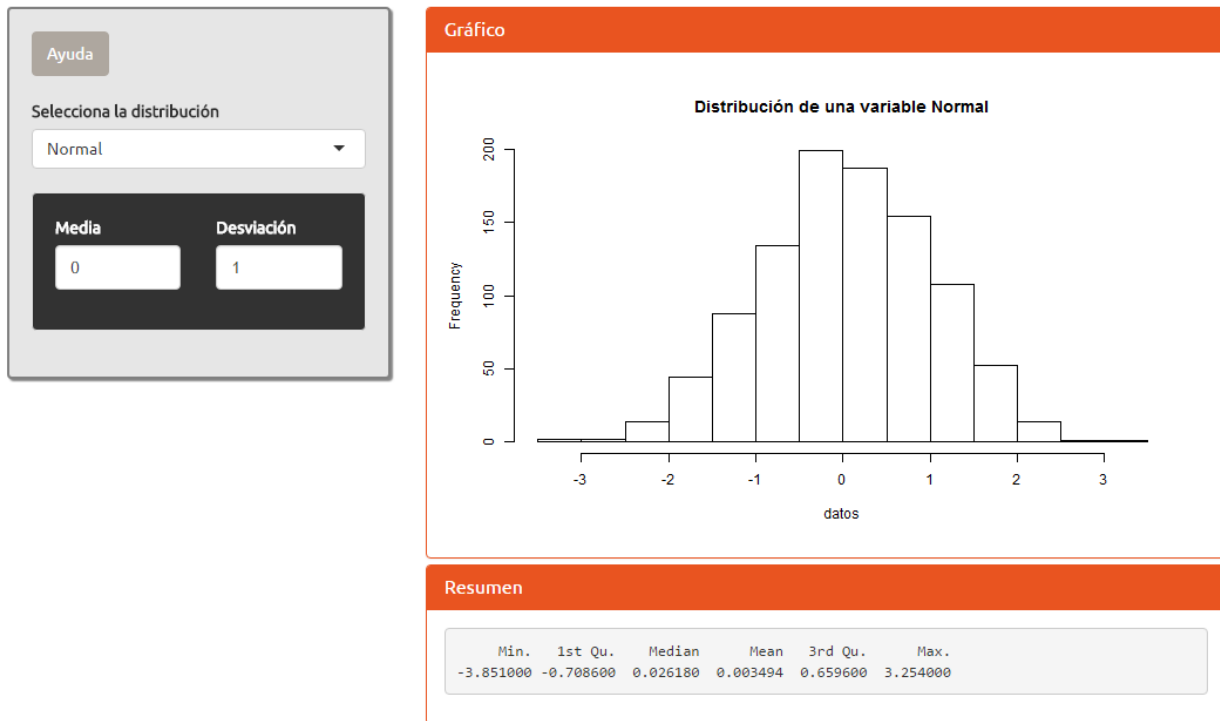


Figure 5: