R SHINY

Lesson 1:

Para hacer una aplicación es necesario ui.R (user interface) y server.R (contiene las instrucciones que el ordenador necesita para la app).

Lesson 2 Build a user-interface:

La estructura MÍNIMA para ui.R es: shinyUI(fluidPage( ))

La estructura MÍNIMA para server.R es: shinyServer(function(input, output) { })

- La función **fluidPage** permite crear un panel que automáticamente se ajusta a las dimensiones de la ventana del usuario. Comandos utilizados con fluidPage: **titlePanel; sidebarLayout; sidebarPanel**(si añades el comando helpText añades texto) y **mainPanel.** Los dos últimos son argumentos necesarios para sidebarLayout.

- La función **navbarPage** da una interfaz multi-página que incluye una barra de navegación.

- La función **fluidRow** y **column** para construir la disposición desde un sistema de cuadrículas.

HTML

p<p>A paragraph of text

h1<h1>A first level header

h2<h2>A second level header

h3<h3>A third level header

h4<h4>A fourth level header

h5<h5>A fifth level header

h6<h6>A sixth level header

a<a>A hyper link

br
A line break (e.g. a blank line)

div A division of text with a uniform style

spanAn in-line division of text with a uniform style

pre<pre>Text 'as is' in a fixed width font

code<code>A formatted block of code

img<img>An image

strongBold text

emItalicized text

HTML Directly passes a character string as HTML codeç

Se puede combinar con el comando **align = "center"**

Lesson 3 Control Widgets:

function	widget
actionButton	Action Button
checkboxGroupInput	A group of check boxes
checkboxInput	A single check box
dateInput	A calendar to aid date selection
dateRangeInput	A pair of calendars for selecting a date range
fileInput	A file upload control wizard
helpText	Help text that can be added to an input form
numericInput	A field to enter numbers
radioButtons	A set of radio buttons
selectInput	A box with choices to select from
sliderInput	A slider bar
submitButton	A submit button
textInput	A field to enter text

Para añadir widgets hay que añadir varios argumentos; los dos primeros son:

- Un nombre para el widget (para poder acceder a él fácilmente)
- Una etiqueta (es lo que aparece en la app)

Ejemplo:

```
selectInput("var",

        label = "Choose a variable to display",

        choices = list("Percent White", "Percent Black",

          "Percent Hispanic", "Percent Asian"),

        selected = "Percent White"),



      sliderInput("range",

        label = "Range of interest:",
```

```
        min = 0, max = 100, value = c(0, 100))
    ),
```

Lesson 4 Display reactive output:

- La función **textOutput** requiere un argumento que shiny emplea como nombre de tu element reactive.

render function	**creates**
`renderImage`	images (saved as a link to a source file)
`renderPlot`	plots
`renderPrint`	any printed output
`renderTable`	data frame, matrix, other table like structures
`renderText`	character strings
`renderUI`	a Shiny tag object or HTML

Output: es una lista de objetos que guarda instrucciones para construir objetos de R en la app

Input: es una segunda lista de objetos que guarda los valores actuales de los widgets de tu app.

Lesson 5 Use R scripts and data

Los sources, librerias y datas se colocan al comienzo de **server.R** ANTES de shinyserver, para que shiny lea esa información una sola vez.

Todos los objetos específicos de usuario han de colocarse fuera de **render calls**, ya que así se leera sólo una vez por usuario (como por ejemplo la información de la sesión).

Cada vez que un usuario use un widget dentro de un **render(shinyserver)**, shiny volverá a leer el código por tanto ha de evitarse introducir código innecesario ya que realentiza la app.

La función **Switch** ayuda para los widgets de elección múltiple, permitiendo cambiar los valores de un widget en expresiones R.

```
args <- switch(input$var,
        "Percent White" = list(counties$white, "darkgreen", "% White")
,
        "Percent Black" = list(counties$black, "black", "% Black"),
        "Percent Hispanic" = list(counties$hispanic, "darkorange", "%
Hispanic"),
        "Percent Asian" = list(counties$asian, "darkviolet", "% Asian"
))
```

Lesson 6 Use reactive expressions.

Permiten limitar que vuelve a leerse un objeto. La expresión reactiva actualizará el valor del widget cada vez que el widget original cambie. Se emplea la función **reactive.**

```
dataInput <- reactive({

  getSymbols(input$symb, src = "yahoo",

    from = input$dates[1],

    to = input$dates[2],

    auto.assign = FALSE)

})
```

Estas expresiones permiten guardar el valor anterior en tu ordenador, la próxima vez que llames a la expresión te devuelve el dato almacenado sin tener que computar.

Lesson 7 Share your apps

Run GitHub: introducir server.R y ui.R en el repositorio y luego los usuarios pueden lanzarlo utilizando: **runGitHub("nombre del repositorio","nombre del usuario")**

Página Web

Shinyapps.io

Shinyserver