

Desarrollo de aplicaciones web con shiny

Ana D. Maldonado

Departamento de Matemáticas
Área de Estadística e Investigación Operativa
Universidad de Almería
ana.d.maldonado@ual.es





Shiny

Ana D. Maldonado
ana.d.maldonado@ual.es

[Introducción](#)

[Estructura de una app](#)

[Widgets](#)

[Outputs](#)

[Reactividad](#)

[Layout](#)

[Publicación](#)

1. Introducción
2. Estructura de una aplicación Shiny
3. Widgets (inputs)
4. Salida reactiva (outputs)
5. Expresiones reactivas
6. Layout
7. Publicación



¿Qué es shiny?

Shiny

Ana D. Maldonado
ana.d.maldonado@ual.es

[Introducción](#)

[Estructura de una app](#)

[Widgets](#)

[Outputs](#)

[Reactividad](#)

[Layout](#)

[Publicación](#)

shiny es un paquete de R diseñado para crear aplicaciones web interactivas.



Para empezar a usar shiny, necesitamos instalar R, RStudio y el propio paquete:

```
# Instalar paquete shiny  
install.packages("shiny")
```

```
# Cargar paquete  
library("shiny")
```



Introducción

Shiny

Ana D. Maldonado
ana.d.maldonado@ual.es

[Introducción](#)

[Estructura de una app](#)

[Widgets](#)

[Outputs](#)

[Reactividad](#)

[Layout](#)

[Publicación](#)

Las aplicaciones de shiny están contenidas en un único script llamado `app.R`. Este script tiene 3 partes:

```
ui <- fluidPage()  
  
server <- function(input, output) {}  
  
shinyApp(ui = ui, server = server)
```

Interfaz:

controla el diseño y apariencia de la app

Server:

Instrucciones para construir la app

Crea la aplicación

También es posible escribir el objeto `ui` y la función `server` en scripts separados llamados `ui.R` y `server.R`, respectivamente.



Introducción

Shiny

Ana D. Maldonado
ana.d.maldonado@ual.es

[Introducción](#)

[Estructura de una app](#)

[Widgets](#)

[Outputs](#)

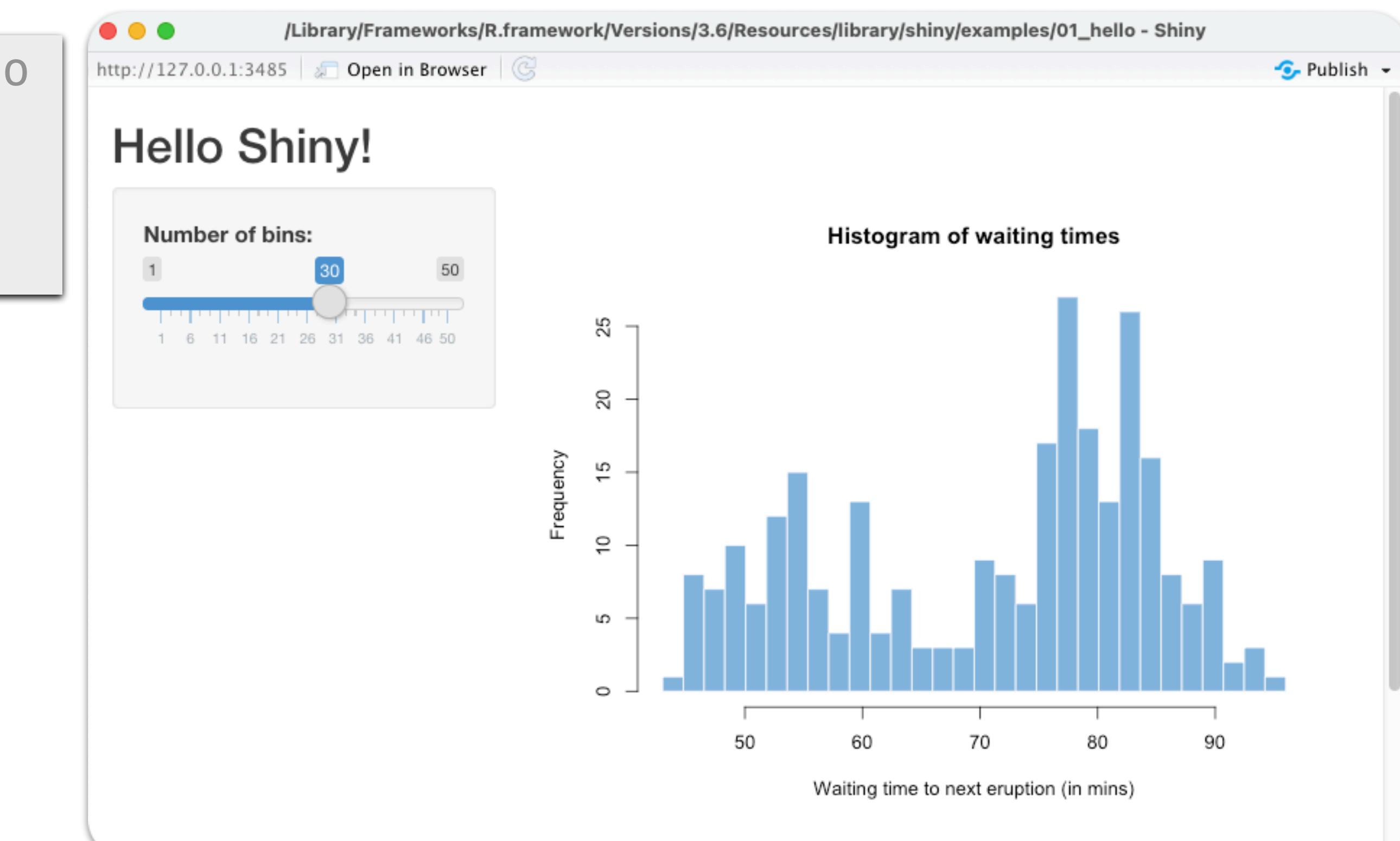
[Reactividad](#)

[Layout](#)

[Publicación](#)

```
# Ejecutar aplicación de ejemplo  
runExample("01_hello")
```

El paquete shiny contiene 11 ejemplos de aplicaciones que sirven para ilustrar cómo funciona shiny. Por ejemplo, la aplicación **Hello Shiny** muestra el histograma del dataset faithful, donde el usuario puede modificar el número de intervalos:





Shiny

Ana D. Maldonado
ana.d.maldonado@ual.es

[Introducción](#)

[Estructura de una app](#)

[Widgets](#)

[Outputs](#)

[Reactividad](#)

[Layout](#)

[Publicación](#)

1. Introducción
2. Estructura de una aplicación Shiny
 1. User Interface
 2. Server
 3. Widgets (inputs)
 4. Salida reactiva (outputs)
 5. Expresiones reactivas
 6. Layout
 7. Publicación



Estructura de una aplicación de shiny

Shiny

Ana D. Maldonado
ana.d.maldonado@ual.es

[Introducción](#)

[Estructura de una app](#)

[Widgets](#)

[Outputs](#)

[Reactividad](#)

[Layout](#)

[Publicación](#)

```
# Define UI for app that draws a histogram ----
ui <- fluidPage(

  # App title ----
  titlePanel("Hello Shiny!"),

  # Sidebar layout with input and output definitions ----
  sidebarLayout(

    # Sidebar panel for inputs ----
    sidebarPanel(

      # Input: Slider for the number of bins ----
      sliderInput(inputId = "bins",
                  label = "Number of bins:",
                  min = 1,
                  max = 50,
                  value = 30)

    ),

    # Main panel for displaying outputs ----
    mainPanel(

      # Output: Histogram ----
      plotOutput(outputId = "distPlot")

    )
  )
)
```



Estructura de una aplicación de shiny

Shiny

Ana D. Maldonado
ana.d.maldonado@ual.es

[Introducción](#)

[Estructura de una app](#)

[Widgets](#)

[Outputs](#)

[Reactividad](#)

[Layout](#)

[Publicación](#)

```
# Define UI for app that draws a histogram ----  
ui <- fluidPage(  
  
  # App title ----  
  titlePanel("Hello Shiny!"),  
  
  # Sidebar layout with input and output definitions ----  
  sidebarLayout(  
  
    # Sidebar panel for inputs ----  
    sidebarPanel(  
  
      # Input: Slider for the number of bins ----  
      sliderInput(inputId = "bins",  
                  label = "Number of bins:",  
                  min = 1,  
                  max = 50,  
                  value = 30)  
  
    ),  
  
    # Main panel for displaying outputs ----  
    mainPanel(  
  
      # Output: Histogram ----  
      plotOutput(outputId = "distPlot")  
  
    )  
  )  
)
```

Crea un diseño “fluid page”: el diseño de la página se ajusta automáticamente al tamaño de la ventana del navegador



Estructura de una aplicación de shiny

Shiny

Ana D. Maldonado
ana.d.maldonado@ual.es

[Introducción](#)

[Estructura de una app](#)

[Widgets](#)

[Outputs](#)

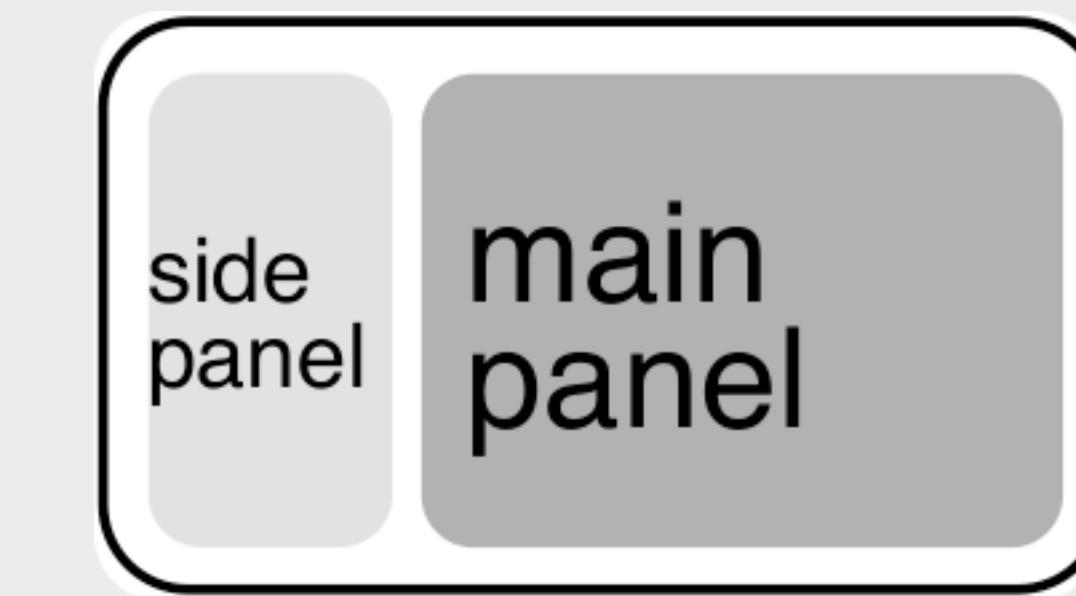
[Reactividad](#)

[Layout](#)

[Publicación](#)

```
# Define UI for app that draws a histogram ----  
ui <- fluidPage(  
  
  # App title ----  
  titlePanel("Hello Shiny!"),  
  
  # Sidebar layout with input and output definitions ----  
  sidebarLayout(  
  
    # Sidebar panel for inputs ----  
    sidebarPanel(  
  
      # Input: Slider for the number of bins ----  
      sliderInput(inputId = "bins",  
                  label = "Number of bins:",  
                  min = 1,  
                  max = 50,  
                  value = 30)  
  
    ),  
  
    # Main panel for displaying outputs ----  
    mainPanel(  
  
      # Output: Histogram ----  
      plotOutput(outputId = "distPlot")  
  
    )  
  )  
)
```

Crea un diseño con barra lateral y panel principal. Siempre toma 2 argumentos: sidebarPanel y mainPanel





Estructura de una aplicación de shiny

Shiny

Ana D. Maldonado
ana.d.maldonado@ual.es

[Introducción](#)

[Estructura de una app](#)

[Widgets](#)

[Outputs](#)

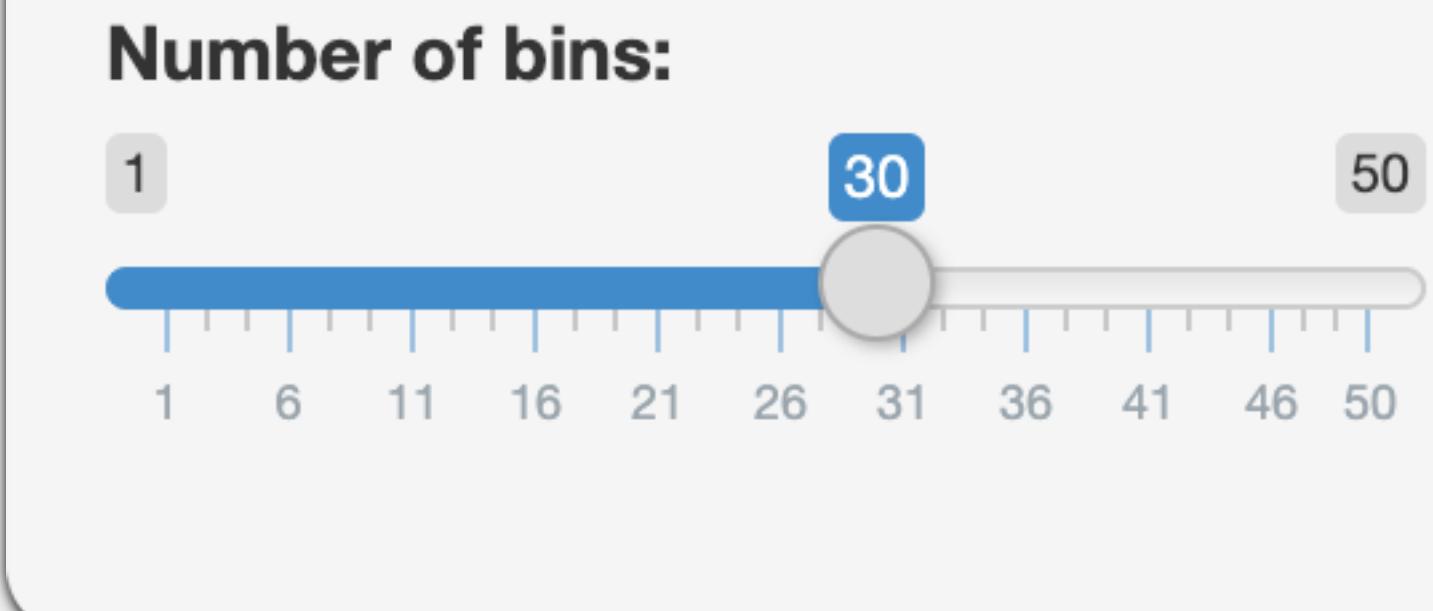
[Reactividad](#)

[Layout](#)

[Publicación](#)

```
# Define UI for app that draws a histogram ----  
ui <- fluidPage(  
  
  # App title ----  
  titlePanel("Hello Shiny!"),  
  
  # Sidebar layout with input and output definitions ----  
  sidebarLayout(  
  
    # Sidebar panel for inputs ----  
    sidebarPanel(  
  
      # Input: Slider for the number of bins ----  
      sliderInput(inputId = "bins",  
                  label = "Number of bins:",  
                  min = 1,  
                  max = 50,  
                  value = 30)  
  
    ),  
  
    # Main panel for displaying outputs ----  
    mainPanel(  
  
      # Output: Histogram ----  
      plotOutput(outputId = "distPlot")  
  
    )  
  )  
)
```

Barra lateral: lugar donde se ponen los widgets (**inputs**) con los que el usuario puede interaccionar





Estructura de una aplicación de shiny

Shiny

Ana D. Maldonado
ana.d.maldonado@ual.es

[Introducción](#)

[Estructura de una app](#)

[Widgets](#)

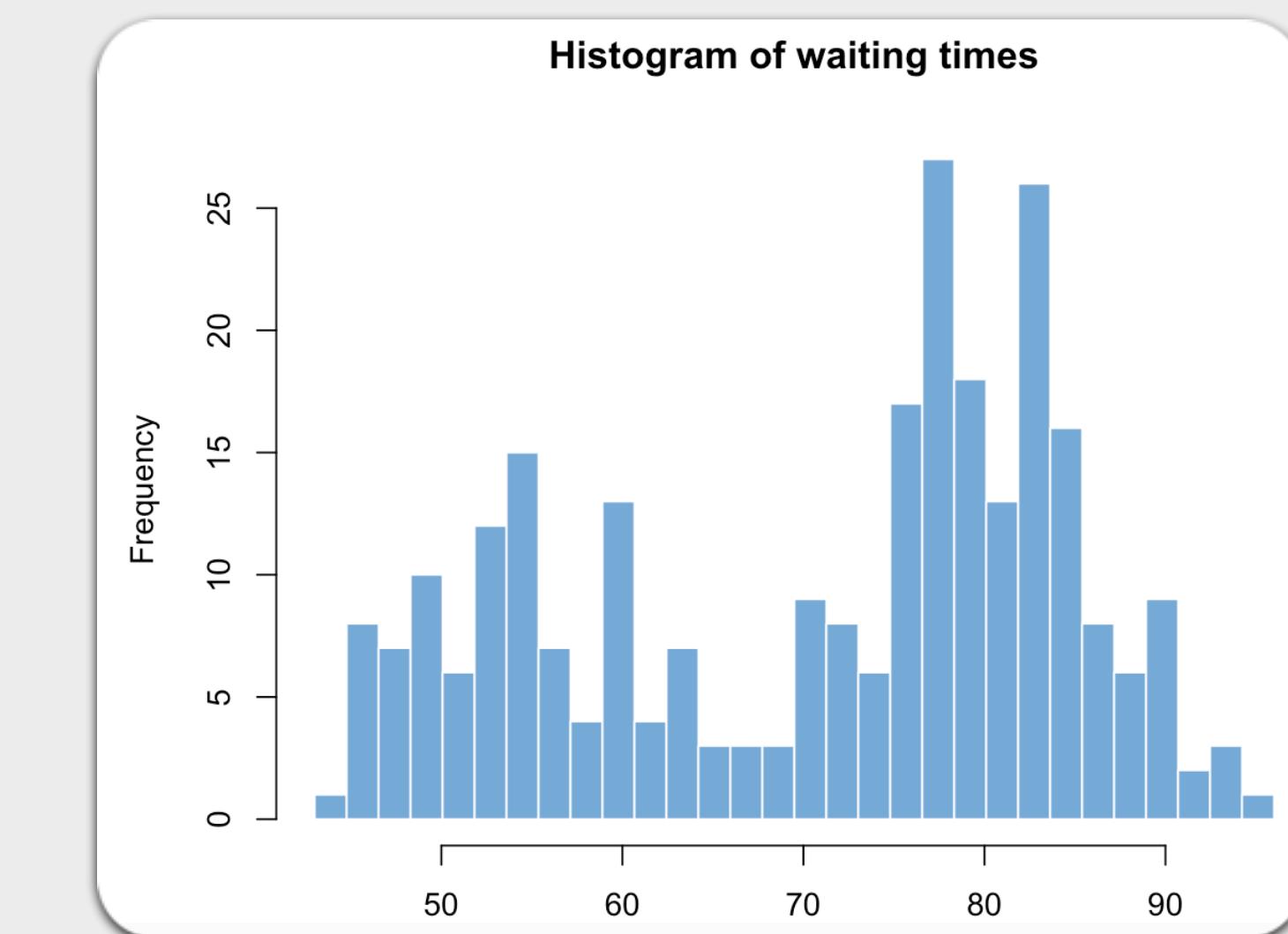
[Outputs](#)

[Reactividad](#)

[Layout](#)

[Publicación](#)

```
# Define UI for app that draws a histogram ----  
ui <- fluidPage(  
  
  # App title ----  
  titlePanel("Hello Shiny!"),  
  
  # Sidebar layout with input and output definitions ----  
  sidebarLayout(  
  
    # Sidebar panel for inputs ----  
    sidebarPanel(  
  
      # Input: Slider for the number of bins ----  
      sliderInput(inputId = "bins",  
                  label = "Number of bins:",  
                  min = 1,  
                  max = 50,  
                  value = 30)  
  
    ),  
  
    # Main panel for displaying outputs ----  
    mainPanel(  
  
      # Output: Histogram ----  
      plotOutput(outputId = "distPlot")  
    )  
  )  
)
```



Panel principal: lugar donde se muestran los resultados (**outputs**), que se crean en la función server



Estructura de una aplicación de shiny

Shiny

Ana D. Maldonado
ana.d.maldonado@ual.es

[Introducción](#)

[Estructura de una app](#)

[Widgets](#)

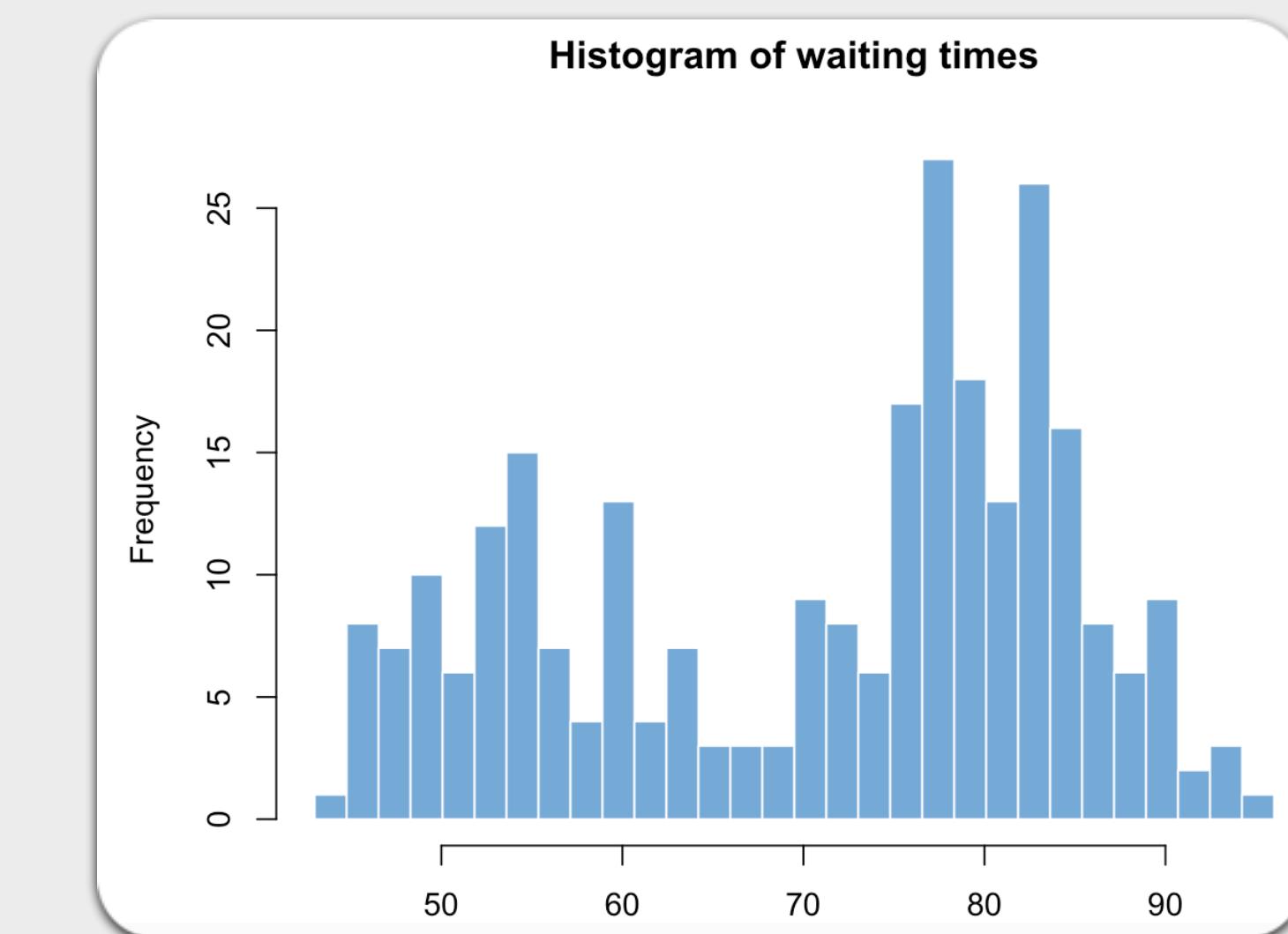
[Outputs](#)

[Reactividad](#)

[Layout](#)

[Publicación](#)

```
# Define UI for app that draws a histogram ----  
ui <- fluidPage(  
  
  # App title ----  
  titlePanel("Hello Shiny!"),  
  
  # Sidebar layout with input and output definitions ----  
  sidebarLayout(  
  
    # Sidebar panel for inputs ----  
    sidebarPanel(  
  
      # Input: Slider for the number of bins ----  
      sliderInput(inputId = "bins",  
                  label = "Number of bins:",  
                  min = 1,  
                  max = 50,  
                  value = 30)  
  
    ),  
  
    # Main panel for displaying outputs ----  
    mainPanel(  
  
      # Output: Histogram ----  
      plotOutput(outputId = "distPlot")  
  
    )  
  )  
)
```



Panel principal: lugar donde se muestran los resultados (**outputs**), que se crean en la función server

UI recibe de server un objeto de salida llamado “distPlot”. La función plotOutput construye una salida de tipo “plot”, a partir de “distPlot”.



Shiny

Ana D. Maldonado
ana.d.maldonado@ual.es

[Introducción](#)

[Estructura de una app](#)

[Widgets](#)

[Outputs](#)

[Reactividad](#)

[Layout](#)

[Publicación](#)

1. Introducción
2. Estructura de una aplicación Shiny
 1. User Interface
 2. Server
 3. Widgets (inputs)
 4. Salida reactiva (outputs)
 5. Expresiones reactivas
 6. Layout
 7. Publicación



Estructura de una aplicación de shiny

Shiny

Ana D. Maldonado
ana.d.maldonado@ual.es

[Introducción](#)

[Estructura de una app](#)

[Widgets](#)

[Outputs](#)

[Reactividad](#)

[Layout](#)

[Publicación](#)

```
# Define server logic required to draw a histogram ----
server <- function(input, output) {

  # Histogram of the Old Faithful Geyser Data ----
  # with requested number of bins
  # This expression that generates a histogram is wrapped in a call
  # to renderPlot to indicate that:
  #
  # 1. It is "reactive" and therefore should be automatically
  #    re-executed when inputs (input$bins) change
  # 2. Its output type is a plot
  output$distPlot <- renderPlot({

    x      <- faithful$waiting
    bins   <- seq(min(x), max(x), length.out = input$bins + 1)

    hist(x, breaks = bins, col = "#75AADB", border = "white",
          xlab = "Waiting time to next eruption (in mins)",
          main = "Histogram of waiting times")

  })
}
```



Estructura de una aplicación de shiny

Shiny

Ana D. Maldonado
ana.d.maldonado@ual.es

[Introducción](#)

[Estructura de una app](#)

[Widgets](#)

[Outputs](#)

[Reactividad](#)

[Layout](#)

[Publicación](#)

```
# Define server logic required to draw a histogram ----
server <- function(input, output) {

  # Histogram of the Old Faithful Geyser Data ----
  # with requested number of bins
  # This expression that generates a histogram is wrapped in a call
  # to renderPlot to indicate that:
  #
  # 1. It is "reactive" and therefore should be automatically
  #    re-executed when inputs (input$bins) change
  # 2. Its output type is a plot
  output$distPlot <- renderPlot({

    x      <- faithful$waiting
    bins   <- seq(min(x), max(x), length.out = input$bins + 1)

    hist(x, breaks = bins, col = "#75AADB", border = "white",
          xlab = "Waiting time to next eruption (in mins)",
          main = "Histogram of waiting times")

  })
}
```

Contiene las instrucciones para construir y actualizar los objetos

output



Estructura de una aplicación de shiny

Shiny

Ana D. Maldonado
ana.d.maldonado@ual.es

[Introducción](#)

[Estructura de una app](#)

[Widgets](#)

[Outputs](#)

[Reactividad](#)

[Layout](#)

[Publicación](#)

```
# Define server logic required to draw a histogram ----  
server <- function(input, output) {  
  
  # Histogram of the Old Faithful Geyser Data ----  
  # with requested number of bins  
  # This expression that generates a histogram is wrapped in a call  
  # to renderPlot to indicate that:  
  #  
  # 1. It is "reactive" and therefore should be automatically  
  #    re-executed when inputs (input$bins) change  
  # 2. Its output type is a plot  
  output$distPlot <- renderPlot({  
  
    x      <- faithful$waiting  
    bins   <- seq(min(x), max(x), length.out = input$bins + 1)  
  
    hist(x, breaks = bins, col = "#75AADB", border = "white",  
          xlab = "Waiting time to next eruption (in mins)",  
          main = "Histogram of waiting times")  
  })  
}
```

Genera un gráfico “reactivo”
(dinámico)



Estructura de una aplicación de shiny

Shiny

Ana D. Maldonado
ana.d.maldonado@ual.es

[Introducción](#)

[Estructura de una app](#)

[Widgets](#)

[Outputs](#)

[Reactividad](#)

[Layout](#)

[Publicación](#)

```
# Define server logic required to draw a histogram ----  
server <- function(input, output) {  
  
  # Histogram of the Old Faithful Geyser Data ----  
  # with requested number of bins  
  # This expression that generates a histogram is wrapped in a call  
  # to renderPlot to indicate that:  
  #  
  # 1. It is "reactive" and therefore should be automatically  
  #    re-executed when inputs (input$bins) change  
  # 2. Its output type is a plot  
  output$distPlot <- renderPlot({  
  
    x   <- faithful$waiting  
    bins <- seq(min(x), max(x), length.out = input$bins + 1)  
  
    hist(x, breaks = bins, col = "#75AADB", border = "white",  
          xlab = "Waiting time to next eruption (in mins)",  
          main = "Histogram of waiting times")  
  })  
}
```

Genera un gráfico “reactivo”
(dinámico)

Las instrucciones para construir el histograma se
guardan en output\$distPlot. Este elemento
corresponde con plotOutput("distplot") del UI.



Estructura de una aplicación de shiny

Shiny

Ana D. Maldonado
ana.d.maldonado@ual.es

Introducción

Estructura de una app

Widgets

Outputs

Reactividad

Layout

Publicación

```
# Define server logic required to draw a histogram ----  
server <- function(input, output) {  
  
  # Histogram of the Old Faithful Geyser Data ----  
  # with requested number of bins  
  # This expression that generates a histogram is wrapped in a call  
  # to renderPlot to indicate that:  
  #  
  # 1. It is "reactive" and therefore should be automatically  
  #    re-executed when inputs (input$bins) change  
  # 2. Its output type is a plot  
  output$distPlot <- renderPlot({  
  
    x      <- faithful$waiting  
    bins   <- seq(min(x), max(x), length.out = input$bins + 1)  
  
    hist(x, breaks = bins, col = "#75AADB", border = "white",  
          xlab = "Waiting time to next eruption (in mins)",  
          main = "Histogram of waiting times")  
  })  
}
```

Genera un gráfico “reactivo”
(dinámico)

input\$bins corresponde con sliderInput("bins",...) del UI.
Cada vez que el usuario mueva el deslizador,
el histograma se actualizará



¡A PRACTICAR!

Shiny

Ana D. Maldonado
ana.d.maldonado@ual.es

[Introducción](#)

[Estructura de una app](#)

[Widgets](#)

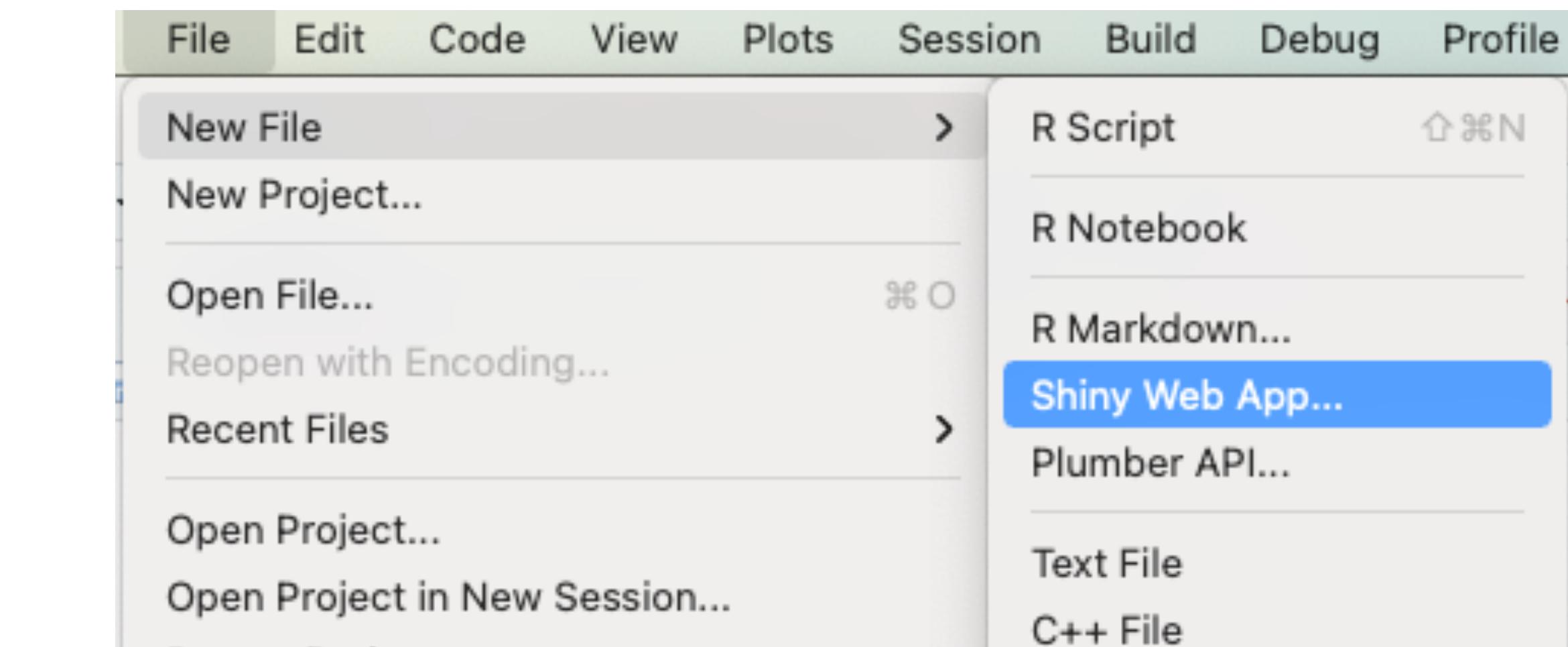
[Outputs](#)

[Reactividad](#)

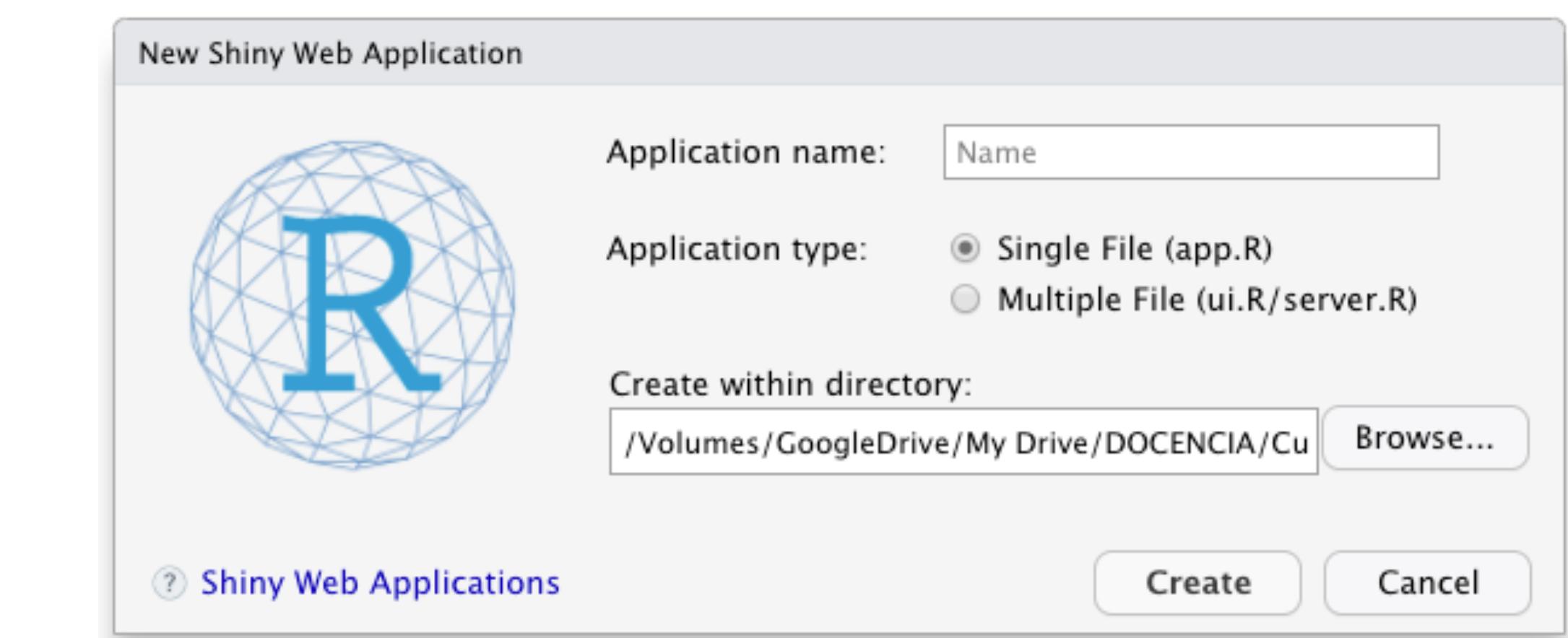
[Layout](#)

[Publicación](#)

- Crea un nuevo script Shiny Web App



- En *Application name*, escribe App-1
- En *Application type*, selecciona Single File
- En *Create within directory*, elige el directorio donde crear la carpeta App-1 que contendrá el archivo app.R.





¡A PRACTICAR!

Shiny

Ana D. Maldonado
ana.d.maldonado@ual.es

[Introducción](#)

[Estructura de una app](#)

[Widgets](#)

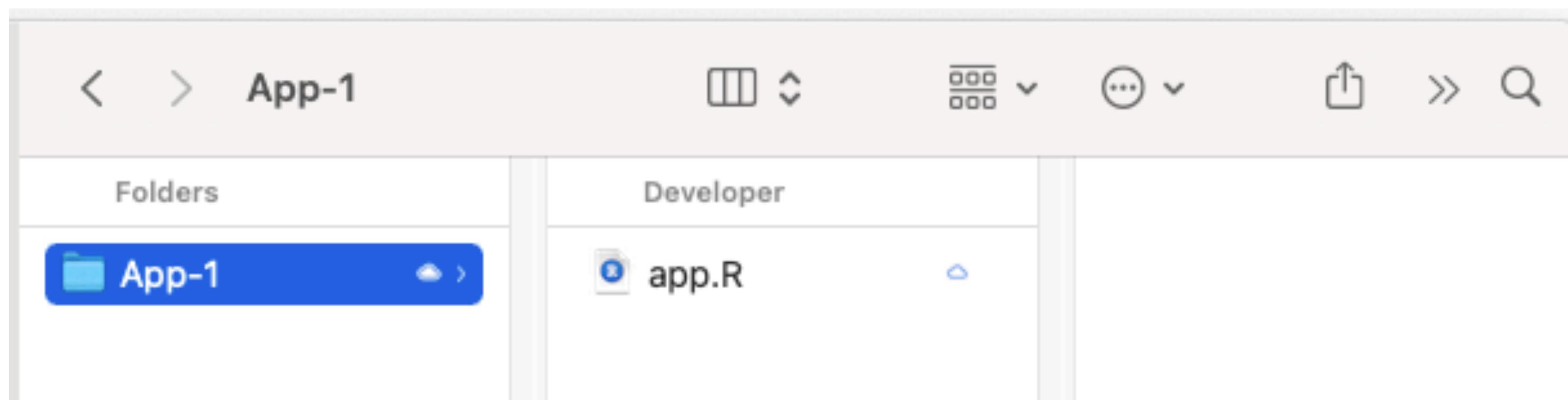
[Outputs](#)

[Reactividad](#)

[Layout](#)

[Publicación](#)

- La carpeta y el archivo aparecen en el directorio indicado.



- Cuando se crea un archivo de shiny, por defecto viene con una versión de la aplicación Hello Shiny!
- Abre el archivo app.R en RStudio y pulsa sobre el botón Run App ▾ .
- Para que se muestre el código que hay debajo de la app, ejecuta en un script o la consola:

```
runApp("App-1", display.mode = "showcase")
```

- Recuerda establecer el directorio de trabajo primero!



¡A PRACTICAR!

Shiny

Ana D. Maldonado
ana.d.maldonado@ual.es

[Introducción](#)

[Estructura de una app](#)

[Widgets](#)

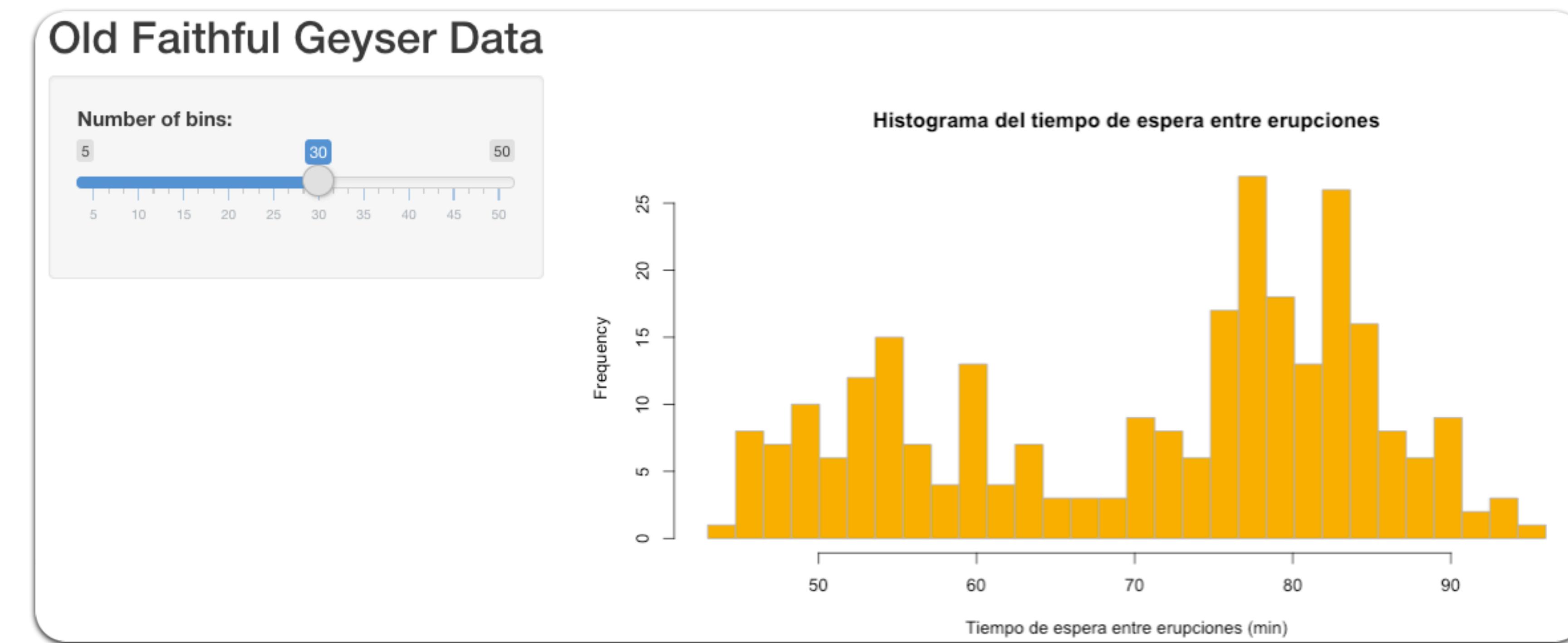
[Outputs](#)

[Reactividad](#)

[Layout](#)

[Publicación](#)

- Cambia el valor mínimo de la barra deslizadora a 5.
- Cambia el color del histograma: barras color naranja y borde gris oscuro.
- Cambia el nombre del histograma.
- Cambia la etiqueta del eje x del histograma.





¡A PRACTICAR! (Solución)

Shiny

Ana D. Maldonado
ana.d.maldonado@ual.es

[Introducción](#)

[Estructura de una app](#)

[Widgets](#)

[Outputs](#)

[Reactividad](#)

[Layout](#)

[Publicación](#)

```
# Define UI for application that draws a histogram ----
ui <- fluidPage

  # Application title ----
  titlePanel("Old Faithful Geyser Data"),

  # Sidebar layout with slider input for number of bins ----
  sidebarLayout(
    sidebarPanel(
      sliderInput(inputId = "bins",
                  label = "Number of bins:",
                  min = 5,
                  max = 50,
                  value = 30)
    ),
    # Show a plot of the generated distribution ----
    mainPanel(
      plotOutput(outputId = "distPlot")
    )
  )
}
```



¡A PRACTICAR! (Solución)

Shiny

Ana D. Maldonado
ana.d.maldonado@ual.es

[Introducción](#)

[Estructura de una app](#)

[Widgets](#)

[Outputs](#)

[Reactividad](#)

[Layout](#)

[Publicación](#)

```
# Define server logic required to draw a histogram ----
server <- function(input, output) {

  output$distPlot <- renderPlot({

    # generate bins based on input$bins from ui.R
    x      <- faithful[, 2]
    bins <- seq(min(x), max(x), length.out = input$bins + 1)

    # draw the histogram with the specified number of bins
    hist(x, breaks = bins, col = "orange", border = "darkgrey",
          main = "Histograma del tiempo de espera",
          xlab = "Tiempo de espera entre erupciones (min)")

  })
}
```



Shiny

Ana D. Maldonado
ana.d.maldonado@ual.es

[Introducción](#)

[Estructura de una app](#)

[Widgets](#)

[Outputs](#)

[Reactividad](#)

[Layout](#)

[Publicación](#)

1. Introducción
2. Estructura de una aplicación Shiny
3. Widgets (inputs)
4. Salida reactiva (outputs)
5. Expresiones reactivas
6. Layout
7. Publicación



Widgets

Shiny

Ana D. Maldonado
ana.d.maldonado@ual.es

[Introducción](#)

[Estructura de una app](#)

[Widgets](#)

[Outputs](#)

[Reactividad](#)

[Layout](#)

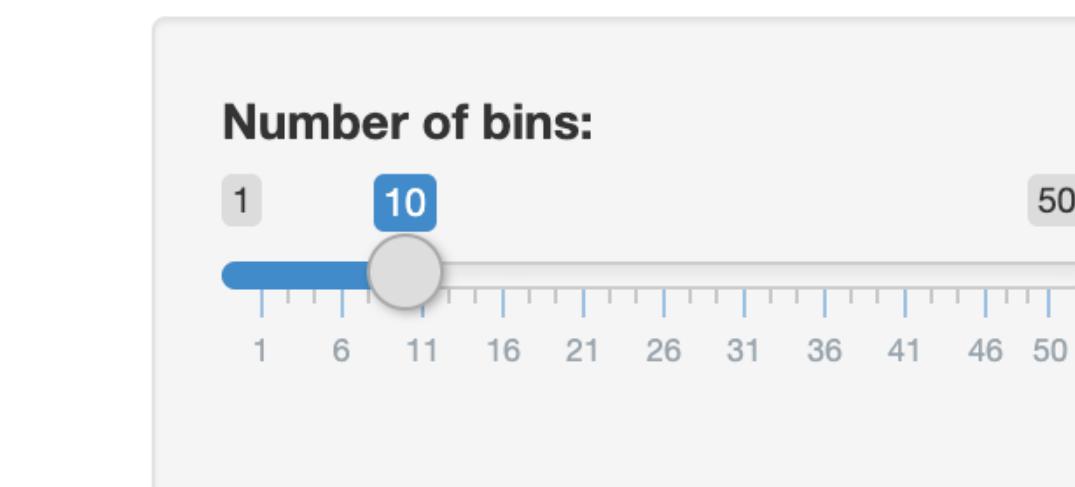
[Publicación](#)

Los widgets son elementos de la app con los que los usuarios pueden interactuar. Estos widgets recogen un valor dado por el usuario, que se almacena en una lista llamada `input`. Los valores de los widgets se guardan con el nombre se se especifica en el argumento `inputId`.

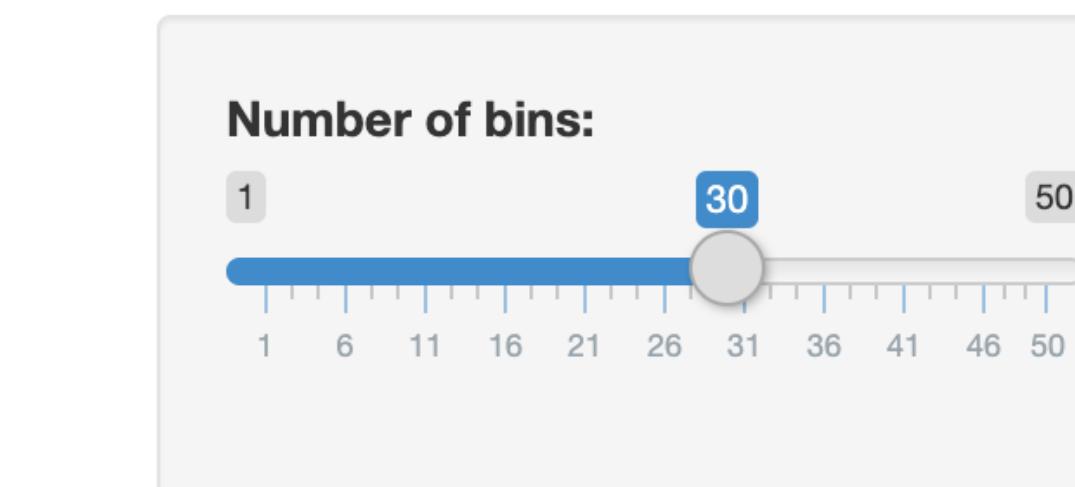
Por ejemplo, si nuestra app tiene 1 widget llamado “bins”, sus valores se almacenan en la lista `input` como `input$bins`.

```
# Input: Slider for the number of bins ----  
sliderInput(inputId = "bins",  
            label = "Number of bins:",  
            min = 1,  
            max = 50,  
            value = 30)
```

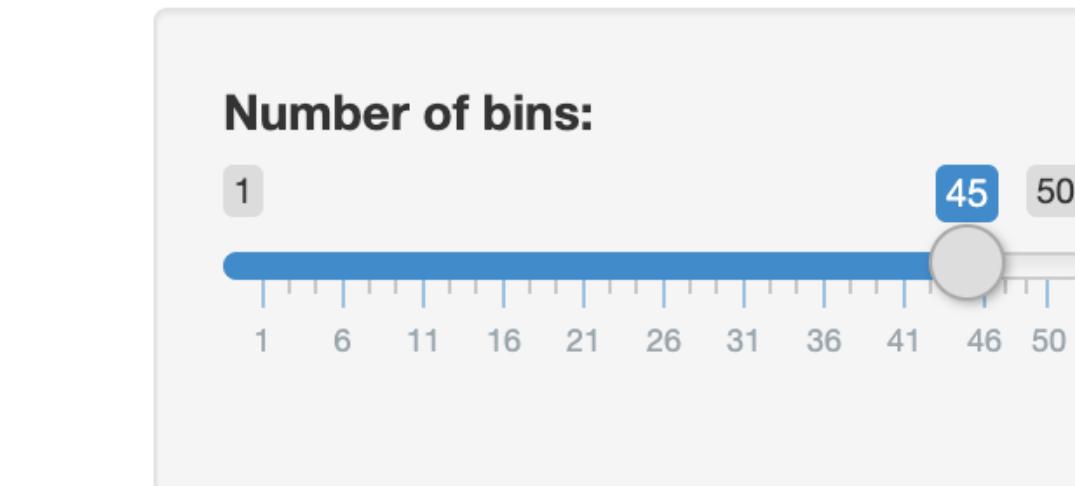
`input$bins`



input\$bins = 10



input\$bins = 30



input\$bins = 45



Widgets

Shiny

Ana D. Maldonado
ana.d.maldonado@ual.es

[Introducción](#)

[Estructura de una app](#)

[Widgets](#)

[Outputs](#)

[Reactividad](#)

[Layout](#)

[Publicación](#)



actionButton(inputId, label, icon, ...)



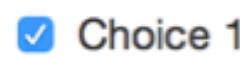
numericInput(inputId, label, value, min, max, step)



actionLink(inputId, label, icon, ...)



passwordInput(inputId, label, value)



checkboxGroupInput(inputId, label, choices, selected, inline)



radioButtons(inputId, label, choices, selected, inline)



checkboxInput(inputId, label, value)



selectInput(inputId, label, choices, selected, multiple, selectize, width, size) (also **selectizeInput()**)



dateInput(inputId, label, value, min, max, format, startview, weekstart, language)



sliderInput(inputId, label, min, max, value, step, round, format, locale, ticks, animate, width, sep, pre, post)



dateRangeInput(inputId, label, start, end, min, max, format, startview, weekstart, language, separator)



submitButton(text, icon)
(Prevents reactions across entire app)



fileInput(inputId, label, multiple, accept)



textInput(inputId, label, value)



Widgets

Shiny

Ana D. Maldonado
ana.d.maldonado@ual.es

[Introducción](#)

[Estructura de una app](#)

[Widgets](#)

[Outputs](#)

[Reactividad](#)

[Layout](#)

[Publicación](#)



`actionButton(inputId, label, icon, ...)`



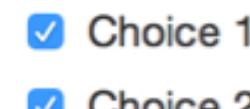
`numericInput(inputId, label, value, min, max, step)`



`actionLink(inputId, label, icon, ...)`



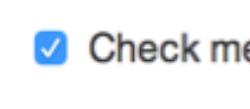
`passwordInput(inputId, label, value)`



`checkboxGroupInput(inputId, label, choices, selected, inline)`



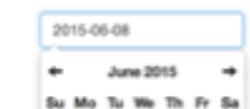
`radioButtons(inputId, label, choices, selected, inline)`



`checkboxInput(inputId, label, value)`



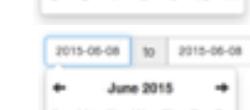
`selectInput(inputId, label, choices, selected, multiple, selectize, width, size) (also selectizeInput())`



`dateInput(inputId, label, value, min, max, format, startview, weekstart, language)`



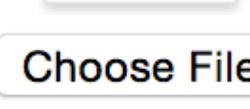
`sliderInput(inputId, label, min, max, value, step, round, format, locale, ticks, animate, width, sep, pre, post)`



`dateRangeInput(inputId, label, start, end, min, max, format, startview, weekstart, language, separator)`



`submitButton(text, icon)`
(Prevents reactions across entire app)



`fileInput(inputId, label, multiple, accept)`



`textInput(inputId, label, value)`

inputId: es el nombre del widget y se usa para acceder su valor en la función server.



Widgets

Shiny

Ana D. Maldonado
ana.d.maldonado@ual.es

[Introducción](#)

[Estructura de una app](#)

[Widgets](#)

[Outputs](#)

[Reactividad](#)

[Layout](#)

[Publicación](#)



`actionButton(inputId, label, icon, ...)`



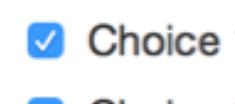
`numericInput(inputId, label, value, min, max, step)`



`actionLink(inputId, label, icon, ...)`



`passwordInput(inputId, label, value)`



`checkboxGroupInput(inputId, label, choices, selected, inline)`



`radioButtons(inputId, label, choices, selected, inline)`



`checkboxInput(inputId, label, value)`



`selectInput(inputId, label, choices, selected, multiple, selectize, width, size) (also selectizeInput())`



`dateInput(inputId, label, value, min, max, format, startview, weekstart, language)`



`sliderInput(inputId, label, min, max, value, step, round, format, locale, ticks, animate, width, sep, pre, post)`



`dateRangeInput(inputId, label, start, end, min, max, format, startview, weekstart, language, separator)`



`submitButton(text, icon)`
(Prevents reactions across entire app)



`fileInput(inputId, label, multiple, accept)`



`textInput(inputId, label, value)`

inputId: es el nombre del widget y se usa para acceder su valor en la función server.

label: es la etiqueta del widget que se muestra en la app. Si no queremos que aparezca nada, ponemos "".



Widgets

Shiny

Ana D. Maldonado
ana.d.maldonado@ual.es

[Introducción](#)

[Estructura de una app](#)

[Widgets](#)

[Outputs](#)

[Reactividad](#)

[Layout](#)

[Publicación](#)

<p>Action</p> <p><code>actionButton(inputId, label, icon, ...)</code></p> <p><code>actionLink(inputId, label, icon, ...)</code></p> <p><code>checkboxGroupInput(inputId, label, choices, selected, inline)</code></p> <p><code>checkboxInput(inputId, label, value)</code></p> <p><code>dateInput(inputId, label, value, min, max, format, startview, weekstart, language)</code></p> <p><code>dateRangeInput(inputId, label, start, end, min, max, format, startview, weekstart, language, separator)</code></p> <p><code>fileInput(inputId, label, multiple, accept)</code></p>	<p>.....</p> <p>1</p> <p>.....</p> <p>Choice A Choice B Choice C</p> <p>Choice 1 Choice 1 Choice 2</p> <p>0 5 10</p> <p>Apply Changes</p> <p>Enter text</p>	<p><code>numericInput(inputId, label, value, min, max, step)</code></p> <p><code>passwordInput(inputId, label, value)</code></p> <p><code>radioButtons(inputId, label, choices, selected, inline)</code></p> <p><code>selectInput(inputId, label, choices, selected, multiple, selectize, width, size) (also <code>selectizeInput()</code>)</code></p> <p><code>sliderInput(inputId, label, min, max, value, step, round, format, locale, ticks, animate, width, sep, pre, post)</code></p> <p><code>submitButton(text, icon)</code> (Prevents reactions across entire app)</p> <p><code>textInput(inputId, label, value)</code></p>
---	---	--

inputId: es el nombre del widget y se usa para acceder su valor en la función server.

label: es la etiqueta del widget que se muestra en la app. Si no queremos que aparezca nada, ponemos “”.

El resto de argumentos varía, según el widget: choices, value, min, max, ...



¡A PRACTICAR!

Shiny

Ana D. Maldonado
ana.d.maldonado@ual.es

[Introducción](#)

[Estructura de una app](#)

[Widgets](#)

[Outputs](#)

[Reactividad](#)

[Layout](#)

[Publicación](#)

- Añade un widget a App-1 que permita seleccionar la variable que queremos representar en el histograma.

Plot variable:

eruptions

waiting

- Añade un widget que permita seleccionar el tipo de gráfico, con las opciones histograma y diagrama de dispersión.

Select plot:

Histogram

Histogram

Scatterplot



¡A PRACTICAR! (Solución)

Shiny

Ana D. Maldonado
ana.d.maldonado@ual.es

[Introducción](#)

[Estructura de una app](#)

[Widgets](#)

[Outputs](#)

[Reactividad](#)

[Layout](#)

[Publicación](#)

```
# Define UI for app that draws a histogram ----
ui <- fluidPage

  # Application title ----
  titlePanel("Old Faithful Geyser Data"),

  # Sidebar with a slider input for number of bins ----
  sidebarLayout(
    sidebarPanel(
      sliderInput(inputId = "bins",
                  label = "Number of bins:",
                  min = 5,
                  max = 50,
                  value = 30),

    # selectInput y radioButtons pueden servir para seleccionar la variable
    radioButtons("variable", "Plot variable:",
                choices = colnames(faithful))

  ),

  # Show a plot of the generated distribution ----
  mainPanel(
    plotOutput(outputId = "distPlot")
  )
)
```



¡A PRACTICAR! (Solución)

Shiny

Ana D. Maldonado
ana.d.maldonado@ual.es

[Introducción](#)

[Estructura de una app](#)

[Widgets](#)

[Outputs](#)

[Reactividad](#)

[Layout](#)

[Publicación](#)

```
# Define UI for app that draws a histogram ----
ui <- fluidPage(

  # Application title ----
  titlePanel("Old Faithful Geyser Data"),

  # Sidebar with a slider input for number of bins ----
  sidebarLayout(
    sidebarPanel(
      sliderInput(inputId = "bins",
                  label = "Number of bins:",
                  min = 5,
                  max = 50,
                  value = 30),

      radioButtons("variable", "Plot variable:",
                  choices = colnames(faithful)),

      selectInput("plotType", "Select plot:",
                  choices = c("Histogram", "Scatterplot"))

    ),

    # Show a plot of the generated distribution ----
    mainPanel(
      plotOutput(outputId = "distPlot")
    )
  )
)
```



Shiny

Ana D. Maldonado
ana.d.maldonado@ual.es

[Introducción](#)

[Estructura de una app](#)

[Widgets](#)

[Outputs](#)

[Reactividad](#)

[Layout](#)

[Publicación](#)

1. Introducción
2. Estructura de una aplicación Shiny
3. Widgets (inputs)
4. Salida reactiva (outputs)
5. Expresiones reactivas
6. Layout
7. Publicación



Outputs

Shiny

Ana D. Maldonado
ana.d.maldonado@ual.es

[Introducción](#)

[Estructura de una app](#)

[Widgets](#)

[Outputs](#)

[Reactividad](#)

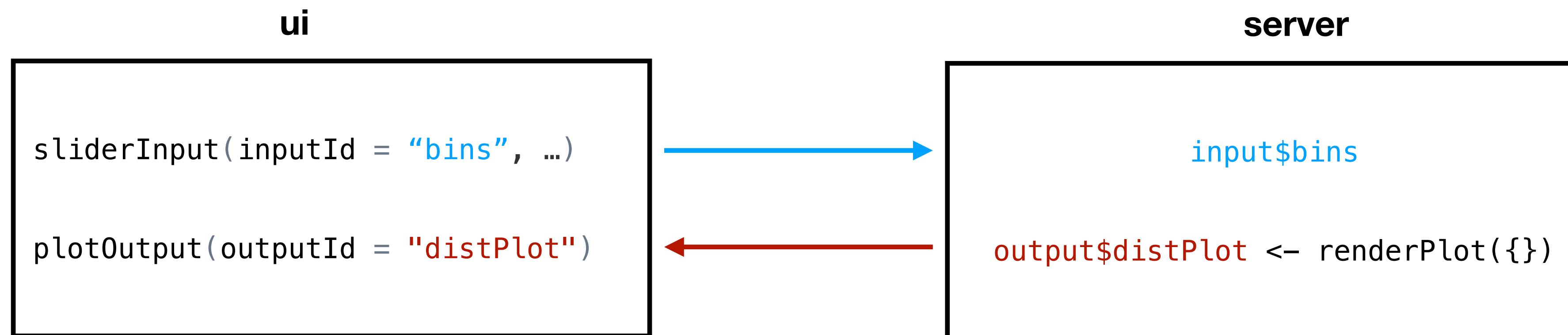
[Layout](#)

[Publicación](#)

Una salida reactiva es un *output* que se actualiza cuando el usuario cambia el valor de un widget. Para crear una salida reactiva, necesitamos:

1. En el UI, crear un widget (input), mediante alguna función `*Input()`, por ejemplo `sliderInput()`.
2. En el server, usar una función `render*`(), por ejemplo `renderPlot()`, para indicar cómo construir el objeto de salida. Estas instrucciones se guardan en la lista `output`, que tiene una entrada por cada objeto reactivo en la aplicación. Si las instrucciones contienen algún valor **input**, el **output** será reactivo.
3. En el UI, usar alguna función `*Output()`, por ejemplo `plotOutput()`, para colocar el output reactivo en la aplicación final.

Conectando inputs con outputs se crea la reactividad en Shiny.





Outputs

Shiny

Ana D. Maldonado
ana.d.maldonado@ual.es

[Introducción](#)

[Estructura de una app](#)

[Widgets](#)

[Outputs](#)

[Reactividad](#)

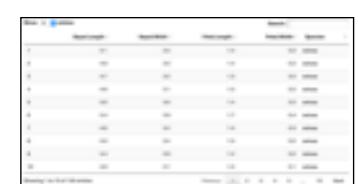
[Layout](#)

[Publicación](#)

Las funciones que convierten objetos de R en outputs para la interfaz de usuario se eligen según el tipo de salida que queramos:

Outputs - render*() and *Output() functions work together to add R output to the UI

Tipo de output



`DT::renderDataTable(expr,
options, callback, escape,
env, quoted)`

works
with

`dataTableOutput(outputId, icon, ...)`

Tabla interactiva

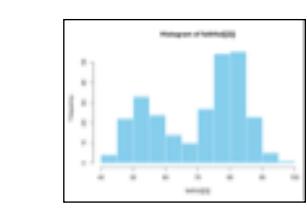
<https://shiny.rstudio.com/gallery/basic-datatable.html>



`renderImage(expr, env, quoted, deleteFile)`

Imagen

<https://shiny.rstudio.com/gallery/image-output.html>



`renderPlot(expr, width, height, res, ..., env,
quoted, func)`

`imageOutput(outputId, width, height, click,
dblclick, hover, hoverDelay, hoverDelayType,
brush, clickId, hoverId, inline)`

Gráfico

<https://shiny.rstudio.com/gallery/image-output.html>

```
'data.frame': 3 obs. of  2 variables:  
 $ Sepal.Length: num 5.1 4.9 4.7  
 $ Sepal.Width : num 3.5 3 3.2
```

`renderPrint(expr, env, quoted, func,
width)`

`plotOutput(outputId, width, height, click,
dblclick, hover, hoverDelay, hoverDelayType,
brush, clickId, hoverId, inline)`

Salida de código

<https://shiny.rstudio.com/gallery/image-output.html>

	Sepal.Length	Sepal.Width	Petal.Length	Petal.Width	Species
1	5.10	3.50	1.40	0.20	Iris-setosa
2	4.90	3.00	1.40	0.20	Iris-setosa
3	4.70	3.20	1.30	0.20	Iris-setosa
4	4.60	3.10	1.50	0.20	Iris-setosa
5	4.50	3.00	1.40	0.20	Iris-setosa
6	4.40	3.00	1.30	0.20	Iris-setosa

`renderTable(expr,..., env, quoted, func)`

`verbatimTextOutput(outputId)`

Tabla

<https://shiny.rstudio.com/gallery/image-output.html>

foo	bar
1	2
2	3
3	4
4	5
5	6
6	7
7	8
8	9
9	10
10	11
11	12
12	13
13	14
14	15
15	16
16	17
17	18
18	19
19	20
20	21
21	22
22	23
23	24
24	25
25	26
26	27
27	28
28	29
29	30
30	31
31	32
32	33
33	34
34	35
35	36
36	37
37	38
38	39
39	40
40	41
41	42
42	43
43	44
44	45
45	46
46	47
47	48
48	49
49	50
50	51
51	52
52	53
53	54
54	55
55	56
56	57
57	58
58	59
59	60
60	61
61	62
62	63
63	64
64	65
65	66
66	67
67	68
68	69
69	70
70	71
71	72
72	73
73	74
74	75
75	76
76	77
77	78
78	79
79	80
80	81
81	82
82	83
83	84
84	85
85	86
86	87
87	88
88	89
89	90
90	91
91	92
92	93
93	94
94	95
95	96
96	97
97	98
98	99
99	100

`renderText(expr, env, quoted, func)`

`tableOutput(outputId)`

Texto

<https://shiny.rstudio.com/gallery/text-output.html>



`renderUI(expr, env, quoted, func)`

`uiOutput(outputId, inline, container, ...)`

Elementos dinámicos del UI

<https://shiny.rstudio.com/gallery/dynamic-ui.html>



Inputs y Outputs en server

Shiny

Ana D. Maldonado
ana.d.maldonado@ual.es

[Introducción](#)

[Estructura de una app](#)

[Widgets](#)

[Outputs](#)

[Reactividad](#)

[Layout](#)

[Publicación](#)

Inputs:

- Son objetos de “solo lectura”, es decir, no se puede cambiar su valor dentro de la función server.

Errores/
error01

```
ui <- fluidPage(  
  numericInput("count", label = "Number of values", value = 100)  
)  
  
server <- function(input, output) {  
  input$count <- 10  
}  
  
shinyApp(ui, server)  
Error in `$.reactivevalues`(`*tmp*`, count, value = 10) :  
  Attempted to assign value to a read-only reactivevalues object
```

- Solo se puede leer un input dentro de un **contexto reactivo** (por ejemplo, dentro de render*() o reactive()).

Errores/
error02

```
server <- function(input, output) {  
  message("The value of input$count is ", input$count)  
}  
  
shinyApp(ui, server)  
Error in .getReactiveEnvironment()$currentContext() :  
  Operation not allowed without an active reactive context. (You tried to  
  do something that can only be done from inside a reactive expression or  
  observer.)
```



Inputs y Outputs en server

Shiny

Ana D. Maldonado
ana.d.maldonado@ual.es

[Introducción](#)

[Estructura de una app](#)

[Widgets](#)

[Outputs](#)

[Reactividad](#)

[Layout](#)

[Publicación](#)

Outputs:

- Siempre se usa el objeto output junto a una función **render***(). Si olvidamos la función render:

Errores/
error03

```
ui <- fluidPage(  
  textOutput("greeting")  
)  
  
server <- function(input, output) {  
  output$greeting <- renderText("Hello human!")  
}  
  
shinyApp(ui = ui, server = server)  
Error in .subset2(x, "impl")$defineOutput(name, value, label) :  
  Unexpected character output for greeting
```

- No se puede leer un objeto output:

Errores/
error04

```
server <- function(input, output) {  
  message("The greeting is ", output$greeting)  
}  
  
shinyApp(ui, server)  
Error in `$.shinyoutput`(output, greeting) :  
  Reading from shinyoutput object is not allowed.
```



¡A PRACTICAR!

Shiny

Ana D. Maldonado
ana.d.maldonado@ual.es

[Introducción](#)

[Estructura de una app](#)

[Widgets](#)

[Outputs](#)

[Reactividad](#)

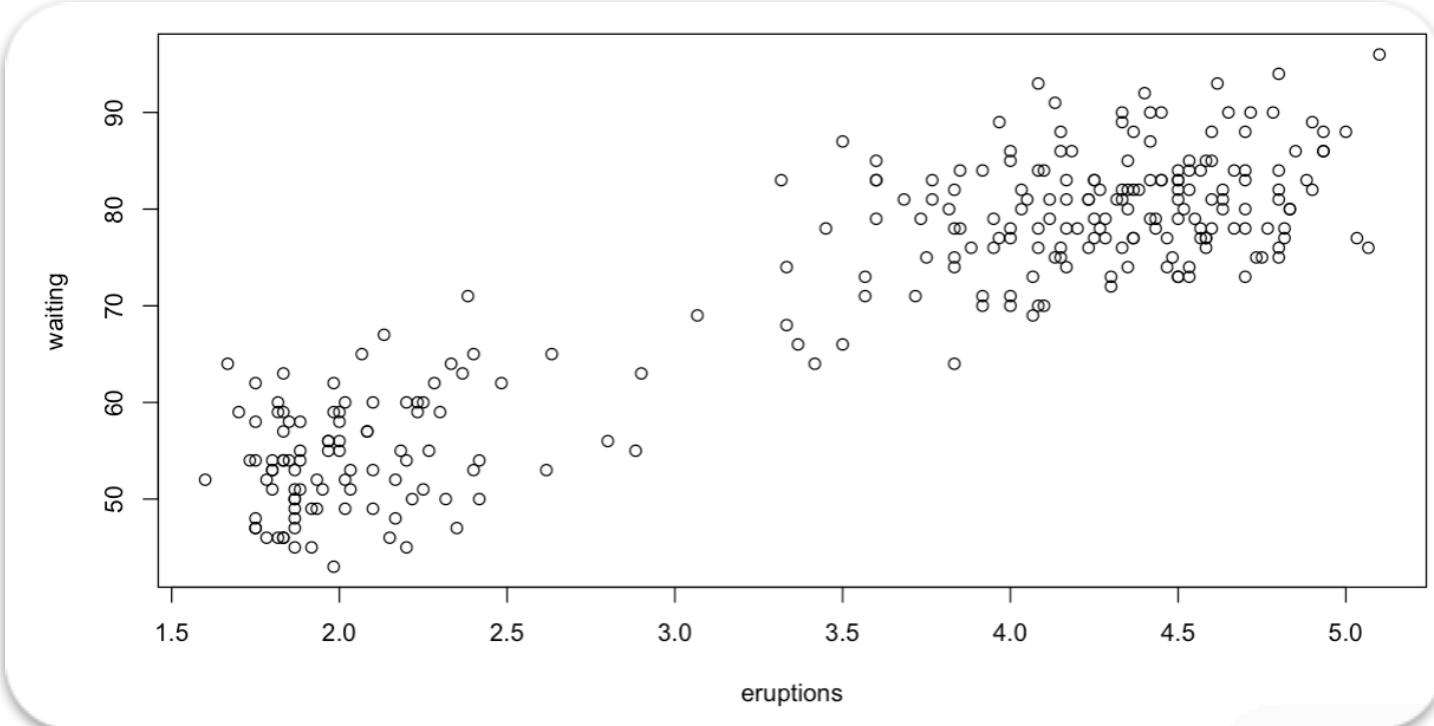
[Layout](#)

[Publicación](#)

- Obtén la salida reactiva de los widgets añadidos anteriormente.

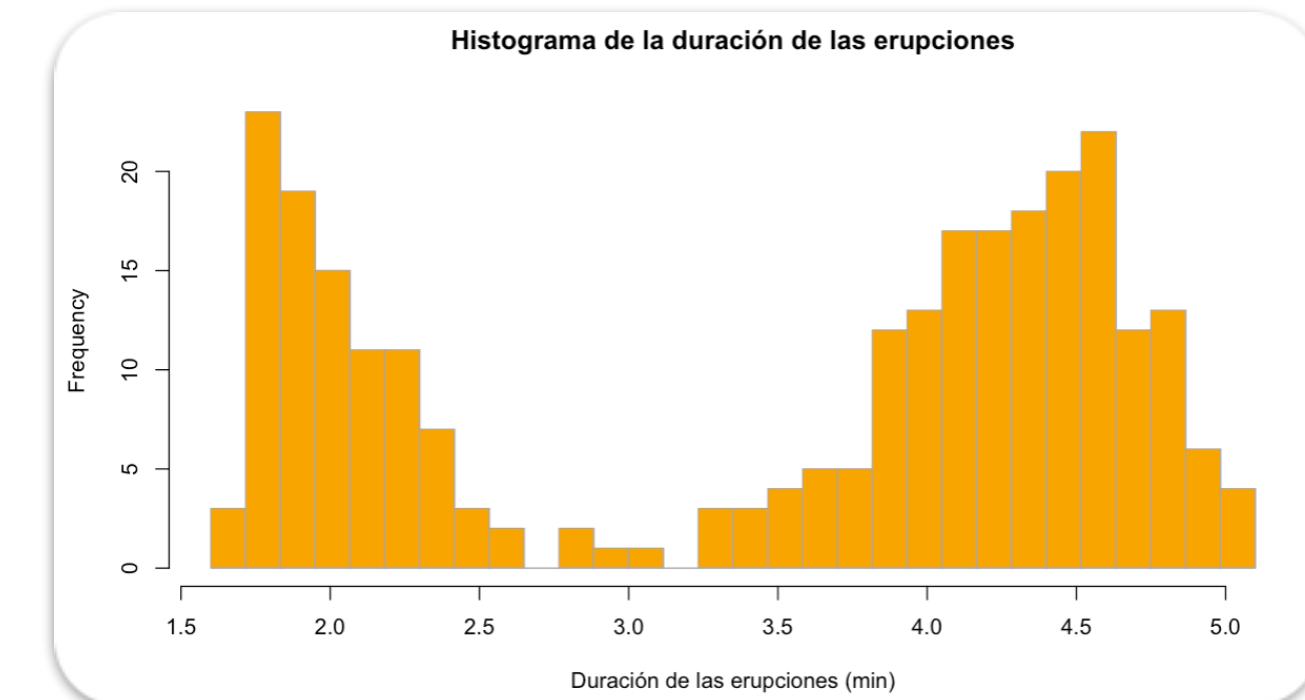
Select plot:

Scatterplot ▾



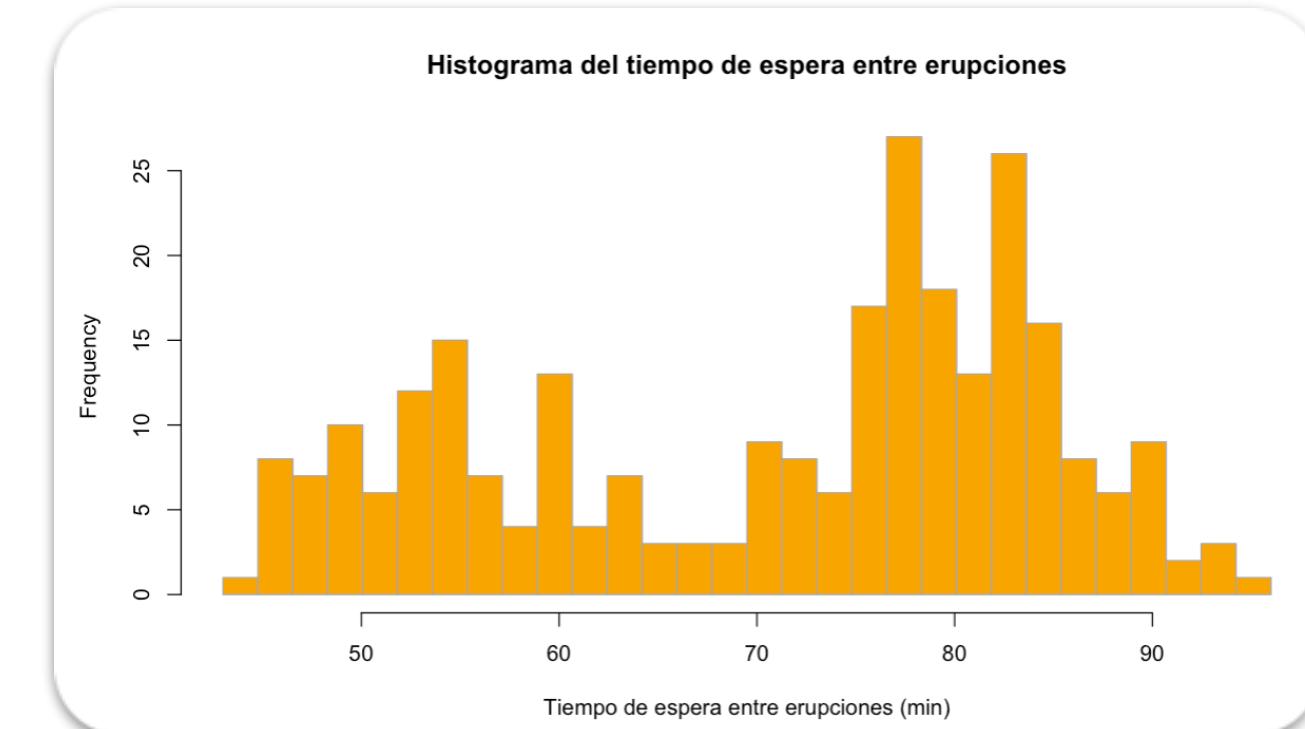
Select plot:

Histogram ▾



Plot variable:
 eruptions
 waiting

Plot variable:
 eruptions
 waiting





¡A PRACTICAR! (Solución)

Shiny

Ana D. Maldonado
ana.d.maldonado@ual.es

[Introducción](#)

[Estructura de una app](#)

[Widgets](#)

[Outputs](#)

[Reactividad](#)

[Layout](#)

[Publicación](#)

```
server <- function(input, output) {  
  
  output$distPlot <- renderPlot({  
    # generate bins based on input$bins from ui.R  
    x   <- faithful[, input$variable]  
    bins <- seq(min(x), max(x), length.out = input$bins + 1)  
  
    # draw the histogram with the specified number of bins  
    hist(x, breaks = bins, col = 'orange', border = 'darkgray',  
          main = "Histograma del tiempo de espera entre erupciones",  
          xlab = "Tiempo de espera entre erupciones (min)")  
  })  
}
```



¡A PRACTICAR! (Solución)

Shiny

Ana D. Maldonado
ana.d.maldonado@ual.es

[Introducción](#)

[Estructura de una app](#)

[Widgets](#)

[Outputs](#)

[Reactividad](#)

[Layout](#)

[Publicación](#)

```
server <- function(input, output) {  
  
  output$distPlot <- renderPlot({  
    if(input$plotType == "Histogram"){  
      # generate bins based on input$bins from ui.R  
      x <- faithful[, input$variable]  
      bins <- seq(min(x), max(x), length.out = input$bins + 1)  
  
      # draw the histogram with the specified number of bins  
      hist(x, breaks = bins, col = 'orange', border = 'darkgray',  
            main = "Histograma del tiempo de espera entre erupciones",  
            xlab = "Tiempo de espera entre erupciones (min)")  
  
    }else{  
      plot(faithful)  
    }  
  })  
}
```



¡A PRACTICAR!

Shiny

Ana D. Maldonado
ana.d.maldonado@ual.es

[Introducción](#)

[Estructura de una app](#)

[Widgets](#)

[Outputs](#)

[Reactividad](#)

[Layout](#)

[Publicación](#)

- Reordena el sidebarPanel para que la elección del gráfico sea lo primero que aparece.
- Muestra los widgets para elegir el número de intervalos y la variable a representar SOLO si la opción ‘Histogram’ ha sido seleccionada.

Select plot:

Histogram

Number of bins:

5 30 50

Plot variable:

eruptions

waiting

```
# Mostar widgets solo si se selecciona el histograma
conditionalPanel(condition = "input.plotType == 'Histogram'",

    # Añade los widgets

)
```



¡A PRACTICAR! (Solución)

Shiny

Ana D. Maldonado
ana.d.maldonado@ual.es

[Introducción](#)

[Estructura de una app](#)

[Widgets](#)

[Outputs](#)

[Reactividad](#)

[Layout](#)

[Publicación](#)

```
# Define UI for app that draws a histogram ----
ui <- fluidPage(

  # Application title ----
  titlePanel("Old Faithful Geyser Data"),

  # Sidebar with a slider input for number of bins ----
  sidebarLayout(
    sidebarPanel(
      selectInput("plotType", "Select plot:",
                 choices = c("Histogram", "Scatterplot")),
      conditionalPanel(condition = "input.plotType == 'Histogram'",
                      sliderInput(inputId = "bins",
                                  label = "Number of bins:",
                                  min = 5,
                                  max = 50,
                                  value = 30),

      radioButtons("variable", "Plot variable:",
                  choices = colnames(faithful))
    ),
    mainPanel(
      plotOutput(outputId = "distPlot")
    )
  )
)
```



Shiny

Ana D. Maldonado
ana.d.maldonado@ual.es

[Introducción](#)

[Estructura de una app](#)

[Widgets](#)

[Outputs](#)

[Reactividad](#)

[Layout](#)

[Publicación](#)

1. Introducción
2. Estructura de una aplicación Shiny
3. Widgets (inputs)
4. Salida reactiva (outputs)
5. Expresiones reactivas
6. Layout
7. Publicación



Expresiones reactivas

Shiny

Ana D. Maldonado
ana.d.maldonado@ual.es

[Introducción](#)

[Estructura de una app](#)

[Widgets](#)

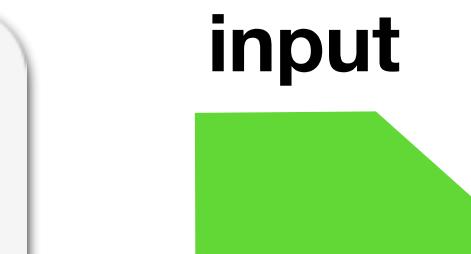
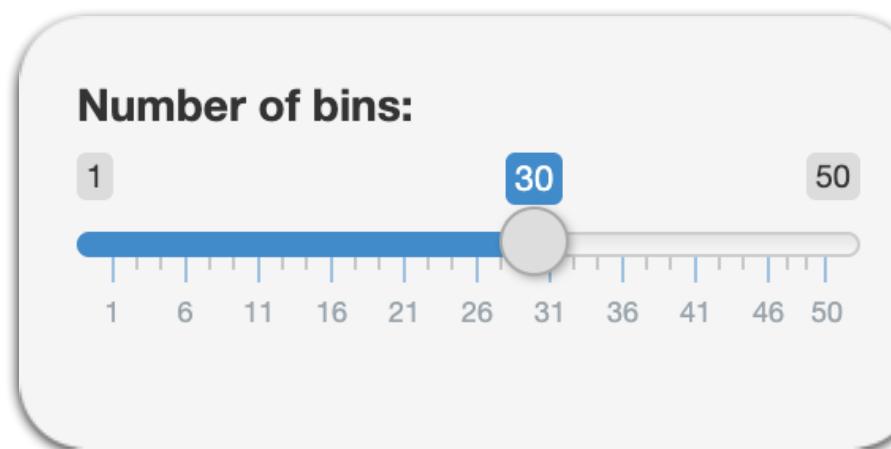
[Outputs](#)

[Reactividad](#)

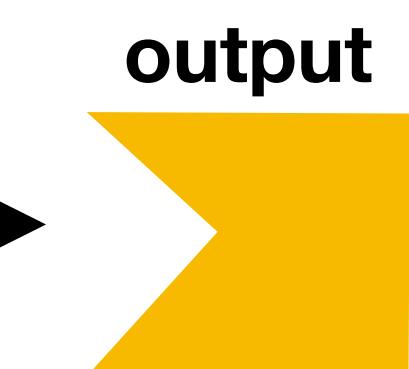
[Layout](#)

[Publicación](#)

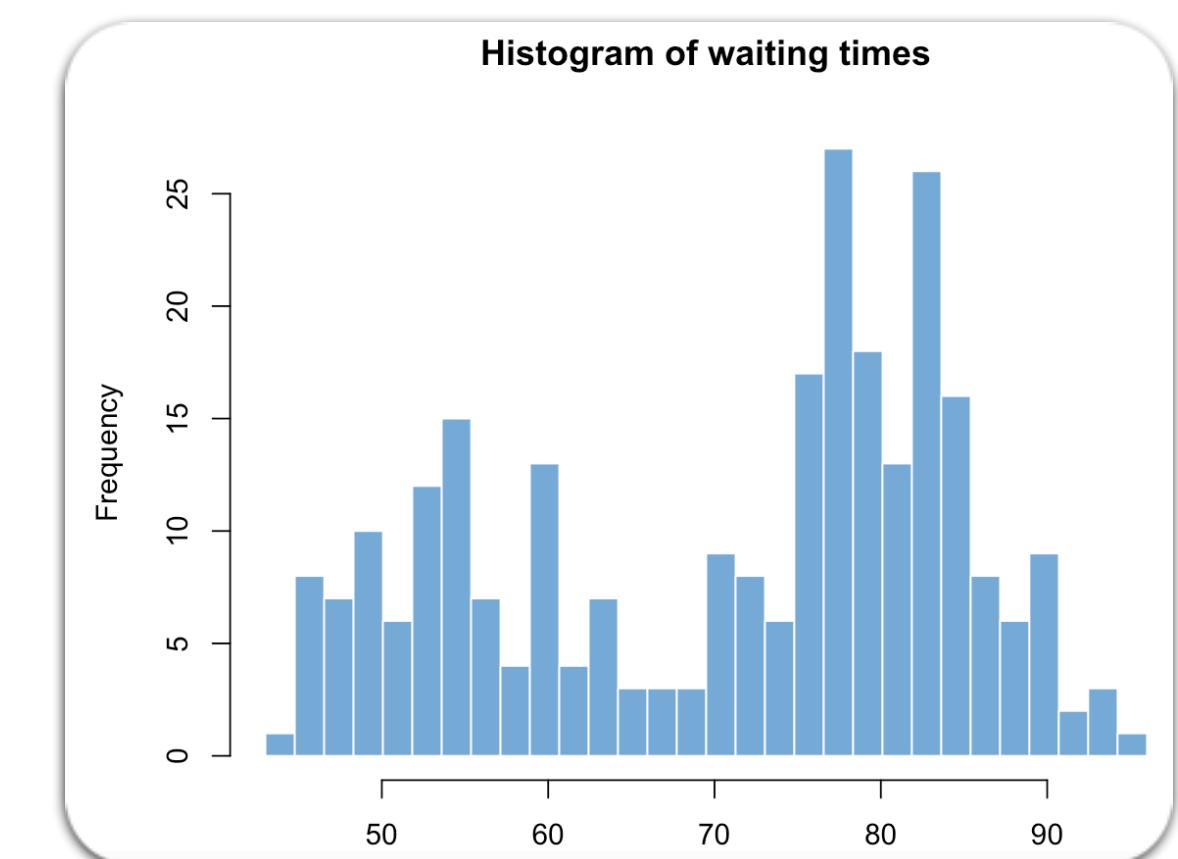
La reactividad en Shiny permite obtener outputs que se actualizan cuando el usuario cambia el valor de un widget. Hasta ahora hemos visto el caso más sencillo:



`input$xxx`



`output$xyz = render*(())`





Expresiones reactivas

Shiny

Ana D. Maldonado
ana.d.maldonado@ual.es

[Introducción](#)

[Estructura de una app](#)

[Widgets](#)

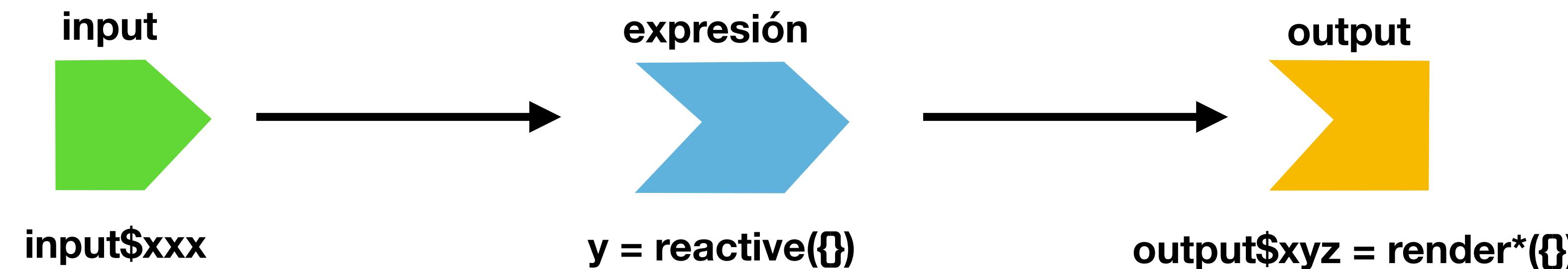
[Outputs](#)

[Reactividad](#)

[Layout](#)

[Publicación](#)

Shiny permite crear expresiones reactivas intermedias que nos ayudan a ganar eficiencia.



- Al igual que los **outputs**, las **expresiones reactivas** dependen de los **inputs**.
- Al igual que los **inputs**, se puede usar el resultado de una **expresión reactivas** en un **output**.



Expresiones reactivas

Shiny

Ana D. Maldonado
ana.d.maldonado@ual.es

[Introducción](#)

[Estructura de una app](#)

[Widgets](#)

[Outputs](#)

[Reactividad](#)

[Layout](#)

[Publicación](#)

input\$bins
input\$variable

```
output$distPlot = renderPlot({  
  x = faithful[,input$variable]  
  
  bins = seq(min(x), max(x),  
            length.out = input$bins + 1)  
  
  hist(x, breaks = bins)  
})
```

```
output$breaks = renderPrint({  
  x = faithful[, input$variable]  
  
  seq(min(x), max(x),  
       length.out = input$bins + 1)  
})
```

Tenemos mucho código duplicado!

Old Faithful Geyser Data

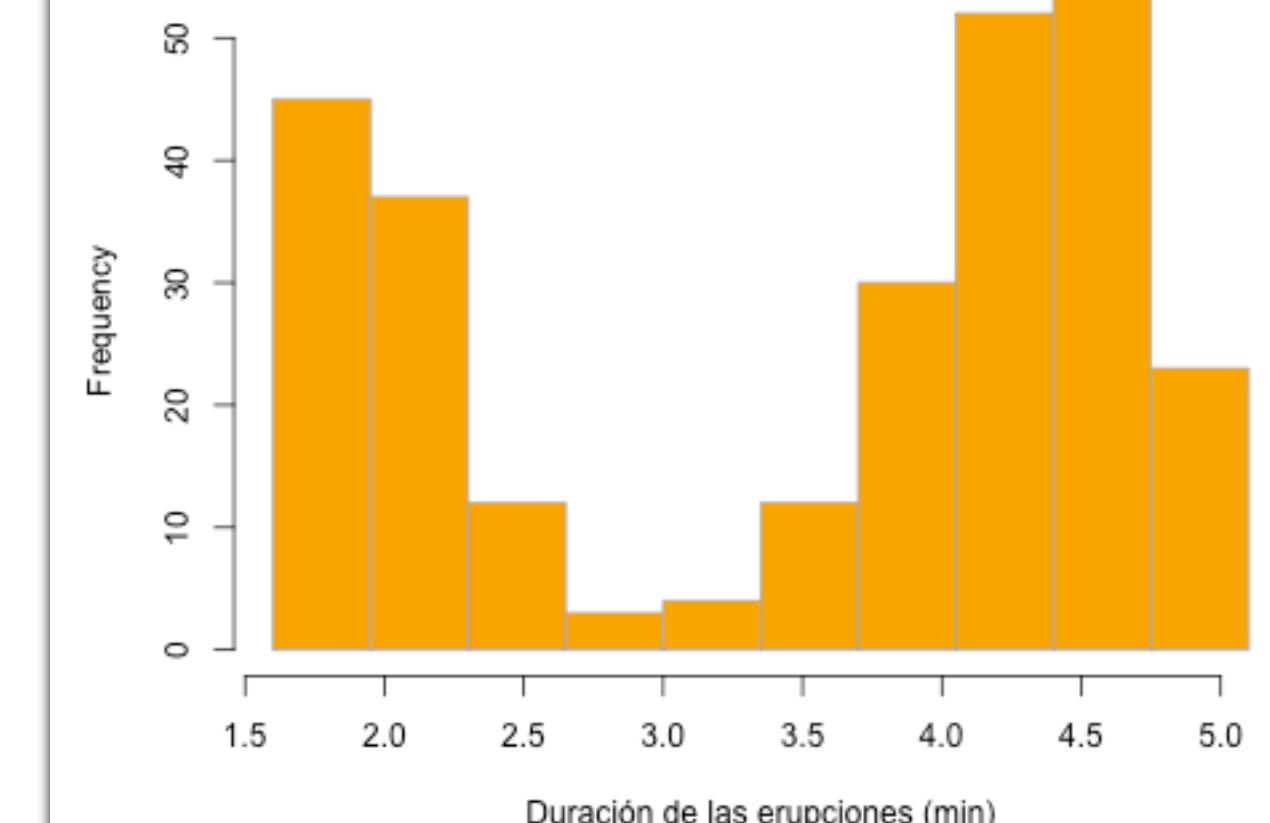
Number of bins:

5 10 50

Plot variable:

eruptions waiting

Histograma de la duración de las erupciones



[1] 1.60 1.95 2.30 2.65 3.00 3.35 3.70 4.05 4.40 4.75 5.10



Expresiones reactivas

Shiny

Ana D. Maldonado
ana.d.maldonado@ual.es

[Introducción](#)

[Estructura de una app](#)

[Widgets](#)

[Outputs](#)

[Reactividad](#)

[Layout](#)

[Publicación](#)

Ejemplos/
ejemplo01

input\$bins
input\$variable

```
x = reactive({ faithful[, input$variable] })
bins = reactive({ seq(min(x()), max(x()), length.out = input$bins + 1) })
```

```
output$distPlot = renderPlot({
  hist(x(), breaks = bins())
})
```

```
output$breaks = renderPrint({
  bins()
})
```

Mediante la función **reactive()** podemos crear objetos reactivos intermedios que se pueden reutilizar en cualquier parte de la app.

Old Faithful Geyser Data

Number of bins:

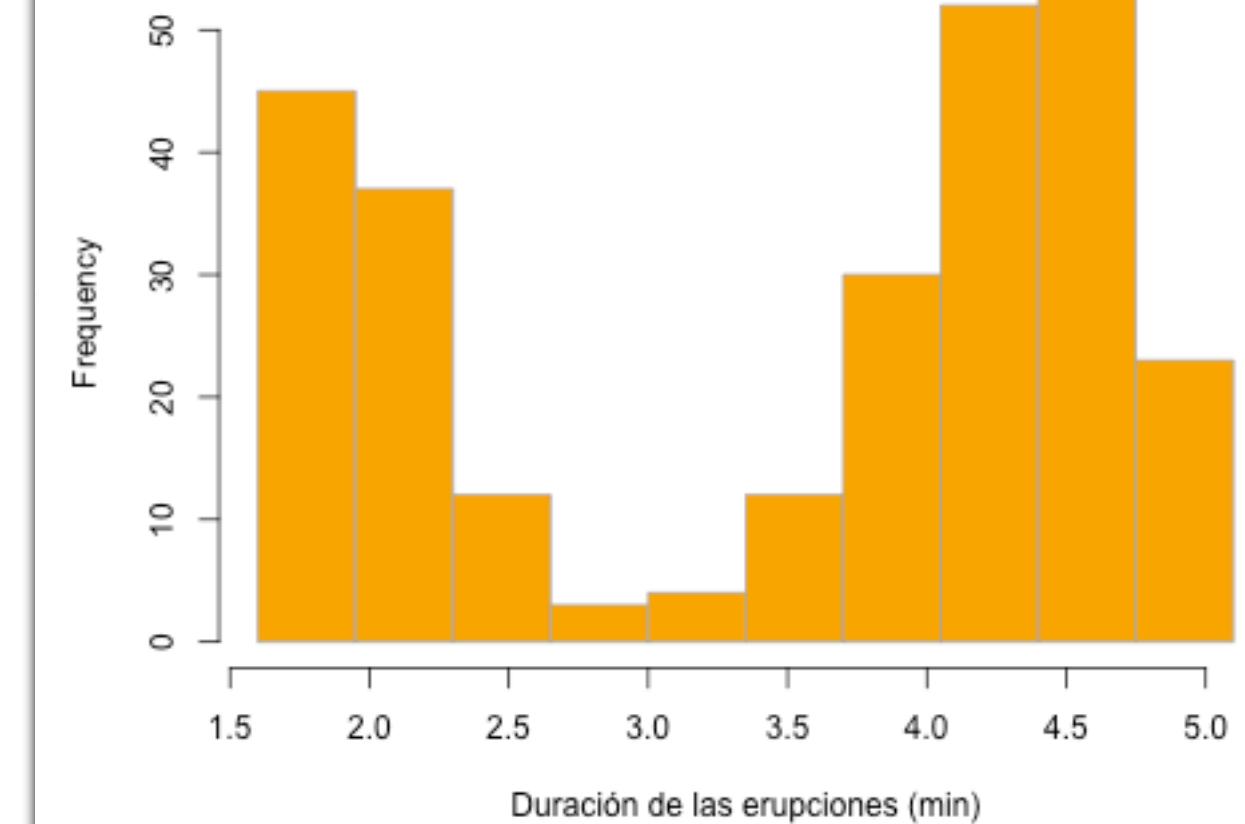
5 10 50

5 10 15 20 25 30 35 40 45 50

Plot variable:

eruptions waiting

Histograma de la duración de las erupciones



[1] 1.60 1.95 2.30 2.65 3.00 3.35 3.70 4.05 4.40 4.75 5.10



Expresiones reactivas

Shiny

Ana D. Maldonado
ana.d.maldonado@ual.es

[Introducción](#)

[Estructura de una app](#)

[Widgets](#)

[Outputs](#)

[Reactividad](#)

[Layout](#)

[Publicación](#)

Una expresión reactiva es, técnicamente, una función, con la diferencia de que almacena el resultado y no lo cambia a menos que cambie alguno de sus inputs. Para llamar a una expresión reactiva, se escribe su nombre seguido de paréntesis, al igual que con una función.

```
# Crear expresión reactiva y guardarla en "objeto" ----  
objeto <- reactive({  
  
    # Expresión  
  
})  
  
# Usar la expresión reactiva "objeto" ----  
objeto()
```

Ventaja: podemos crear nuestra app de forma modular, lo cual nos permite reutilizar código.



Expresiones reactivas

Shiny

Ana D. Maldonado
ana.d.maldonado@ual.es

[Introducción](#)

[Estructura de una app](#)

[Widgets](#)

[Outputs](#)

[Reactividad](#)

[Layout](#)

[Publicación](#)

```
ui <- fluidPage(  
  sliderInput(inputId = "num",  
    label = "Elige un número",  
    value = 25, min = 1, max = 100),  
  
  plotOutput("hist"),  
  
  verbatimTextOutput("summary")  
)  
  
server <- function(input, output) {  
  
  x = function(){rnorm(input$num)}  
  
  output$hist <- renderPlot({  
    hist(x())  
  })  
  output$summary <- renderPrint({  
    summary(x())  
  })  
}  
  
shinyApp(ui = ui, server = server)
```

```
ui <- fluidPage(  
  sliderInput(inputId = "num",  
    label = "Elige un número",  
    value = 25, min = 1, max = 100),  
  
  plotOutput("hist"),  
  
  verbatimTextOutput("summary")  
)  
  
server <- function(input, output) {  
  
  x = reactive({rnorm(input$num)})  
  
  output$hist <- renderPlot({  
    hist(x())  
  })  
  output$summary <- renderPrint({  
    summary(x())  
  })  
}  
  
shinyApp(ui = ui, server = server)
```

¿Existe alguna diferencia en el funcionamiento de estas 2 aplicaciones?

Ejemplos/
ejemplo02



¡A PRACTICAR!

Shiny

Ana D. Maldonado
ana.d.maldonado@ual.es

[Introducción](#)

[Estructura de una app](#)

[Widgets](#)

[Outputs](#)

[Reactividad](#)

[Layout](#)

[Publicación](#)

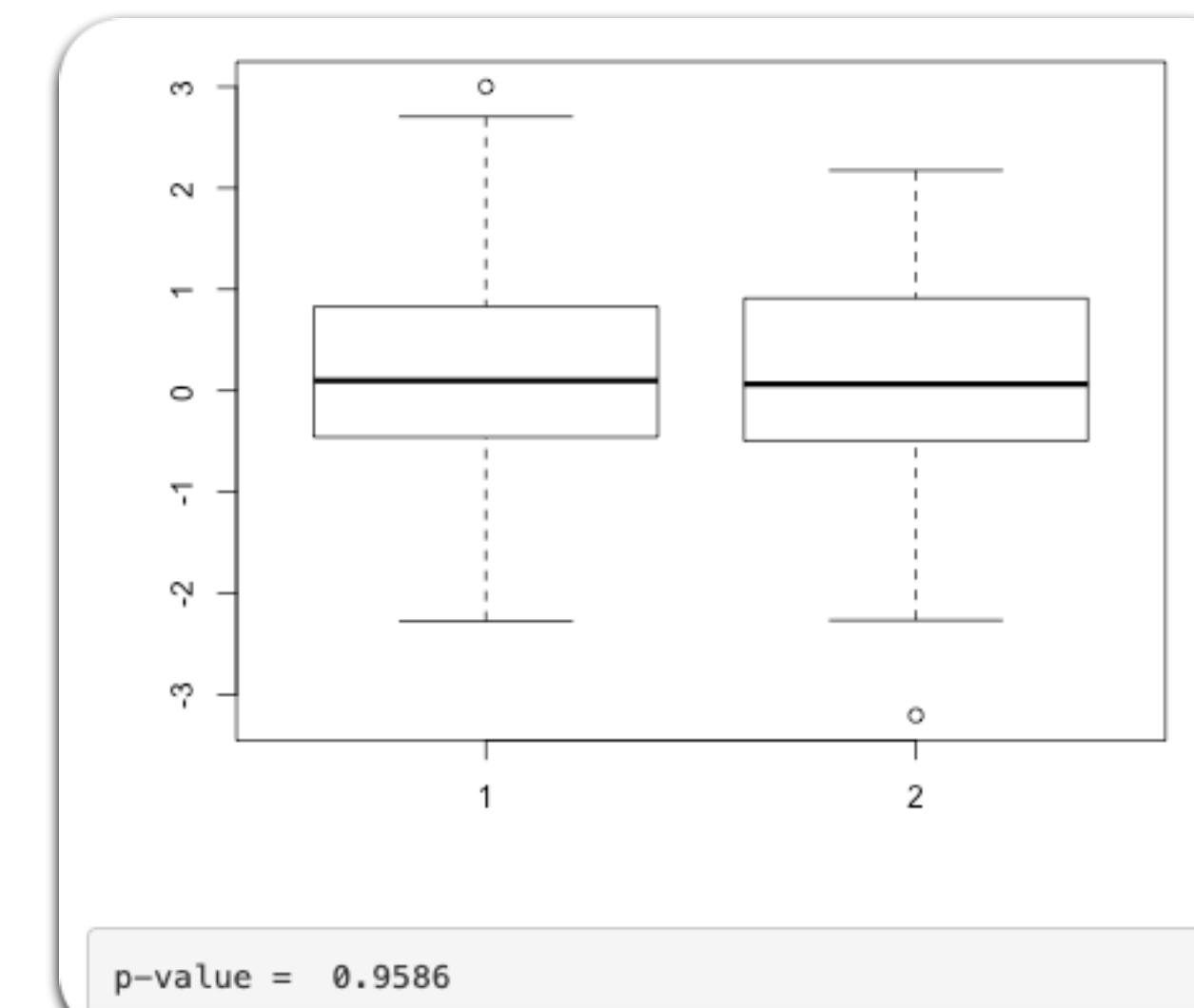
Escribe una aplicación (App-2) que simule 2 conjuntos de datos y los compare mediante un gráfico y un test de hipótesis.

- Generar 2 muestras que sigan una distribución normal, cuyo tamaño y parámetros sean especificados por el usuario.
- Construir un diagrama de cajas y bigotes.
- Realizar un contraste de hipótesis para comparar las medias de 2 distribuciones.

Como resultado, la aplicación devuelve el gráfico y el p-valor del contraste.

```
# Generar una muestra normal —  
x1 = rnorm(n1, mu1, sigma1) # n, mu y sigma son inputs  
x2 = rnorm(n2, mu2, sigma2) # n, mu y sigma son inputs  
  
# Construir un diagrama de cajas y bigotes ----  
boxplot(x1, x2)  
  
# Realizar el contraste de hipótesis —  
t.test(x1, x2)  
  
# Mostrar p-valor —  
paste("p-value = " , round(t.test(x1, x2)$p.value,4))
```

Distribución 1	Distribución 2
n	n
100	100
μ	μ
0	0
σ	σ
1	1





¡A PRACTICAR! (Solución)

Shiny

Ana D. Maldonado
ana.d.maldonado@ual.es

[Introducción](#)

[Estructura de una app](#)

[Widgets](#)

[Outputs](#)

[Reactividad](#)

[Layout](#)

[Publicación](#)

```
ui <- fluidPage(  
  titlePanel("Comparar 2 distribuciones"),  
  sidebarLayout(  
    sidebarPanel(  
      fluidRow(  
        column(6,  
          "Distribución 1",  
          numericInput("n1", label = "n", value = 100, min = 1),  
          numericInput("mean1", label = "μ", value = 0, step = 0.1),  
          numericInput("sd1", label = "σ", value = 1, min = 0.1, step = 0.1),  
        ),  
        column(6,  
          "Distribución 2",  
          numericInput("n2", label = "n", value = 100, min = 1),  
          numericInput("mean2", label = "μ", value = 0, step = 0.1),  
          numericInput("sd2", label = "σ", value = 1, min = 0.1, step = 0.1)  
        )  
      )  
    ),  
    mainPanel(  
      plotOutput("boxplot"),  
      verbatimTextOutput("ttest")  
    )  
  )  
)
```



¡A PRACTICAR! (Solución)

Shiny

Ana D. Maldonado
ana.d.maldonado@ual.es

Introducción

Estructura de una app

Widgets

Outputs

Reactividad

Layout

Publicación

```
ui <- fluidPage(  
  titlePanel("Comparar 2 distribuciones"),  
  sidebarLayout(  
    sidebarPanel(  
      fluidRow(  
        column(6,  
          "Distribución 1",  
          numericInput("n1", label = "n", value = 100, min = 1),  
          numericInput("mean1", label = "μ", value = 0, step = 0.1),  
          numericInput("sd1", label = "σ", value = 1, min = 0.1, step = 0.1),  
        ),  
        column(6,  
          "Distribución 2",  
          numericInput("n2", label = "n", value = 100, min = 1),  
          numericInput("mean2", label = "μ", value = 0, step = 0.1),  
          numericInput("sd2", label = "σ", value = 1, min = 0.1, step = 0.1)  
        )  
      ),  
      mainPanel(  
        plotOutput("boxplot"),  
        verbatimTextOutput("ttest")  
      )  
    )  
)
```

inputs

outputs



¡A PRACTICAR! (Solución incorrecta)

Shiny

Ana D. Maldonado
ana.d.maldonado@ual.es

[Introducción](#)

[Estructura de una app](#)

[Widgets](#)

[Outputs](#)

[Reactividad](#)

[Layout](#)

[Publicación](#)

```
server <- function(input, output, session) {  
  output$boxplot <- renderPlot({  
    x1 <- rnorm(input$n1, input$mean1, input$sd1)  
    x2 <- rnorm(input$n2, input$mean2, input$sd2)  
  
    boxplot(x1, x2)  
  })  
  
  output$ttest <- renderText({  
    x1 <- rnorm(input$n1, input$mean1, input$sd1)  
    x2 <- rnorm(input$n2, input$mean2, input$sd2)  
  
    paste("p-value = " , round(t.test(x1, x2)$p.value,4))  
  })  
}
```

¿Cuál es el problema del server? ¿Cómo podríamos solucionarlo?



¡A PRACTICAR! (Solución)

Shiny

Ana D. Maldonado
ana.d.maldonado@ual.es

[Introducción](#)

[Estructura de una app](#)

[Widgets](#)

[Outputs](#)

[Reactividad](#)

[Layout](#)

[Publicación](#)

```
server <- function(input, output, session) {  
  output$boxplot <- renderPlot({  
    x1 <- rnorm(input$n1, input$mean1, input$sd1)  
    x2 <- rnorm(input$n2, input$mean2, input$sd2)  
  
    boxplot(x1, x2)  
  })  
  
  output$ttest <- renderText({  
    x1 <- rnorm(input$n1, input$mean1, input$sd1)  
    x2 <- rnorm(input$n2, input$mean2, input$sd2)  
  
    paste("p-value = " , round(t.test(x1, x2)$p.value,4))  
  })  
}
```

- Se actualizan tanto x1 como x2 cada vez que uno de los inputs (n1, mean1, sd1, n2, mean2 o sd2) cambia.
- El gráfico y el contraste de hipótesis usan datos diferentes.

¿Cuál es el problema del server? ¿Cómo podríamos solucionarlo?



¡A PRACTICAR! (Solución)

Shiny

Ana D. Maldonado
ana.d.maldonado@ual.es

[Introducción](#)

[Estructura de una app](#)

[Widgets](#)

[Outputs](#)

[Reactividad](#)

[Layout](#)

[Publicación](#)

```
server <- function(input, output, session) {  
  
  x1 <- reactive({rnorm(input$n1, input$mean1, input$sd1)})  
  x2 <- reactive({rnorm(input$n2, input$mean2, input$sd2)})  
  
  output$boxplot <- renderPlot({  
    boxplot(x1(), x2())  
  })  
  
  output$ttest <- renderText({  
    paste("p-value = " , round(t.test(x1(), x2())$p.value,4))  
  })  
}
```

x1 y x2 solo se actualizan cuando cambian sus respectivos inputs.

El gráfico y el contraste de hipótesis usan los mismos datos.



Retrasar reactividad

Shiny

Ana D. Maldonado
ana.d.maldonado@ual.es

[Introducción](#)

[Estructura de una app](#)

[Widgets](#)

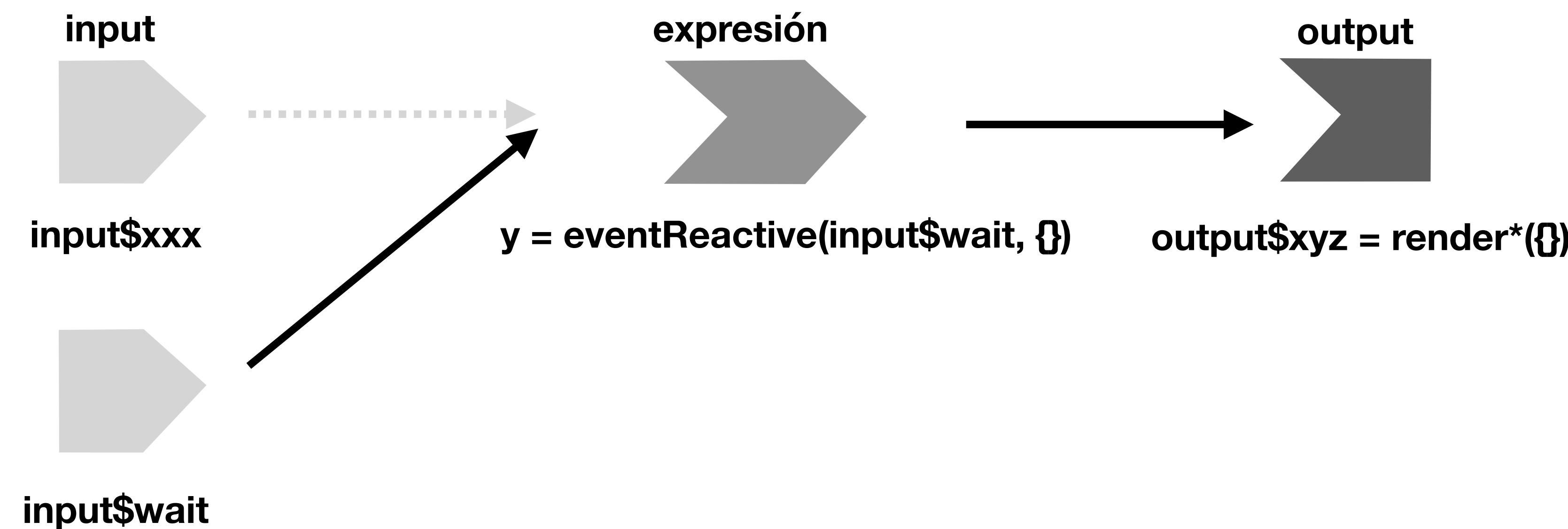
[Outputs](#)

[Reactividad](#)

[Layout](#)

[Publicación](#)

Shiny permite retrasar la ejecución de una expresión reactiva hasta que un input específico cambia de valor.



- La función eventReactive() toma 2 argumentos: el primero es el input o inputs con los que se activa, y el segundo, encerrado entre llaves, es el código que se ejecuta (que puede depender de otros inputs).
- Al igual que con reactive(), el resultado se puede guardar y re-usar en otras partes del código.



Retrasar reactividad

Shiny

Ana D. Maldonado
ana.d.maldonado@ual.es

[Introducción](#)

[Estructura de una app](#)

[Widgets](#)

[Outputs](#)

[Reactividad](#)

[Layout](#)

[Publicación](#)

Supón que quieres que alguna parte de tu aplicación SOLO se ejecute cuando el usuario pulse un botón. Esto se puede hacer usando el widget `actionButton()` y `eventReactive()`.

```
ui <- fluidPage(  
  helpText("Adivina un número del 1 al 10"),  
  actionButton("go", "Click me!"),  
  textOutput("out")  
)  
  
server <- function(input, output) {  
  y = eventReactive(input$go, {sample(1:10,1)})  
  output$out <- renderText({y()})  
}  
  
shinyApp(ui = ui, server = server)
```

Ejemplos/
ejemplo03



¡A PRACTICAR!

Shiny

Ana D. Maldonado
ana.d.maldonado@ual.es

[Introducción](#)

[Estructura de una app](#)

[Widgets](#)

[Outputs](#)

[Reactividad](#)

[Layout](#)

[Publicación](#)

Añade un `actionButton()` a App-2 para que no se generen las muestras hasta que el usuario pulse el botón “Actualizar”.

Comparar 2 distribuciones

Distribución 1	Distribución 2
n	n
100	100
μ	μ
0	0
σ	σ
1	1
Actualizar	



¡A PRACTICAR!

Shiny

Ana D. Maldonado
ana.d.maldonado@ual.es

[Introducción](#)

[Estructura de una app](#)

[Widgets](#)

[Outputs](#)

[Reactividad](#)

[Layout](#)

[Publicación](#)

```
ui <- fluidPage(  
  titlePanel("Comparar 2 distribuciones"),  
  sidebarLayout(  
    sidebarPanel(  
      fluidRow(  
        column(6,  
          "Distribución 1",  
          numericInput("n1", label = "n", value = 100, min = 1),  
          numericInput("mean1", label = " $\mu$ ", value = 0, step = 0.1),  
          numericInput("sd1", label = " $\sigma$ ", value = 1, min = 0.1, step = 0.1),  
        ),  
        column(6,  
          "Distribución 2",  
          numericInput("n2", label = "n", value = 100, min = 1),  
          numericInput("mean2", label = " $\mu$ ", value = 0, step = 0.1),  
          numericInput("sd2", label = " $\sigma$ ", value = 1, min = 0.1, step = 0.1)  
        )  
      ),  
      actionButton("go", "Actualizar")  
    ),  
    mainPanel(  
      plotOutput("boxplot"),  
      verbatimTextOutput("ttest")  
    )  
  )  
)
```



¡A PRACTICAR!

Shiny

Ana D. Maldonado
ana.d.maldonado@ual.es

[Introducción](#)

[Estructura de una app](#)

[Widgets](#)

[Outputs](#)

[Reactividad](#)

[Layout](#)

[Publicación](#)

```
server <- function(input, output, session) {  
  
  x1 <- eventReactive(input$go, {rnorm(input$n1, input$mean1, input$sd1)})  
  x2 <- eventReactive(input$go, {rnorm(input$n2, input$mean2, input$sd2)})  
  
  output$boxplot <- renderPlot({  
    boxplot(x1(), x2())  
  })  
  
  output$ttest <- renderText({  
    paste("p-value = " , round(t.test(x1(), x2())$p.value,4))  
  })  
}
```



Actualizar opciones de widgets

Shiny

Ana D. Maldonado
ana.d.maldonado@ual.es

[Introducción](#)

[Estructura de una app](#)

[Widgets](#)

[Outputs](#)

[Reactividad](#)

[Layout](#)

[Publicación](#)

Para actualizar los valores preestablecidos de un widget, se pueden usar las funciones `update*()`, dentro de un contexto reactivo, normalmente, una función `observe()`. Además, la función `server()` deberá incluir el argumento `session`. Ejemplo: un `selectInput` depende del `input` de otro `selectInput` para mostrar sus opciones.

Choose dataset

Choose dataset

faithful

Choose variable

eruptions

waiting

Choose dataset

Choose dataset

iris

Choose variable

|

Sepal.Length

Sepal.Width

Petal.Length

Petal.Width

Species

Choose dataset

Choose dataset

mtcars

Choose variable

|

mpg

cyl

disp

hp

drat

wt

qsec

vs

Ejemplos/
ejemplo04



Actualizar opciones de widgets

Shiny

Ana D. Maldonado
ana.d.maldonado@ual.es

[Introducción](#)

[Estructura de una app](#)

[Widgets](#)

[Outputs](#)

[Reactividad](#)

[Layout](#)

[Publicación](#)

```
ui <- fluidPage(  
  
  # Application title  
  titlePanel("Choose dataset"),  
  
  sidebarLayout(  
    # Panel lateral  
    sidebarPanel(  
      selectInput("dataset", "Choose dataset",  
                 choices = c("faithful", "iris", "mtcars")),  
  
      selectInput("var", "Choose variable", choices = "", multiple = T),  
  
      actionButton("go", "Actualizar")  
    ),  
  
    # Panel principal  
    mainPanel(  
      plotOutput("plot")  
    )  
  )  
)
```



Actualizar opciones de widgets

Shiny

Ana D. Maldonado
ana.d.maldonado@ual.es

[Introducción](#)

[Estructura de una app](#)

[Widgets](#)

[Outputs](#)

[Reactividad](#)

[Layout](#)

[Publicación](#)

```
ui <- fluidPage(  
  # Application title  
  titlePanel("Choose dataset"),  
  
  sidebarLayout(  
    # Panel lateral  
    sidebarPanel(  
      selectInput("dataset", "Choose dataset",  
                 choices = c("faithful", "iris", "mtcars")),  
  
      selectInput("var", "Choose variable", choices = "", multiple = T),  
  
      actionButton("go", "Actualizar")  
    ),  
  
    # Panel principal  
    mainPanel(  
      plotOutput("plot")  
    )  
  )
```

Input para elegir el conjunto
de datos



Actualizar opciones de widgets

Shiny

Ana D. Maldonado
ana.d.maldonado@ual.es

[Introducción](#)

[Estructura de una app](#)

[Widgets](#)

[Outputs](#)

[Reactividad](#)

[Layout](#)

[Publicación](#)

```
ui <- fluidPage(  
  
  # Application title  
  titlePanel("Choose dataset"),  
  
  sidebarLayout(  
    # Panel lateral  
    sidebarPanel(  
      selectInput("dataset", "Choose dataset",  
                 choices = c("faithful", "iris", "mtcars")),  
  
      selectInput("var", "Choose variable", choices = "", multiple = T),  
  
      actionButton("go", "Actualizar")  
    ),  
    # Panel principal  
    mainPanel(  
      plotOutput("plot")  
    )  
  )  
)
```

Input para elegir la variable
del conjunto de datos
seleccionado.



Actualizar opciones de widgets

Shiny

Ana D. Maldonado
ana.d.maldonado@ual.es

[Introducción](#)

[Estructura de una app](#)

[Widgets](#)

[Outputs](#)

[Reactividad](#)

[Layout](#)

[Publicación](#)

```
ui <- fluidPage(  
  
  # Application title  
  titlePanel("Choose dataset"),  
  
  sidebarLayout(  
    # Panel lateral  
    sidebarPanel(  
      selectInput("dataset", "Choose dataset",  
                 choices = c("faithful", "iris", "mtcars")),  
  
      selectInput("var", "Choose variable", choices = "", multiple = T),  
  
      actionButton("go", "Actualizar")  
    ),  
  
    # Panel principal  
    mainPanel(  
      plotOutput("plot")  
    )  
  )  
)
```

A priori, no tenemos
opciones.



Actualizar opciones de widgets

Shiny

Ana D. Maldonado
ana.d.maldonado@ual.es

[Introducción](#)

[Estructura de una app](#)

[Widgets](#)

[Outputs](#)

[Reactividad](#)

[Layout](#)

[Publicación](#)

```
ui <- fluidPage(  
  
  # Application title  
  titlePanel("Choose dataset"),  
  
  sidebarLayout(  
    # Panel lateral  
    sidebarPanel(  
      selectInput("dataset", "Choose dataset",  
                 choices = c("faithful", "iris", "mtcars")),  
  
      selectInput("var", "Choose variable", choices = "", multiple = T),  
  
      actionButton("go", "Actualizar")  
    ),  
  
    # Panel principal  
    mainPanel(  
      plotOutput("plot")  
    )  
  )  
)
```

Este argumento permite elegir varias opciones.



Actualizar opciones de widgets

Shiny

Ana D. Maldonado
ana.d.maldonado@ual.es

[Introducción](#)

[Estructura de una app](#)

[Widgets](#)

[Outputs](#)

[Reactividad](#)

[Layout](#)

[Publicación](#)

```
server <- function(input, output, session) {  
  
  datasetInput <- reactive({  
    switch(input$dataset,  
      "faithful" = faithful,  
      "iris" = iris,  
      "mtcars" = mtcars)  
  })  
  
  observe({ updateSelectInput(session, "var", choices = colnames(datasetInput())) })  
  
  plotData <- eventReactive(input$go, {  
    datasetInput() [,input$var]  
  })  
  output$plot <- renderPlot({  
  
    plot(plotData())  
  })  
}
```



Actualizar opciones de widgets

Shiny

Ana D. Maldonado
ana.d.maldonado@ual.es

[Introducción](#)

[Estructura de una app](#)

[Widgets](#)

[Outputs](#)

[Reactividad](#)

[Layout](#)

[Publicación](#)

```
server <- function(input, output, session) {  
  
  datasetInput <- reactive({  
    switch(input$dataset,  
      "faithful" = faithful,  
      "iris" = iris,  
      "mtcars" = mtcars)  
  })  
  
  observe({ updateSelectInput(session, "var", choices = colnames(datasetInput())) })  
  
  plotData <- eventReactive(input$go, {  
    datasetInput() [,input$var]  
  })  
  output$plot <- renderPlot({  
  
    plot(plotData())  
  })  
}
```

Permite actualizar opciones
de los widgets



Actualizar opciones de widgets

Shiny

Ana D. Maldonado
ana.d.maldonado@ual.es

[Introducción](#)

[Estructura de una app](#)

[Widgets](#)

[Outputs](#)

[Reactividad](#)

[Layout](#)

[Publicación](#)

```
server <- function(input, output, session) {  
  
  datasetInput <- reactive({  
    switch(input$dataset,  
      "faithful" = faithful,  
      "iris" = iris,  
      "mtcars" = mtcars)  
  })  
  
  observe({ updateSelectInput(session, "var", choices = colnames(datasetInput())) })  
  
  plotData <- eventReactive(input$go, {  
    datasetInput() [,input$var]  
  })  
  output$plot <- renderPlot({  
  
    plot(plotData())  
  })  
}
```

Según el valor de input\$dataset, el objeto datasetInput() será igual a faithful, iris, o mtcars.



Actualizar opciones de widgets

Shiny

Ana D. Maldonado
ana.d.maldonado@ual.es

Introducción

Estructura de una app

Widgets

Outputs

Reactividad

Layout

Publicación

```
server <- function(input, output, session) {  
  
  datasetInput <- reactive({  
    switch(input$dataset,  
      "faithful" = faithful,  
      "iris" = iris,  
      "mtcars" = mtcars)  
  })  
  
  observe({ updateSelectInput(session, "var", choices = colnames(datasetInput())) })  
  
  plotData <- eventReactive(input$go, {  
    datasetInput() [,input$var]  
  })  
  output$plot <- renderPlot({  
  
    plot(plotData())  
  })  
}
```

updateSelectInput permite actualizar las opciones de selectInput.



Actualizar opciones de widgets

Shiny

Ana D. Maldonado
ana.d.maldonado@ual.es

Introducción

Estructura de una app

Widgets

Outputs

Reactividad

Layout

Publicación

```
server <- function(input, output, session) {  
  
  datasetInput <- reactive({  
    switch(input$dataset,  
      "faithful" = faithful,  
      "iris" = iris,  
      "mtcars" = mtcars)  
  })  
  
  observe({ updateSelectInput(session, "var", choices = colnames(datasetInput())) })  
  
  plotData <- eventReactive(input$go, {  
    datasetInput()[,input$var]  
  })  
  output$plot <- renderPlot({  
    plot(plotData())  
  })  
}
```

plotData() es un data.frame que contiene solo las variables seleccionadas (input\$var) del conjunto de datos seleccionado (input\$dataset)



Shiny

Ana D. Maldonado
ana.d.maldonado@ual.es

[Introducción](#)

[Estructura de una app](#)

[Widgets](#)

[Outputs](#)

[Reactividad](#)

[Layout](#)

[Publicación](#)

1. Introducción
2. Estructura de una aplicación Shiny
3. Widgets (inputs)
4. Salida reactiva (outputs)
5. Expresiones reactivas
6. Layout
7. Publicación



Layout: tabsetPanel

Shiny

Ana D. Maldonado
ana.d.maldonado@ual.es

[Introducción](#)

[Estructura de una app](#)

[Widgets](#)

[Outputs](#)

[Reactividad](#)

[Layout](#)

[Publicación](#)

Hasta ahora hemos trabajado con el mismo diseño de web: una única página que contiene todo. Cuando la app empieza a tener un cierto tamaño, es recomendable crear distintas pestañas. Esto se puede hacer usando las funciones `tabsetPanel()` y `tabPanel()`. <https://shiny.rstudio.com/gallery/tabssets.html>

```
ui <- fluidPage(
  titlePanel("Old Faithful Geyser Data"),
  tabsetPanel(
    tabPanel("Primera pestaña",
      # Sidebar with a slider input for number of bins
      sidebarLayout(
        sidebarPanel(
          sliderInput("bins",
            "Number of bins:",
            min = 1,
            max = 50,
            value = 30)
        ),
        # Show a plot of the generated distribution
        mainPanel(
          plotOutput("distPlot")
        )
      )),
    tabPanel("Segunda pestaña"),
    tabPanel("Tercera pestaña")
  )
)
```

Ejemplos/
ejemplo05



Layout: tabsetPanel

Shiny

Ana D. Maldonado
ana.d.maldonado@ual.es

[Introducción](#)

[Estructura de una app](#)

[Widgets](#)

[Outputs](#)

[Reactividad](#)

[Layout](#)

[Publicación](#)

Hasta ahora hemos trabajado con el mismo diseño de web: una única página que contiene todo. Cuando la app empieza a tener un cierto tamaño, es recomendable crear distintas pestañas. Esto se puede hacer usando las funciones `tabsetPanel()` y `tabPanel()`.

```
ui <- fluidPage(  
  titlePanel("Old Faithful Geyser Data"),  
  tabsetPanel(  
    tabPanel("Primera pestaña",  
      # Sidebar with a slider input for number of  
      sidebarLayout(  
        sidebarPanel(  
          sliderInput("bins",  
            "Number of bins:",  
            min = 1,  
            max = 50,  
            value = 30)  
        ),  
        # Show a plot of the generated distribution  
        mainPanel(  
          plotOutput("distPlot")  
        )  
      ),  
      tabPanel("Segunda pestaña"),  
      tabPanel("Tercera pestaña")  
    )  
)
```

Crea el entorno para definir distintas pestañas. Se pueden poner varios y en cualquier parte



Layout: tabsetPanel

Shiny

Ana D. Maldonado
ana.d.maldonado@ual.es

[Introducción](#)

[Estructura de una app](#)

[Widgets](#)

[Outputs](#)

[Reactividad](#)

[Layout](#)

[Publicación](#)

Hasta ahora hemos trabajado con el mismo diseño de web: una única página que contiene todo. Cuando la app empieza a tener un cierto tamaño, es recomendable crear distintas pestañas. Esto se puede hacer usando las funciones `tabsetPanel()` y `tabPanel()`.

```
ui <- fluidPage(  
  titlePanel("Old Faithful Geyser Data"),  
  tabsetPanel(  
    tabPanel("Primera pestaña",  
      # Sidebar with a slider input for number of bins  
      sidebarLayout(  
        sidebarPanel(  
          sliderInput("bins",  
            "Number of bins:",  
            min = 1,  
            max = 50,  
            value = 30)  
        ),  
        # Show a plot of the generated distribution  
        mainPanel(  
          plotOutput("distPlot")  
        )  
      ),  
      tabPanel("Segunda pestaña"),  
      tabPanel("Tercera pestaña")  
    )  
)
```

Primera pestaña. Contiene una función `sidebarLayout()`, que contiene un `sidebarPanel()` y un `mainPanel()`.



Layout: tabsetPanel

Shiny

Ana D. Maldonado
ana.d.maldonado@ual.es

[Introducción](#)

[Estructura de una app](#)

[Widgets](#)

[Outputs](#)

[Reactividad](#)

[Layout](#)

[Publicación](#)

Hasta ahora hemos trabajado con el mismo diseño de web: una única página que contiene todo. Cuando la app empieza a tener un cierto tamaño, es recomendable crear distintas pestañas. Esto se puede hacer usando las funciones `tabsetPanel()` y `tabPanel()`.

```
ui <- fluidPage(  
  titlePanel("Old Faithful Geyser Data"),  
  tabsetPanel(  
    tabPanel("Primera pestaña",  
      # Sidebar with a slider input for number of bins  
      sidebarLayout(  
        sidebarPanel(  
          sliderInput("bins",  
            "Number of bins:",  
            min = 1,  
            max = 50,  
            value = 30)  
        ),  
        # Show a plot of the generated distribution  
        mainPanel(  
          plotOutput("distPlot")  
        )  
      ),  
      tabPanel("Segunda pestaña"),  
      tabPanel("Tercera pestaña")  
    )  
)
```

Primera pestaña. Contiene una función `sidebarLayout()`, que contiene un `sidebarPanel()` y un `mainPanel()`.



Layout: tabsetPanel

Shiny

Ana D. Maldonado
ana.d.maldonado@ual.es

[Introducción](#)

[Estructura de una app](#)

[Widgets](#)

[Outputs](#)

[Reactividad](#)

[Layout](#)

[Publicación](#)

Hasta ahora hemos trabajado con el mismo diseño de web: una única página que contiene todo. Cuando la app empieza a tener un cierto tamaño, es recomendable crear distintas pestañas. Esto se puede hacer usando las funciones `tabsetPanel()` y `tabPanel()`.

```
ui <- fluidPage(  
  titlePanel("Old Faithful Geyser Data"),  
  tabsetPanel(  
    tabPanel("Primera pestaña",  
      # Sidebar with a slider input for number of bins  
      sidebarLayout(  
        sidebarPanel(  
          sliderInput("bins",  
            "Number of bins:",  
            min = 1,  
            max = 50,  
            value = 30)  
        ),  
        # Show a plot of the generated distribution  
        mainPanel(  
          plotOutput("distPlot")  
        )  
      )  
    ),  
    tabPanel("Segunda pestaña"),  
    tabPanel("Tercera pestaña")  
)
```

Se podría añadir otro `tabsetPanel()` dentro del panel principal:

```
mainPanel(  
  tabsetPanel(  
    tabPanel("Histograma",  
      plotOutput("distPlot")  
    ),  
    tabPanel("Dispersión",  
      plotOutput("plot2")  
    )  
)
```



Layout: tabsetPanel

Shiny

Ana D. Maldonado
ana.d.maldonado@ual.es

[Introducción](#)

[Estructura de una app](#)

[Widgets](#)

[Outputs](#)

[Reactividad](#)

[Layout](#)

[Publicación](#)

Hasta ahora hemos trabajado con el mismo diseño de web: una única página que contiene todo. Cuando la app empieza a tener un cierto tamaño, es recomendable crear distintas pestañas. Esto se puede hacer usando las funciones `tabsetPanel()` y `tabPanel()`.

```
ui <- fluidPage(  
  titlePanel("Old Faithful Geyser Data"),  
  tabsetPanel(  
    tabPanel("Primera pestaña",  
      # Sidebar with a slider input for number of bins  
      sidebarLayout(  
        sidebarPanel(  
          sliderInput("bins",  
            "Number of bins:",  
            min = 1,  
            max = 50,  
            value = 30)  
        ),  
        # Show a plot of the generated distribution  
        mainPanel(  
          plotOutput("distPlot")  
        )  
      ),  
      tabPanel("Segunda pestaña"),  
      tabPanel("Tercera pestaña")  
    )  
)
```

Old Faithful Geyser Data

Primera pestaña

Segunda pestaña

Tercera pestaña

Resto de pestañas. Pueden contener una función `sidebarLayout()`, al igual que la primera.



¡A PRACTICAR!

Shiny

Ana D. Maldonado
ana.d.maldonado@ual.es

[Introducción](#)

[Estructura de una app](#)

[Widgets](#)

[Outputs](#)

[Reactividad](#)

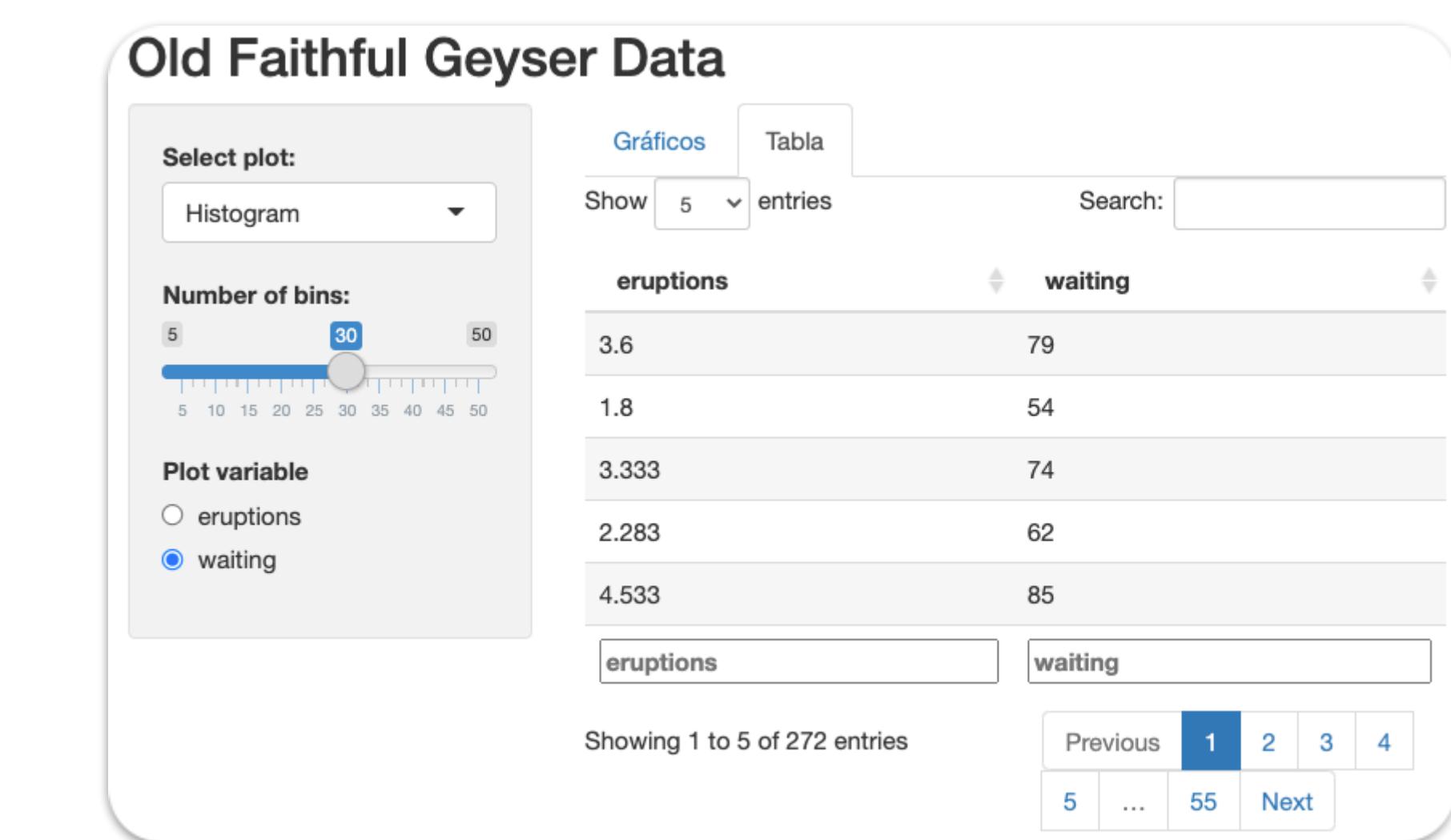
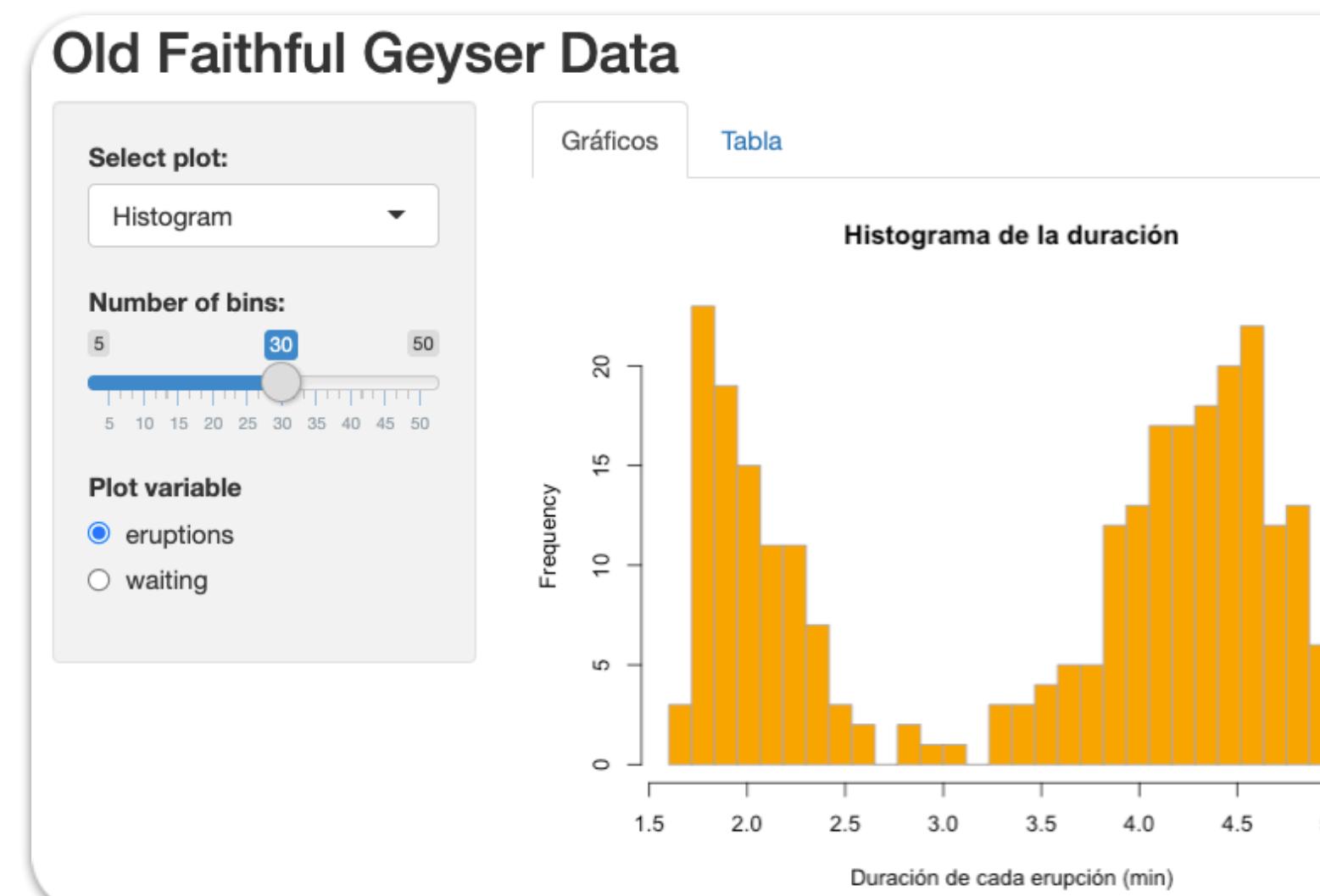
[Layout](#)

[Publicación](#)

- Añade dos pestañas al main panel de App-1: una contendrá los gráficos, y la otra una tabla interactiva que muestre las variables de faithful.

```
output$tabla <- renderDataTable({  
  # Añade los datos  
  ...  
}, options = list(# opciones para configurar la tabla dinámica  
  lengthMenu = list(c(5, 15, -1), c('5', '15', 'All')),  
  pageLength = 5)  
)
```

- Por otro lado, haz que el título y etiqueta del eje x del histograma varíe en función de la variable seleccionada.





¡A PRACTICAR! (Solución)

Shiny

Ana D. Maldonado
ana.d.maldonado@ual.es

[Introducción](#)

[Estructura de una app](#)

[Widgets](#)

[Outputs](#)

[Reactividad](#)

[Layout](#)

[Publicación](#)

```
ui <- fluidPage(  
  
  titlePanel("Old Faithful Geyser Data"),  
  
  sidebarLayout(  
    sidebarPanel(  
      selectInput("plotType", "Select plot:",  
                 choices = c("Histogram", "Scatterplot")),  
      conditionalPanel(condition = "input.plotType == 'Histogram'",  
                      sliderInput(inputId = "bins",  
                                 label = "Number of bins:",  
                                 min = 5,  
                                 max = 50,  
                                 value = 30),  
  
      radioButtons("variable", "Plot variable:",  
                  choices = colnames(faithful))  
    ),  
    mainPanel(  
      tabsetPanel(  
        tabPanel("Gráficos",  
                 plotOutput("distPlot"))  
        ,  
        tabPanel("Tabla",  
                 dataTableOutput("tabla"))  
      )  
    )  
  )  
)
```



¡A PRACTICAR! (Solución)

Shiny

Ana D. Maldonado
ana.d.maldonado@ual.es

[Introducción](#)

[Estructura de una app](#)

[Widgets](#)

[Outputs](#)

[Reactividad](#)

[Layout](#)

[Publicación](#)

```
server <- function(input, output) {  
  
  output$distPlot <- renderPlot({  
    if(input$tipo == "Histogram"){  
      x     <- faithful[, input$variable]  
      bins <- seq(min(x), max(x), length.out = input$bins + 1)  
  
      # cambia títulos histograma  
      main = ifelse(input$variable == "waiting",  
                    "Histograma del tiempo de espera",  
                    "Histograma de la duración")  
      xlab = ifelse(input$variable == "waiting",  
                    "Tiempo de espera entre erupciones (min)",  
                    "Duración de cada erupción (min)")  
      hist(x, breaks = bins, col = 'orange', border = 'darkgrey',  
            main = main,  
            xlab = xlab)  
    }else{  
      plot(faithful)  
    }  
  
  })  
  # tabla dinámica  
  output$tabla <- renderDataTable({  
    faithful  
  },options = list(  
    lengthMenu = list(c(5, 15, -1), c('5', '15', 'All')),  
    pageLength = 5  
  ))  
}
```



Layout: navbarPage

Shiny

Ana D. Maldonado
ana.d.maldonado@ual.es

[Introducción](#)

[Estructura de una app](#)

[Widgets](#)

[Outputs](#)

[Reactividad](#)

[Layout](#)

[Publicación](#)

Una alternativa a fluidPage() es navbarPage() que proporciona un entorno para usar directamente tabPanel(), creando un menú horizontal. <https://shiny.rstudio.com/gallery/navbar-example.html>

```
ui <- navbarPage(  
  title = "Old Faithful Geyser Data",  
  
  tabPanel("Primera pestaña",  
    # Sidebar with a slider input for number of bins  
    sidebarLayout(  
      sidebarPanel(  
        sliderInput("bins",  
          "Number of bins:",  
          min = 1,  
          max = 50,  
          value = 30)  
      ),  
  
      # Show a plot of the generated distribution  
      mainPanel(  
        plotOutput("distPlot")  
      )  
    ),  
    tabPanel("Segunda pestaña"),  
    navbarMenu("Subpanel",  
      tabPanel("Sub1"),  
      tabPanel("Sub2")  
    )  
)
```

Ejemplos/
ejemplo06



Layout: navbarPage

Shiny

Ana D. Maldonado
ana.d.maldonado@ual.es

[Introducción](#)

[Estructura de una app](#)

[Widgets](#)

[Outputs](#)

[Reactividad](#)

[Layout](#)

[Publicación](#)

Una alternativa a fluidPage() es navbarPage() que proporciona un entorno para usar directamente tabPanel(), creando un menú horizontal.

```
ui <- navbarPage(  
  title = "Old Faithful Geyser Data",  
  
  tabPanel("Primera pestaña",  
    # Sidebar with a slider input for number of bins  
    sidebarLayout(  
      sidebarPanel(  
        sliderInput("bins",  
          "Number of bins:",  
          min = 1,  
          max = 50,  
          value = 30)  
      ),  
  
      # Show a plot of the generated distribution  
      mainPanel(  
        plotOutput("distPlot")  
      )  
    ),  
  ),  
  tabPanel("Segunda pestaña"),  
  navbarMenu("Subpanel",  
    tabPanel("Sub1"),  
    tabPanel("Sub2")  
  )  
)
```

No es necesario usar tabsetPanel dentro de navbarPage.



Layout: navbarPage

Shiny

Ana D. Maldonado
ana.d.maldonado@ual.es

[Introducción](#)

[Estructura de una app](#)

[Widgets](#)

[Outputs](#)

[Reactividad](#)

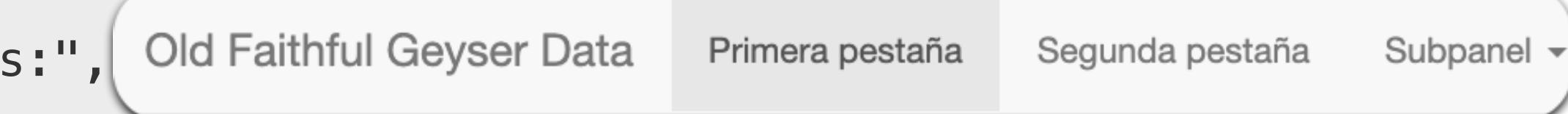
[Layout](#)

[Publicación](#)

Una alternativa a fluidPage() es navbarPage() que proporciona un entorno para usar directamente tabPanel(), creando un menú horizontal.

```
ui <- navbarPage(  
  title = "Old Faithful Geyser Data",  
  
  tabPanel("Primera pestaña",  
    # Sidebar with a slider input for number of bins  
    sidebarLayout(  
      sidebarPanel(  
        sliderInput("bins",  
          "Number of bins:",  
          min = 1,  
          max = 50,  
          value = 30)  
      ),  
      mainPanel(  
        plotOutput("distPlot")  
      )  
    ),  
    tabPanel("Segunda pestaña"),  
    navbarMenu("Subpanel",  
      tabPanel("Sub1"),  
      tabPanel("Sub2"))  
  ))
```

Cada tabPanel aparece en un menú horizontal. Dentro del tabPanel, se puede diseñar cada página como se quiera.





Layout: navbarPage

Shiny

Ana D. Maldonado
ana.d.maldonado@ual.es

[Introducción](#)

[Estructura de una app](#)

[Widgets](#)

[Outputs](#)

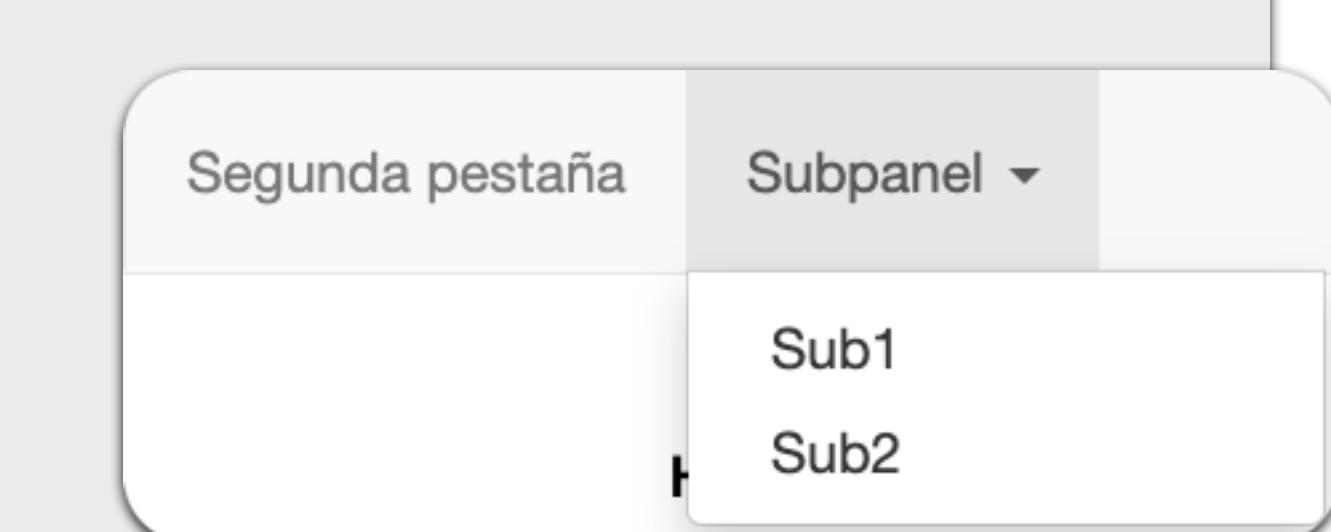
[Reactividad](#)

[Layout](#)

[Publicación](#)

Una alternativa a fluidPage() es navbarPage() que proporciona un entorno para usar directamente tabPanel(), creando un menú horizontal.

```
ui <- navbarPage(  
  title = "Old Faithful Geyser Data",  
  
  tabPanel("Primera pestaña",  
    # Sidebar with a slider input for number of bins  
    sidebarLayout(  
      sidebarPanel(  
        sliderInput("bins",  
          "Number of bins:",  
          min = 1,  
          max = 50,  
          value = 30)  
      ),  
      mainPanel(  
        plotOutput("distPlot")  
      )  
    ),  
    tabPanel("Segunda pestaña"),  
    navbarMenu("Subpanel",  
      tabPanel("Sub1"),  
      tabPanel("Sub2"))  
  ))
```



navbarMenu() crea una pestaña con un menú desplegable. Cada una de estas páginas se pueden llenar con sidebarLayout.



¡A PRACTICAR!

Shiny

Ana D. Maldonado
ana.d.maldonado@ual.es

[Introducción](#)

[Estructura de una app](#)

[Widgets](#)

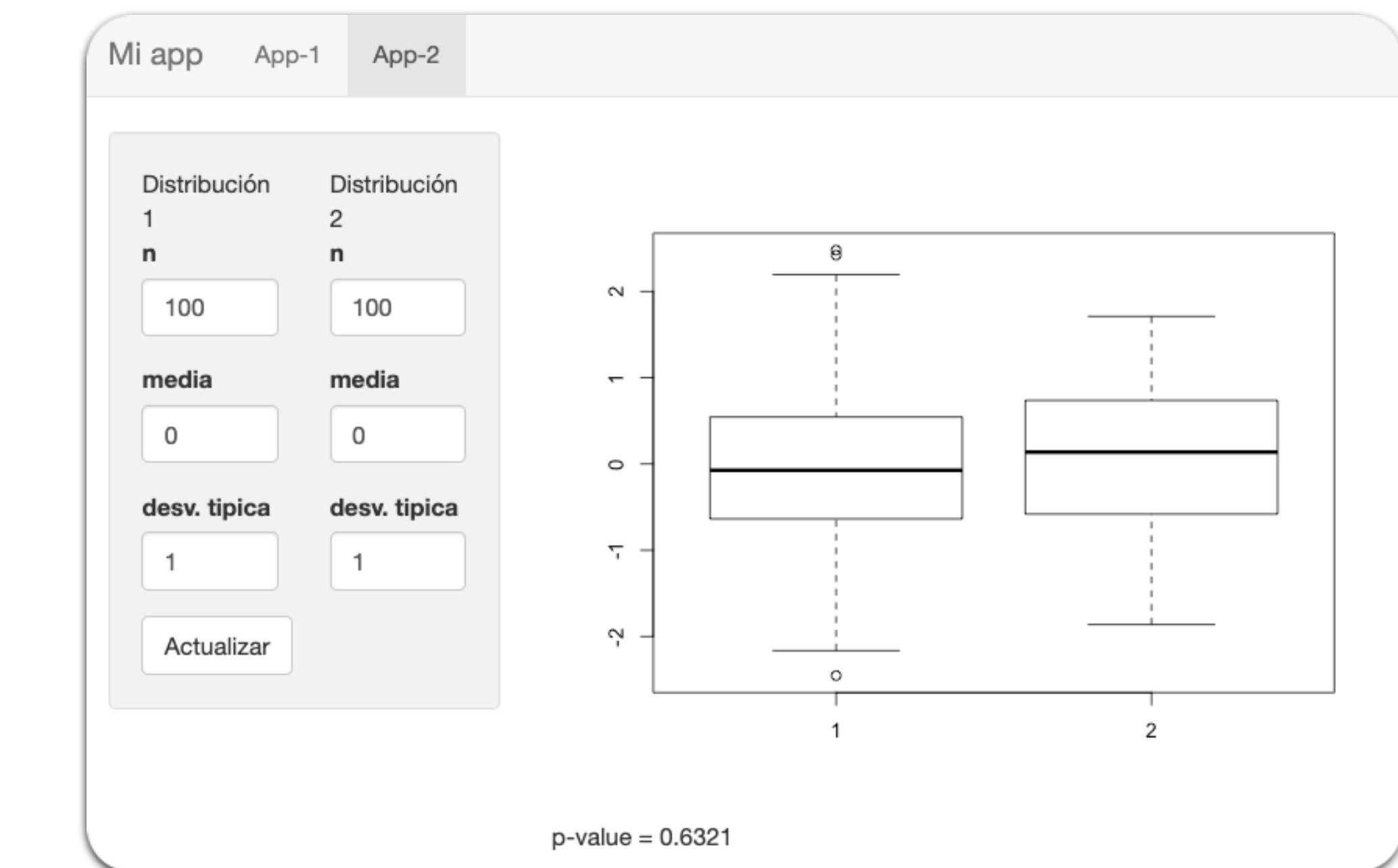
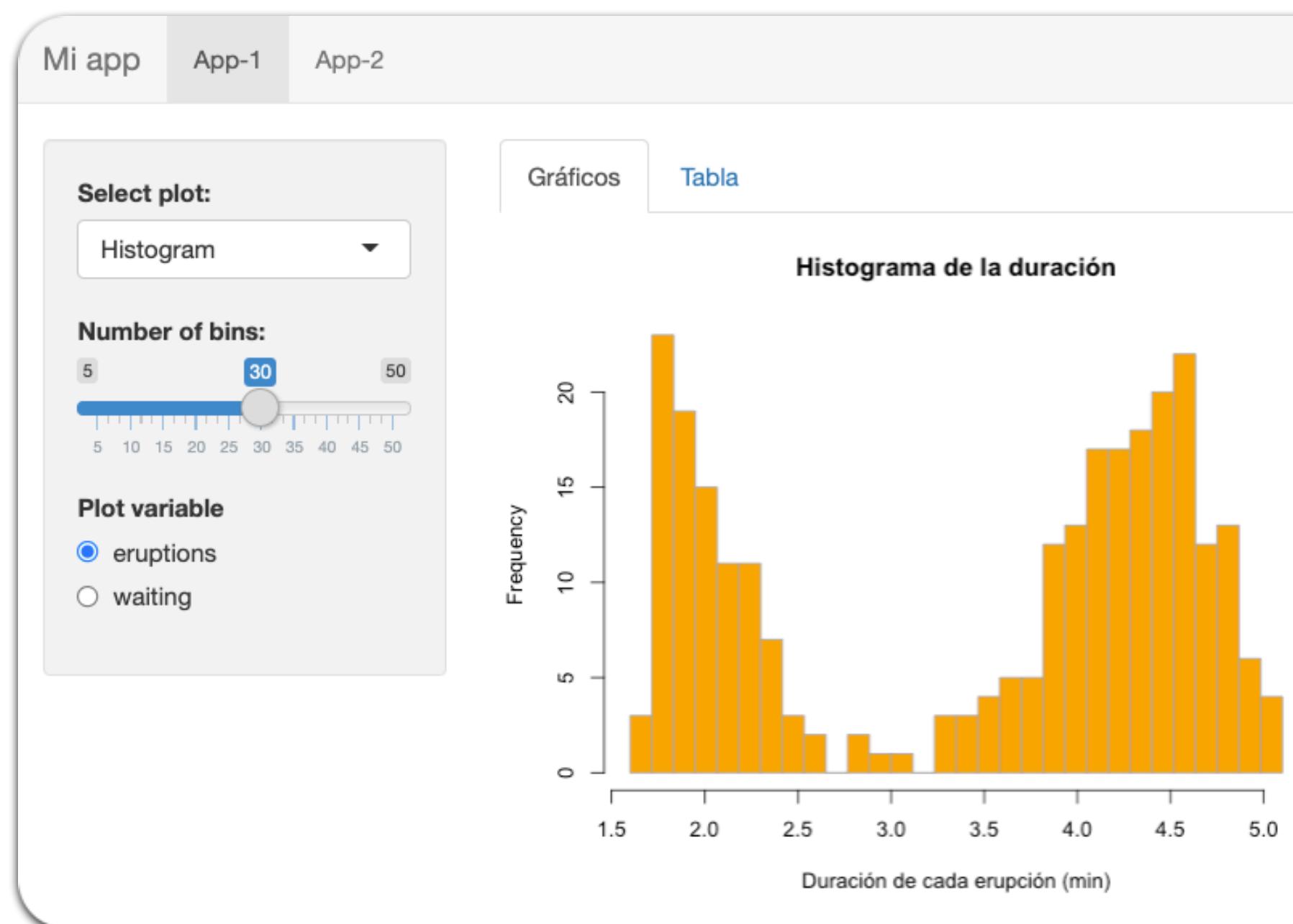
[Outputs](#)

[Reactividad](#)

[Layout](#)

[Publicación](#)

- Crea una nueva aplicación (App-3) con el contenido de App-1 y App-2, usando el layout navbarPage.





¡A PRACTICAR! (Solución)

Shiny

Ana D. Maldonado
ana.d.maldonado@ual.es

[Introducción](#)

[Estructura de una app](#)

[Widgets](#)

[Outputs](#)

[Reactividad](#)

[Layout](#)

[Publicación](#)

```
ui <- navbarPage("Mi app",
                 tabPanel("App-1",
                          # Copiar el contenido de sidebarLayout() de App-1
                          ),
                 tabPanel("App-2",
                          # Copiar el contenido de sidebarLayout() de App-2
                          )
                 )

server <- function(input, output) {
  # Copiar todo el contenido del sever de App-1

  # Copiar todo el contenido del sever de App-2

}

# Run the application
shinyApp(ui = ui, server = server)
```



Temas

Shiny

Ana D. Maldonado
ana.d.maldonado@ual.es

[Introducción](#)

[Estructura de una app](#)

[Widgets](#)

[Outputs](#)

[Reactividad](#)

[Layout](#)

[Publicación](#)

Shiny tiene disponibles una serie de temas que modifican el aspecto de la app relativo a colores, tamaño y tipo de fuente. Podemos ver los temas disponibles en:

<https://shiny.rstudio.com/gallery/shiny-theme-selector.html>

Para usar uno de estos temas en nuestra app, debemos instalar el paquete shinythemes.

```
ui = fluidPage(theme = shinytheme("united") ,  
               ...  
)  
  
ui = navbarPage(theme = shinytheme("cerulean") ,  
                ...  
)
```



Shiny

Ana D. Maldonado
ana.d.maldonado@ual.es

[Introducción](#)

[Estructura de una app](#)

[Widgets](#)

[Outputs](#)

[Reactividad](#)

[Layout](#)

[Publicación](#)

1. Introducción
2. Estructura de una aplicación Shiny
3. Widgets (inputs)
4. Salida reactiva (outputs)
5. Expresiones reactivas
6. Layout
7. Publicación



Publicación de Shiny apps

Shiny

Ana D. Maldonado
ana.d.maldonado@ual.es

[Introducción](#)

[Estructura de una app](#)

[Widgets](#)

[Outputs](#)

[Reactividad](#)

[Layout](#)

[Publicación](#)

Una vez que hemos terminado nuestra shiny app y estamos listos para publicarla, la manera más sencilla de hacerlo es mediante shinyapps.io.

Existe una versión gratuita, que está limitada a 5 apps y 25 horas de uso mensual.



FREE	STARTER	BASIC	STANDARD	PROFESSIONAL
\$ 0 /month	\$ 9 /month (or \$100/year)	\$ 39 /month (or \$440/year)	\$ 99 /month (or \$1,100/year)	\$ 299 /month (or \$3,300/year)
New to Shiny? Deploy your applications for FREE.	More applications. More active hours!	Take your users to the next level!	Password protection? Authenticate your users!	Professional has it all! Personalize your domains.
5 Applications	25 Applications	Unlimited Applications	Unlimited Applications	Unlimited Applications
25 Active Hours	100 Active Hours	500 Active Hours	2,000 Active Hours	10,000 Active Hours
<input checked="" type="checkbox"/> Community Support	<input checked="" type="checkbox"/> Premium Email Support	<input checked="" type="checkbox"/> Performance Boost	<input checked="" type="checkbox"/> Performance Boost	<input checked="" type="checkbox"/> Authentication
<input type="checkbox"/> RStudio Branding		<input checked="" type="checkbox"/> Premium Email Support	<input checked="" type="checkbox"/> Premium Email	<input checked="" type="checkbox"/> Account Sharing
				<input checked="" type="checkbox"/> Performance Boost

Existen otras opciones, pero requieren tener conocimiento sobre configuración de servidores:
<https://www.rstudio.com/products/shiny/shiny-server>



Publicación de Shiny apps

Shiny

Ana D. Maldonado
ana.d.maldonado@ual.es

[Introducción](#)

[Estructura de una app](#)

[Widgets](#)

[Outputs](#)

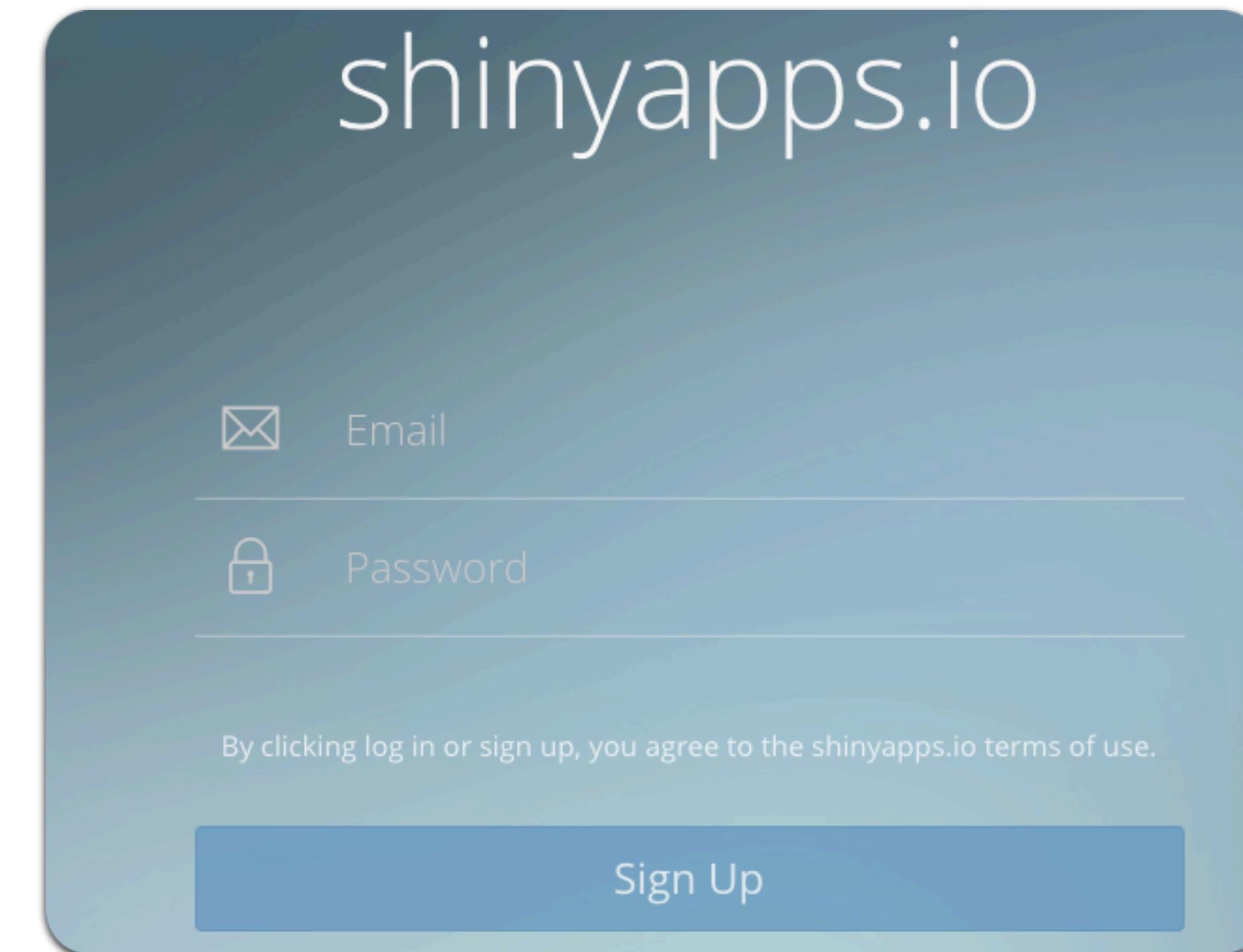
[Reactividad](#)

[Layout](#)

[Publicación](#)

Para usar [shinyapps.io](https://www.shinyapps.io), es necesario crear una cuenta de usuario:

<https://www.shinyapps.io/admin/#/signup>





Publicación de Shiny apps

Shiny

Ana D. Maldonado
ana.d.maldonado@ual.es

[Introducción](#)

[Estructura de una app](#)

[Widgets](#)

[Outputs](#)

[Reactividad](#)

[Layout](#)

[Publicación](#)

Una vez introducimos un email y contraseña, aparece una ventana para que elijamos un nombre de usuario.

ACCOUNT SETUP

Let's get started

You'll need an account before you can deploy any applications. Account names can contain letters, numbers and hyphens, but can't start with a hyphen or a number, and can't end with a hyphen. Pick an account name below to proceed.

https://account.shinyapps.io

Save



Publicación de Shiny apps

Shiny

Ana D. Maldonado
ana.d.maldonado@ual.es

[Introducción](#)

[Estructura de una app](#)

[Widgets](#)

[Outputs](#)

[Reactividad](#)

[Layout](#)

[Publicación](#)

Si ya tienes una cuenta: <https://www.shinyapps.io/admin/#/login>

The screenshot shows the shinyapps.io dashboard with the following interface elements:

- Header:** shinyapps.io, Help, Account: admaldonado, user icon.
- Left Sidebar:** Dashboard, Applications, Account.
- Center Content:**
 - WHAT'S NEW?**: Empty section.
 - APPLICATIONS ONLINE**: Shows 4 applications online.
 - Running: 0
 - Sleeping: 4
 - Archived: 0
 - RECENT APPLICATIONS**: Table of recent applications.

ID	Name	Status
1737011	visualizaciones	Sleeping
2649704	rPACI	Sleeping
2188549	App_dados	Sleeping
1726874	MoTBFs_App	Sleeping



Publicación de Shiny apps

Shiny

Ana D. Maldonado
ana.d.maldonado@ual.es

[Introducción](#)

[Estructura de una app](#)

[Widgets](#)

[Outputs](#)

[Reactividad](#)

[Layout](#)

[Publicación](#)

```
5 # Find out more about building applications with Shiny here:  
6 #  
7 #     http://shiny.rstudio.com/  
8 #  
9  
10 library(shiny)  
11  
12 # Define UI for application that draws a histogram  
13 ui <- fluidPage(  
14  
15     # Application title  
16     titlePanel("Old Faithful Geyser Data"),  
17  
18     # Sidebar with a slider input for number of bins  
19     ...  
20 )
```



Publicación de Shiny apps

Shiny

Ana D. Maldonado
ana.d.maldonado@ual.es

[Introducción](#)

[Estructura de una app](#)

[Widgets](#)

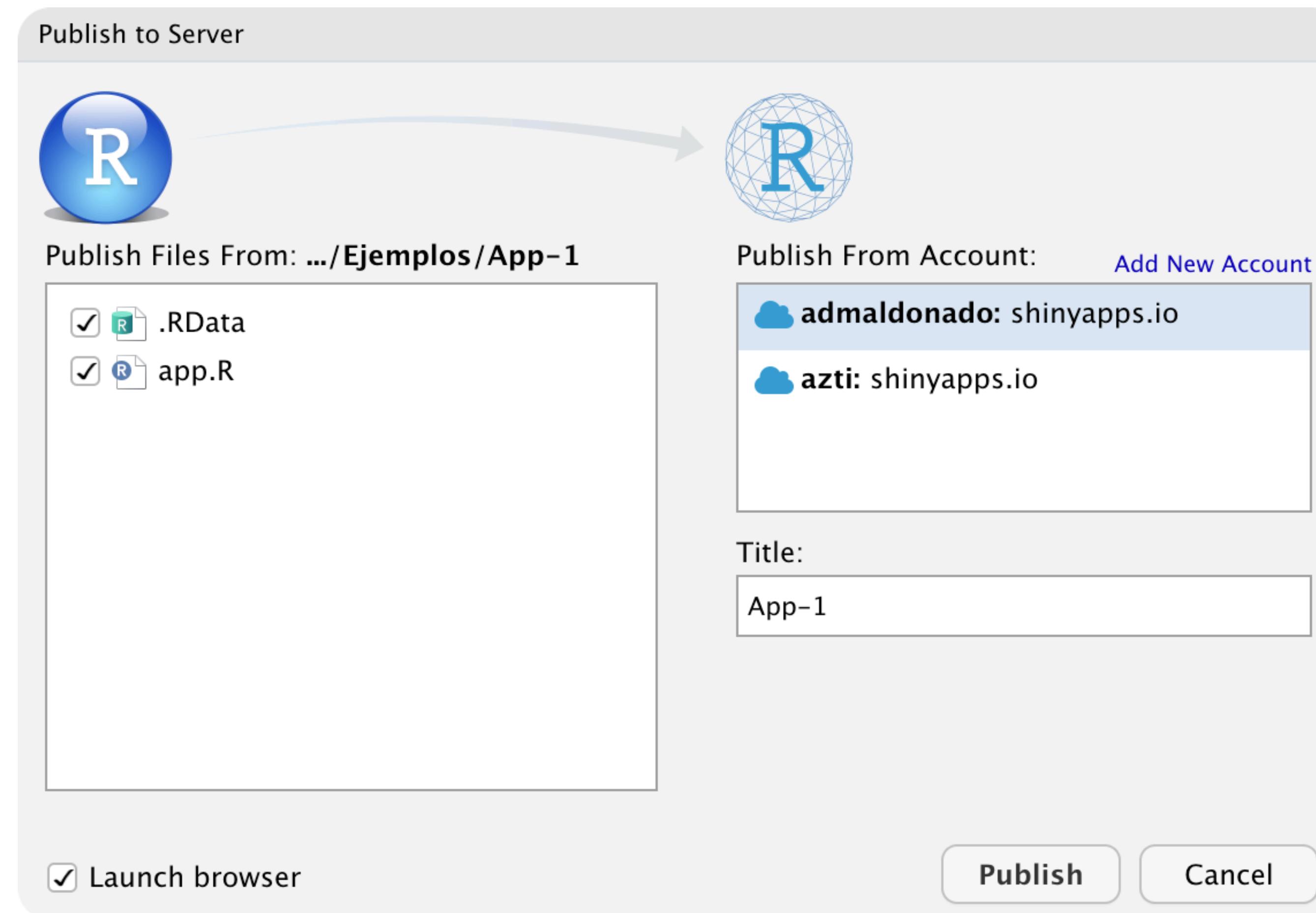
[Outputs](#)

[Reactividad](#)

[Layout](#)

[Publicación](#)

Si es la primera app que se publica, el recuadro “Publishing From Account” estará vacío. En este caso, clicar sobre “Add New Account”.





Publicación de Shiny apps

Shiny

Ana D. Maldonado
ana.d.maldonado@ual.es

[Introducción](#)

[Estructura de una app](#)

[Widgets](#)

[Outputs](#)

[Reactividad](#)

[Layout](#)

[Publicación](#)

En la nueva ventana que se abre, seleccionamos ShinyApps.io.

The screenshot shows a modal dialog box titled "Connect Account". It contains two options: "ShinyApps.io" and "RStudio Connect". The "ShinyApps.io" option is highlighted with a blue box. To the right of the dialog, a sidebar shows a list of accounts: "aldonado: shinyapps.io" and "shinyapps.io". The "Add New Account" button in the sidebar is also highlighted with a blue box. At the bottom of the dialog are "Publish" and "Cancel" buttons, and at the bottom of the sidebar is a "Cancel" button.

Connect Account

Connect Account

ShinyApps.io
A cloud service run by RStudio. Publish Shiny applications and interactive documents to the Internet. >

RStudio Connect
RStudio Connect is a server product from RStudio for secure sharing of applications, reports, plots, and APIs. >

From Account:

aldonado: shinyapps.io

shinyapps.io

Add New Account

Cancel

Publish Cancel



Publicación de Shiny apps

Shiny

Ana D. Maldonado
ana.d.maldonado@ual.es

[Introducción](#)

[Estructura de una app](#)

[Widgets](#)

[Outputs](#)

[Reactividad](#)

[Layout](#)

[Publicación](#)

La siguiente pantalla indica que vayamos a nuestra cuenta de ShinyApps, copiemos un token y lo peguemos en el recuadro.

Connect Account

Connect Account

ShinyApps.io
A cloud service run by RStudio. Publish Shiny applications and interactive documents to the Internet.

RStudio Connect
RStudio Connect is a server product from RStudio for secure delivery of applications, reports, plots, and APIs.

Connect ShinyApps.io Account

Go to [your account on ShinyApps](#) and log in.
Click your name, then choose **Tokens** from your account menu.
Click **Show** on the token you want to use, then **Show Secret** and **Copy to Clipboard**. Paste the result here:

Need a ShinyApps.io account? [Get started here.](#)

Connect Account **Cancel**



Publicación de Shiny apps

Shiny

Ana D. Maldonado

ana.d.maldonado@ual.es

Clicamos en “Go to [your account on ShinyApps](#) and log in”. Se abre la web de shinyapps.io en el navegador. Si ya tienes la cuenta creada, pincha sobre “Dashboard”.

The screenshot shows a web browser window for shinyapps.io. The main content area features a large blue banner with the text "Share your Shiny Applications Online" and a "Sign Up" button. Below the banner is a subtext "Deploy your Shiny applications on the Web in minutes" and an image of a computer monitor displaying a map-based Shiny application. At the top of the browser window, there is a navigation bar with links for Home, Features, Pricing, Support, and Dashboard. The "Dashboard" link is highlighted with a blue box. To the right of the browser window, a modal dialog box titled "Connect ShinyApps.io Account" is displayed. It contains instructions: "Go to [your account on ShinyApps](#) and log in.", "Click your name, then choose Tokens from your account menu.", and "Click Show on the token you want to use, then Show Secret and Copy to Clipboard. Paste the result here:". There is a text input field for pasting the secret key. At the bottom of the dialog are "Connect Account" and "Cancel" buttons.

Intro
Estr
Wid
Out
Rea
Lay
Pub

shinyapps.io by RStudio

Home Features Pricing Support Dashboard

Share your Shiny Applications Online

Deploy your Shiny applications on the Web in minutes

Sign Up

shinyapps.io

Connect ShinyApps.io Account

Go to [your account on ShinyApps](#) and log in.

Click your name, then choose **Tokens** from your account menu.

Click **Show** on the token you want to use, then **Show Secret** and **Copy to Clipboard**. Paste the result here:

Need a ShinyApps.io account? [Get started here.](#)

Connect Account Cancel



Publicación de Shiny apps

Shiny

Ana D. Maldonado
ana.d.maldonado@ual.es

[Introducción](#)

[Estructura de una app](#)

[Widgets](#)

[Outputs](#)

[Reactividad](#)

[Layout](#)

[Publicación](#)

Clicamos sobre el icono de perfil y en “Tokens”.

The screenshot shows the shinyapps.io admin dashboard. On the right, a user profile menu is open, showing options: Profile (highlighted with a red box), Tokens (highlighted with a blue box), and Log out.

The main dashboard area displays:

- WHAT'S NEW?**: No new applications.
- RECENT APPLICATIONS**: A table listing recent applications:

ID	Name	Status
1737011	visualizaciones	Sleeping
2649704	rPACI	Sleeping
2188549	App_dados	Sleeping
1726874	MoTBFs_App	Sleeping

On the left sidebar, there are navigation links: Dashboard, Applications, and Account.

In the center, there's a summary card: **4 APPLICATIONS ONLINE**. Below it, categories are listed: Running (0), Sleeping (4), and Archived (0).

© 2020 RStudio, PBC | All Rights Reserved | Terms Of Use



Publicación de Shiny apps

Shiny

Ana D. Maldonado
ana.d.maldonado@ual.es

[Introducción](#)

[Estructura de una app](#)

[Widgets](#)

[Outputs](#)

[Reactividad](#)

[Layout](#)

[Publicación](#)

Si es la primera app que publicas, no habrá ningún token creado. Pulsa sobre “+ Add Token”.

The screenshot shows a web browser window for shinyapps.io. The URL in the address bar is `shinyapps.io/admin/#/tokens`. The page title is "shinyapps.io". On the left, there is a sidebar with links: "Dashboard", "Applications", and "Account". The main content area is titled "TOKENS" and contains two columns: "Token" and "Secret". At the bottom of the main area, there is a "Show 5 entries per page" dropdown and a set of navigation buttons labeled "First", "<", "1", ">", and "Last". In the top right corner of the main content area, there is a green button with a white plus sign and the text "+ Add Token", which is highlighted with a red rectangular box.



Publicación de Shiny apps

Shiny

Ana D. Maldonado
ana.d.maldonado@ual.es

[Introducción](#)

[Estructura de una app](#)

[Widgets](#)

[Outputs](#)

[Reactividad](#)

[Layout](#)

[Publicación](#)

Aparecerá un nuevo token. Pulsa sobre “Show”

The screenshot shows the shinyapps.io admin interface with the URL `shinyapps.io/admin/#/tokens`. The page title is "TOKENS". On the left, there's a sidebar with "Dashboard", "Applications", and "Account". The main area displays a table with two columns: "Token" and "Secret". In the "Secret" column, a long string of characters (XXXXXXXXXXXXXXXXXXXXXX) is shown, with the "Show" button next to it highlighted by a red box. There are also "Add Token" and "Delete" buttons. At the bottom, there's a pagination control showing page 1 of 1, and copyright information: "© 2020 RStudio, PBC | All Rights Reserved | Terms Of Use".



Publicación de Shiny apps

Shiny

Ana D. Maldonado
ana.d.maldonado@ual.es

[Introducción](#)

[Estructura de una app](#)

[Widgets](#)

[Outputs](#)

[Reactividad](#)

[Layout](#)

[Publicación](#)

En la nueva ventana que aparece, pulsa sobre “Show secret”.

The shinyapps package must be authorized to your account using a token and secret. To do this, click the copy button below and we'll copy the whole command you need to your clipboard. Just paste it into your console to authorize your account. Once you've entered the command successfully in R, that computer is now authorized to deploy applications to your shinyapps.io account.

```
rsconnect::setAccountInfo(name='admaldonado',
                           token=[REDACTED]
                           secret='<SECRET>')
```

Show secret

Copy to clipboard

OK



Publicación de Shiny apps

Shiny

Ana D. Maldonado
ana.d.maldonado@ual.es

[Introducción](#)

[Estructura de una app](#)

[Widgets](#)

[Outputs](#)

[Reactividad](#)

[Layout](#)

[Publicación](#)

A continuación, pulsa sobre “Copy to clipboard”.

The `shinyapps` package must be authorized to your account using a token and secret. To do this, click the copy button below and we'll copy the whole command you need to your clipboard. Just paste it into your console to authorize your account. Once you've entered the command successfully in R, that computer is now authorized to deploy applications to your shinyapps.io account.

```
rsconnect::setAccountInfo(name='admaldonado',
                           token=[REDACTED]
                           secret=[REDACTED])
```

[Hide secret](#)

[Copy to clipboard](#)

[OK](#)



Publicación de Shiny apps

Shiny

Ana D. Maldonado
ana.d.maldonado@ual.es

[Introducción](#)

[Estructura de una app](#)

[Widgets](#)

[Outputs](#)

[Reactividad](#)

[Layout](#)

[Publicación](#)

Copia el texto que aparece marcado y vuelve a RStudio.

The screenshot shows a web browser window for shinyapps.io with the URL `shinyapps.io/admin/#/tokens`. A modal dialog box is open, prompting the user to copy the command to the clipboard. The command is:

```
rsconnect::setAccountInfo(name='admaldonado', token='0F97291241C')
```

The modal has two buttons: "Cancel" and "OK". In the background, the shinyapps.io interface shows a list of tokens for the account "admaldonado". One token is highlighted with a red bar, and a "Copy to clipboard" button is visible next to it. Other buttons in the background include "Hide secret", "Add Token", "OK", "Delete", "First", "Last", and a page navigation bar.



Publicación de Shiny apps

Shiny

Ana D. Maldonado
ana.d.maldonado@ual.es

[Introducción](#)

[Estructura de una app](#)

[Widgets](#)

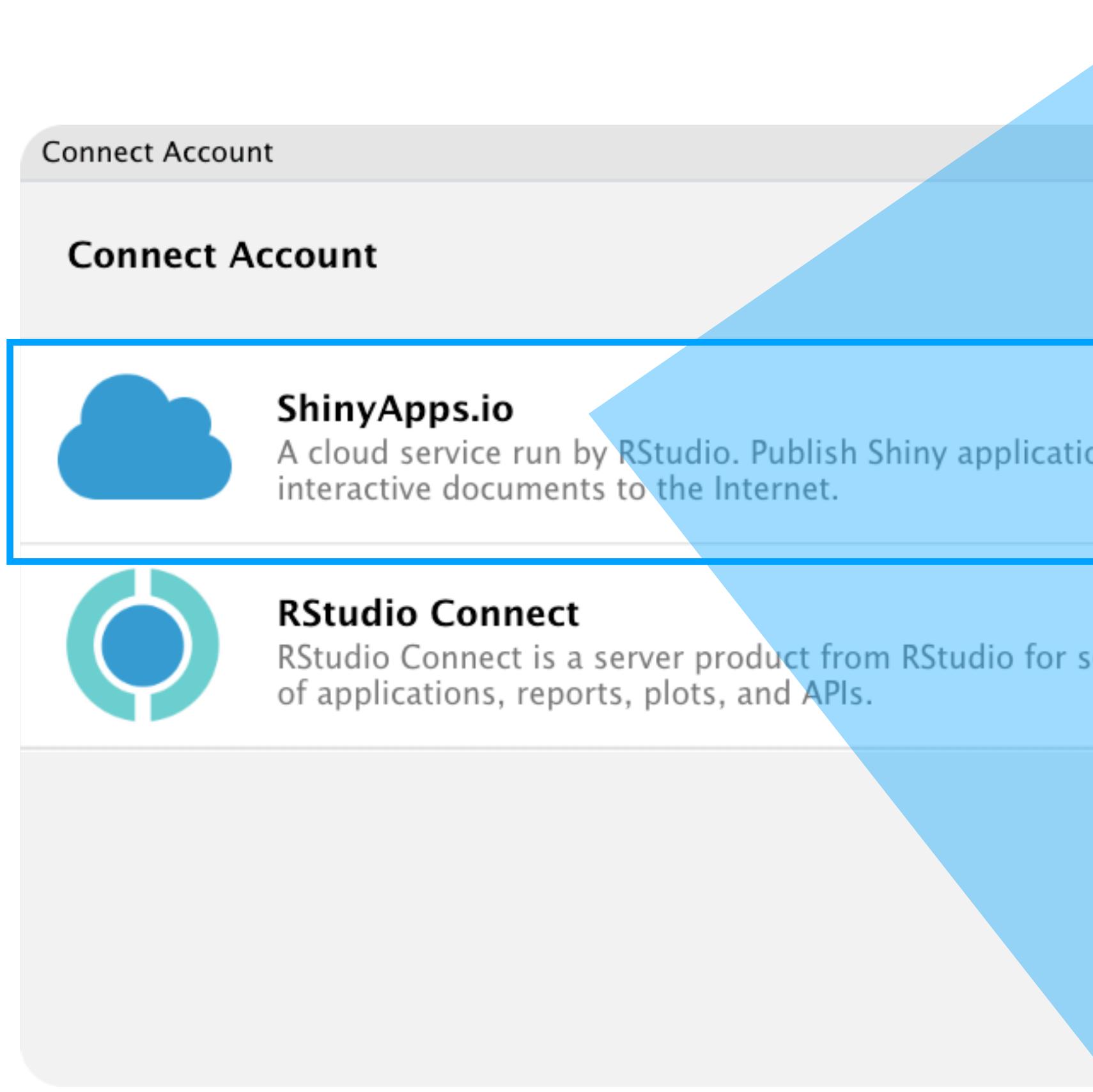
[Outputs](#)

[Reactividad](#)

[Layout](#)

[Publicación](#)

En RStudio y pegamos el token copiado en el recuadro. Pulsamos sobre “Connect Account”.



The screenshot shows the "Connect Account" dialog in RStudio. It has two main sections: "ShinyApps.io" and "RStudio Connect". The "ShinyApps.io" section is highlighted with a blue border. It contains a blue cloud icon and the text: "ShinyApps.io A cloud service run by RStudio. Publish Shiny applications and interactive documents to the Internet." The "RStudio Connect" section contains a teal circular icon and the text: "RStudio Connect RStudio Connect is a server product from RStudio for secure delivery of applications, reports, plots, and APIs." Below the dialog, a large blue arrow points from the "ShinyApps.io" section towards the right side of the slide, where the "Connect ShinyApps.io Account" dialog is shown.

Connect Account

Back

Connect ShinyApps.io Account

Go to [your account on ShinyApps](#) and log in.

Click your name, then choose **Tokens** from your account menu.

Click **Show** on the token you want to use, then **Show Secret** and **Copy to Clipboard**. Paste the result here:

```
rsconnect::setAccountInfo(name='admaldonado',  
token='[REDACTED]',  
secret='[REDACTED]',  
)
```

Need a ShinyApps.io account? [Get started here.](#)

Connect Account **Cancel**



Publicación de Shiny apps

Shiny

Ana D. Maldonado
ana.d.maldonado@ual.es

[Introducción](#)

[Estructura de una app](#)

[Widgets](#)

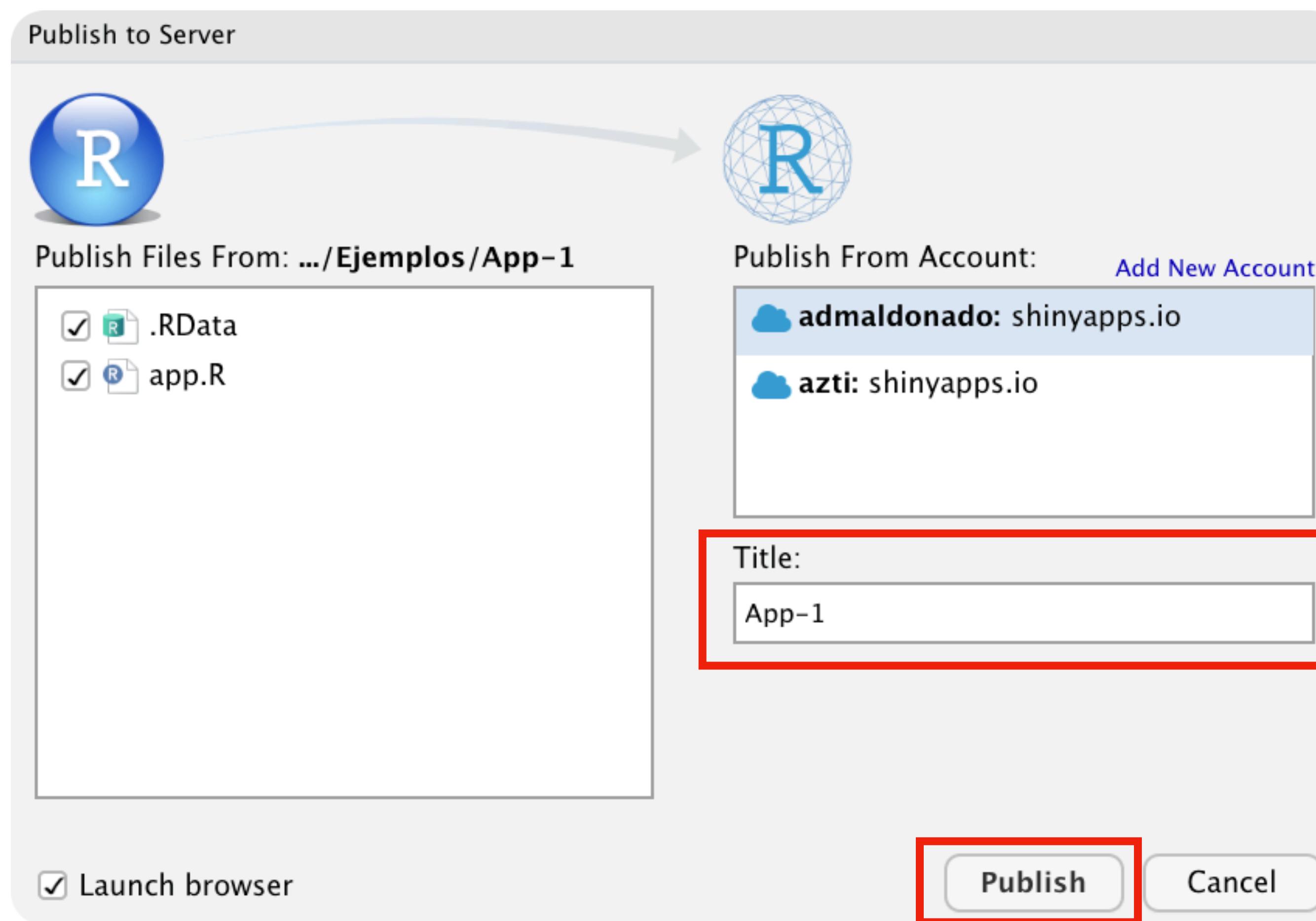
[Outputs](#)

[Reactividad](#)

[Layout](#)

[Publicación](#)

Aparecerá el nombre de nuestra cuenta en el recuadro “Publish From Account”. Elegimos un nombre para nuestra app y pulsamos sobre “Publish”.





Publicación de Shiny apps

Shiny

Ana D. Maldonado
ana.d.maldonado@ual.es

[Introducción](#)

[Estructura de una app](#)

[Widgets](#)

[Outputs](#)

[Reactividad](#)

[Layout](#)

[Publicación](#)

RStudio nos muestra el avance del proceso (puede tardar varios minutos).

```
Console Terminal × Deploy × Jobs ×
.../Ejemplos/App-1

Preparing to deploy application...DONE
Uploading bundle for application: 4208096...DONE
Deploying bundle: 4659358 for application: 4208096 ...
Waiting for task: 942622167
  building: Building image: 5350157
  building: Installing packages
  building: Installing files
  building: Pushing image: 5350157
  deploying: Starting instances
  unstaging: Stopping old instances
Application successfully deployed to https://admaldonado.shinyapps.io/App-1/
Deployment completed: https://admaldonado.shinyapps.io/App-1/
```



Publicación de Shiny apps

Shiny

Ana D. Maldonado
ana.d.maldonado@ual.es

[Introducción](#)

[Estructura de una app](#)

[Widgets](#)

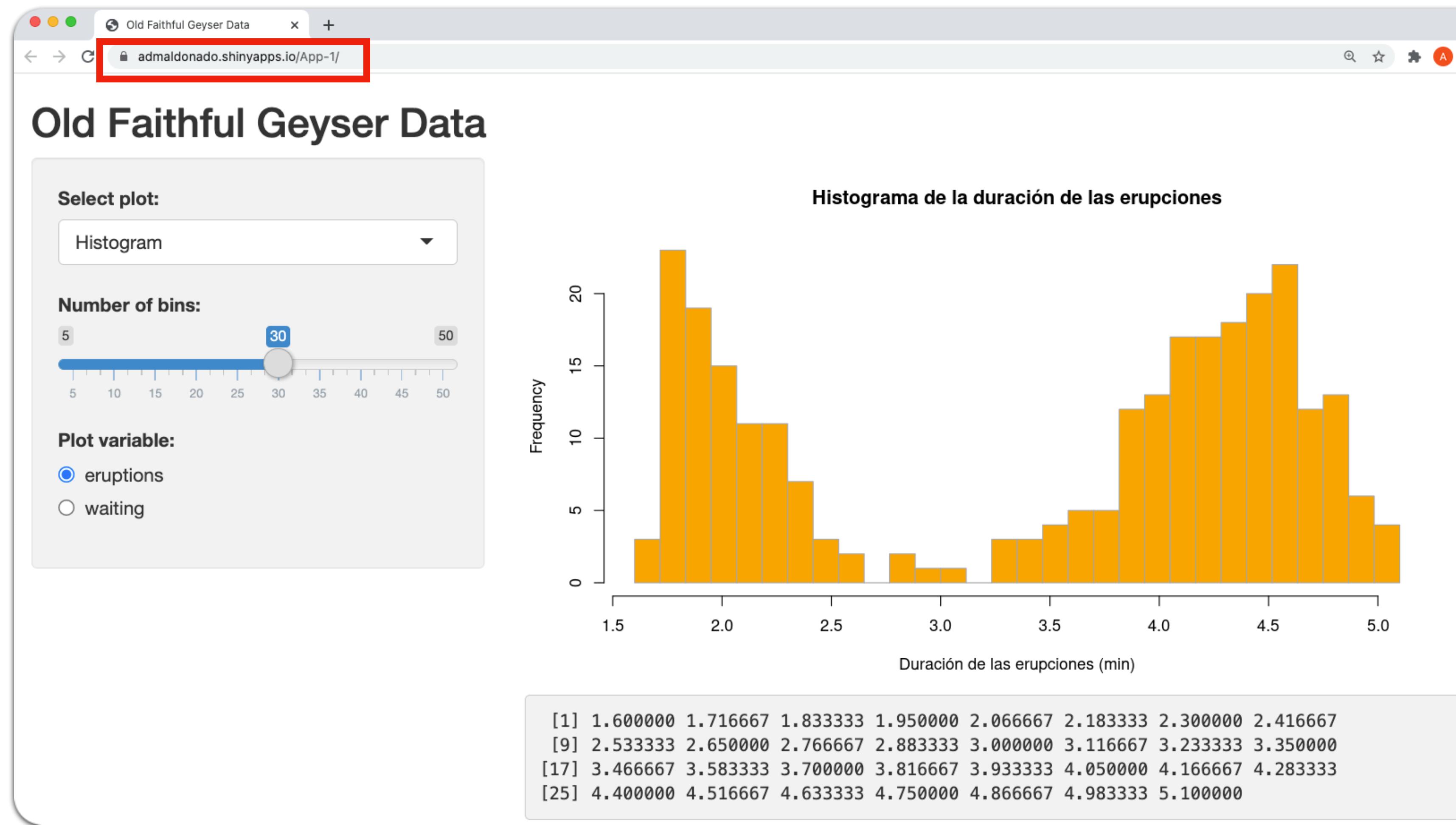
[Outputs](#)

[Reactividad](#)

[Layout](#)

[Publicación](#)

Cuando termina, se abre una ventana en el navegador con nuestra app. ¡Ya puede ser visitada por cualquier usuario!





Publicación de Shiny apps

Shiny

Ana D. Maldonado
ana.d.maldonado@ual.es

[Introducción](#)

[Estructura de una app](#)

[Widgets](#)

[Outputs](#)

[Reactividad](#)

[Layout](#)

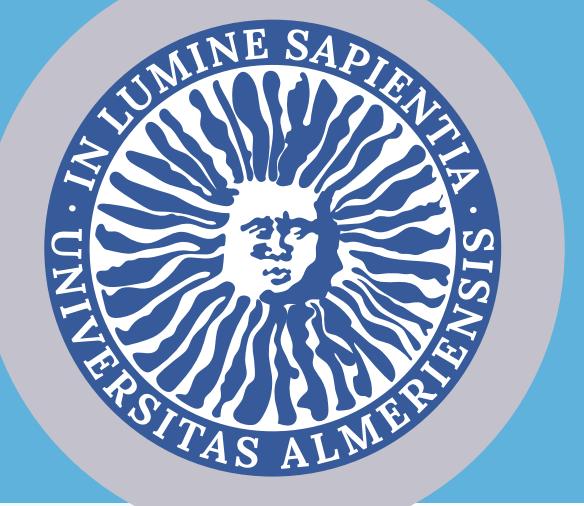
[Publicación](#)

En el dashboard de nuestra cuenta de [shinyapps.io](#) aparecerá la nueva app creada.

The screenshot shows the shinyapps.io admin interface. On the left, there's a sidebar with links for Dashboard, Applications, and Account. The main area has a 'WHAT'S NEW?' section and a 'RECENT APPLICATIONS' table. The table lists five applications:

ID	Name	Status
4208096	App-1	Running
1737011	visualizaciones	Sleeping
2649704	rPACI	Sleeping
2188549	App_dados	Sleeping
1726874	MoTBFs_App	Sleeping

The first application, 'App-1', is highlighted with a red border.



Publicación de Shiny apps

Shiny

Ana D. Maldonado
ana.d.maldonado@ual.es

[Introducción](#)

[Estructura de una app](#)

[Widgets](#)

[Outputs](#)

[Reactividad](#)

[Layout](#)

[Publicación](#)

Si pulsamos sobre “Applications” podemos gestionar nuestras apps.

ID	Name	Status	Instances	Deployed Date	Created Date	Action	Action	Action	Action
4208096	App-1	Running	1	May 31, 2021	May 31, 2021				
2649704	rPACI	Sleeping	1	Aug 7, 2020	Aug 7, 2020				
1726874	MoTBFs_App	Sleeping	1	May 4, 2020	Jan 29, 2020				
2188549	App_dados	Sleeping	1	Apr 26, 2020	Apr 26, 2020				
1737011	visualizaciones	Sleeping	1	Mar 20, 2020	Jan 30, 2020				



Publicación de Shiny apps

Shiny

Ana D. Maldonado
ana.d.maldonado@ual.es

[Introducción](#)

[Estructura de una app](#)

[Widgets](#)

[Outputs](#)

[Reactividad](#)

[Layout](#)

[Publicación](#)

Entrando en una de ellas, disponemos de varios menús donde podemos ver un resumen de los datos de la app (overview), o información sobre la ejecución de la app (logs), muy importante en el caso de que se produzcan errores.

The screenshot shows the shinyapps.io admin interface for an application with ID 4208096, named "App-1". The interface includes a sidebar with "Dashboard", "Applications" (All, Running, Sleeping, Archived), and "Account" sections. The main area has tabs for Overview, Metrics, URLs, Settings, Users, Logs, Restart, Archive, and Delete. The Overview section displays details like Id (4208096), Name (App-1), URL (<https://admaldonado.shinyapps.io/App-1/>), Status (Running), Size (large), Deployed (May 31, 2021), Updated (May 31, 2021), Created (May 31, 2021), and a Download button. The Logs section shows a chart of application usage from May 25 to May 31, 2021, with a total of 0.43 hours. The chart shows a sharp increase on May 31.



Publicación de Shiny apps

Shiny

Ana D. Maldonado
ana.d.maldonado@ual.es

[Introducción](#)

[Estructura de una app](#)

[Widgets](#)

[Outputs](#)

[Reactividad](#)

[Layout](#)

[Publicación](#)

Alternativamente, podemos usar directamente la función `deployApp()` del paquete `rsconnect` para publicar la app.

Para ello, abre un nuevo script de R y guárdalo en el directorio donde está el archivo `app.R` que quieras publicar. Después, copia tu token, establece el directorio de trabajo y llama a `deployApp()`.

```
install.packages(rsconnect)
library(rsconnect)

# Copia y pega el token creado en shinyapps.io
rsconnect::setAccountInfo(name='admaldonado',
                           token='*****',
                           secret='*****')

# Establece tu directorio de trabajo en el directorio donde está el archivo app.R que quieras
# publicar. Para hacerlo de forma automática, el script debe estar guardado en el directorio de la
# app.
setwd(dirname(rstudioapi::getActiveDocumentContext()$path))

# Elige un nombre para al app (appTitle) y especifica tu nombre de usuario de shinyapps.io
# (account).
rsconnect::deployApp(appTitle = 'App-1', account = 'admaldonado')
```



Enlaces de interés

Shiny

Ana D. Maldonado

ana.d.maldonado@ual.es

[Introducción](#)

[Estructura de una app](#)

[Widgets](#)

[Outputs](#)

[Reactividad](#)

[Layout](#)

[Publicación](#)

- Web oficial de Shiny: <https://shiny.rstudio.com/>
- Tutorial de Shiny: <https://shiny.rstudio.com/tutorial/>
- Wickham, H. (2021). *Mastering shiny*. "O'Reilly Media, Inc.": <https://mastering-shiny.org/>
- ¿Estás preparad@ para Shiny? Pon a prueba tu conocimiento sobre R: <https://shiny.rstudio.com/tutorial/quiz/>

Muchas gracias por su atención

