Desarrollo de aplicaciones web con shiny

Curso de Formación del Profesorado

10 y 11 de febrero de 2022

Ana D. Maldonado

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

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?

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")
```

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

Introducción
Estructura de una app
Widgets
Outputs
Reactividad
Layout
Publicación

Introducción

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 = uni, 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.

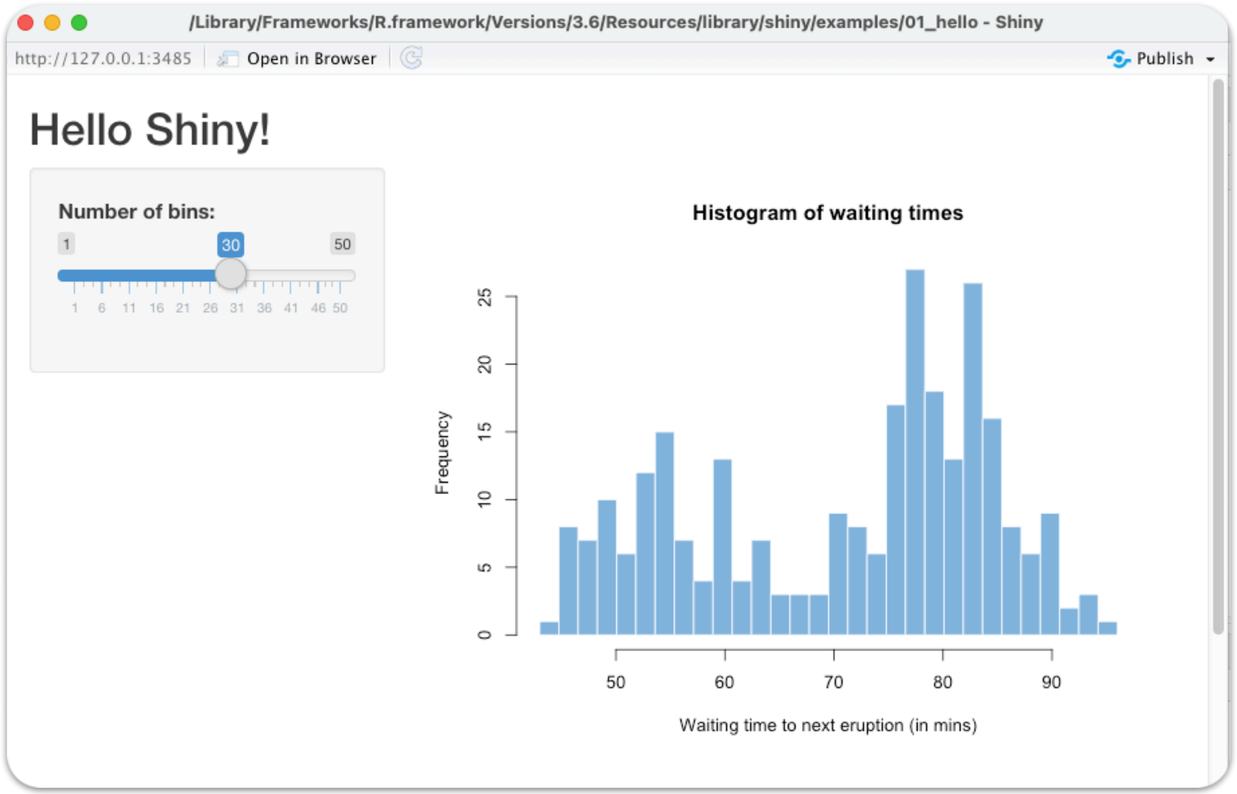
Introducción

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

Introducción
Estructura de una app
Widgets
Outputs
Reactividad
Layout
Publicación

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:

```
# Ejecutar aplicación de ejemplo

  runExample("01_hello")
```



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

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

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

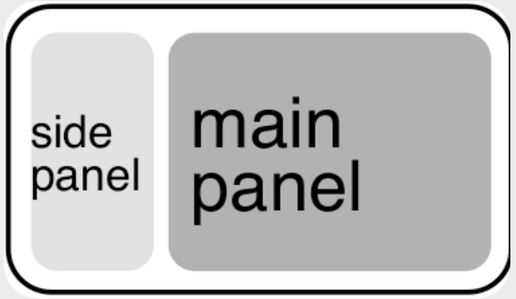
Estructura de una aplicación de shiny

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



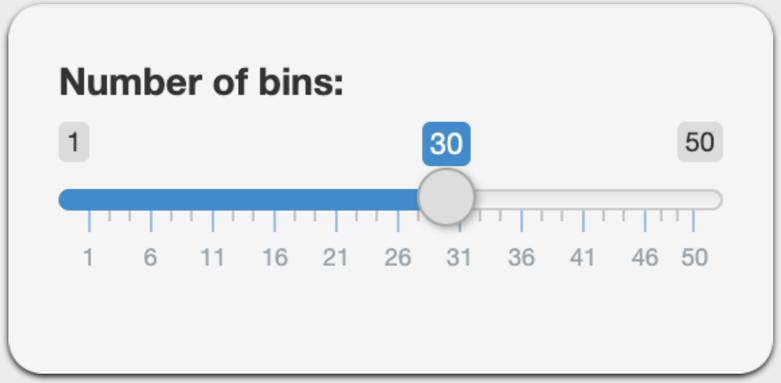
Estructura de una aplicación de shiny

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

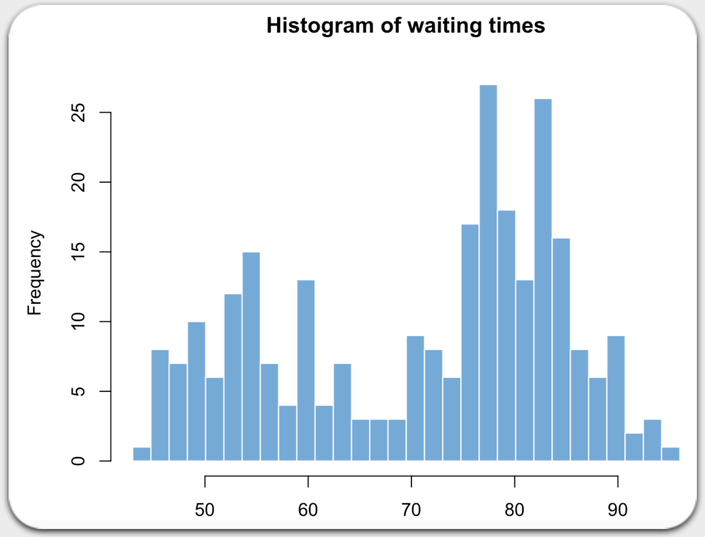


Estructura de una aplicación de shiny

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")

    )
  )
)
```



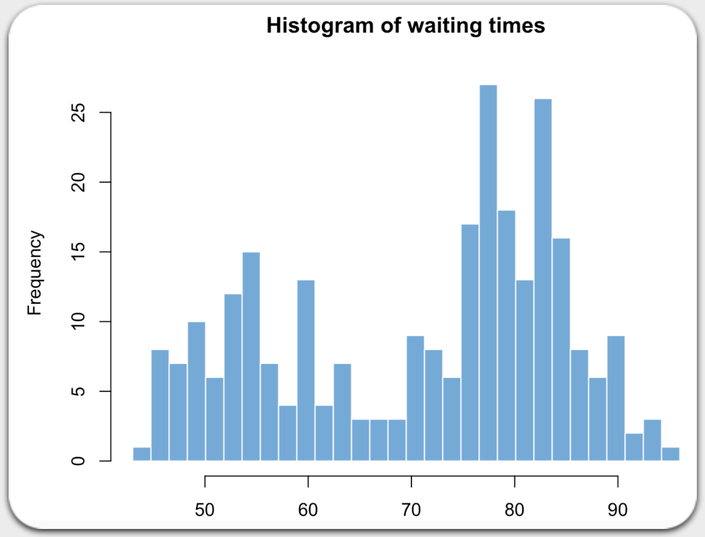
Panel principal: lugar donde se muestran los resultados (**outputs**), que se crean en la función server

Estructura de una aplicación de shiny

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”.

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

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

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

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

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

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)

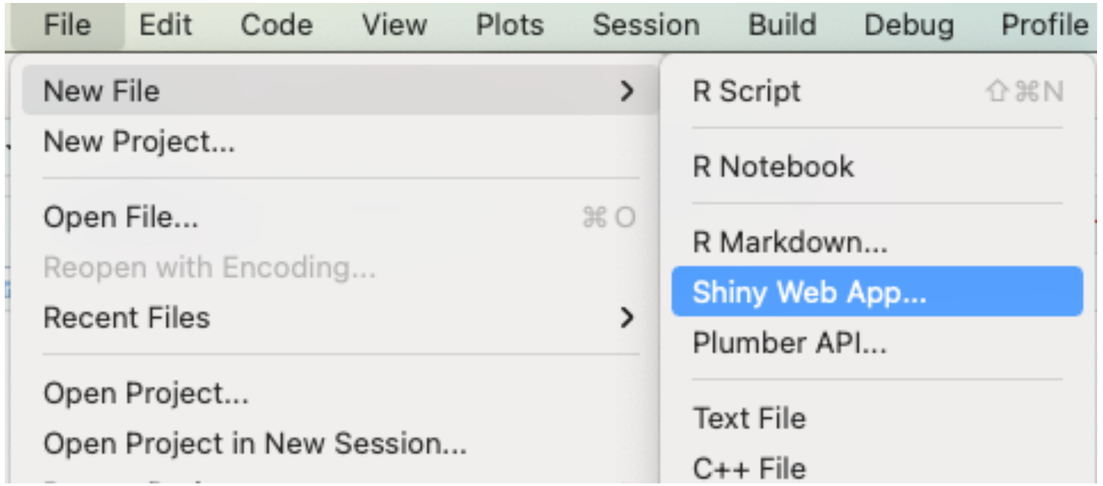
input$bins corresponde con sliderInput(“bins”,…) del UI. Cada vez que el usuario mueva el deslizador, el histograma se actualizará

¡A PRACTICAR!

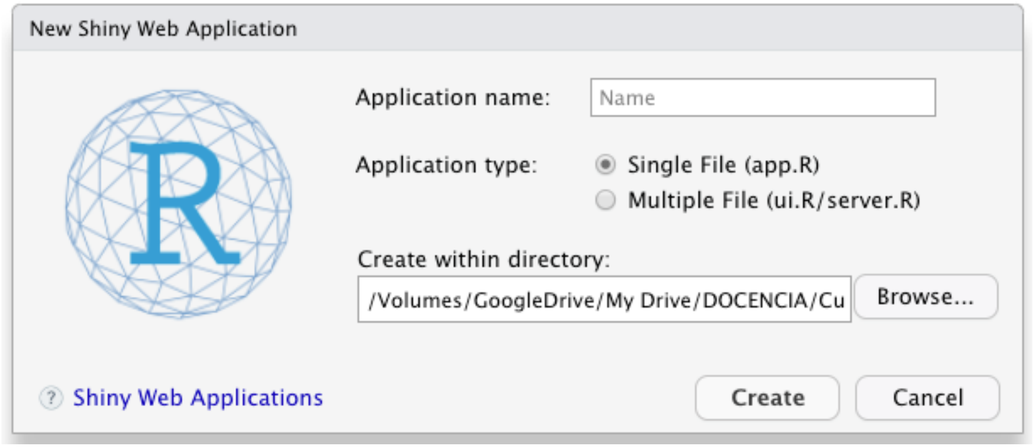
Ana D. Maldonado
ana.d.maldonado@ual.es

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.

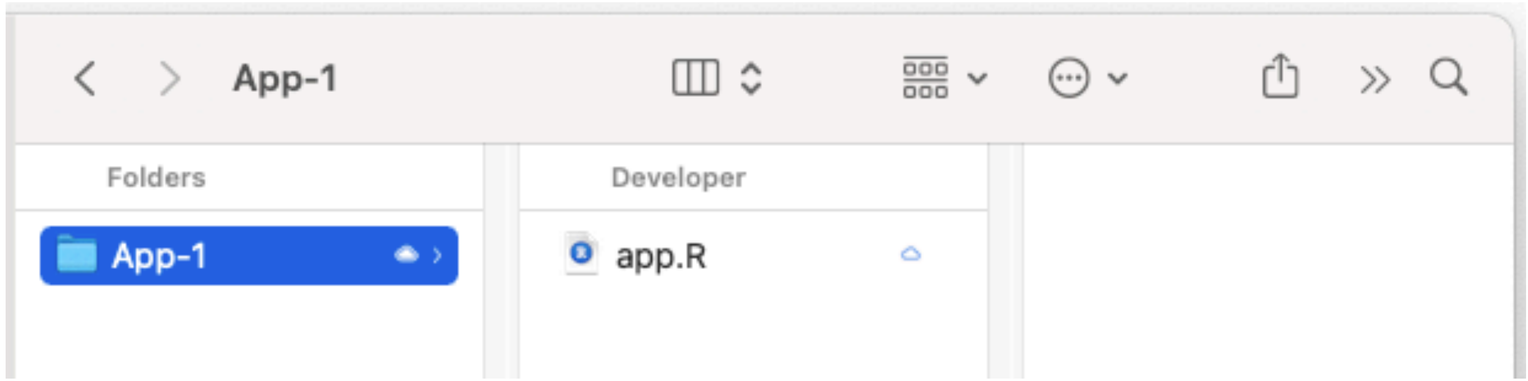


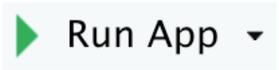
¡A PRACTICAR!

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  .

- 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")
```

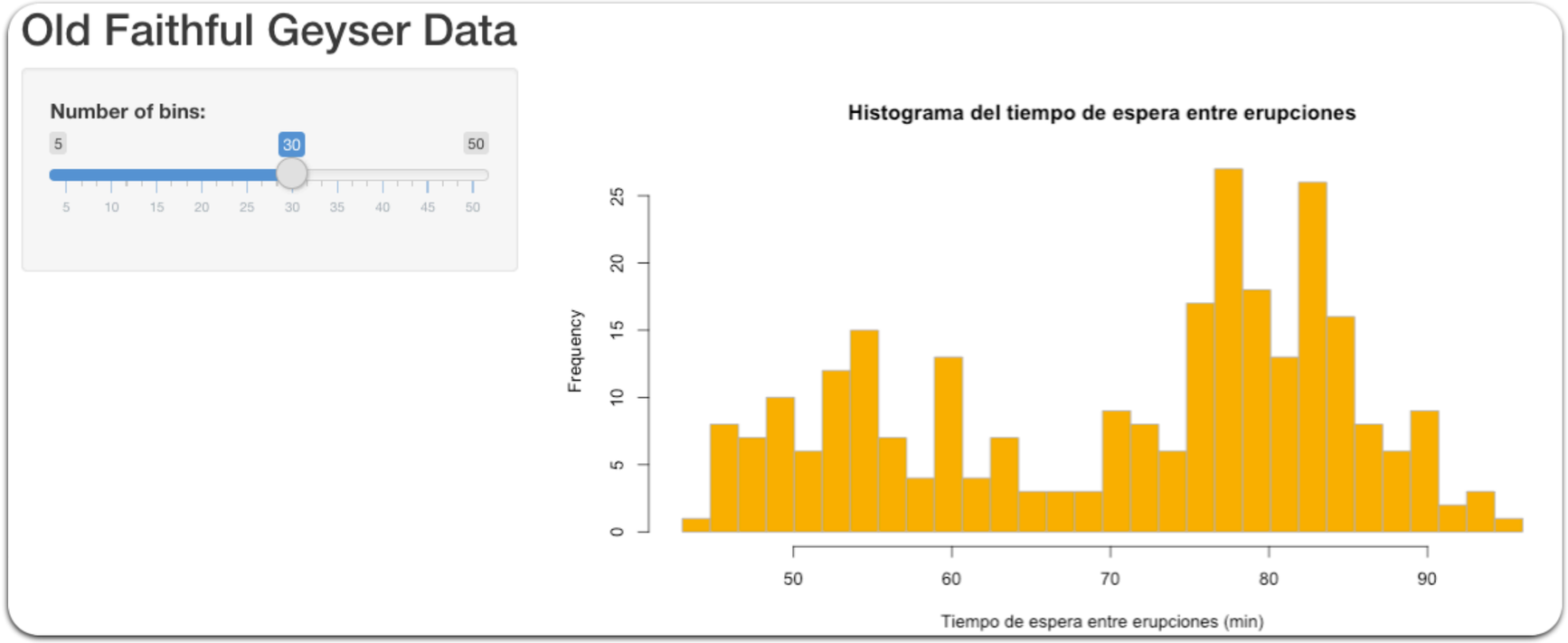
- Recuerda establecer el directorio de trabajo primero!

¡A PRACTICAR!

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.



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

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

Introducción
Estructura de una app
Widgets
Outputs
Reactividad
Layout
Publicación

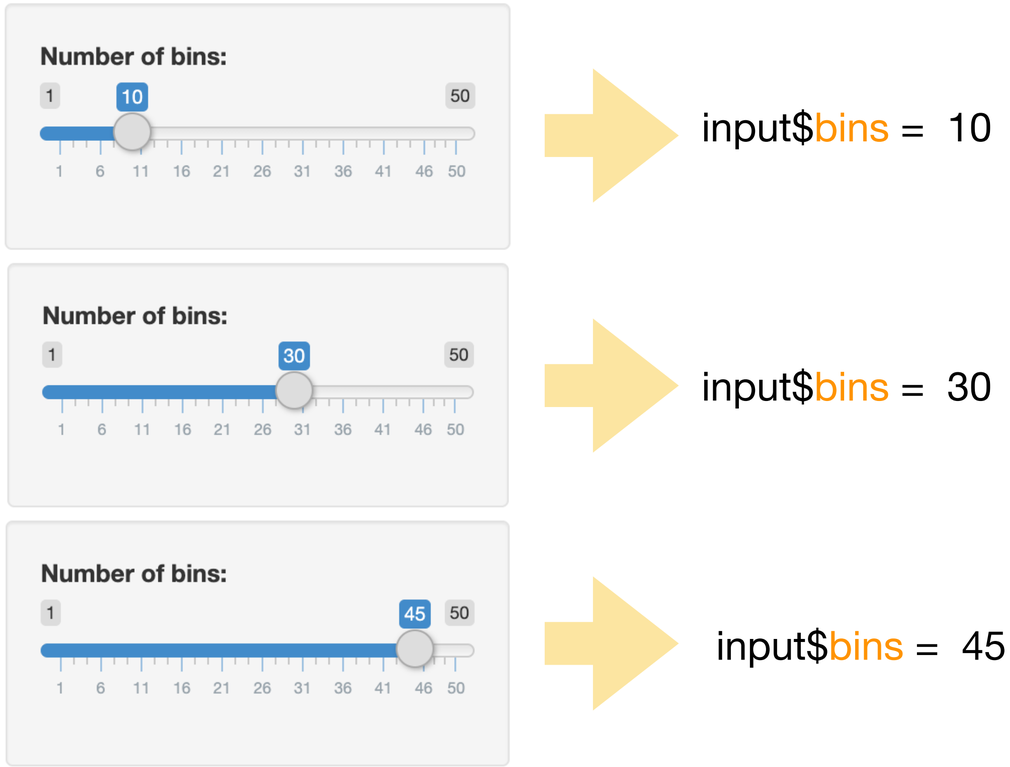
Widgets

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



Widgets

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

Introducción
Estructura de una app
Widgets
Outputs
Reactividad
Layout
Publicación

Existen distintos tipos de widgets: https://shiny.rstudio.com/gallery/widget-gallery.html

Action — **actionButton**(inputId, label, icon, …)

Link — **actionLink**(inputId, label, icon, …)

Choice 1 / Choice 2 / Choice 3 — **checkboxGroupInput**(inputId, label, choices, selected, inline)

Check me — **checkboxInput**(inputId, label, value)

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

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

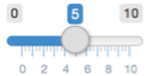
Choose File — **fileInput**(inputId, label, multiple, accept)

1 — **numericInput**(inputId, label, value, min, max, step)

••••••• — **passwordInput**(inputId, label, value)

Choice A / Choice B / Choice C — **radioButtons**(inputId, label, choices, selected, inline)

Choice 1 / Choice 1 / Choice 2 — **selectInput**(inputId, label, choices, selected, multiple, selectize, width, size) (also **selectizeInput()**)



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

Apply Changes — **submitButton**(text, icon)
(Prevents reactions across entire app)

Enter text — **textInput**(inputId, label, value)

Widgets

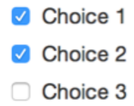
Ana D. Maldonado
ana.d.maldonado@ual.es

Introducción
Estructura de una app
Widgets
Outputs
Reactividad
Layout
Publicación

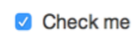
Existen distintos tipos de widgets: https://shiny.rstudio.com/gallery/widget-gallery.html

actionButton(inputId, label, icon, …)

actionLink(inputId, label, icon, …)



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



checkboxInput(inputId, label, value)

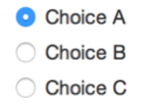
dateInput(inputId, label, value, min, max, format, startview, weekstart, language)

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

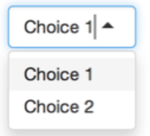
fileInput(inputId, label, multiple, accept)

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

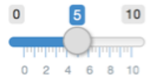
passwordInput(inputId, label, value)



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

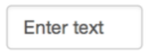


selectInput(inputId, label, choices, selected, multiple, selectize, width, size) (also **selectizeInput()**)



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

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



textInput(inputId, label, value)

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

Widgets

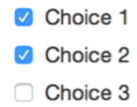
Ana D. Maldonado
ana.d.maldonado@ual.es

Introducción
Estructura de una app
Widgets
Outputs
Reactividad
Layout
Publicación

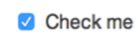
Existen distintos tipos de widgets: https://shiny.rstudio.com/gallery/widget-gallery.html

actionButton(inputId, label, icon, …)

actionLink(inputId, label, icon, …)



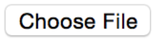
checkboxGroupInput(inputId, label, choices, selected, inline)



checkboxInput(inputId, label, value)

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

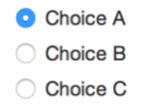
dateRangeInput(inputId, label, start, end, min, max, format, startview, weekstart, language, separator)



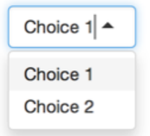
fileInput(inputId, label, multiple, accept)

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

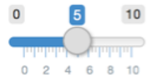
passwordInput(inputId, label, value)



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

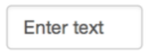


selectInput(inputId, label, choices, selected, multiple, selectize, width, size) (also **selectizeInput()**)



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

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



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

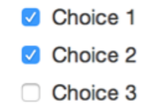
Ana D. Maldonado
ana.d.maldonado@ual.es

Introducción
Estructura de una app
Widgets
Outputs
Reactividad
Layout
Publicación

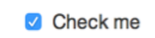
Existen distintos tipos de widgets: https://shiny.rstudio.com/gallery/widget-gallery.html

actionButton(inputId, label, icon, …)

actionLink(inputId, label, icon, …)



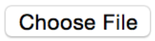
checkboxGroupInput(inputId, label, choices, selected, inline)



checkboxInput(inputId, label, value)

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

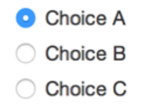
dateRangeInput(inputId, label, start, end, min, max, format, startview, weekstart, language, separator)



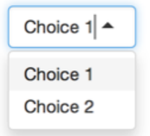
fileInput(inputId, label, multiple, accept)

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

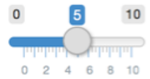
passwordInput(inputId, label, value)



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

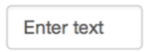


selectInput(inputId, label, choices, selected, multiple, selectize, width, size) (also **selectizeInput()**)



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

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



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 `""`.

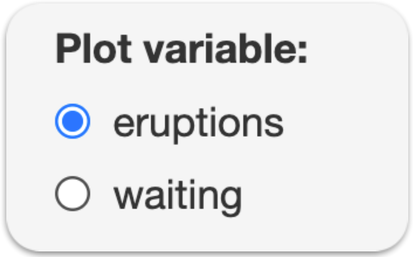
El resto de argumentos varía, según el widget: choices, value, min, max, …

¡A PRACTICAR!

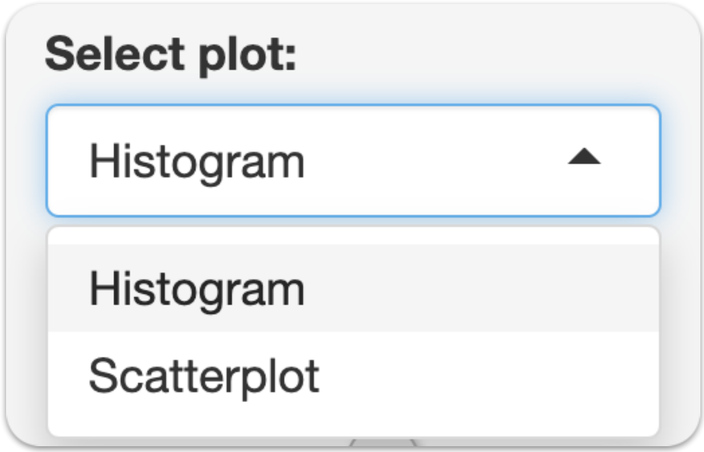
Ana D. Maldonado
ana.d.maldonado@ual.es

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.



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



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

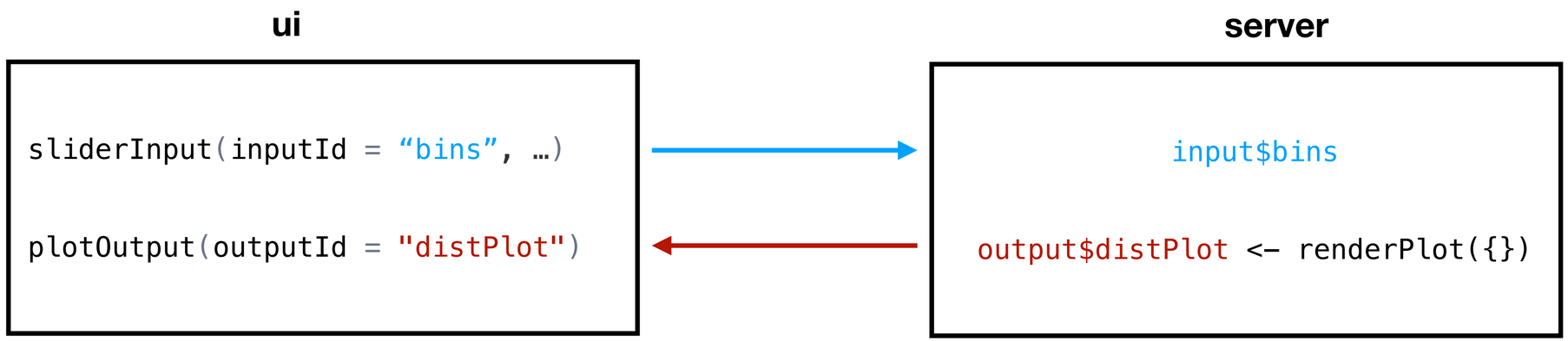
Ana D. Maldonado
ana.d.maldonado@ual.es

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.

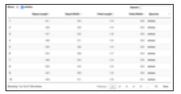
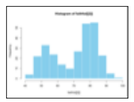
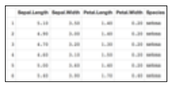


Outputs

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)	**imageOutput**(outputId, width, height, click, dblclick, hover, hoverDelay, hoverDelayType, brush, clickId, hoverId, inline)	Imagen	https://shiny.rstudio.com/gallery/image-output.html
	**renderPlot**(expr, width, height, res, …, env, quoted, func)	**plotOutput**(outputId, width, height, click, dblclick, hover, hoverDelay, hoverDelayType, brush, clickId, hoverId, inline)	Gráfico	
	**renderPrint**(expr, env, quoted, func, width)	**verbatimTextOutput**(outputId)	Salida de código	
	**renderTable**(expr,…, env, quoted, func)	**tableOutput**(outputId)	Tabla	
foo	**renderText**(expr, env, quoted, func)	**textOutput**(outputId, container, inline)	Texto	
	**renderUI**(expr, env, quoted, func)	**uiOutput**(outputId, inline, container, …) & **htmlOutput**(outputId, inline, container, …)	Elementos dinámicos del UI	https://shiny.rstudio.com/gallery/dynamic-ui.html
	server	ui		

Inputs y Outputs en server

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

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

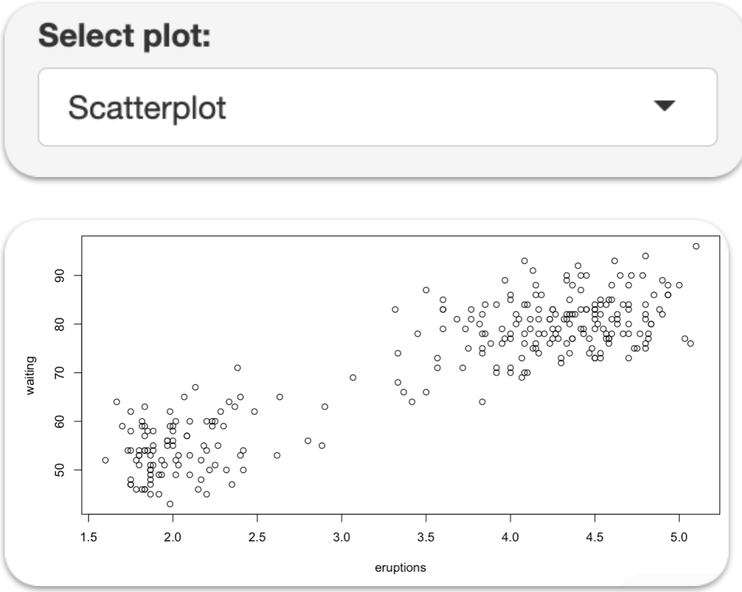
```
server <- function(input, output) {
  message("The greeting is ", output$greeting)
}

shinyApp(ui, server)
Error in `$.shinyoutput`(output, greeting) :
 Reading from shinyoutput object is not allowed.
```

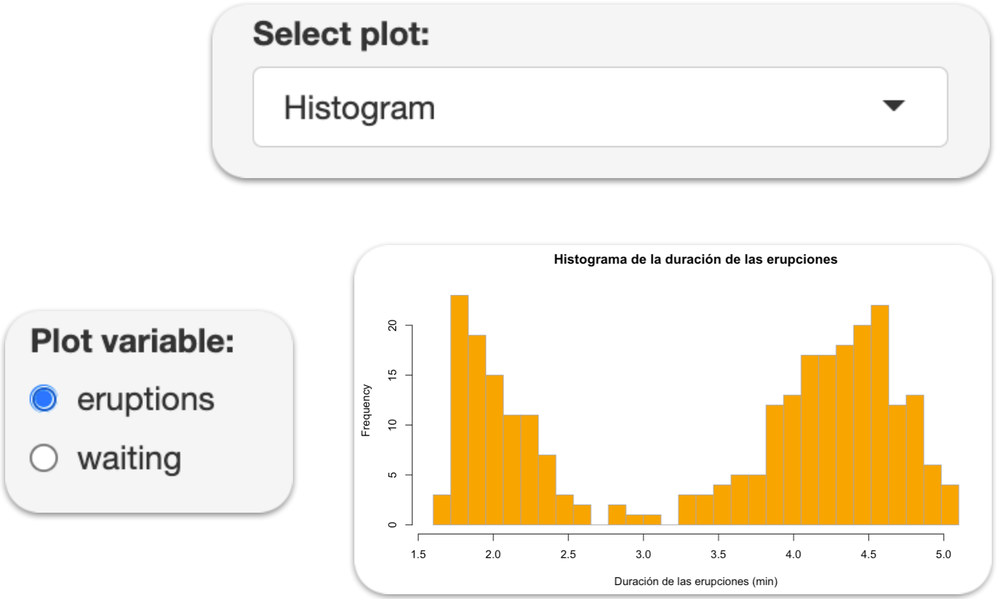
¡A PRACTICAR!

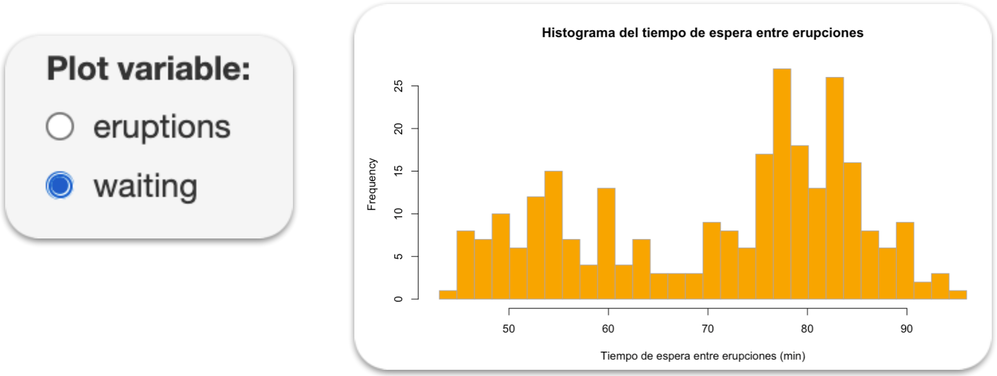
Ana D. Maldonado
ana.d.maldonado@ual.es

Introducción
Estructura de una app
Widgets
Outputs
Reactividad
Layout
Publicación

- Obtén la salida reactiva de los widgets añadidos anteriormente.





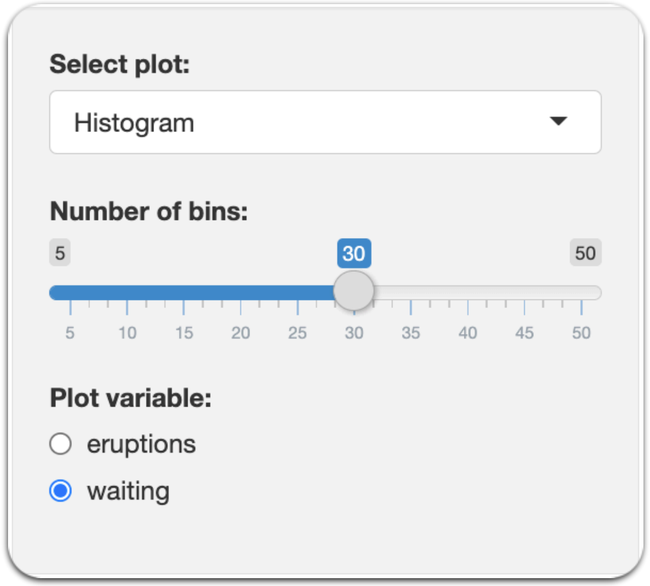


¡A PRACTICAR!

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.




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

     # Añade los widgets

  )
```

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

Introducción
Estructura de una app
Widgets
Outputs
Reactividad
Layout
Publicación

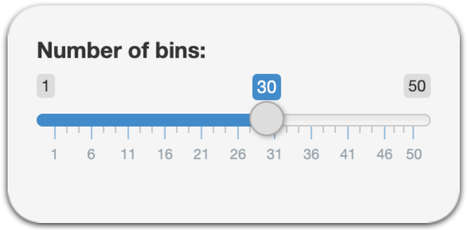
1. Introducción
2. Estructura de una aplicación Shiny
3. Widgets (inputs)
4. Salida reactiva (outputs)
5. Expresiones reactivas
6. Layout
7. Publicación

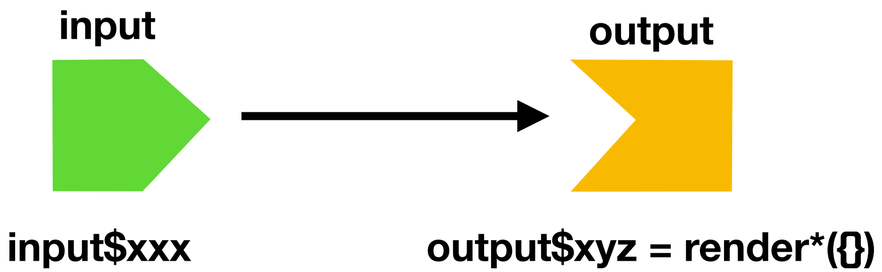
Ana D. Maldonado
ana.d.maldonado@ual.es

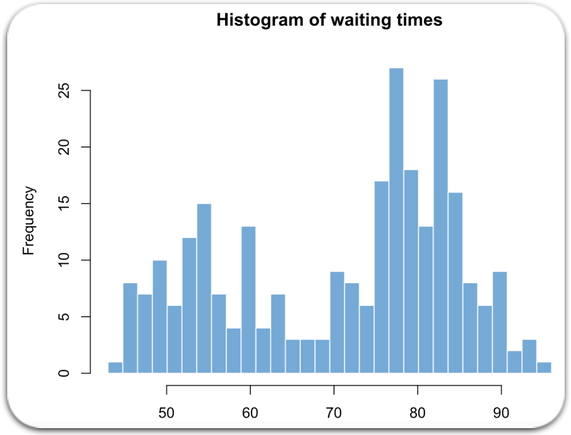
Introducción
Estructura de una app
Widgets
Outputs
Reactividad
Layout
Publicación

Expresiones reactivas

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:





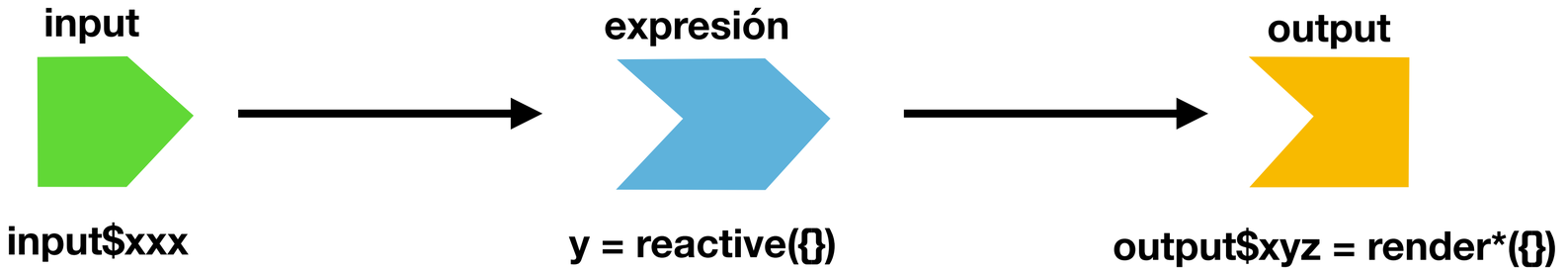


Expresiones reactivas

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 reactiva** en un **output**.

Expresiones reactivas

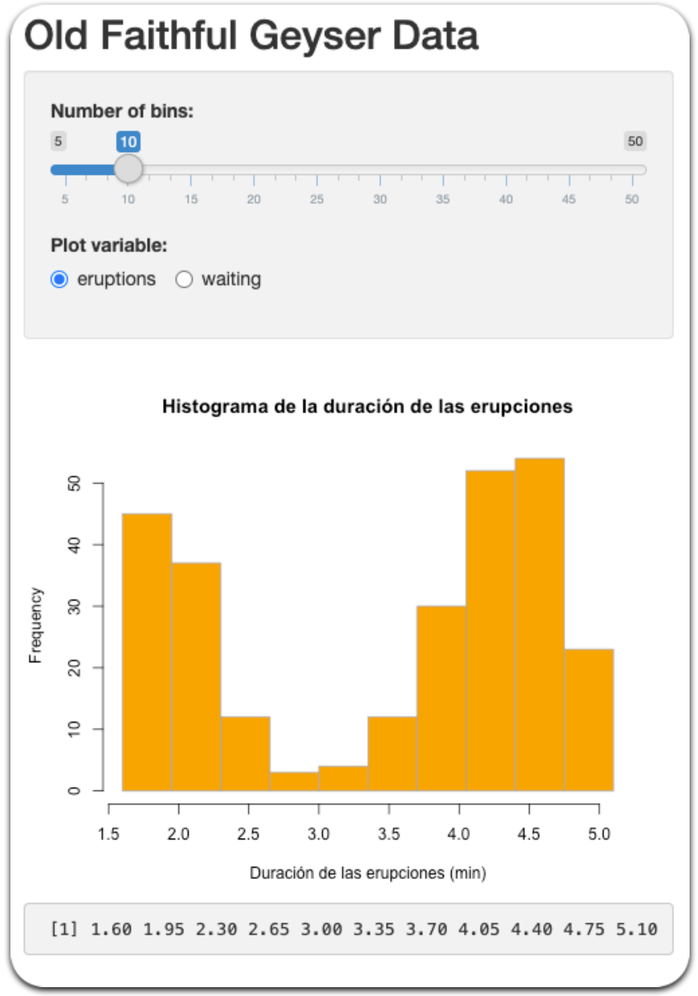
Ana D. Maldonado
ana.d.maldonado@ual.es

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!

Expresiones reactivas

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

Introducción
Estructura de una app
Widgets
Outputs
Reactividad
Layout
Publicación

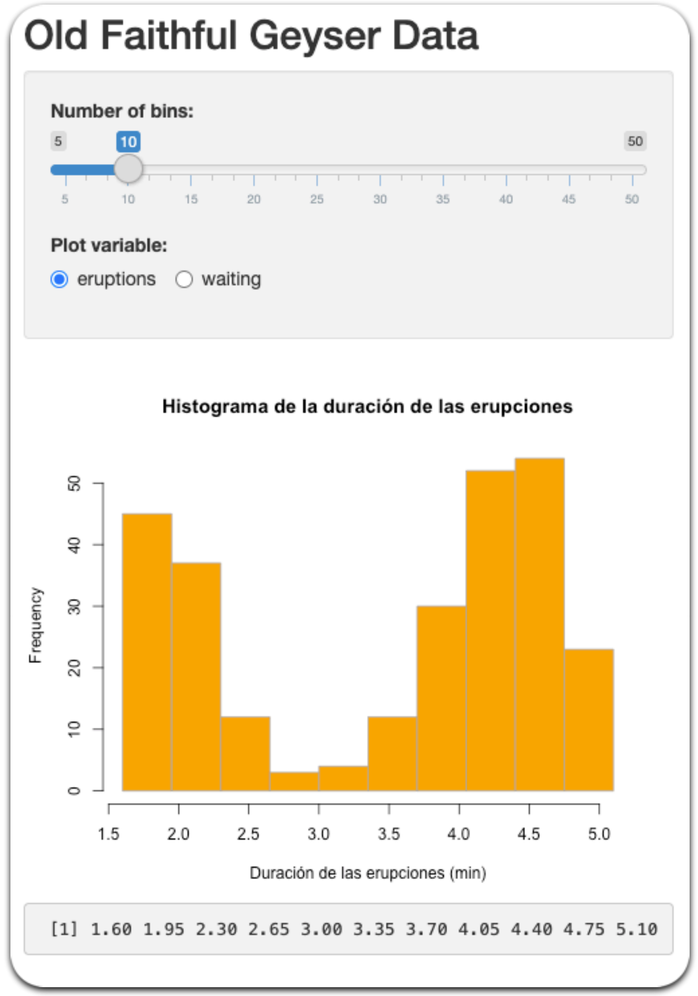
Ejemplos/ ejemplo01

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.

Expresiones reactivas

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

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!

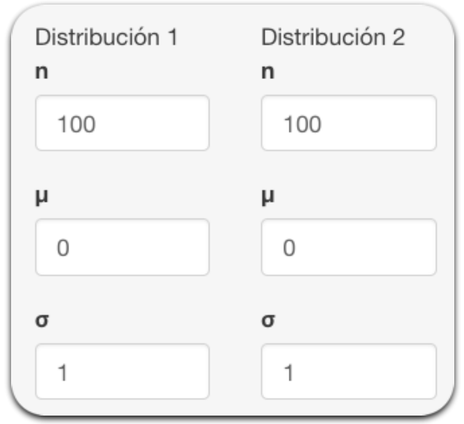
Ana D. Maldonado
ana.d.maldonado@ual.es

Introducción
Estructura de una app
Widgets
Outputs
Reactividad
Layout
Publicación

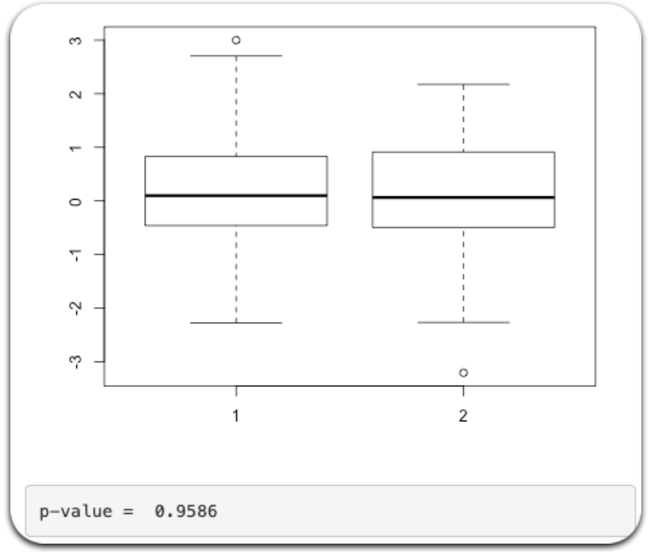
Escribe una aplicación (App-2) que simule 2 conjuntos de datos y los compare mediante un gráfico y un test de hipótesis.

- 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.




```r
# 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))
```



¡A PRACTICAR! (Solución incorrecta)

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))
    })
}
```

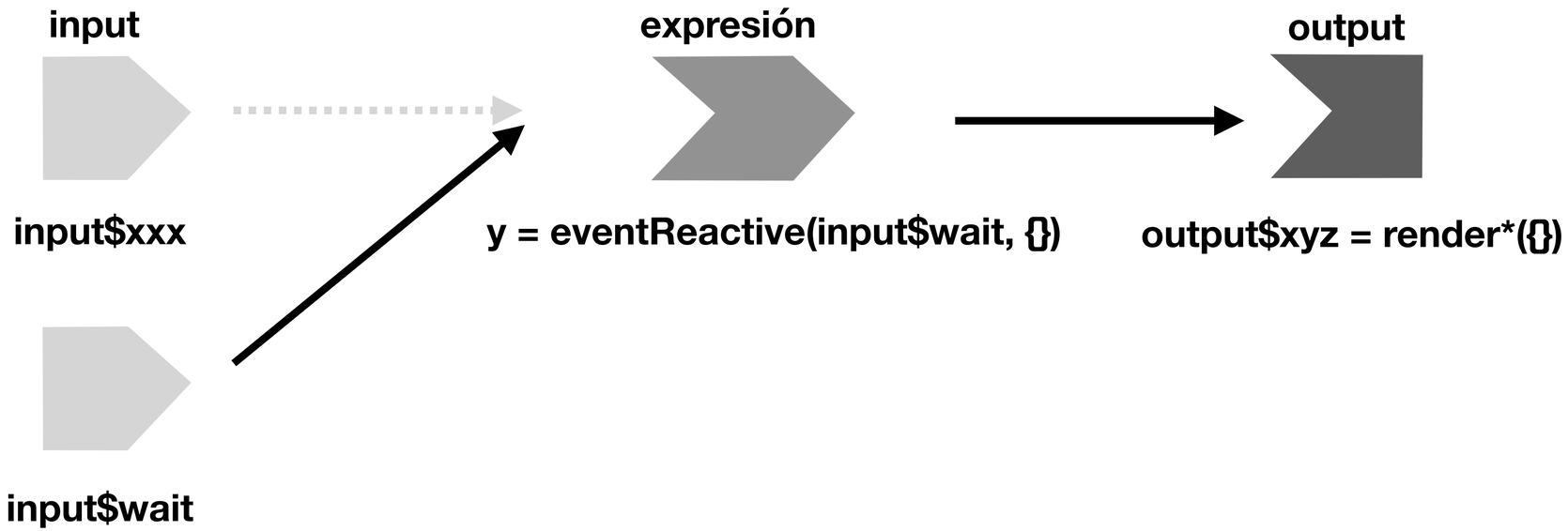
¿Cuál es el problema del server? ¿Cómo podríamos solucionarlo?

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

Introducción
Estructura de una app
Widgets
Outputs
Reactividad
Layout
Publicación

Retrasar reactividad

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.

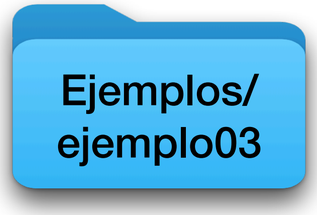
Retrasar reactividad

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)
```



¡A PRACTICAR!

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

Introducción
Estructura de una app
Widgets
Outputs
Reactividad
Layout
Publicación

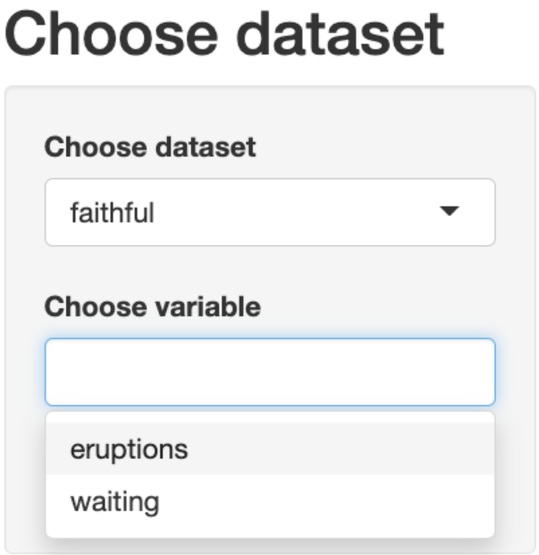
Añade un actionButton() a App-2 para que no se generen las muestras hasta que el usuario pulse el botón "Actualizar".

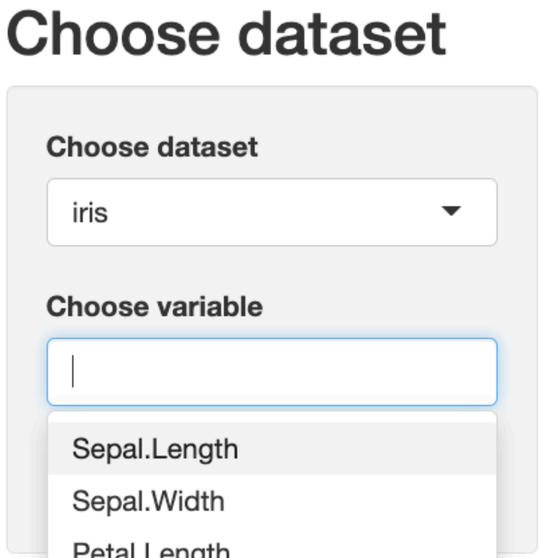
Actualizar opciones de widgets

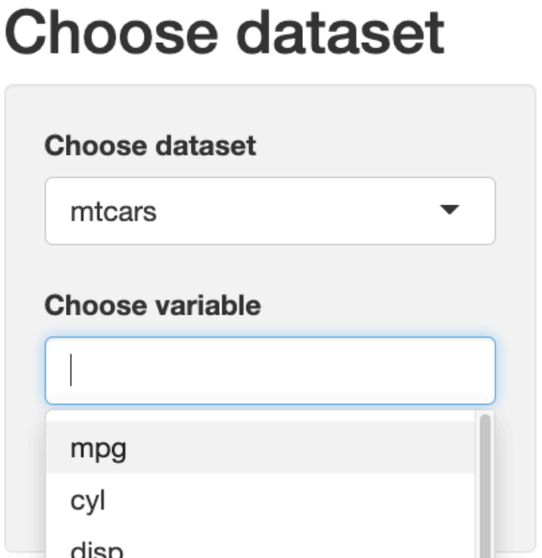
Ana D. Maldonado
ana.d.maldonado@ual.es

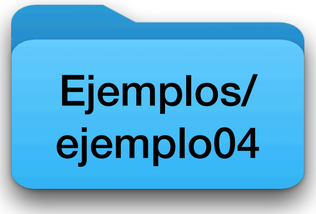
Introducción
Estructura de una app
Widgets
Outputs
Reactividad
Layout
Publicación

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.









Actualizar opciones de widgets

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

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

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

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

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

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

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

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

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

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$var) del conjunto de datos seleccionado (input$datastet)

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

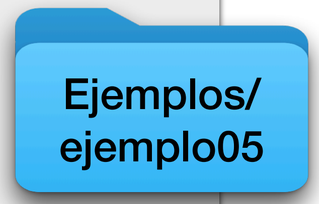
Ana D. Maldonado
ana.d.maldonado@ual.es

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/tabsets.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")
    )

)
```



Layout: tabsetPanel

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

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 bi
                 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

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 bi
               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

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

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")
    )

)
```

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

¡A PRACTICAR!

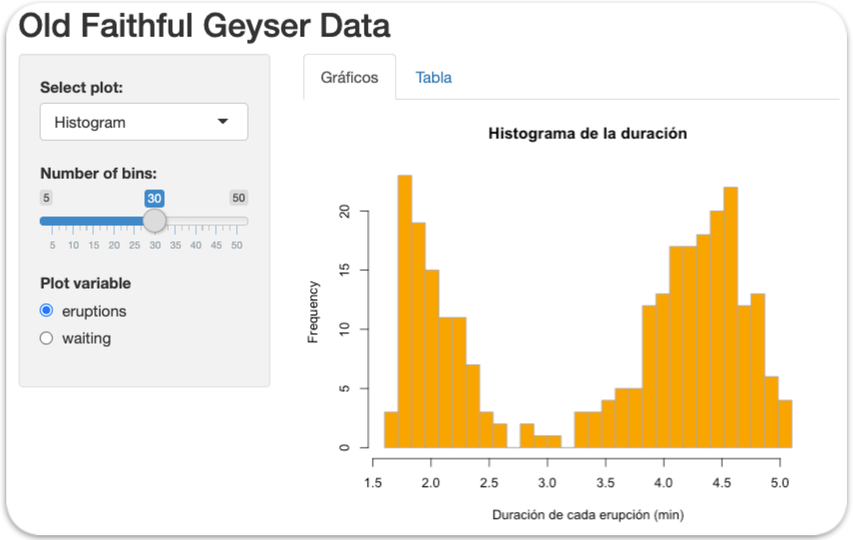
Ana D. Maldonado
ana.d.maldonado@ual.es

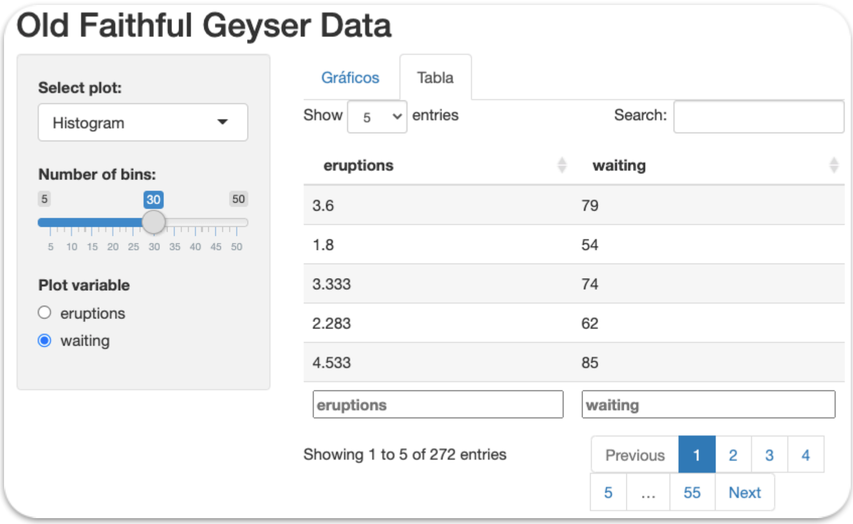
Introducción
Estructura de una app
Widgets
Outputs
Reactividad
Layout
Publicación

- 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.





Layout: navbarPage

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

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.

No es necesario usar tabsetPanel dentro de navbarPage.

```
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")
        )

)
```

Layout: navbarPage

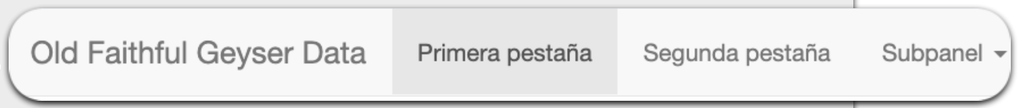
Ana D. Maldonado
ana.d.maldonado@ual.es

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")
    )

)
```

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



Layout: navbarPage

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")
    )

)
```

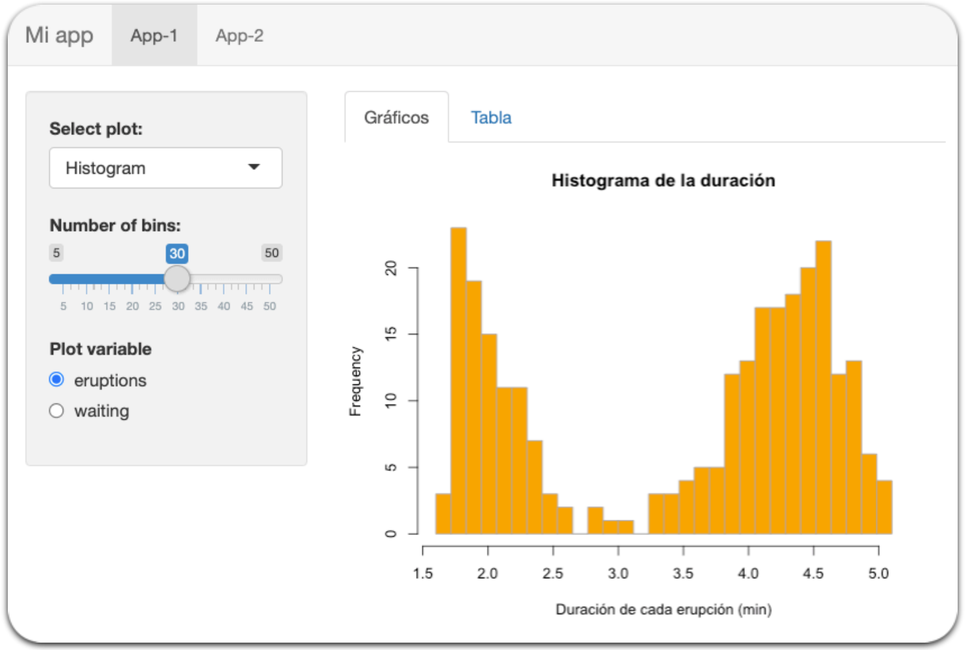
navbarMenu() crea una pestaña con un menú desplegable. Cada una de estas páginas se pueden llenar con sidebarLayout.

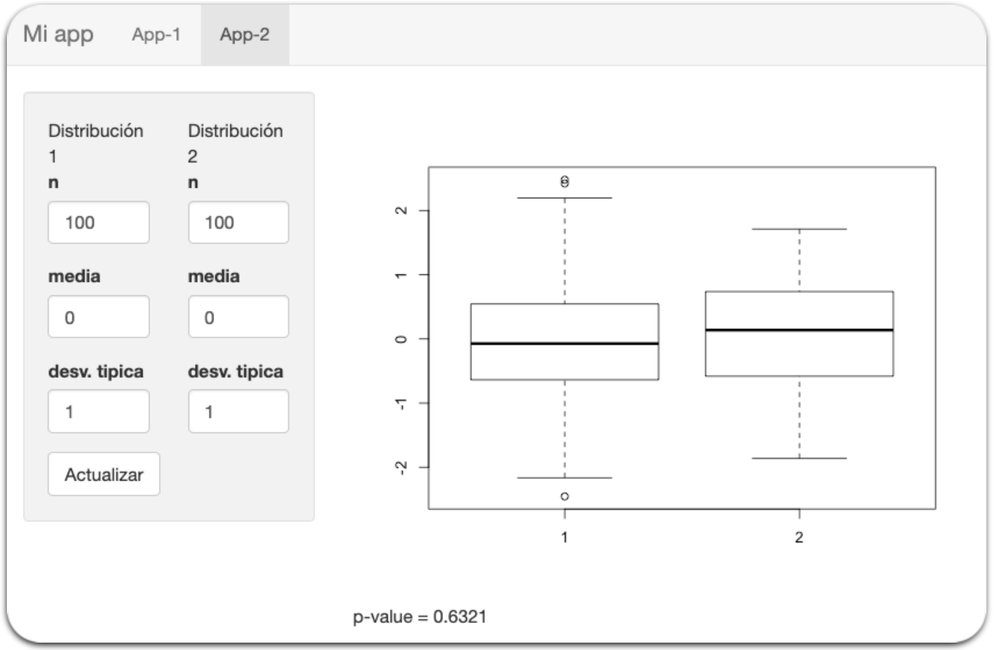
¡A PRACTICAR!

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.





Temas

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"),
    ...
)
```

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

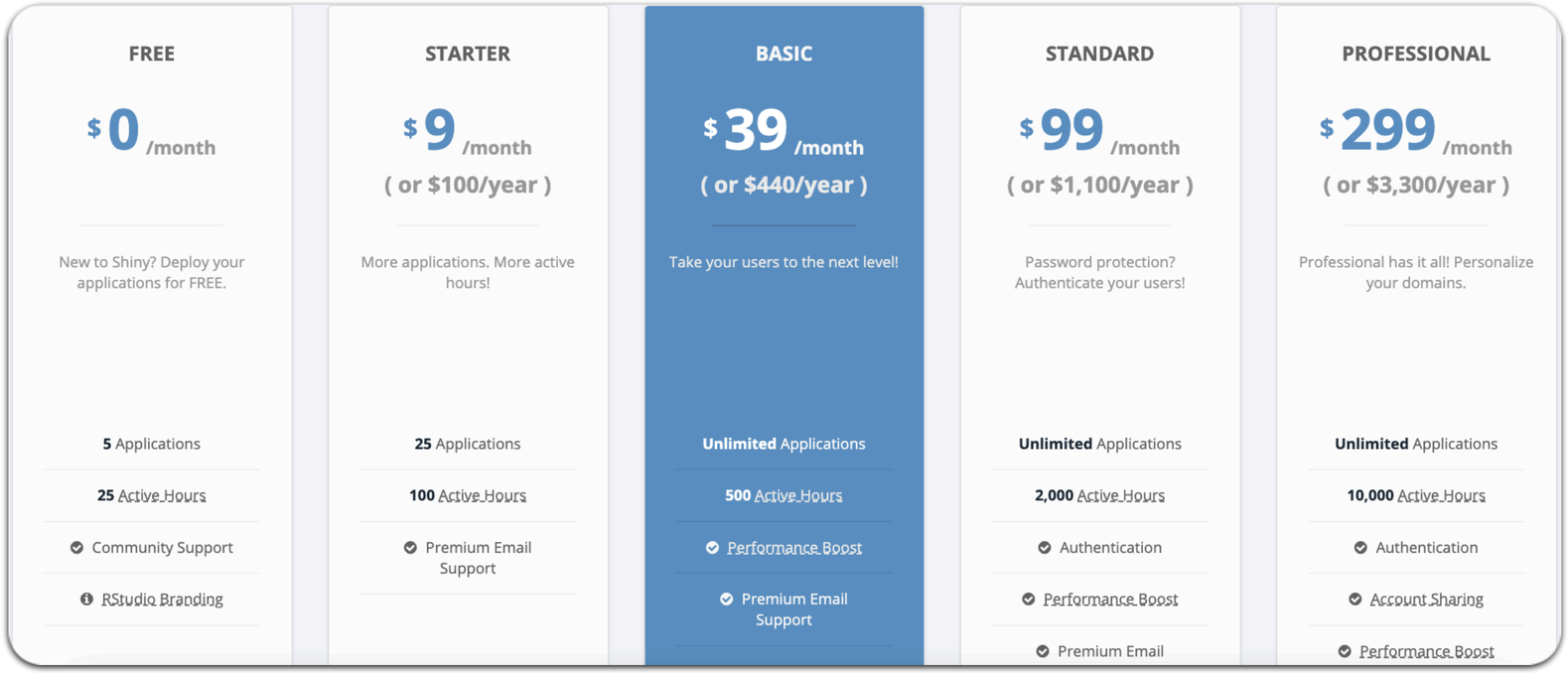
Publicación de Shiny apps

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
Community Support	Premium Email Support	Performance Boost	Authentication	Authentication
RStudio Branding		Premium Email Support	Performance Boost	Account Sharing
			Premium Email	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

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

Introducción
Estructura de una app
Widgets
Outputs
Reactividad
Layout
Publicación

Para usar shinyapps.io, es necesario crear una cuenta de usuario:

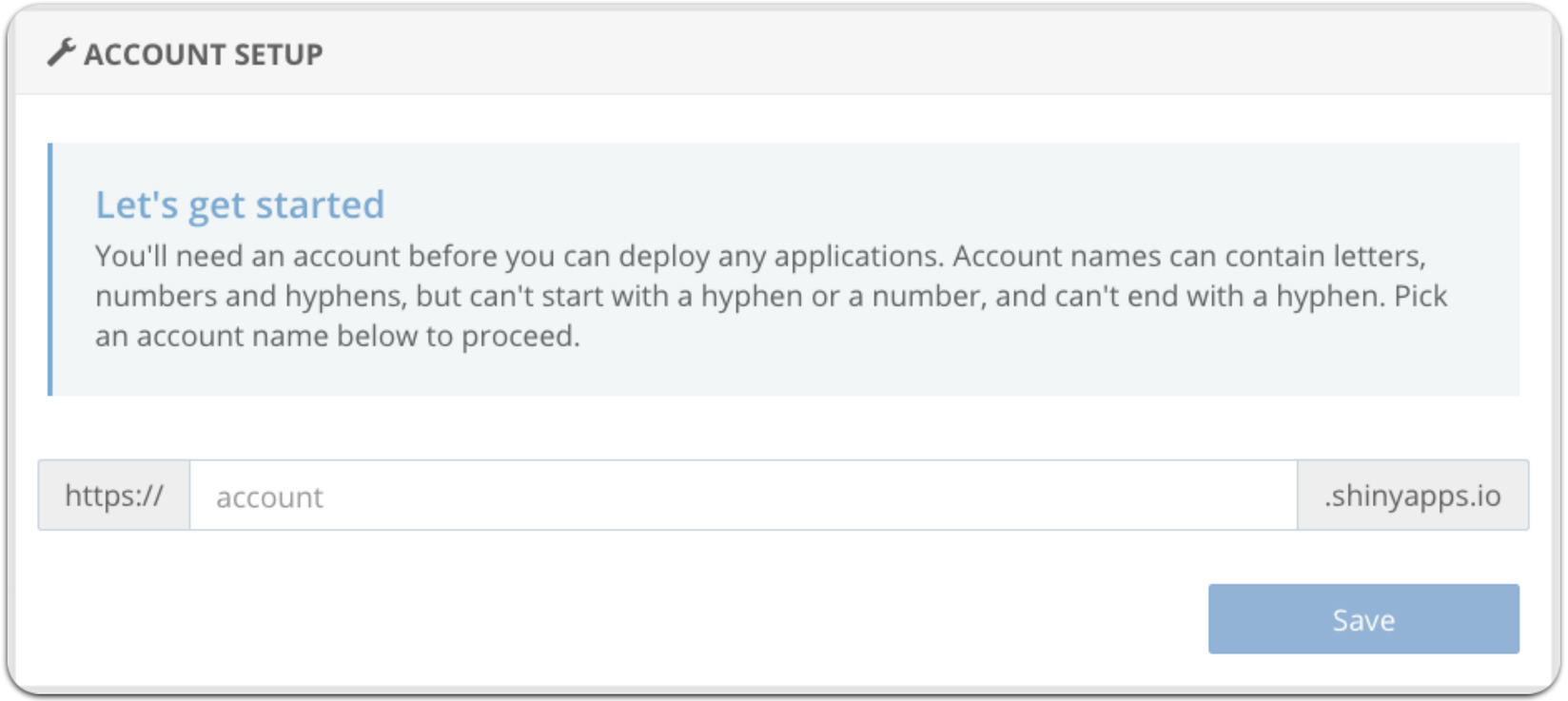
https://www.shinyapps.io/admin/#/signup

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

Introducción
Estructura de una app
Widgets
Outputs
Reactividad
Layout
Publicación

Publicación de Shiny apps

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

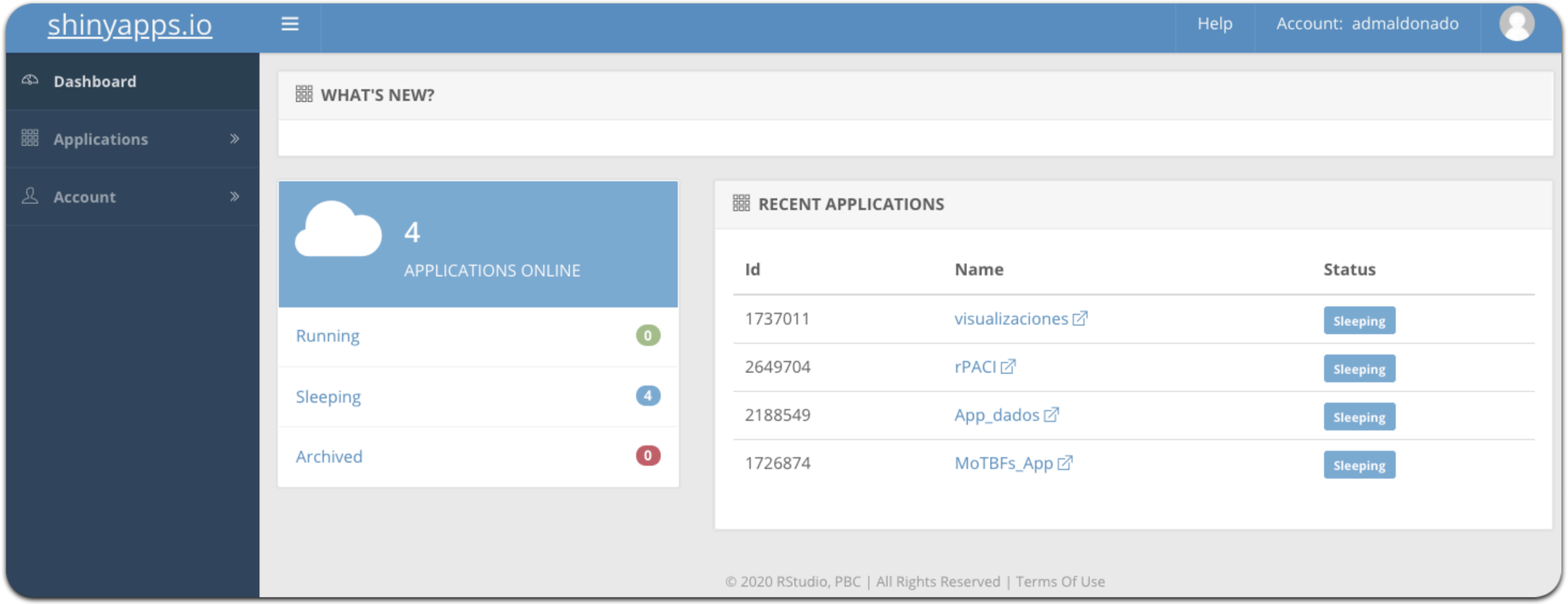


Publicación de Shiny apps

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

Introducción
Estructura de una app
Widgets
Outputs
Reactividad
Layout
Publicación

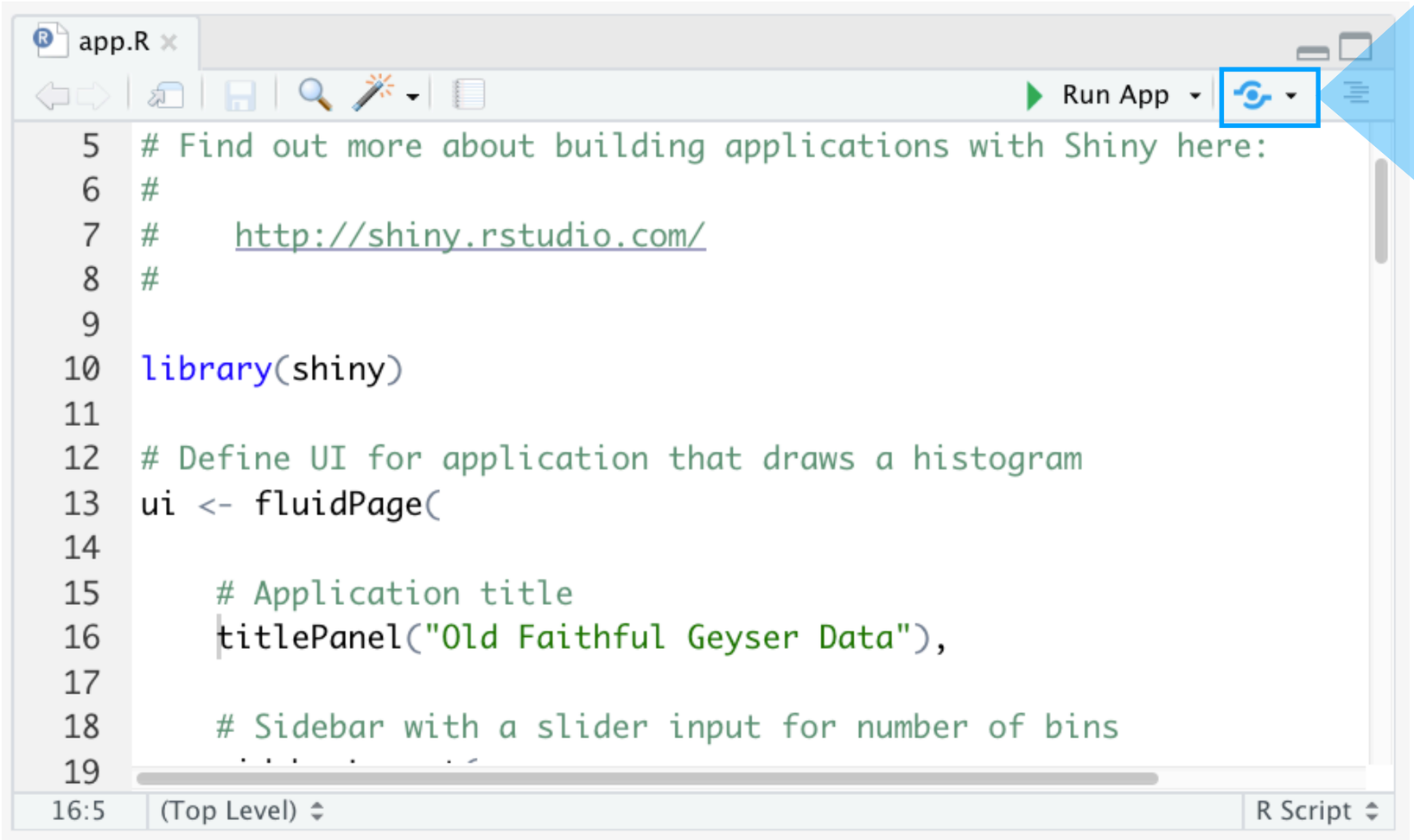
Si ya tienes una cuenta: https://www.shinyapps.io/admin/#/login

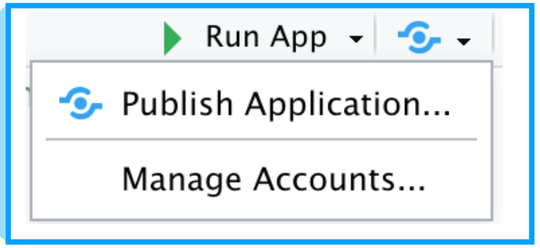


Publicación de Shiny apps

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

Introducción
Estructura de una app
Widgets
Outputs
Reactividad
Layout
Publicación



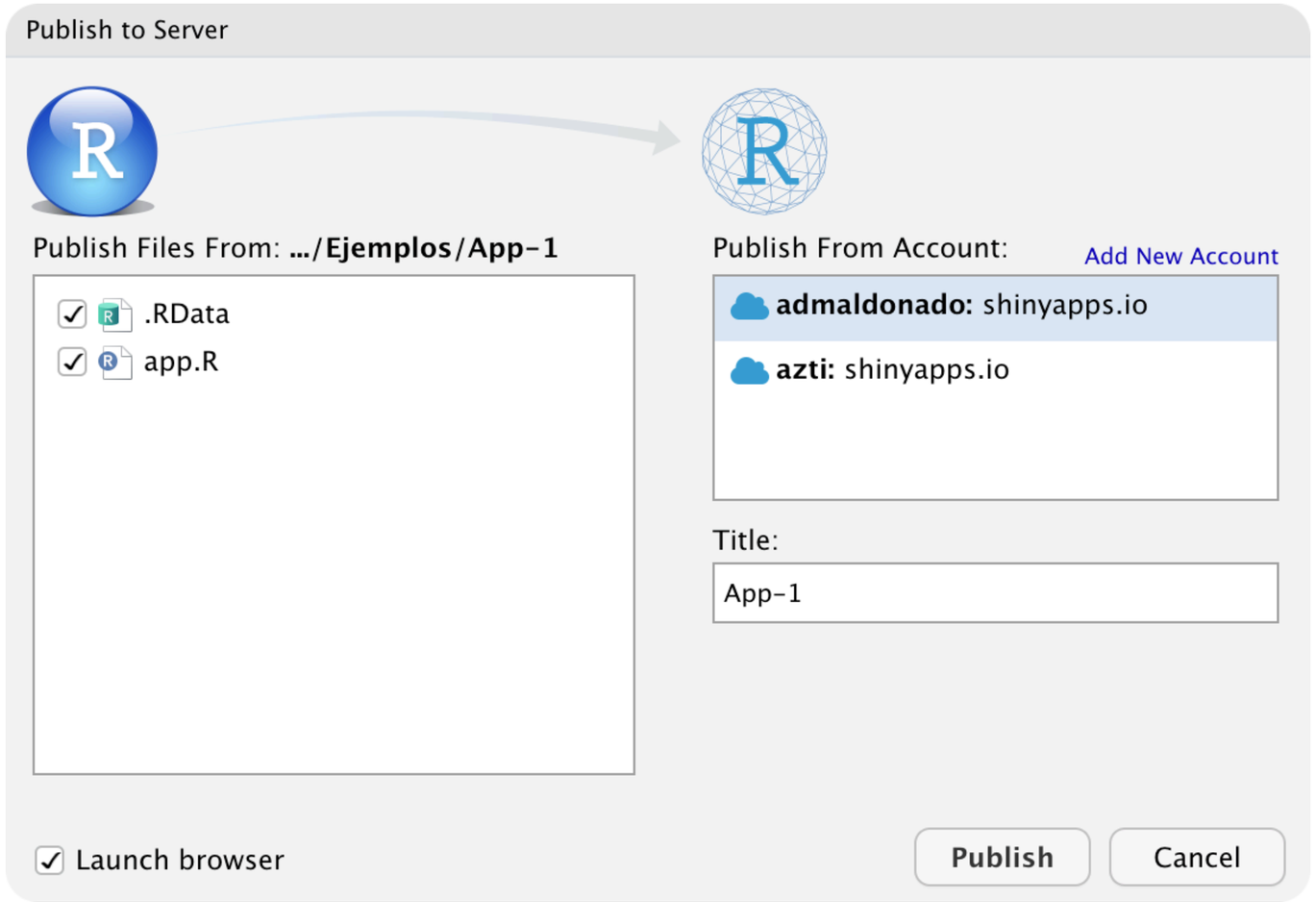


Ana D. Maldonado
ana.d.maldonado@ual.es

Introducción
Estructura de una app
Widgets
Outputs
Reactividad
Layout
Publicación

Publicación de Shiny apps

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

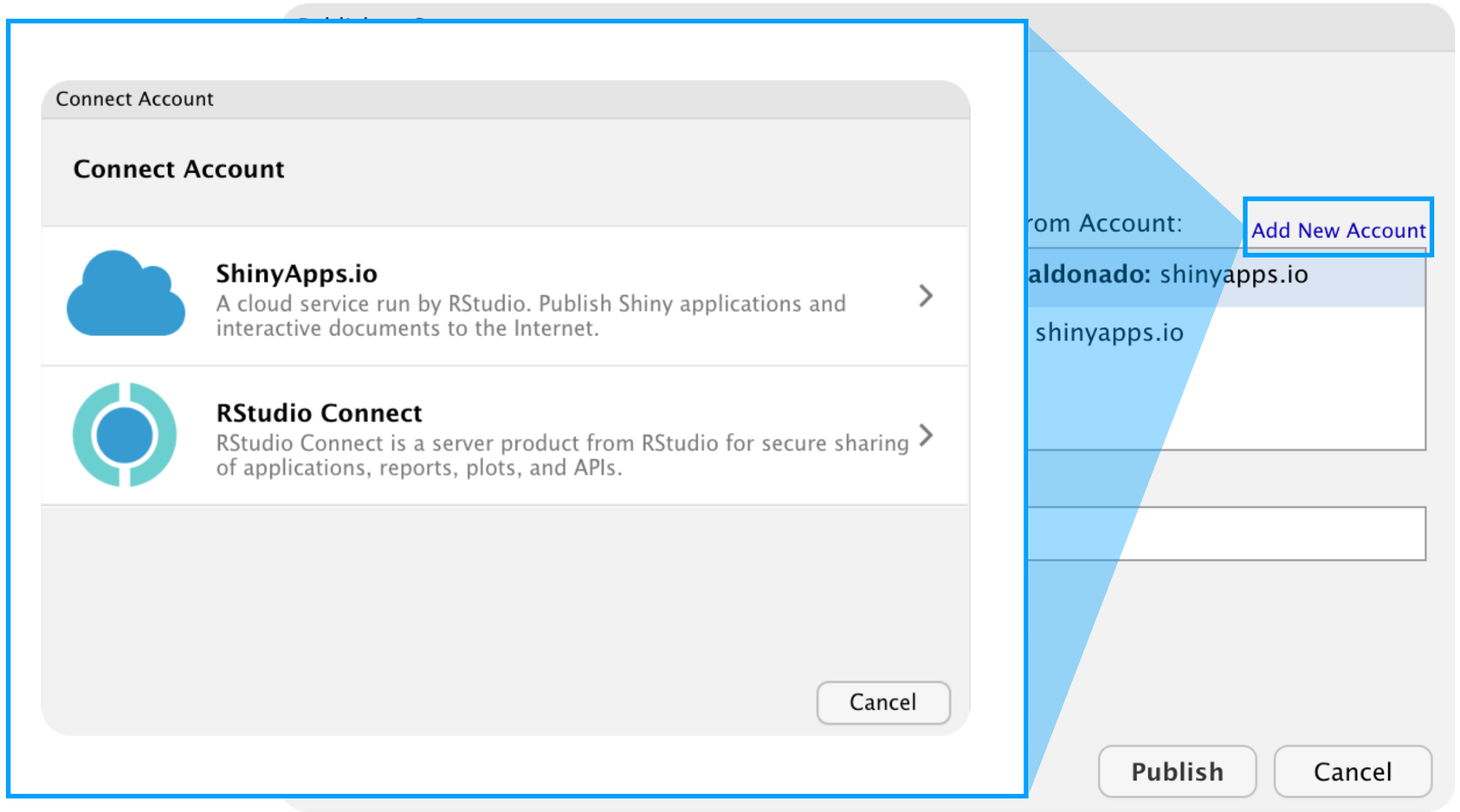


Publicación de Shiny apps

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.

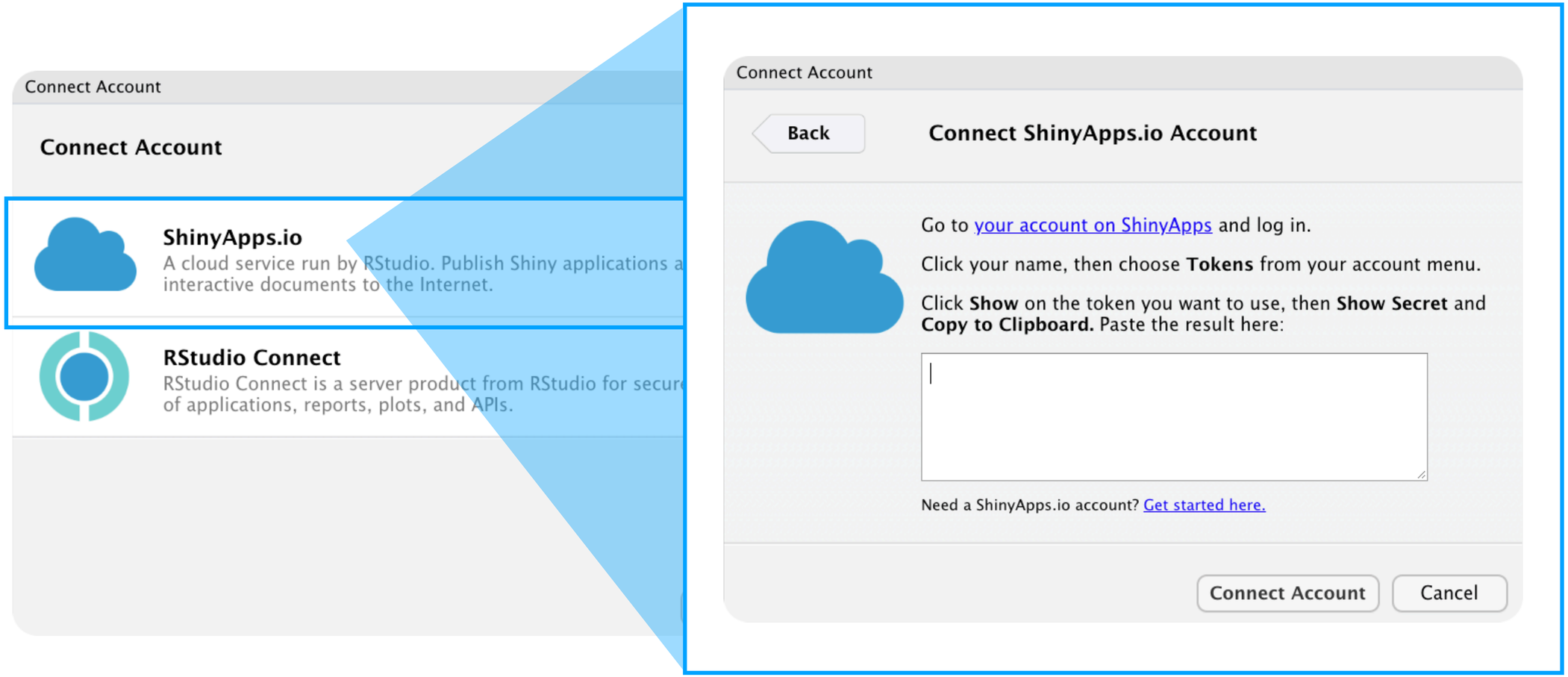


Publicación de Shiny apps

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.

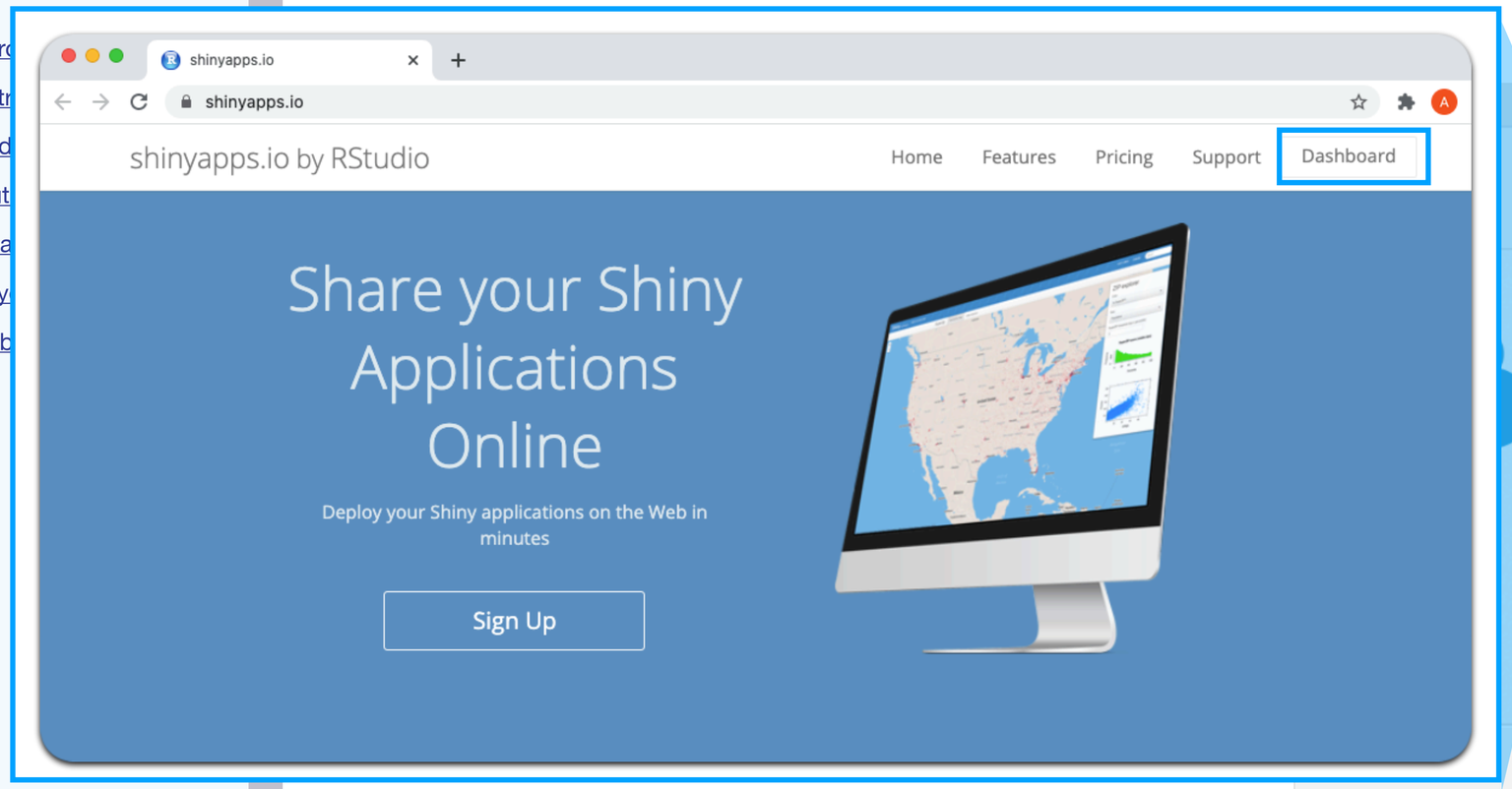


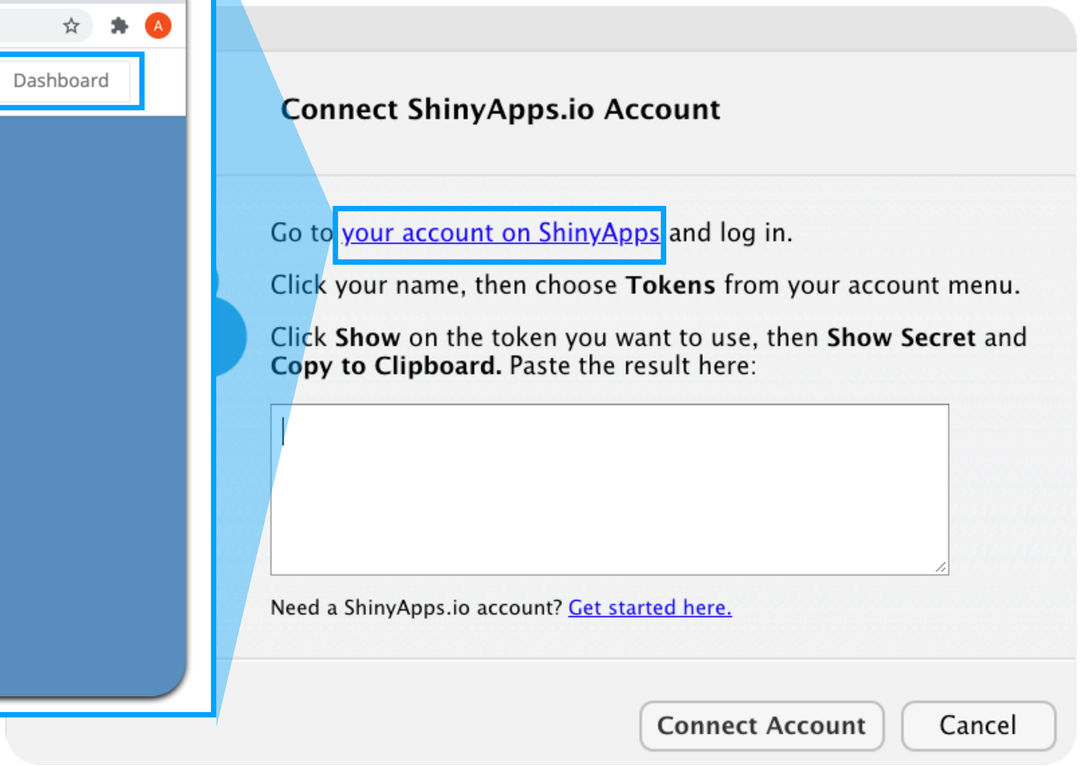
Publicación de Shiny apps

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

Intro
Estr
Wid
Out
Rea
Lay
Pub

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”.



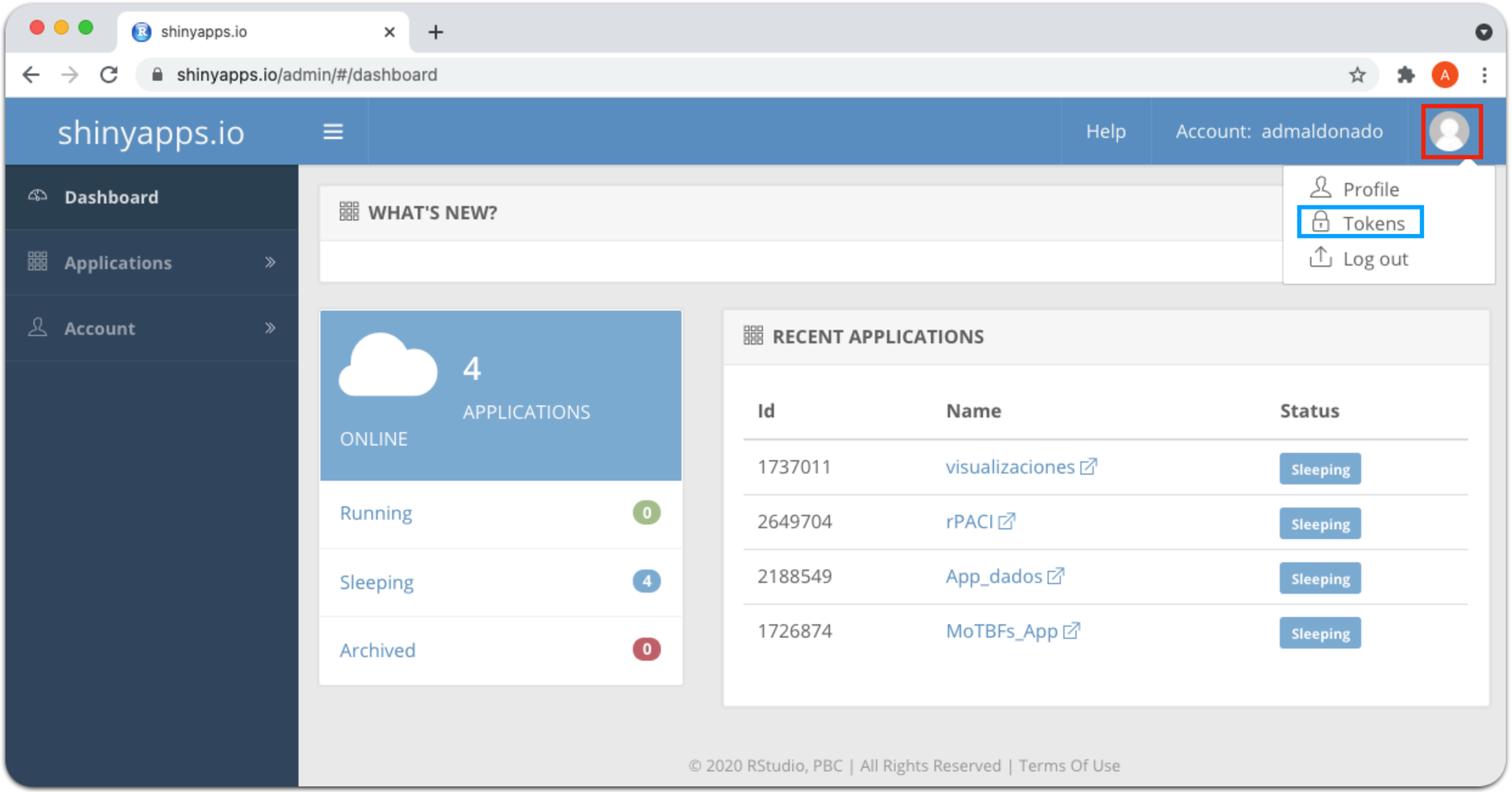


Publicación de Shiny apps

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”.

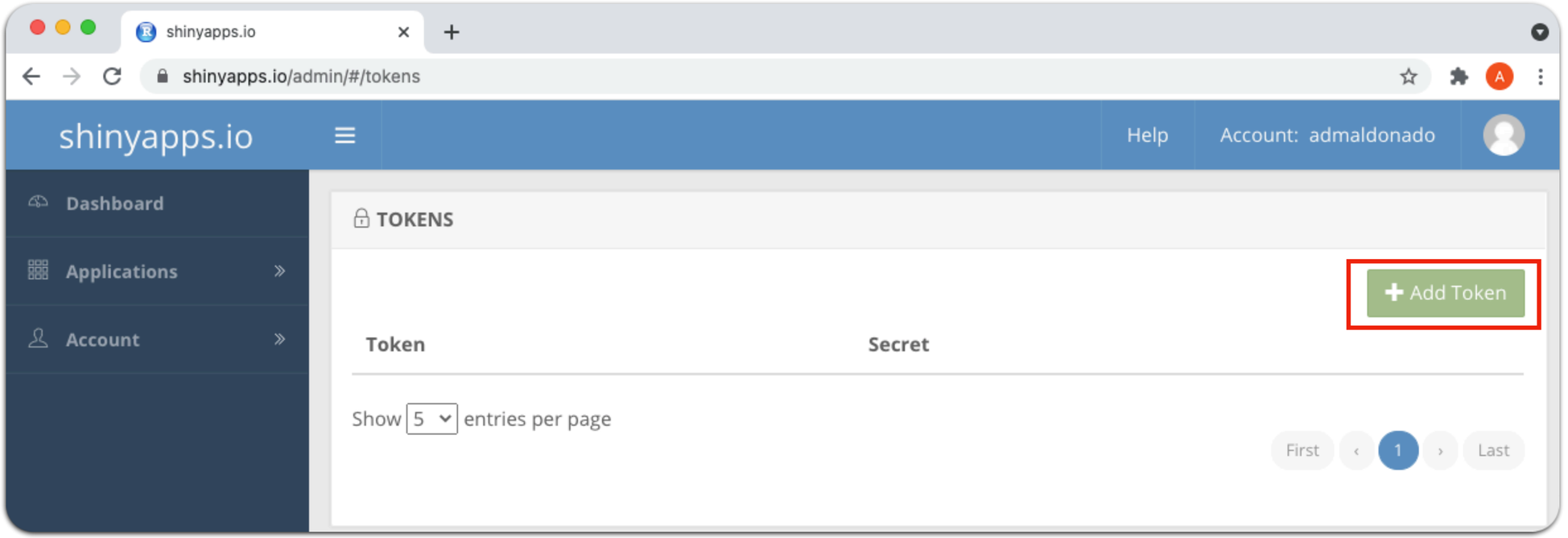


Publicación de Shiny apps

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”.

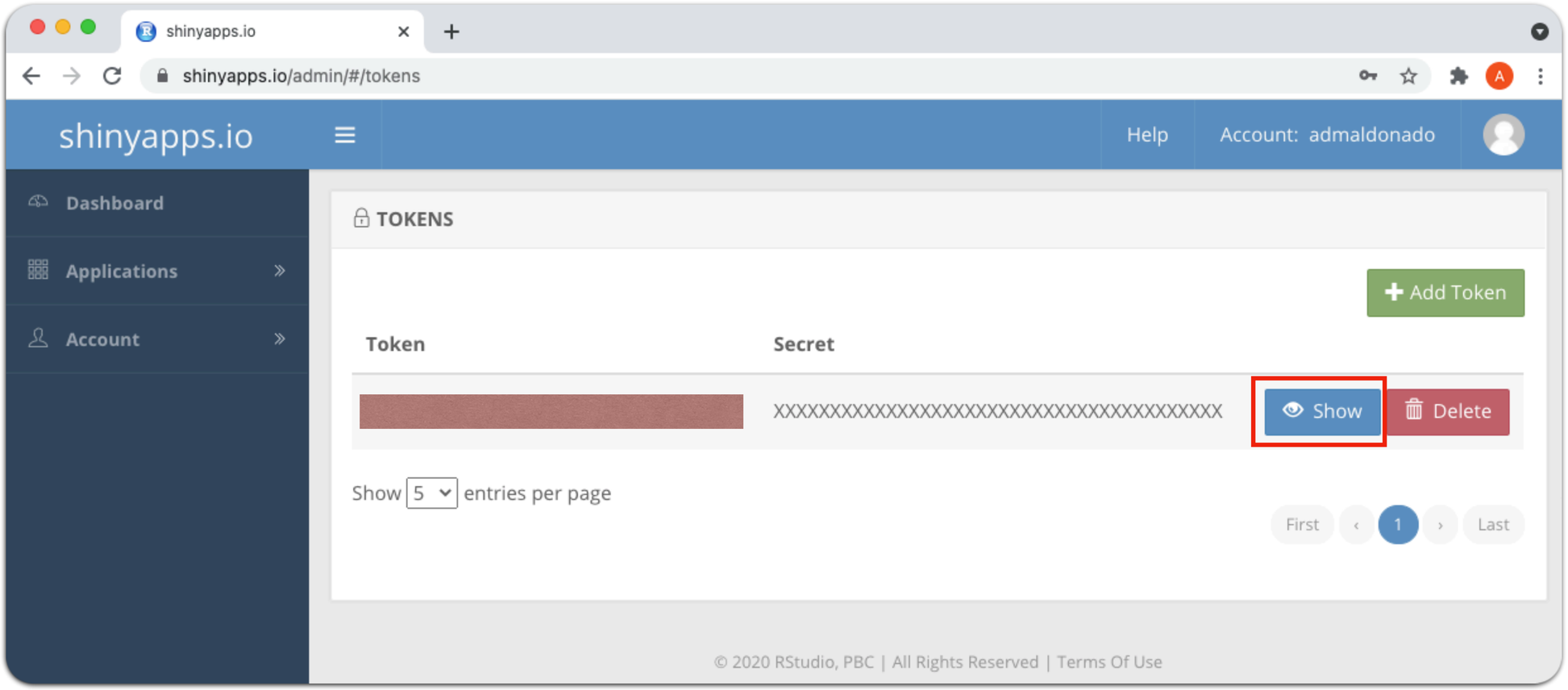


Ana D. Maldonado
ana.d.maldonado@ual.es

Introducción
Estructura de una app
Widgets
Outputs
Reactividad
Layout
Publicación

Publicación de Shiny apps

Aparecerá un nuevo token. Pulsa sobre “Show”

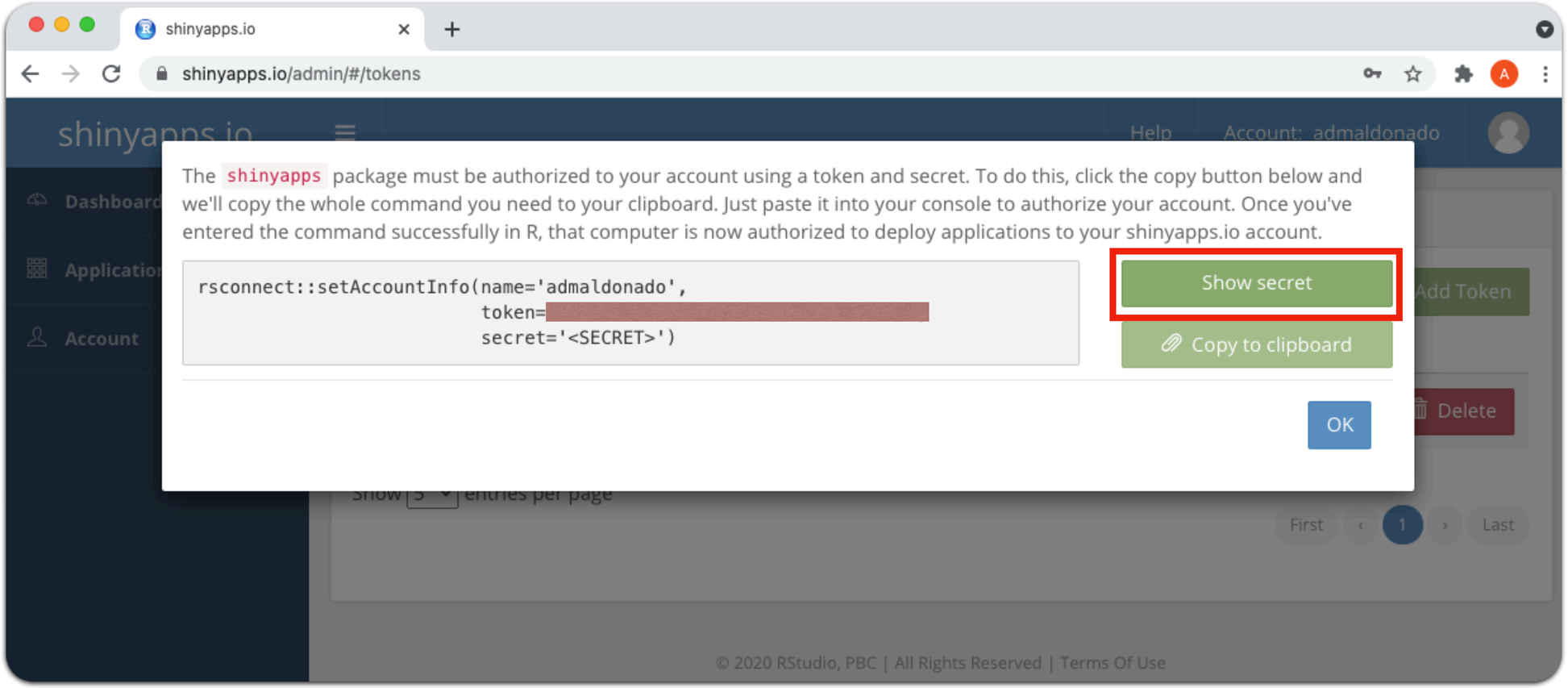


Publicación de Shiny apps

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”.

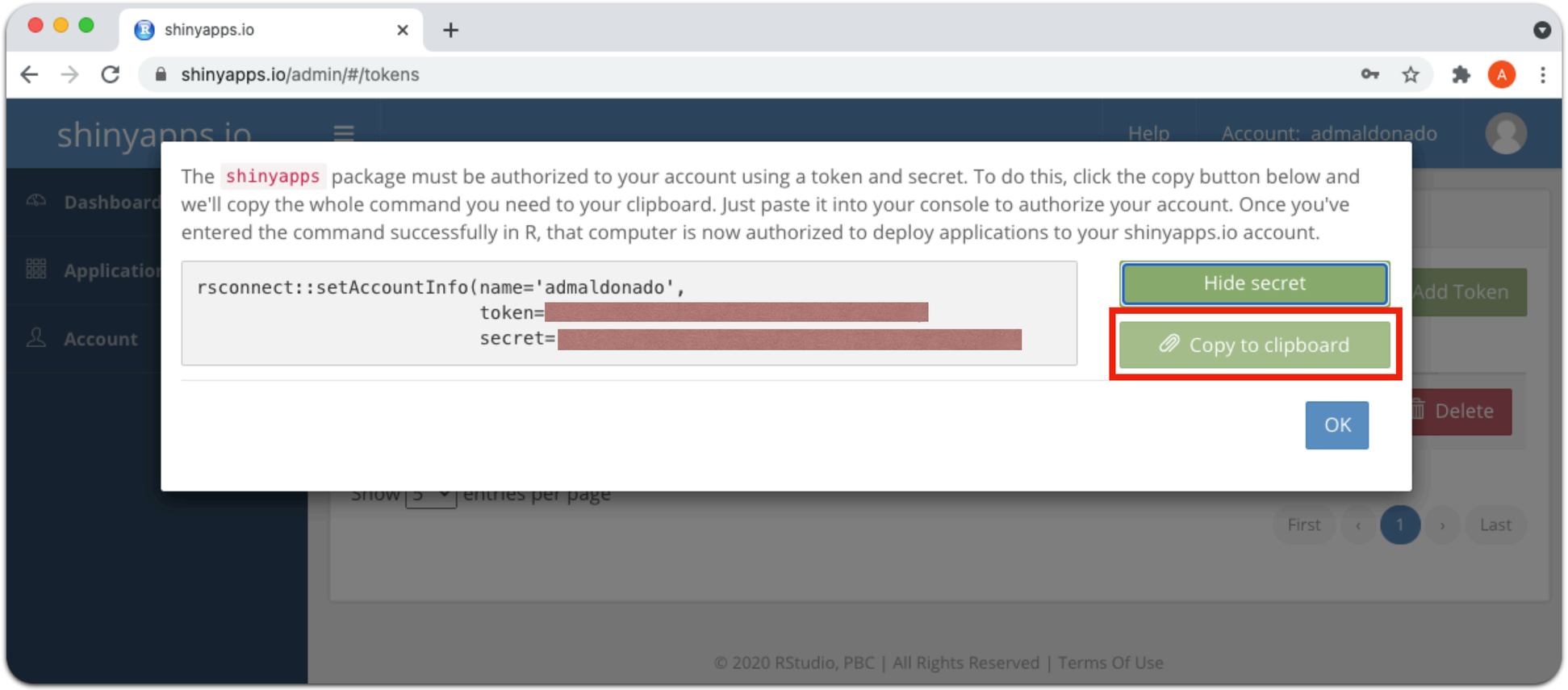


Publicación de Shiny apps

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”.

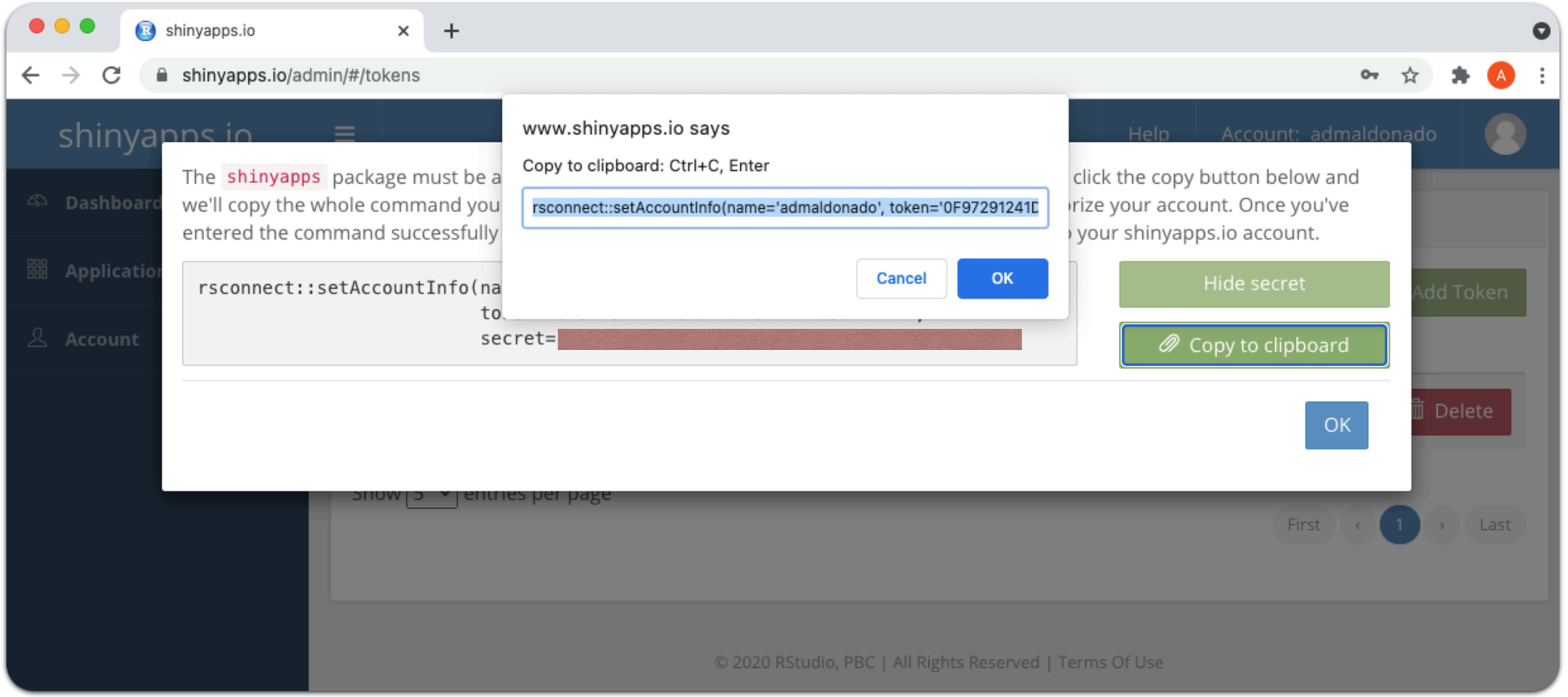


Publicación de Shiny apps

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.

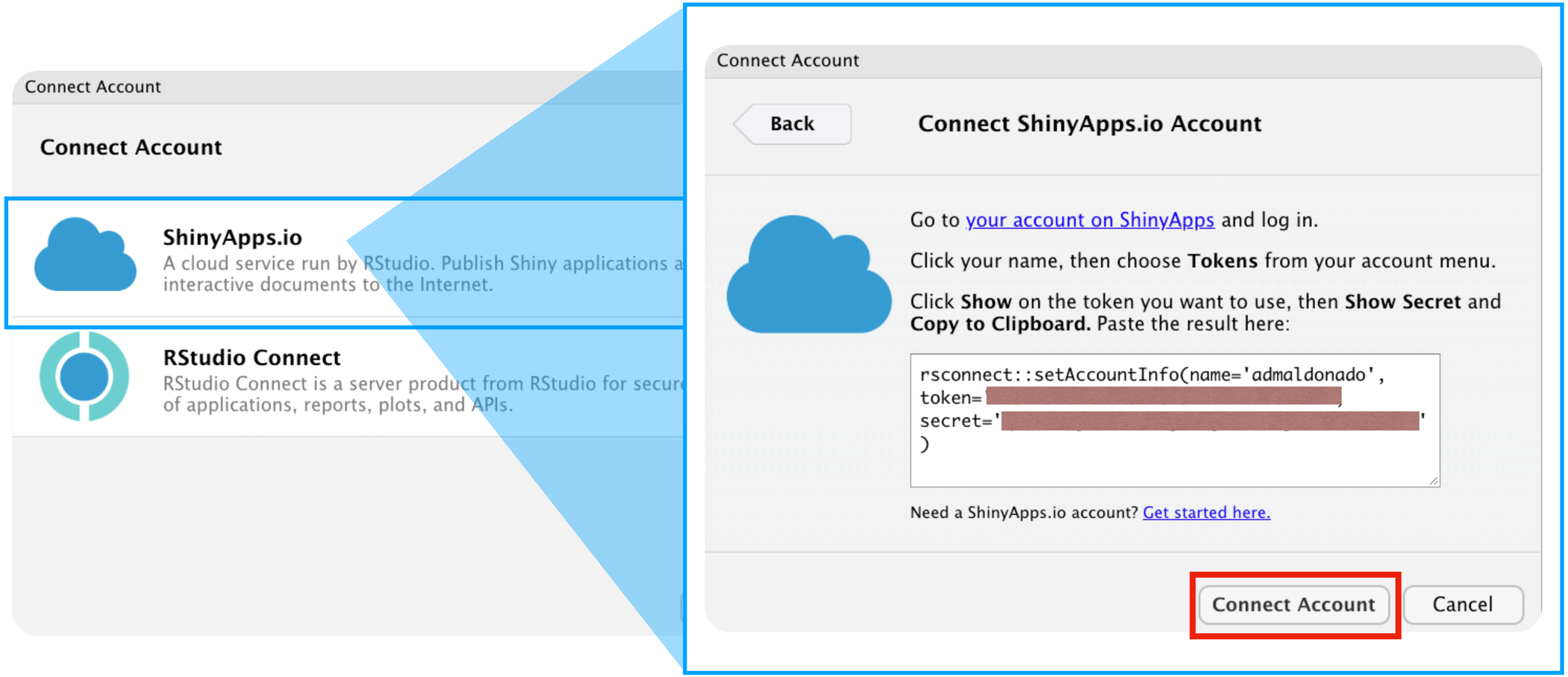


Publicación de Shiny apps

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”.

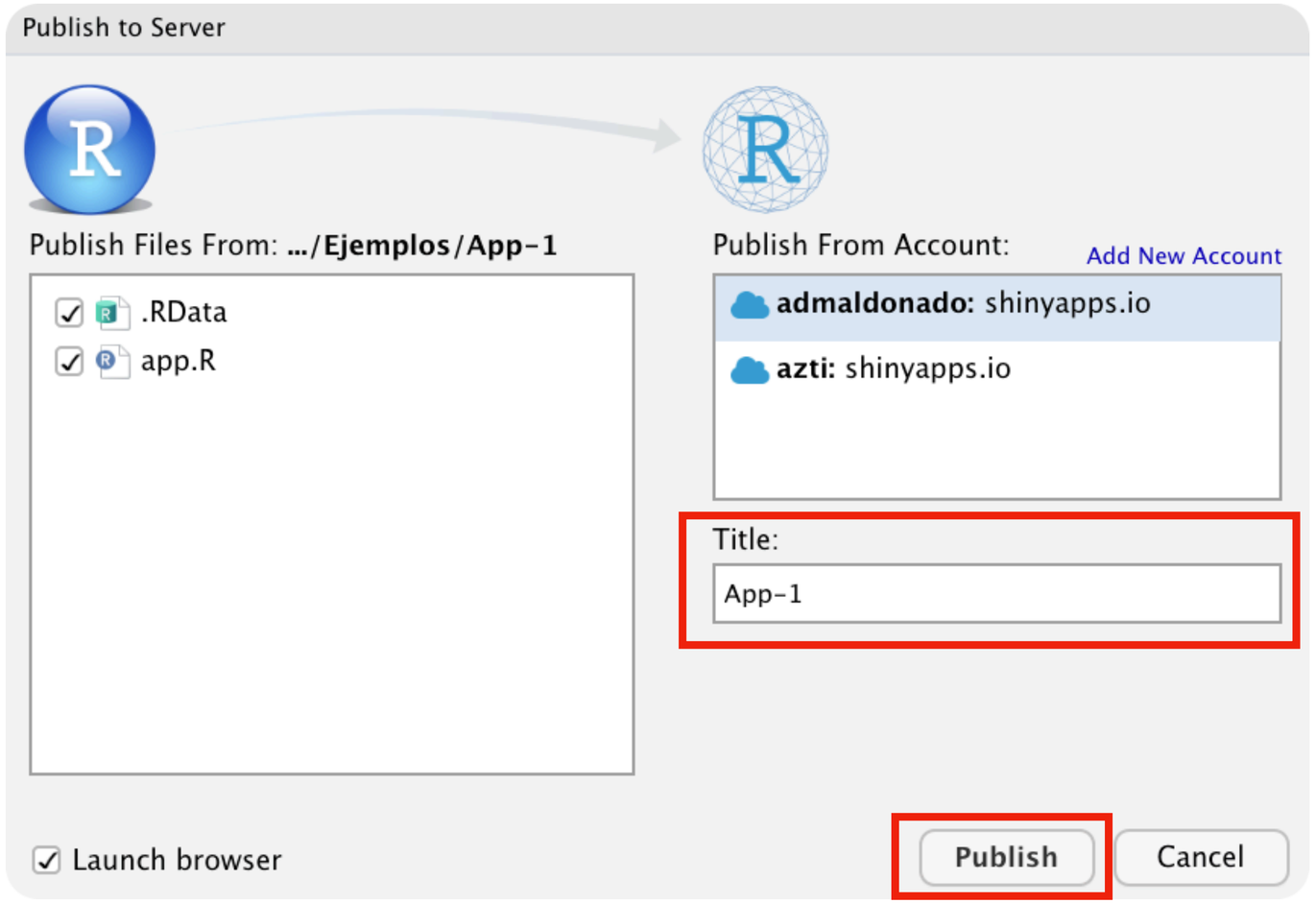


Ana D. Maldonado
ana.d.maldonado@ual.es

Introducción
Estructura de una app
Widgets
Outputs
Reactividad
Layout
Publicación

Publicación de Shiny apps

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

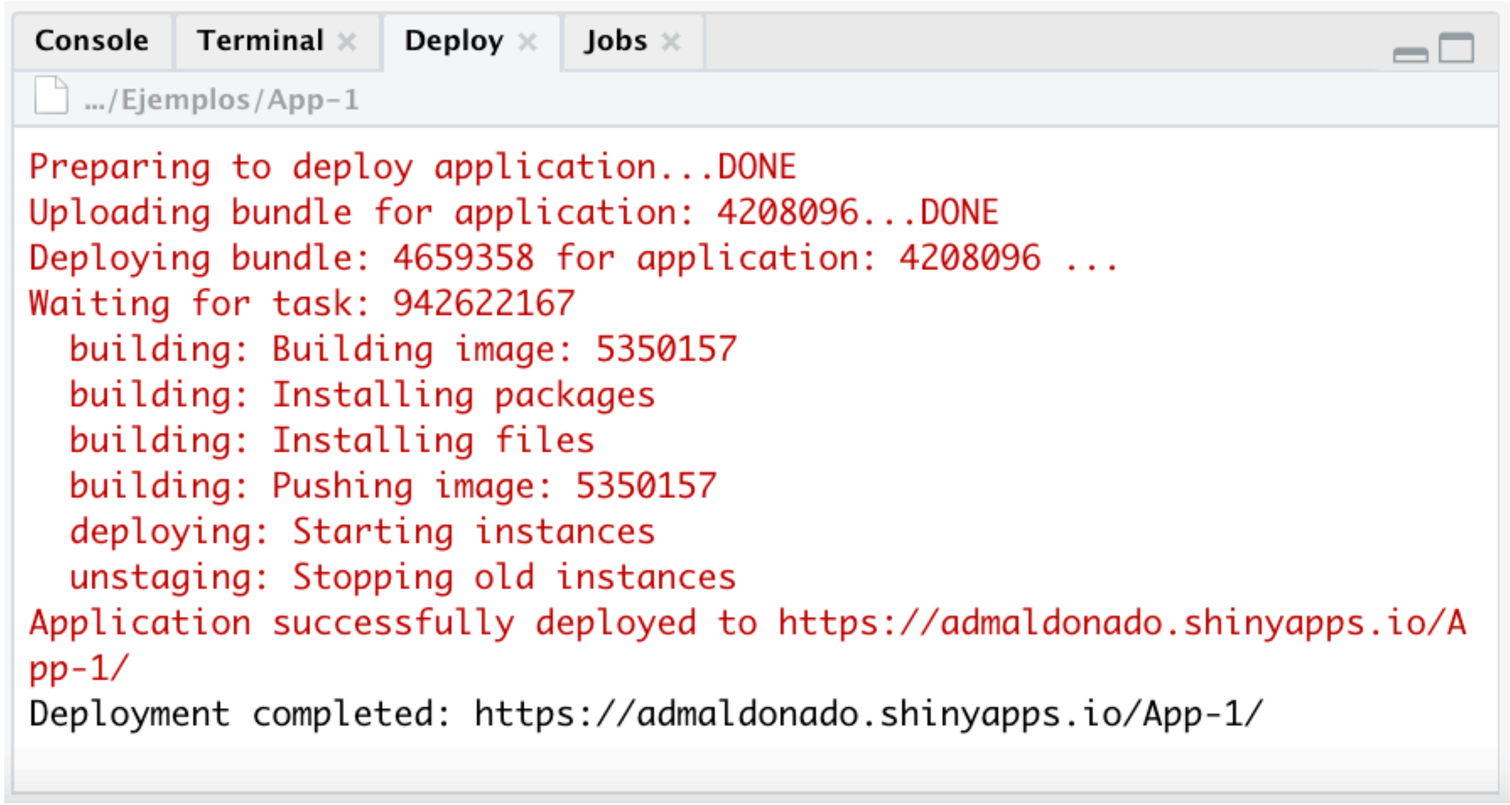


Ana D. Maldonado
ana.d.maldonado@ual.es

Introducción
Estructura de una app
Widgets
Outputs
Reactividad
Layout
Publicación

Publicación de Shiny apps

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

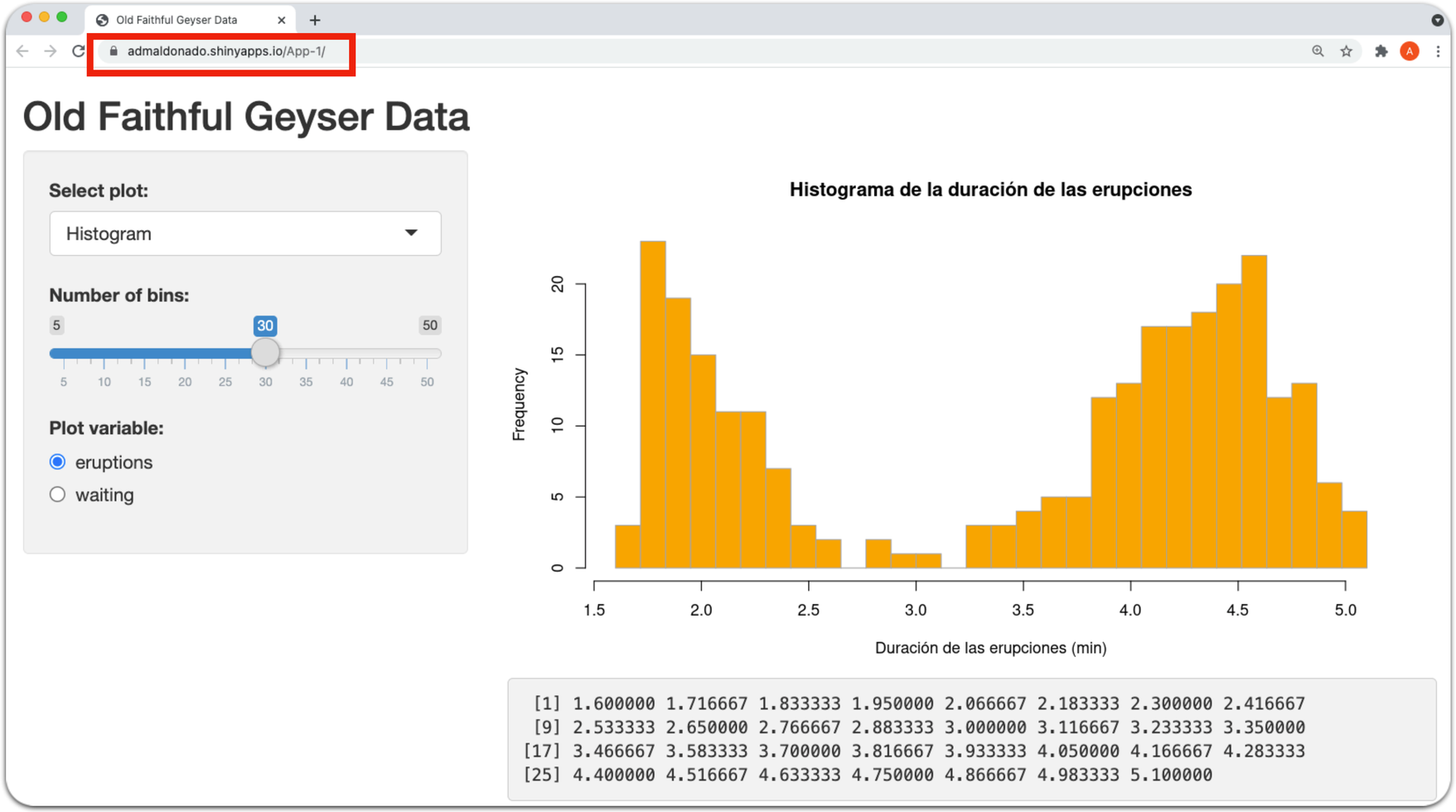


Publicación de Shiny apps

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!

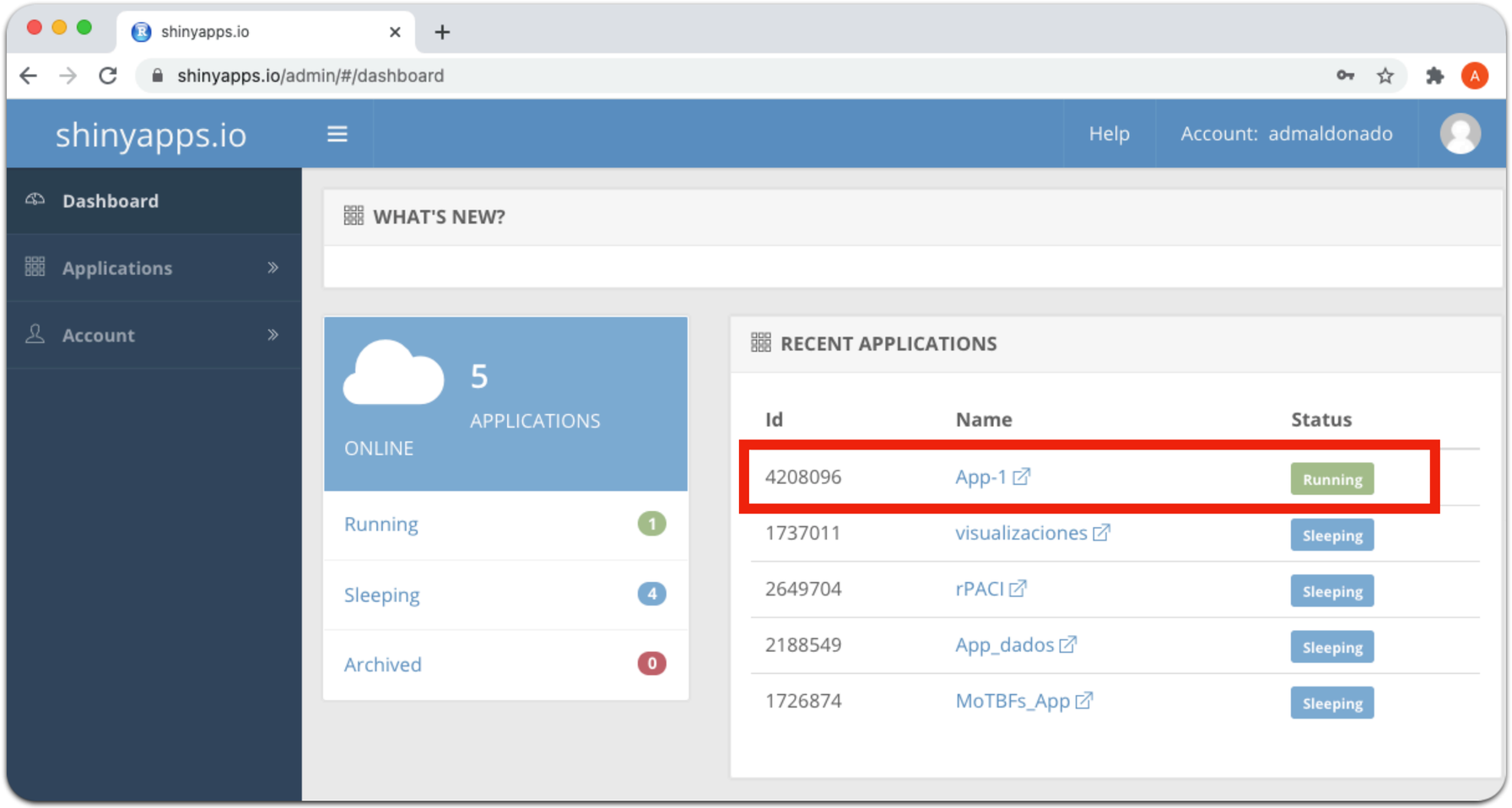


Publicación de Shiny apps

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.

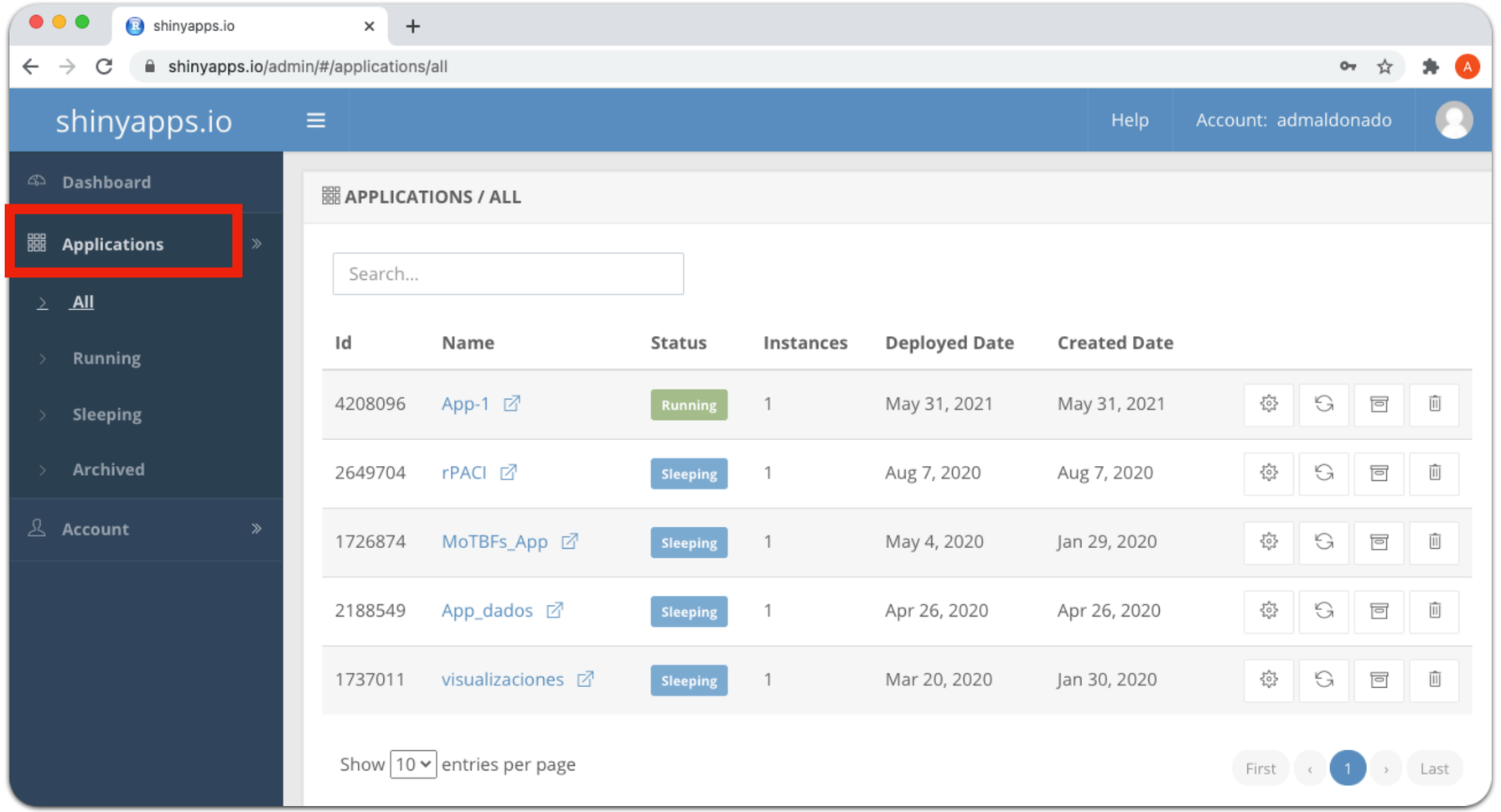


Publicación de Shiny apps

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.

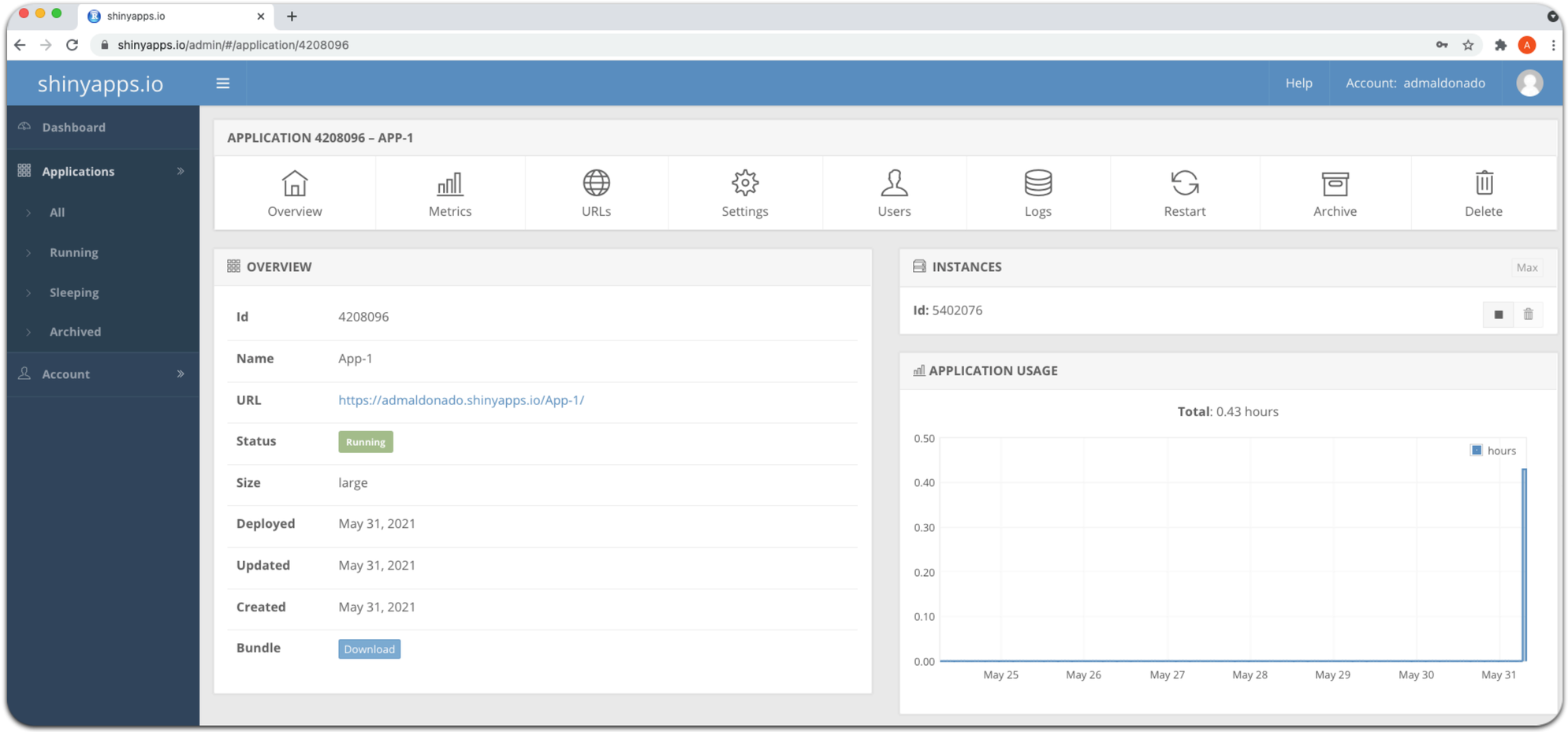


Publicación de Shiny apps

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.



Publicación de Shiny apps

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 quieres 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 quieres
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

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