

CURSO SOBRE BIOHERRAMIENTAS EN BIOESTADISTICA Y BIOINFORMATICA (1ªEdición)

Barcelona, 16, 17 y 18 de Mayo 2017

Cómo crear aplicaciones con Shiny

Parte II: Diseño del formulario

Elementos de entrada

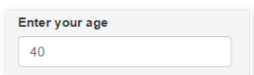
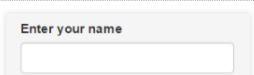

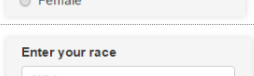



| Tipo | Función | Argumentos | Ejemplo |
|-------------------------|-----------------------------|--|---|
| Entrada numérica | <code>numericInput</code> | <code>inputId, label, value, min, max, step</code> |  |
| Entrada de texto | <code>textInput</code> | <code>inputId, label, value, width</code> |  |
| Lista de opciones | <code>radioButtons</code> | <code>inputId, label, choices, selected, inline, width</code> |  |
| Lista desplegable | <code>selectInput</code> | <code>inputId, label, choices, selected, multiple, selectize, width, size</code> |  |
| Lista desplegable | <code>selectizeInput</code> | <code>...+ options</code> | |
| Número (mínimo, máximo) | <code>sliderInput</code> | <code>inputId, label, min, max, value, step, animate</code> |  |
| Verdadero/Falso | <code>checkboxInput</code> | <code>inputId, label, value, width</code> |  |
| Botón | <code>actionButton</code> | <code>inputId, label, icon, width</code> |  |

Figure 1:

Listas desplegables

- Se usan las funciones `selectInput`, `selectizeInput`.
- Para poder seleccionar más de un ítem `multiple=TRUE`.
- Cuando se seleccionar más de un ítem, se puede visualizar la lista en dos formatos distintos:
 1. **Formato simple**: una opción debajo de la otra. Puede ser interesante hacer que la ventana sea más o menos larga mediante el argumento `size`.
 2. **Formato “selectize”**. Permite buscar las opciones con “texto-sensible”, y añadir más opciones mediante el argumento `options` usando la función `selectizeInput`. Para más información visita la siguiente web

```
library(compareGroups)
data(regicor)

ui <- fluidPage(

  selectInput("lista1", "Una opción", names(regicor)),

  selectInput("lista2", "Varios. Formato simple", names(regicor),
    multiple=TRUE, selectize=FALSE),

  selectInput("lista3", "Formato simple", names(regicor),
    multiple=TRUE, size=ncol(regicor), selectize=FALSE),

  selectizeInput("lista4", "Formato selectize", names(regicor),
    multiple=TRUE,
    options=list(plugins=list('remove_button', 'drag_drop')))

)

server <- function(input, output){}

shinyApp(ui, server)
```

Selecciona una variable

id

Selecciona las variables

id

year

age

sex

Selecciona las variables

id

year

age

sex

smoker

sbp

dbp

histtn

txhtn

chol

hdl

triglyc

ldl

histchol

txchol

height

weight

bmi

phyact

pcs

mcs

cv

tocv

death

todeath

Selecciona las variables

dbp

x

year

x

smoker

x

Elementos de salida

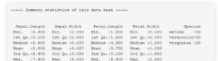
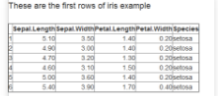
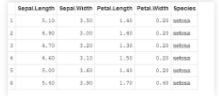
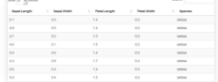
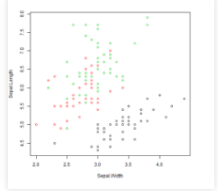
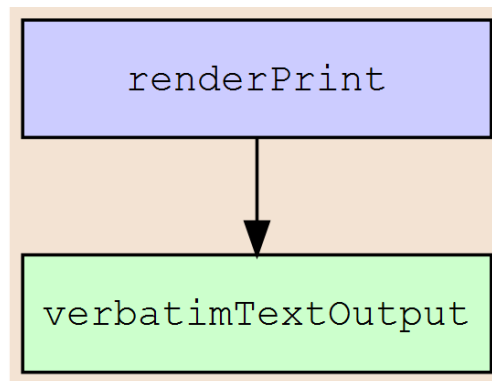
| Aspecto | Función | Argumentos | Ejemplo |
|-------------------------------|---|--|---|
| Texto como en la consola de R | <code>verbatimTextOutput</code> | <code>outputId</code> |  |
| Texto interpretado como HTML | <code>htmlOutput</code> | <code>outputId, inline</code> |  |
| Tabla "simple" | <code>tableOutput</code> | <code>outputId</code> |  |
| Tabla "compleja" | <code>dataTableOutput</code> | <code>outputId</code> |  |
| Gráfico | <code>plotOutput</code> <code>imageOutput</code> | <code>outputId, width, height, click, ...</code> |  |

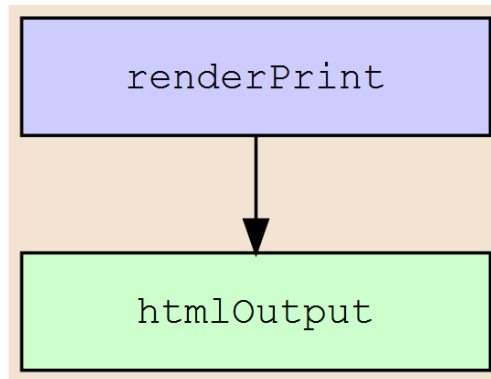
Figure 2:

Texto interpretado en R



```
ui <- fluidPage(  
  verbatimTextOutput("result")  
)  
  
server <- function(input, output) {  
  output$result<-renderPrint({  
    summary(iris)  
  })  
}  
  
shinyApp(ui = ui, server = server)
```

Texto interpretado en HTML



```
library(xtable)

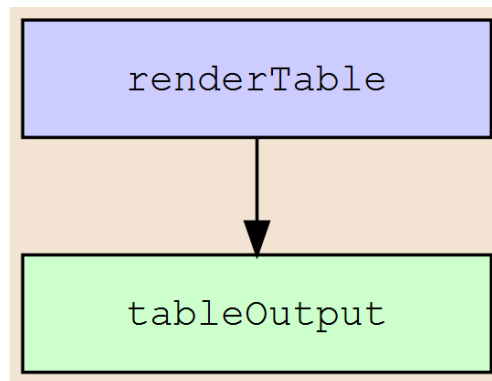
ui <- fluidPage(
  htmlOutput("result")
)

server <- function(input, output) {
  output$result<-renderPrint({
    print(xtable(head(iris)), type = "html")
  })
}

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

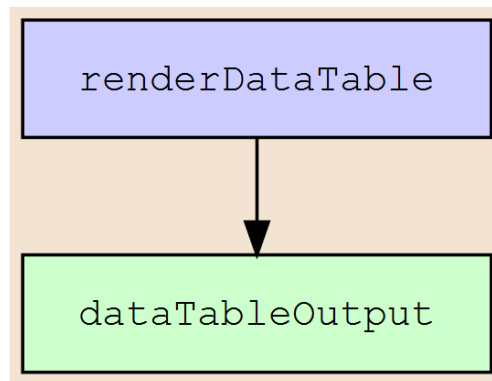
Ejercicio: substituye `htmlOutput` por `verbatimTextOutput`

Tabla Simple



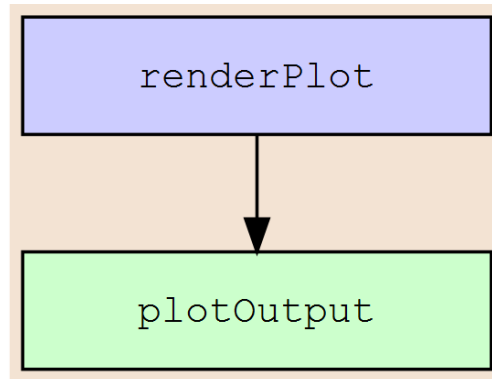
```
ui <- fluidPage(  
  tableOutput("result")  
)  
  
server <- function(input, output) {  
  output$result<-renderTable({  
    head(iris)  
  })  
}  
  
shinyApp(ui = ui, server = server)
```

Tabla Compleja



```
ui <- fluidPage(  
  dataTableOutput("result")  
)  
  
server <- function(input, output) {  
  output$result<-renderDataTable({  
    iris  
  })  
}  
  
shinyApp(ui = ui, server = server)
```


Gráficos



```
ui <- fluidPage(  
  plotOutput("result")  
)  
  
server <- function(input, output) {  
  
  output$result<-renderPlot({  
    plot(Sepal.Length ~ Sepal.Width,  
        col = Species, data = iris)  
  }, width = 500, height = 500)  
  
}  
  
shinyApp(ui = ui, server = server)
```

¿Cómo quedaría la figura si no se especifica la anchura ni la altura?

Disposición de los elementos. Layout

Paneles izquierdo y derecho

```
fluidPage(  
  sidebarLayout(  
    sidebarPanel(...),  
    mainPanel(...)  
  )  
)
```

- Es la opción más común
- Equivalentemente, también se puede usar la función `bootstrapPage` en lugar de `fluidPage`.

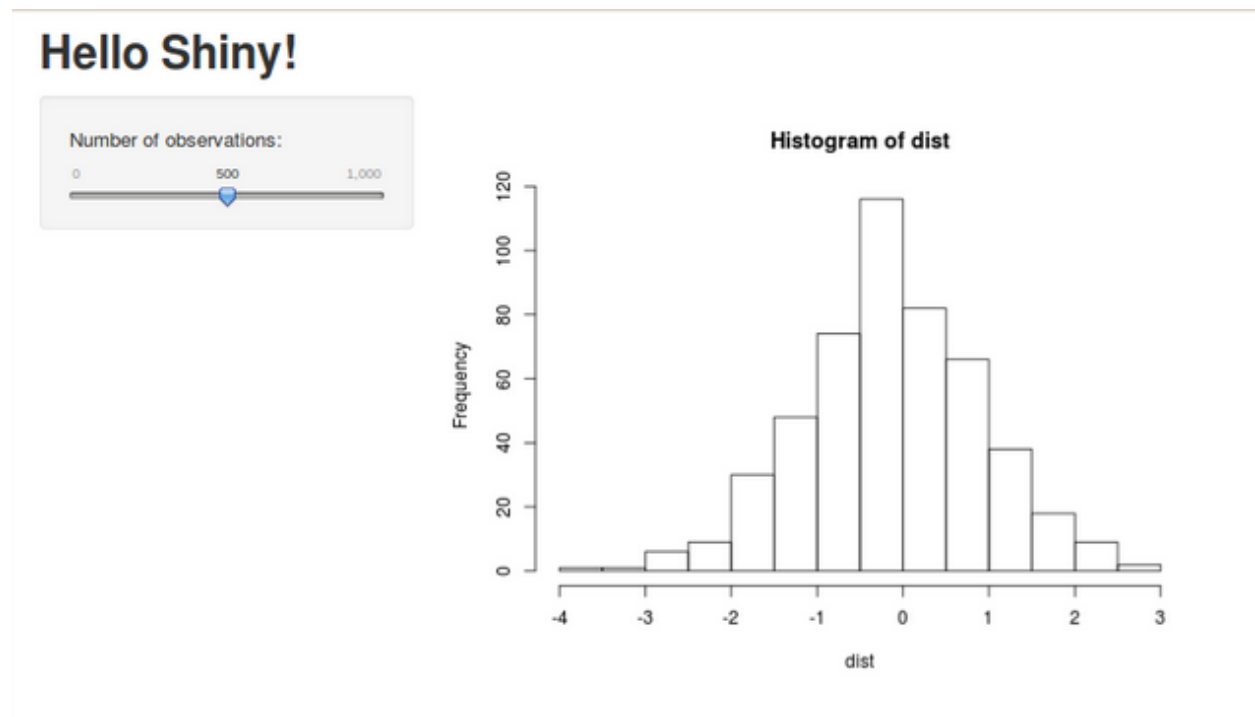


Figure 3:

Especificando filas y columnas

```
fluidPage(  
  
  fluidRow(  
    column(4, ...),  
    column(8, ...)  
  ),  
  
  fluidRow(  
    column(6, ...),  
    ...  
  ),  
  ...  
)
```

- Es la opción más flexible
- Se puede especificar la anchura de las columnas
- No se puede especificar la altura de las filas
- La suma de las columnas ha de ser 12.
- Se pueden poner tantas columnas como se quiera.
- Cada fila se especifica con **fluidRow**.

Ejercicio: Completa el código para disponer los elementos como en este ejemplo

```
ui <- shinyUI(fluidPage(  
  
  titlePanel("Grid example"),  
  ...  
  ...  
  
  submitButton("submit", "")  
  
))  
  
server <- function(input, output) {}  
  
shinyApp(ui = ui, server = server)
```

Grid example

Enter your name

Enter your age

Enter your gender

☒ Male
☐ Female

This is a very long text. This is a very long text. This is a very long text. This is a very long text. This is a very long text.

submit

Figure 4:

Menú general

```
navbarPage(  
  tabPanel(...),  
  tabPanel(...),  
)
```

- No es tan usado como los anteriores.
- El funcionamiento es el mismo que para las pestañas que se explicará a continuación.

| | | | |
|--------------|---------|------------|-------------|
| Menu general | Alumnos | Profesores | Asignaturas |
|--------------|---------|------------|-------------|

Nombre y apellidos

Edad

Sexo

- ☒ chico
☐ chica

| | | | |
|--------------|---------|------------|-------------|
| Menu general | Alumnos | Profesores | Asignaturas |
|--------------|---------|------------|-------------|

Nombre y apellidos

Número de asignaturas

Departamento

Pestañas

```
ui <- fluidPage(  
  tabsetPanel(id = "menu",  
    tabPanel("Tabla",  
      ...  
    ),  
    tabPanel("Resumen",  
      ...  
    )  
  )  
)
```

- Disposición de los elementos uno **detrás** de otro.
- Para crear un conjunto de pestañas se usa la función `tabsetPanel`.
- Para la creación de cada pestaña individualmente se usa la función `tabPanel`.

| Tabla | | Resumen | | | |
|-------|--------------|-------------|--------------|-------------|---------|
| | Sepal.Length | Sepal.Width | Petal.Length | Petal.Width | Species |
| 1 | 5.10 | 3.50 | 1.40 | 0.20 | setosa |
| 2 | 4.90 | 3.00 | 1.40 | 0.20 | setosa |
| 3 | 4.70 | 3.20 | 1.30 | 0.20 | setosa |
| 4 | 4.60 | 3.10 | 1.50 | 0.20 | setosa |
| 5 | 5.00 | 3.60 | 1.40 | 0.20 | setosa |
| 6 | 5.40 | 3.90 | 1.70 | 0.40 | setosa |

| Tabla | Resumen |
|---------------|--------------|
| Sepal.Length | Sepal.Width |
| Min. :4.300 | Min. :2.00 |
| 1st Qu.:5.100 | 1st Qu.:2.80 |
| Median :5.800 | Median :3.00 |
| Mean :5.843 | Mean :3.05 |
| 3rd Qu.:6.400 | 3rd Qu.:3.30 |
| Max. :7.900 | Max. :4.40 |

Desplegables: Unión de varias pestañas en una

```
ui <- fluidPage(  
  tabsetPanel(id="menu", type="pills",  
    tabPanel("Pestaña1",  
      ...  
    ),  
    navbarMenu("Desplegable",  
      tabPanel("opción1",  
        ...  
      ),  
      tabPanel("opción2",  
        ...  
      )  
    )  
  )  
)
```

- Función `navbar` cuyos argumentos son las pestañas.
- Se usa dentro de `tabsetPanel` para crear pestañas.
- Si una pestaña no tiene desplegable se usará igualmente la función `tabPanel`.
- Para la función `tabsetPanel` se puede especificar el estilo con el argumento “type”.

| resumen | | Tabla ▼ | | | | | | | |
|--------------|--------|-------------|--------------|-------------|---------|---------|--------|------------|-----|
| Sepal.Length | | Sepal.Width | Petal.Length | Petal.Width | Species | | | | |
| Min. | :4.300 | Min. | :2.000 | Min. | :1.000 | Min. | :0.100 | setosa | :50 |
| 1st Qu. | :5.100 | 1st Qu. | :2.800 | 1st Qu. | :1.600 | 1st Qu. | :0.300 | versicolor | :50 |
| Median | :5.800 | Median | :3.000 | Median | :4.350 | Median | :1.300 | virginica | :50 |
| Mean | :5.843 | Mean | :3.057 | Mean | :3.758 | Mean | :1.199 | | |
| 3rd Qu. | :6.400 | 3rd Qu. | :3.300 | 3rd Qu. | :5.100 | 3rd Qu. | :1.800 | | |
| Max. | :7.900 | Max. | :4.400 | Max. | :6.900 | Max. | :2.500 | | |

Figure 5:

| resumen | | | | | |
|---------|--------------|-------------|--------------|-------------|---------|
| Tabla ▼ | | | | | |
| | Sepal.Length | Sepal.Width | Petal.Length | Petal.Width | Species |
| 1 | 5.10 | 3.50 | 1.40 | 0.20 | setosa |
| 2 | 4.90 | 3.00 | 1.40 | 0.20 | setosa |
| 3 | 4.70 | 3.20 | 1.30 | 0.20 | setosa |
| 4 | 4.60 | 3.10 | 1.50 | 0.20 | setosa |
| 5 | 5.00 | 3.60 | 1.40 | 0.20 | setosa |
| 6 | 5.40 | 3.90 | 1.70 | 0.40 | setosa |

Figure 6:

Paneles condicionales

- Se usan cuando algunos elementos tienen que aparecer o no en función del valor de un input.
- Función `conditionalPanel`.
- Primer argumento: carácter. Condición lógica escrita en **formato “javascript”**.
- Segundo argumento: elementos del formulario que aparecen si la condición se cumple.

```
conditionalPanel(
  condition = "input.tipo==1",
  ...
)
```

Ejemplo 1

```
ui <- fluidPage(  
  radioButtons("tipo",  
    "¿Qué quieres mostrar?",  
    c("Tabla"=1, "Resumen"=2)),  
  conditionalPanel(  
    condition = "input.tipo==1",  
    tableOutput("result1")  
  ),  
  conditionalPanel(  
    condition = "input.tipo==2",  
    verbatimTextOutput("result2")  
  )  
)  
  
server <- function(input, output) {  
  output$result1 <- renderTable(  
    head(iris)  
  )  
  output$result2 <- renderPrint(  
    summary(iris)  
  )  
}
```

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

¿Qué quieres mostrar?

☒ Tabla

☐ Resumen

| | Sepal.Length | Sepal.Width | Petal.Length | Petal.Width | Species |
|---|--------------|-------------|--------------|-------------|---------|
| 1 | 5.10 | 3.50 | 1.40 | 0.20 | setosa |
| 2 | 4.90 | 3.00 | 1.40 | 0.20 | setosa |
| 3 | 4.70 | 3.20 | 1.30 | 0.20 | setosa |
| 4 | 4.60 | 3.10 | 1.50 | 0.20 | setosa |
| 5 | 5.00 | 3.60 | 1.40 | 0.20 | setosa |
| 6 | 5.40 | 3.90 | 1.70 | 0.40 | setosa |

Figure 7:

Ejercicio: Fíjate que esta opción podría ser una alternativa a `tabsetPanel`. ¿Cómo lo harías?

¿Qué quieres mostrar?

- ☐ Tabla
- ☒ Resumen

| Sepal.Length | Sepal.Width | Petal.Length | Petal.Width | Species |
|---------------|---------------|---------------|---------------|---------------|
| Min. :4.300 | Min. :2.000 | Min. :1.000 | Min. :0.100 | setosa :50 |
| 1st Qu.:5.100 | 1st Qu.:2.800 | 1st Qu.:1.600 | 1st Qu.:0.300 | versicolor:50 |
| Median :5.800 | Median :3.000 | Median :4.350 | Median :1.300 | virginica :50 |
| Mean :5.843 | Mean :3.057 | Mean :3.758 | Mean :1.199 | |
| 3rd Qu.:6.400 | 3rd Qu.:3.300 | 3rd Qu.:5.100 | 3rd Qu.:1.800 | |
| Max. :7.900 | Max. :4.400 | Max. :6.900 | Max. :2.500 | |

Figure 8:

Ejemplo 2

```
ui <- fluidPage(  
  
  checkboxInput("ayuda", "Ayuda"),  
  
  conditionalPanel(  
    condition = "input.ayuda",  
    helpText("Esto es una  
      explicación sobre como  
      funciona el aplicativo.")  
  )  
  
)  
  
server <- function(input, output) {}  
  
shinyApp(ui = ui, server = server)
```

☒ Ayuda

Esto es una explicación sobre como funciona el aplicativo.

☐ Ayuda

El texto aparece o desaparece.

Ejemplo 3

```
library(shiny)

ui <- fluidPage(

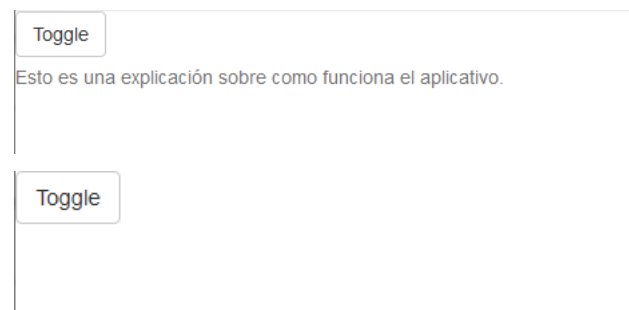
  actionButton("toggle", "Toggle"),

  conditionalPanel(
    condition = "input.toggle%2==0",
    helpText("Esto es una explicación
sobre como funciona el aplicativo.")
  )

)

server <- function(input, output) {}

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



Ejemplo 4

```
ui <- fluidPage(  
  
  selectInput("format", "Formato",  
    c("SPSS"=1, "EXCEL"=2, "TXT"=3, "R"=4)),  
  
  conditionalPanel(  
    condition = "input.format==3",  
    wellPanel(  
      radioButtons("sep", "Separador",  
        c(",", ";", "tab"), inline = TRUE),  
      radioButtons("dec", "Decimal",  
        c(".", ","), inline = TRUE),  
      checkboxInput("header", "Cabecera")  
    )  
  )  
  
)  
  
server <- function(input, output) {}  
  
shinyApp(ui = ui, server = server)
```

Fíjate en la función `wellPanel` para crear un *frame* alrededor de los elementos, y el argumento `inline` de la función `radioButtons` para disponer las opciones (botones) en horizontal.

Formato

SPSS

Formato

TXT

Separador

☒ , ☐ ; ☐ tab

Decimal

☒ . ☐ ,

☐ Cabecera

Ejemplo 5

```
ui <- fluidPage(  
  sidebarLayout(  
    sidebarPanel(  
      conditionalPanel(  
        condition = "input.menu=='panel 1'",  
        "Ahora está activo el panel 1"  
      ),  
      conditionalPanel(  
        condition = "input.menu=='panel 2'",  
        "Ahora está activo el panel 2"  
      )  
    ),  
    mainPanel(  
      tabsetPanel(id = "menu",  
        tabPanel("panel 1",  
          "Texto en el panel 1"  
        ),  
        tabPanel("panel 2",  
          "Texto en el panel 2"  
        )  
      )  
    )  
  )  
)  
  
server <- function(input, output) {}  
  
shinyApp(ui = ui, server = server)
```

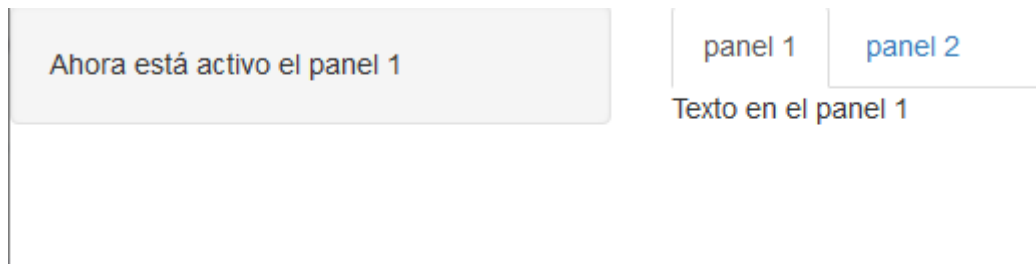


Figure 9:

El resultado varía según está activo una pestaña u otra.

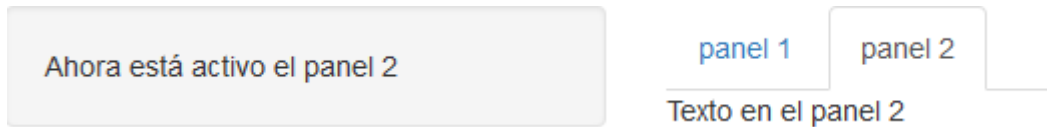


Figure 10:

Ejercicio

Completa el código de la parte UI para colocar las elementos y pestañas del siguiente formulario.

A la **izquierda del formulario** debe aparecer un panel donde:

- Se escoja una distribución entre
- exponencial
- normal
- binomial
- Según la distribución aparezcan los inputs de los parámetros apropiados para que el usuario introduzca sus valores

A la **derecha del formulario** aparezcan dos pestañas:

- La primera debe contener un histograma de los datos generados según la distribución (1000 datos).
- La segunda pestaña debe contener un **summary** de los datos generados.

Ejemplo Distribución

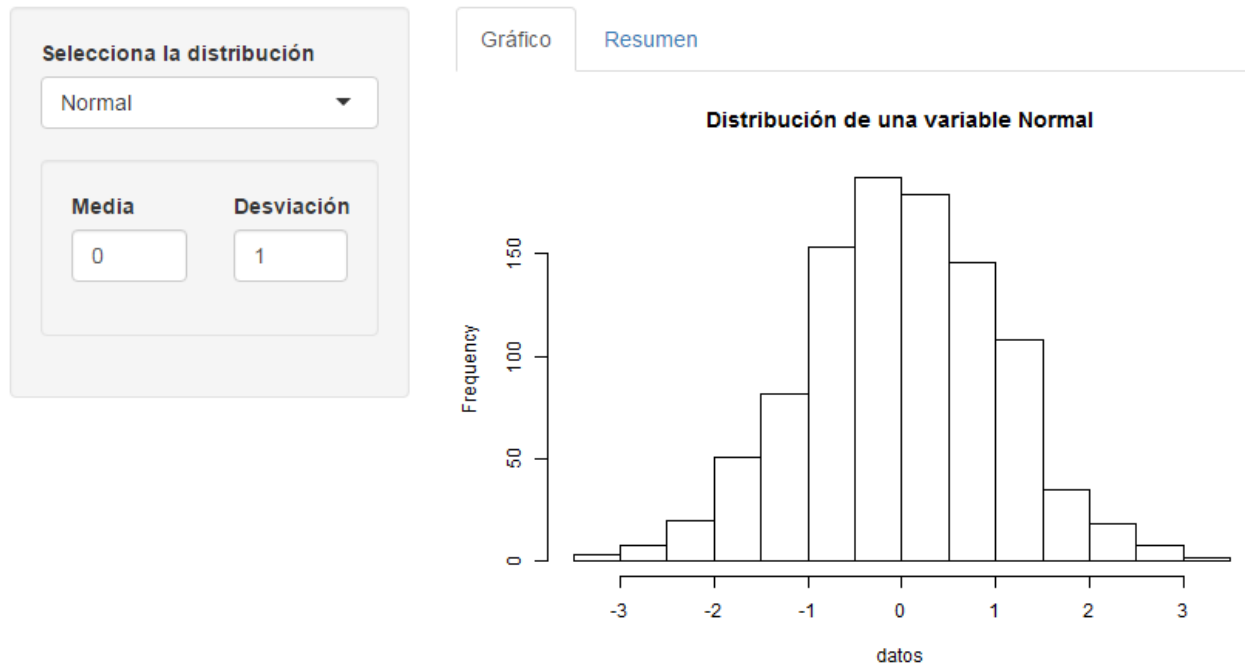


Figure 11:

```
ui <- fluidPage(  
  titlePanel("Ejemplo Distribución"),  
  sidebarLayout(  
    sidebarPanel(  
      ## rellenar  
    ),  
    mainPanel(  
      ## rellenar  
    )  
  )  
)  
  
server <- function(input, output) {  
  output$resumen <- renderPrint({  
    if (input$distr=="Normal")  
      datos <- rnorm(1000, input$media, input$sd)  
    if (input$distr=="Exponencial")  
      datos <- rexp(1000, input$lambda)  
    if (input$distr=="Binomial")  
      datos <- rbinom(1000, input$n, input$p)  
    summary(datos)  
  })  
}
```

```

output$grafico <- renderPlot({
  if (input$distr=="Normal")
    datos <- rnorm(1000, input$media, input$sd)
  if (input$distr=="Exponencial")
    datos <- rexp(1000, input$lambda)
  if (input$distr=="Binomial")
    datos <- rbinom(1000, input$n, input$p)
  if (input$distr=="Binomial"){
    datosfact <- factor(datos, levels=0:input$n)
    barplot(table(datosfact))
  } else {
    hist(datos, main="")
  }
  title(paste("Distribución ", input$distr))
})
}

```