Shiny: Visualizacion de datos con R

Diplomado en Data Science 22, MatPUC

Joshua Kunst Fuentes jbkunst@gmail.com

Version 2022

Johnson

Jue 21/7 - Sáb 23/7 - Jue 28/7 - Jue 4/8

Touring

Mie 27/7 - Mie 02/8 - Mie 10/8 - Mie 17/8

Programa

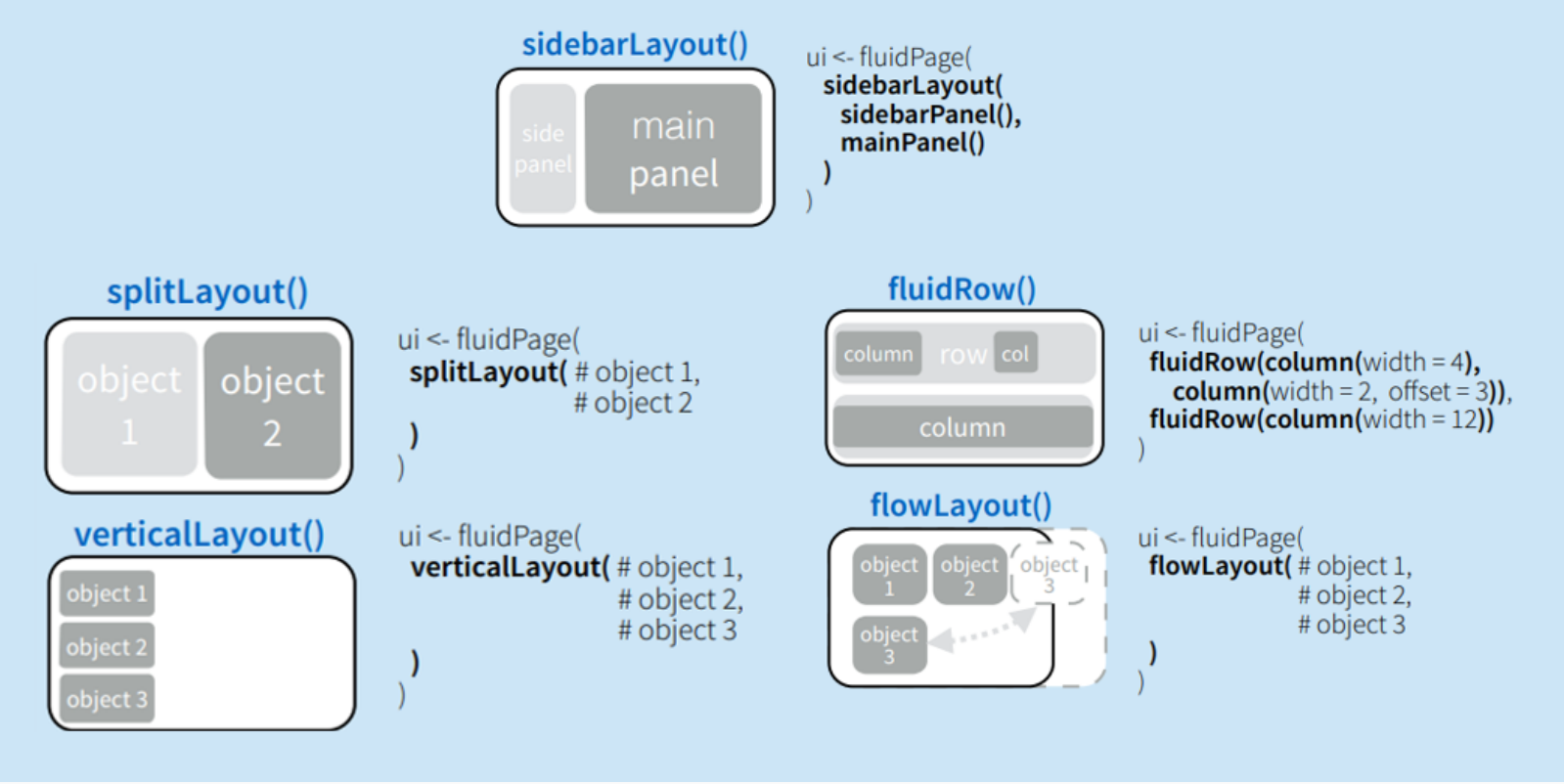
- Clase 1
  - Aplicación (web), ejemplos.
  - Introducción a shiny: Interfaz usuario y servidor
  - Reactividad
- Clase 2
  - Layouts
  - Integración HTMLWidgets
- Clase 3
  - Repaso/Ejercicios
  - Templates y diseño
  - Compartir una app
- Clase 4
  - Expresiones reactivas
  - Extensiones shiny

Layouts

Tipos de Layouts

Laytous se refiere a la disposición de elementos -como inputs, textos, outputs- en nuestra app.

Dependiendo de las necesidades puede ser convenientes algunos tipos de layuts sobre otros.



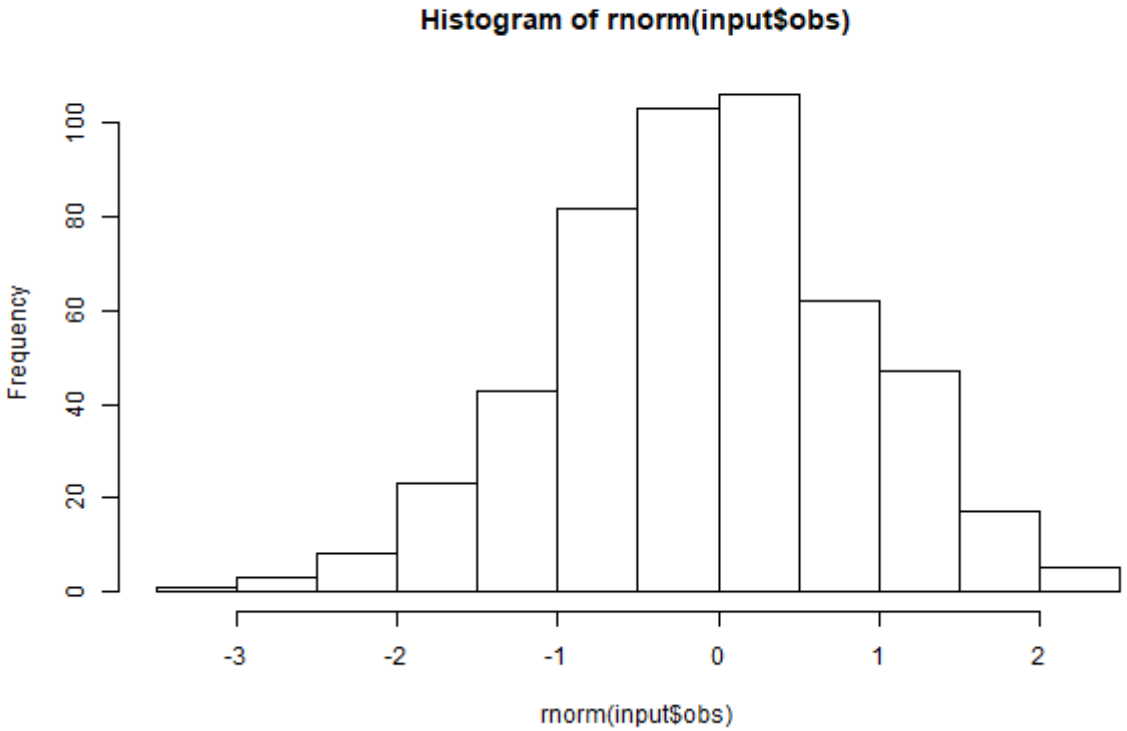
sidebarLayout

Hello Shiny!

Number of observations:

0 500 1,000

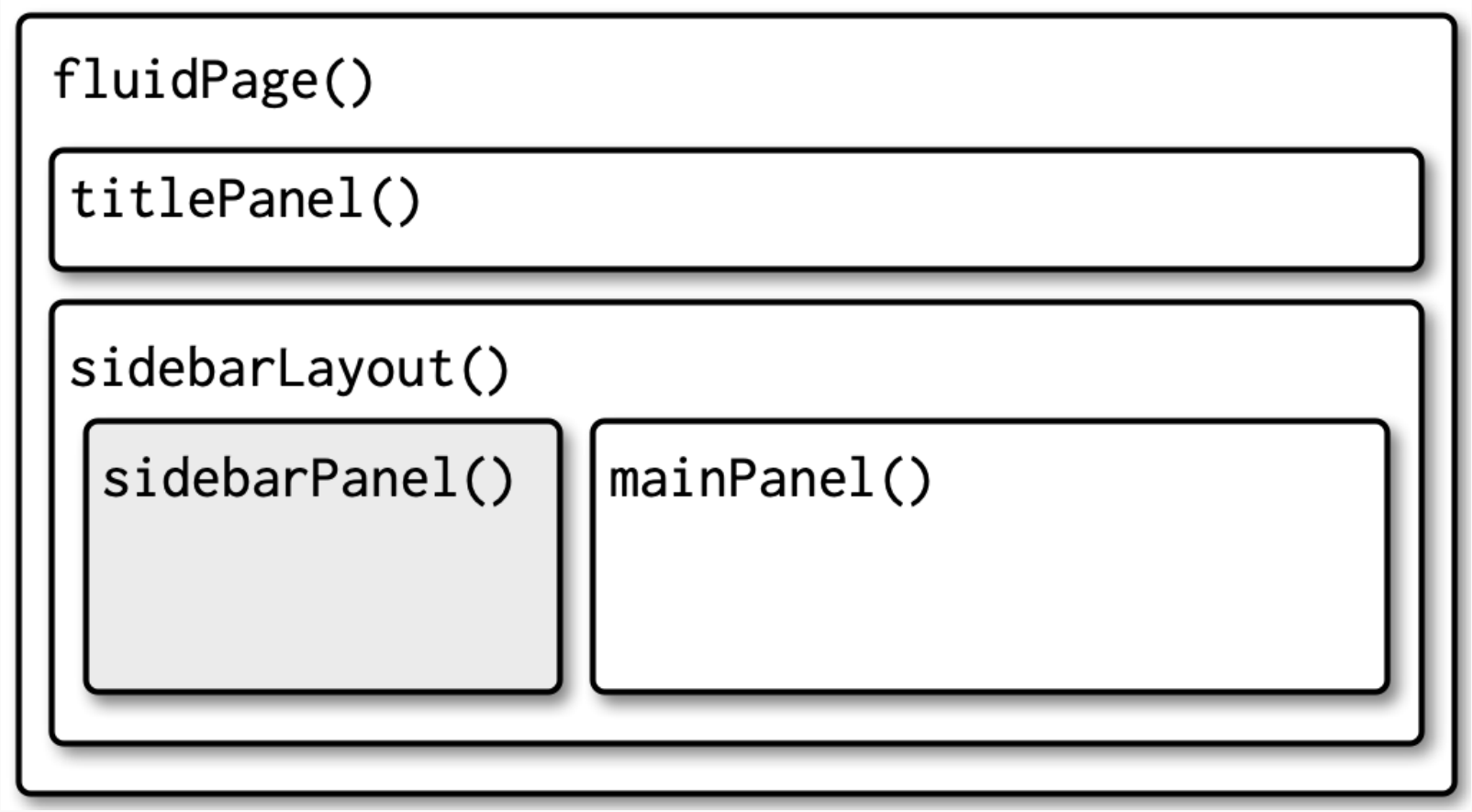
0 100 200 300 400 500 600 700 800 900 1,000



sidebarLayout: Estructura

Es una distribución simple donde lo primero en aparecer es los controles o inputs.

Luego viene el panel principal el cual posee mayor espacio.



sidebarLayout: Ejemplo código

```
library(shiny)

ui ← fluidPage(
  titlePanel("Hello Shiny!"),
  sidebarLayout(
    sidebarPanel(
      sliderInput("obs", "Number of observations:", min = 0, max = 1000, value = 500)
    ),
    mainPanel(plotOutput("distPlot"))
  )
)

server ← function(input, output) {
  output$distPlot ← renderPlot({ hist(rnorm(input$obs)) })
}

shinyApp(ui, server)
```

Paneles de Navegación

Tipos de Paneles de Navegación

Es posible que nuestra aplicación vaya creciendo, en dicho caso, Los paneles nos permiten organizar distintas secciones de la aplicación.

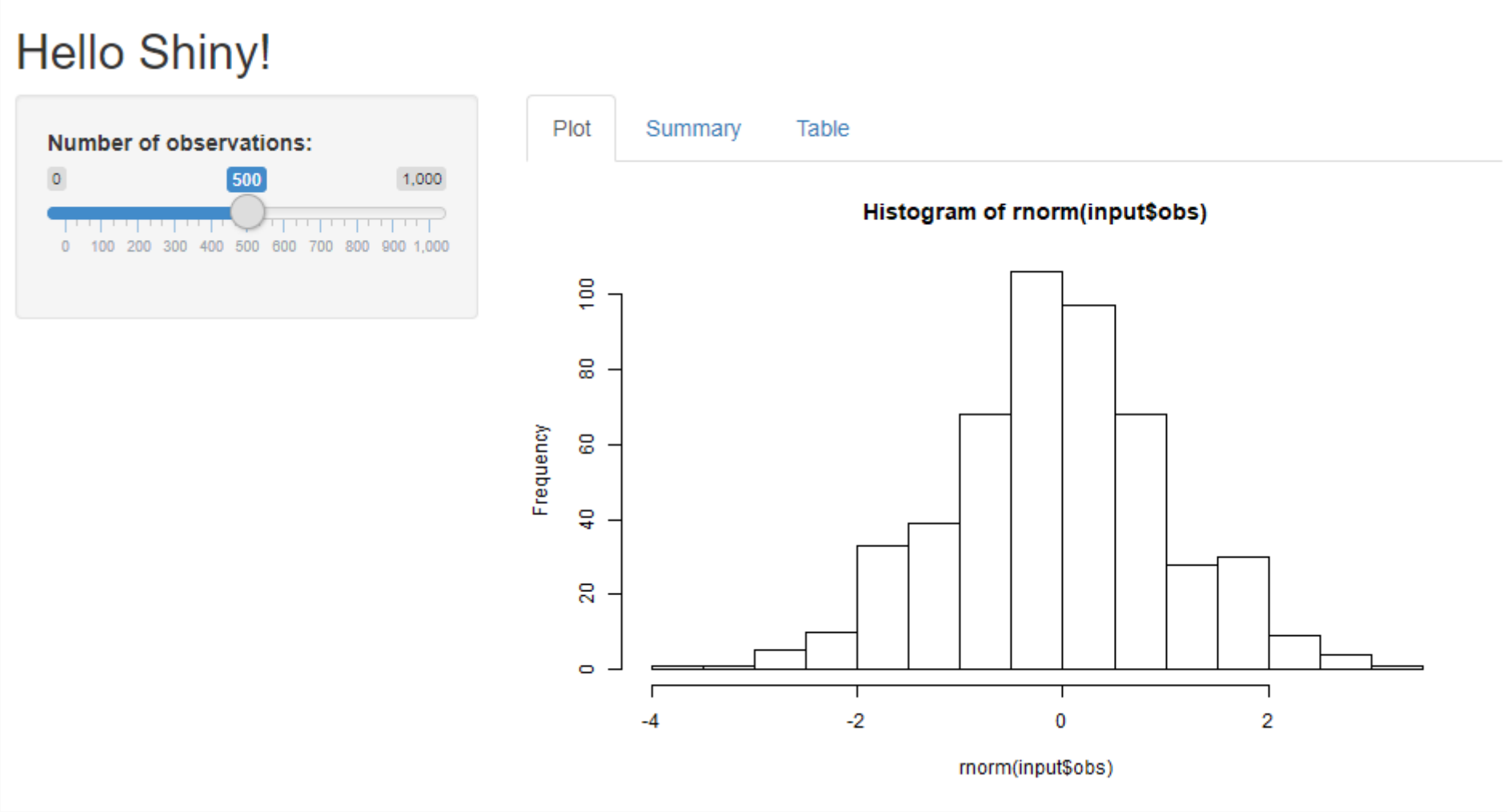
tabsetPanel

Los tabs (`tabsetPanel`) son útiles para separar secciones *similares* en nuestra app. A diferencia de por ejemplo `navbarPage`.

```
library(shiny)

ui ← fluidPage(
  titlePanel("Hello Shiny!"),
  sidebarLayout(
    sidebarPanel(
      sliderInput("obs", "Number of observations:", min = 0, max = 1000, value = 500)
    ),
    mainPanel(
      tabsetPanel(
        tabPanel("Plot", plotOutput("plot")),
        tabPanel("Summary", verbatimTextOutput("summary")),
        tabPanel("Table", tableOutput("tabla"))
      )
    )
  )
)

server ← function(input, output) {
  output$plot ← renderPlot({ hist(rnorm(input$obs)) })
  output$summary ← renderText({ input$obs })
  output$tabla ← renderTable({ data.frame(input$obs) })
}

shinyApp(ui, server)
```

tabsetPanel

Los tabs (`tabsetPanel`) son útiles para separar secciones *similares* en nuestra app. A diferencia de por ejemplo `navbarPage`.



Ejercicio: Modificar la estructura de la aplicación

Nivel *fácil, para calentar las manos*: Para el ejemplo de `sidebarLayout`: Modifique la aplicación para que posea un layout de tipo `flowLayout`.

Nivel *entretenido*: Para el ejemplo de `tabsetPanel` agregue:

- Un tab más con un gráfico de líneas.
- Un tab que utilice la función `summary` para mostra resúmen de los datos.
- Modifique la applicación para utilizar la función `navbarPage`.

Solución 1

```
library(shiny)

ui ← fluidPage(
  titlePanel("Hello Shiny!"),
  flowLayout(
    sliderInput("obs",
                "Number of observations:",
                min = 0,
                max = 1000,
                value = 500),
    plotOutput("distPlot")
    )
  )

server ← function(input, output) {
  output$distPlot ← renderPlot({
    hist(rnorm(input$obs))
    })
}

shinyApp(ui, server)
```

Solución 2

```
library(shiny)
library(ggplot2)

ui ← navbarPage(
  title = "Hello Shiny!",
  tabPanel("Plot",
           sliderInput("obs",
           "Number of observations:",
           min = 0, max = 1000, value = 500),
           plotOutput("plot")
           ),
  tabPanel("Línea", plotOutput("linea")),
  tabPanel("Summary", verbatimTextOutput("summary")),
  tabPanel("Table", tableOutput("tabla"))
)
```

```
server ← function(input, output) {
  output$plot ← renderPlot({
    hist(rnorm(input$obs))
    })
  output$summary ← renderPrint({
    summary(rnorm(input$obs))
    })
  output$tabla ← renderTable({
    data.frame(rnorm(input$obs))
    })
  output$linea ← renderPlot({
    x ← rnorm(input$obs)
    qplot(x = 1:input$obs, x, geom = "line")
  })


}

shinyApp(ui, server)
```

HTMLWidgets

leaflet

```
library(leaflet)

leaflet(datos) ▷
  addTiles() ▷
  addMarkers(
    lng = ~longitud,
    lat = ~latitud,
    popup = ~as.character(magnitud_2),
    label = ~as.character(`fecha_local_lugar`)
  ) ▷
  addProviderTiles("Esri.WorldImagery")
```

Pero como usar HTMLWidgets en nuestra App?

Cada uno de los HTMLWidgets presentados, en su documentación detallan como usarlos en una aplicación shiny.

De forma general en cada paquete existirá una función tipo:

- `*Output` para colocarla en el `ui`.
- `render*` para definirla en el `server`

A modo de ejemplo, el paquete `leaflet` tiene `leafletOutput()` y `renderLeaflet()`.

Los anterior está documentado en https://rstudio.github.io/leaflet/shiny.html.

Ejercicio: Agregar HTMLWidgets a la app

A la aplicación obtenida de modificar la de `tabsetPanel` (Nivel *entretenido*), modificar gráficos y tablas para incluir algunos HTMLWidgets.

Solución

```
library(shiny)
library(ggplot2)
library(plotly)
library(DT)

ui ← navbarPage(
  title = "Hello Shiny!",
  tabPanel("Plot",
           sliderInput("obs",
           "Number of observations:",
           min = 0, max = 1000, value = 500),
           plotOutput("plot")
           ),
  tabPanel("Línea", plotlyOutput("linea")),
  tabPanel("Summary", verbatimTextOutput("summary")),
  tabPanel("Table", DTOutput("tabla"))
)
```

```
server ← function(input, output) {
  output$plot ← renderPlot({
    hist(rnorm(input$obs))
    })
  output$summary ← renderPrint({
    summary(rnorm(input$obs))
    })
  output$tabla ← renderDT({
    d ← data.frame(rnorm(input$obs))
    datatable(d)
    })
  output$linea ← renderPlotly({
    y ← rnorm(input$obs)
    p ← qplot(x = 1:input$obs, y = y, geom = "line")
    ggplotly(p)
  })

}

shinyApp(ui, server)
```

Solución v2

```
library(shiny)
library(ggplot2)
library(plotly)
library(DT)

ui ← navbarPage(
  title = "Hello Shiny!",
  tabPanel("Plot",
           sliderInput("obs",
           "Number of observations:",
                min = 0, max = 1000, value = 500),
    plotlyOutput("plot")
  ),
  tabPanel('Linea', plotOutput('linea')),
  tabPanel("Summary", textOutput("summary")),
  tabPanel("Table", dataTableOutput("tabla")),
  tabPanel('Resumen', verbatimTextOutput('resumen'))
)
```

```
server ← function(input, output) {
  output$plot ← renderPlotly({
    n ← input$obs
    x ← rnorm(n)
    p ← qplot(x, geom = "histogram")
    ggplotly(p)
  })

  output$summary ← renderText({ input$obs })
  output$tabla ← renderDataTable({
    df ← data.frame(1:input$obs)
    datatable(df)
    })
  output$resumen ← renderPrint({
    summary(rnorm(input$obs))
  } )
  output$linea ← renderPlot({
    plot(rnorm(input$obs), type = 'l')
  })
}
shinyApp(ui, server)
```