Shiny
Visualizacion de datos con R
Slides 2

Diplomado en Data Science, MatPUC

Joshua Kunst Fuentes jbkunst@gmail.com https://jkunst.com/blog/

Shiny: Visualizacion de datos con R

Version 2025

Clases

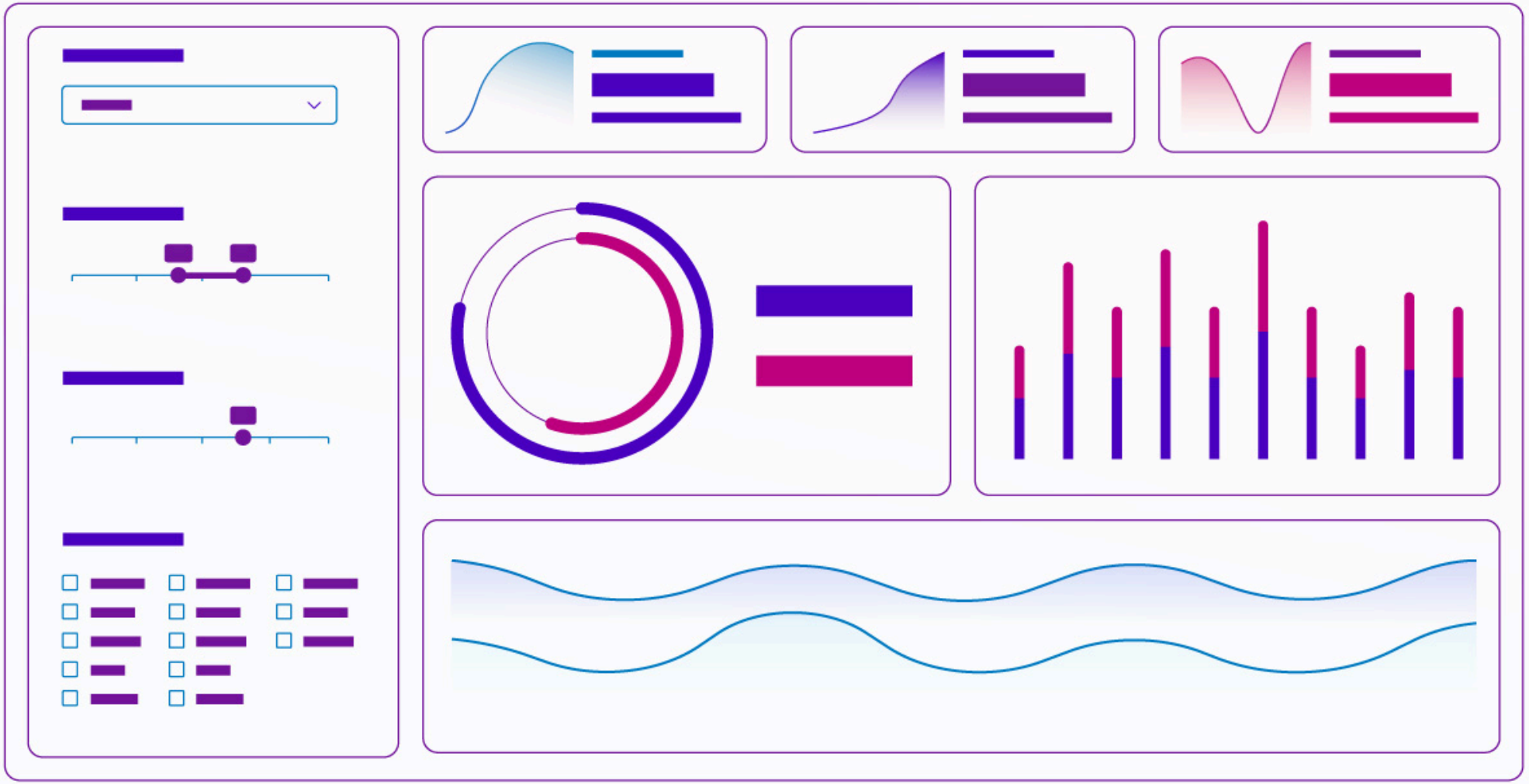
- Martes 18 de noviembre
- Jueves 20 de Noviembre

Programa

- Slides 1
  - Aplicación (web), ejemplos.
  - Introducción a shiny: Interfaz usuario y servidor
- Slides 2
  - Layouts
  - Integración HTMLWidgets
- Slides 3
  - Temas, templates y diseño
  - Compartir una app
  - Shiny en Python
- Slides 4
  - Expresiones reactivas
  - Orden de ejecución
  - Extensiones shiny

Layouts

Layouts



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.

Revisemos el ejemplo *Chicago Flights* que viene en el paquete {bslib}:

```
# install.packages(c("histoslider", "DT", "fontawesome", "bsicons", "pak"))
# pak::pak("cpsievert/chiflights22")

shiny::runApp(system.file("examples-shiny", "flights", package = "bslib"))
```

Layouts

Como mencionamos anteriormente utilizaremos para esto el paquete {bslib}.

Lo importante es considerar que los elementos que revisaremos se pueden utilizar dentro de otros. La idea de armar un layout es generalmente anidar elementos.

Acá nos centraremos solamente en el *ui* pues al modificar la disposición de los elementos no afectaremos al *server*.

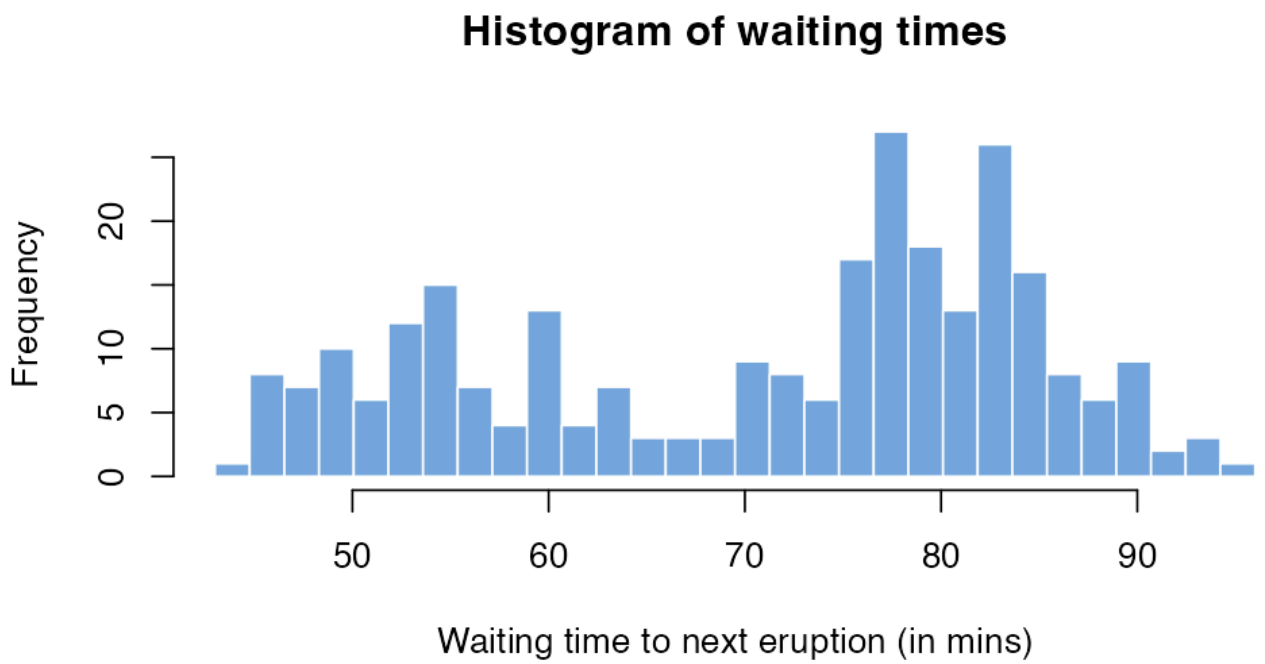
page_sidebar

Hello Shiny!

Number of bins:

1 30 50

1 6 11 16 21 26 31 36 41 46 50



page_sidebar

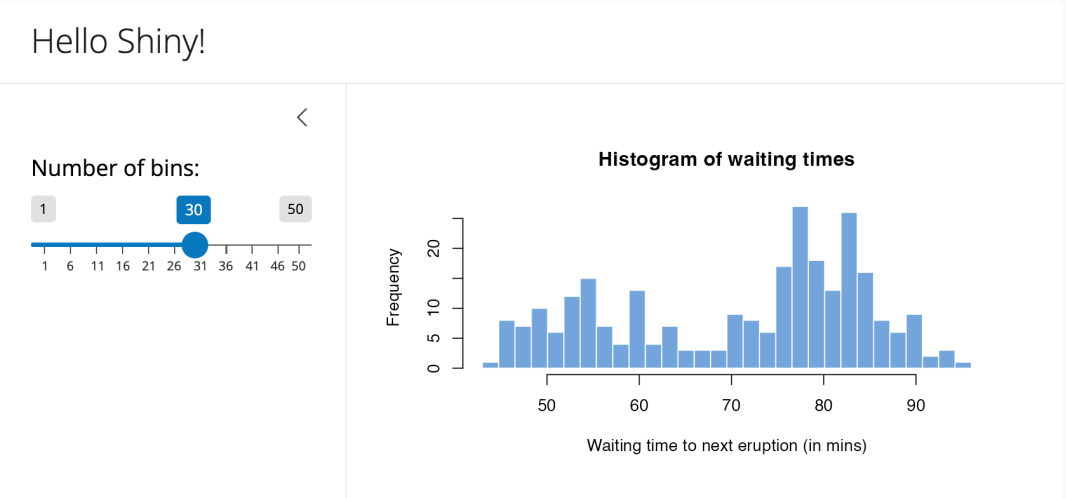
```
library(shiny)
library(bslib)

ui <- page_sidebar(

  title = "Hello Shiny!",

  sidebar = sidebar(
    sliderInput(
      "bins", label = "Number of bins:",
      min = 1, value = 30, max = 50
    )
  ),

  plotOutput("distPlot")
)
```



https://shiny.posit.co/r/articles/build/layout-guide/

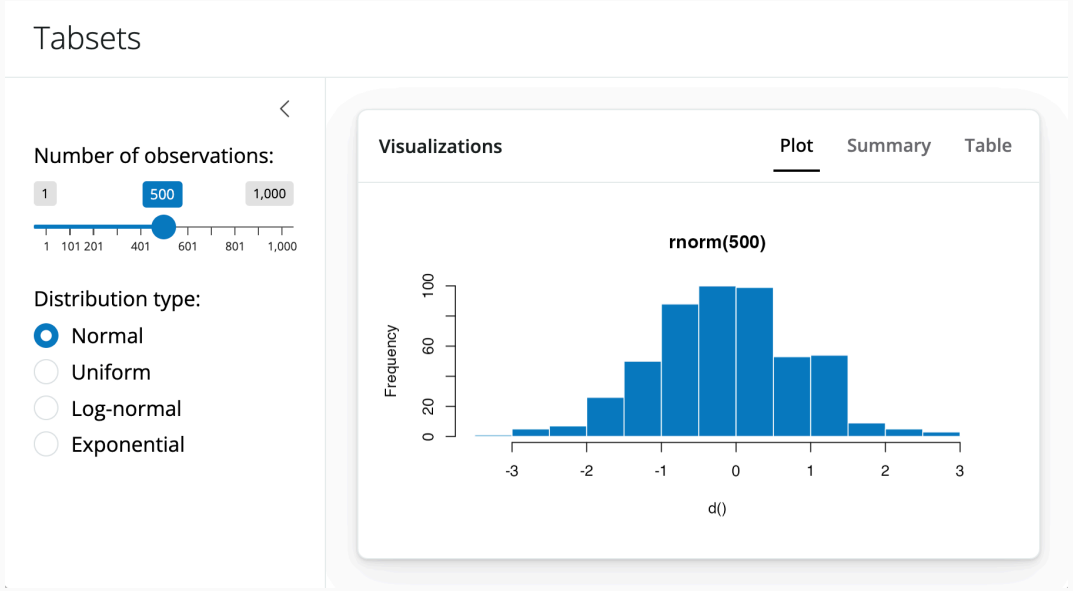
page_navbar


```
ui <- page_navbar(
  title = "My App",
  bg = "#2D89C8",
  inverse = TRUE,
  nav_panel(title = "One", p("First page content.")),
  nav_panel(title = "Two", p("Second page content.")),
  nav_panel(title = "Three", p("Third page content.")),
  nav_spacer(),
  nav_menu(
    title = "Links",
    align = "right",
    nav_item(tags$a("Posit", href = "https://posit.co")),
    nav_item(tags$a("Shiny", href = "https://shiny.posit.co"))
  )
)
```

navset card underline/pill/tab

```
library(shiny)
library(bslib)

ui <- page_sidebar(
  title = "Tabsets",
  sidebar = sidebar(
    ...
  ),
  navset_card_underline(
    title = "Visualizations",
    nav_panel("Plot", plotOutput("plot")),
    nav_panel("Summary", tableOutput("summary")),
    nav_panel("Table", tableOutput("table"))
  )
)
```



accordion

```
ui <- page_sidebar(
  title = "Penguins dashboard",
  sidebar = sidebar(
    accordion(
      accordion_panel(
        "Primary controls",
        varSelectInput(...)
      ),
      accordion_panel(
        "Other controls",
        "Other controls go here"
      )
    )
  ),
  accordion(
    accordion_panel("Bill Length", plotOutput("p1")),
    accordion_panel("Bill Depth", plotOutput("p2")),
    accordion_panel("Body Mass", plotOutput("p3"))
  )
)
```

https://shiny.posit.co/r/articles/build/layout-guide/

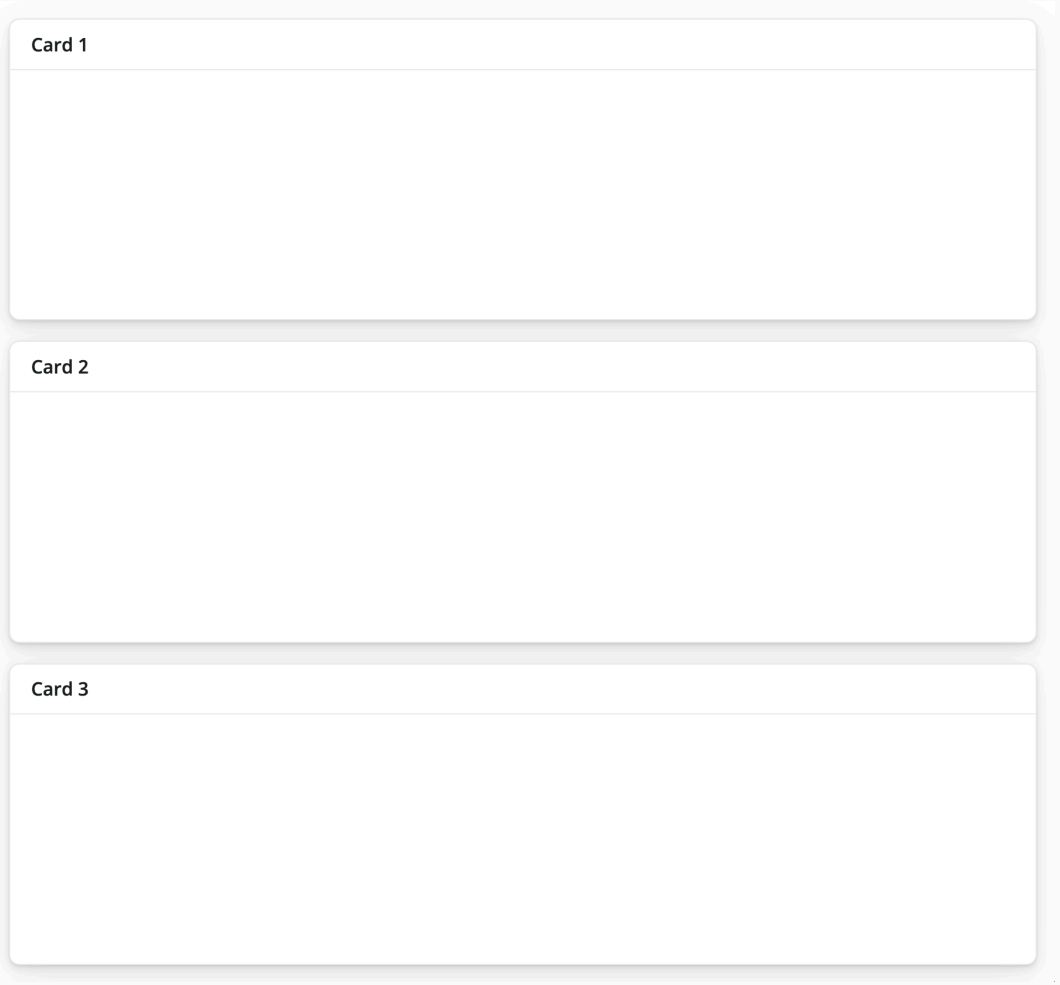
Multiples columnas: Layout por filas

`page_fillable()` permite crear una app sin barra lateral. Los componentes dentro de page_fillable() se ajustan automáticamente para ocupar el espacio vertical y horizontal disponible. Por defecto, Shiny coloca cada nuevo componente debajo del anterior, como si cada uno fuera una nueva fila.

```
ui <- page_fillable(

  card(card_header("Card 1")),
  card(card_header("Card 2")),
  card(card_header("Card 3"))

)
```

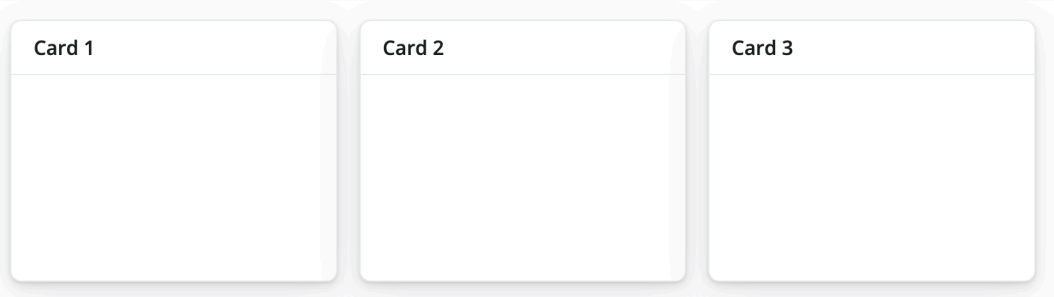


Multiples columnas: Layout por columnas

```
ui <- page_fillable(

  layout_columns(
      card(card_header("Card 1")),
      card(card_header("Card 2")),
      card(card_header("Card 3"))
  )

)
```

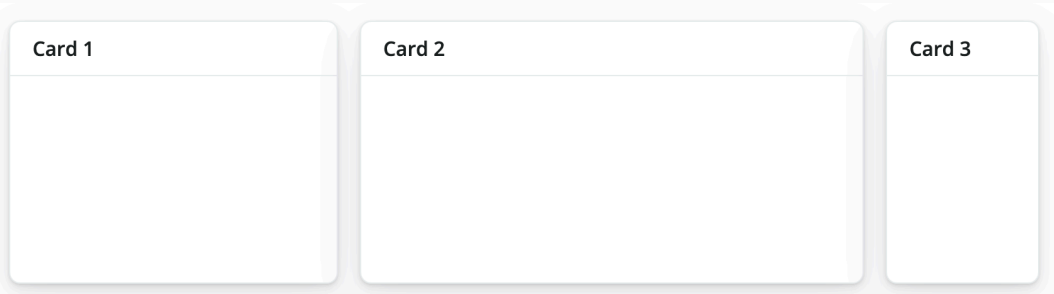


Multiples columnas: Layout por columnas (anchos)

```r
ui <- page_fillable(

  layout_columns(
     card(card_header("Card 1")),
     card(card_header("Card 2")),
     card(card_header("Card 3")),
     col_widths = c(4, 6, 2)
  )

)
```

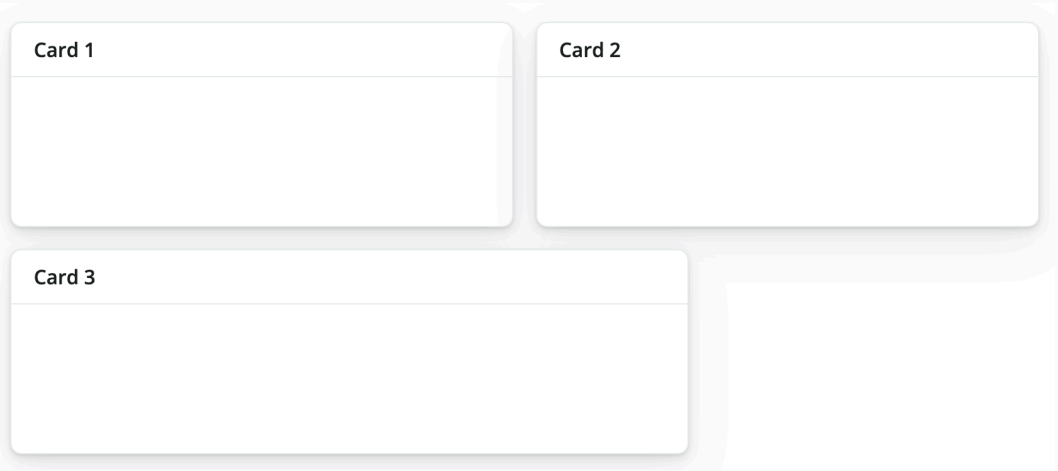


Multiples columnas: Layout por columnas (anchos 2)

```
ui <- page_fillable(

  layout_columns(
     card(card_header("Card 1")),
     card(card_header("Card 2")),
     card(card_header("Card 3")),
     col_widths = c(6, 6, 8)
  )

)
```



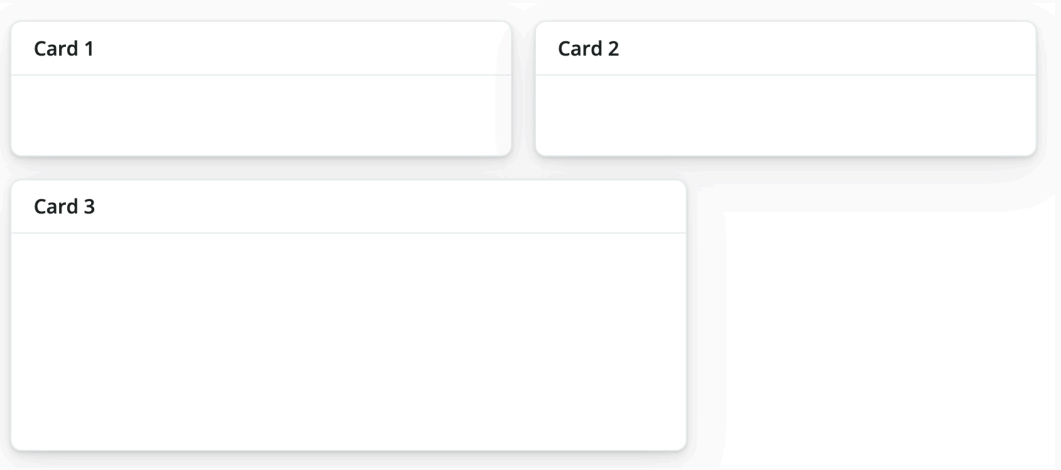
Si los elementos superan las 12 unidades de ancho, automáticamente pasan a una nueva fila.

Multiples columnas: Distintas altos por filas

```
ui <- page_fillable(

  layout_columns(
     card(card_header("Card 1")),
     card(card_header("Card 2")),
     card(card_header("Card 3")),
     col_widths = c(6, 6, 8),
     row_heights = c(1, 2)
  )

)
```

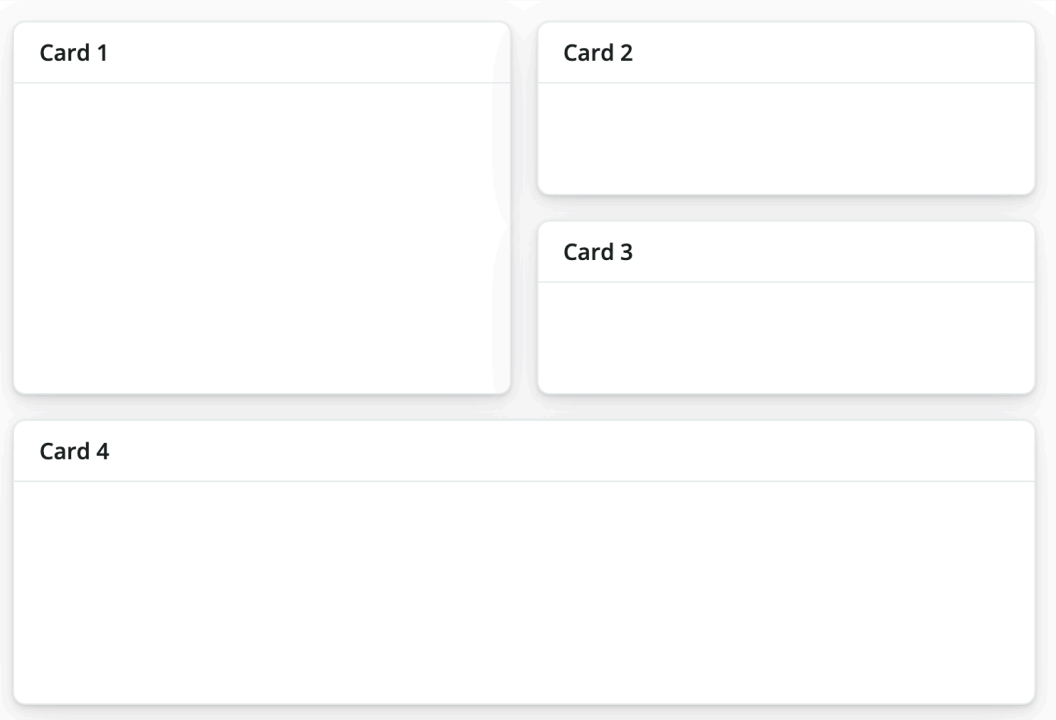


Si los elementos superan las 12 unidades de ancho, automáticamente pasan a una nueva fila.

Multiples columnas: Layout mixto

Se pueden combinar y anidarpara crear diseños personalizados basados en grillas.

```
ui <- page_fillable(
  layout_columns(
    card(card_header("Card 1")),
    layout_columns(
      card(card_header("Card 2")),
      card(card_header("Card 3")),
      col_widths = c(12, 12)
      )
  ),
  card(card_header("Card 4"))
)
```



Ejercicio: Distribuyendo outputs

Tomar la siguiente app e implementar un `navset_card_*`, `accordion` y un layout mixto.

```
# UI
library(shiny)
library(bslib)

datos <- rnorm(100)

ui <- page_sidebar(
  title = "Mi primer app",
  sidebar = sidebar(
    sliderInput("nrand", "Simulaciones", min = 50,
                max = 100, value = 70),
    selectInput("col", "Color", c("red", "blue", "black")),
    checkboxInput("punto", "Puntos:", value = FALSE)
    ),
  plotOutput("grafico"),
  plotOutput("grafico2"),
  tableOutput("tabla")
  )
```

```
# Server
server <- function(input, output) {
  output$grafico <- renderPlot({
    plot(
      head(datos, input$nrand),
      type =  ifelse(input$punto, "b", "l"),
      col = input$col
      )
  })

  output$grafico2 <- renderPlot({
    hist(head(datos, input$nrand), col = input$col)
  })

  output$tabla <- renderTable({
    x <- head(datos, input$nrand)
    data.frame(mean(x), sd(x), length(x))
  })

}

shinyApp(ui, server)
```

Ejercicio: Distribuyendo outputs (solución)

Solamente modificamos la `ui`. El resto de implementaciones son similares.

```
# UI
library(shiny)
library(bslib)

datos <- rnorm(100)

ui <- page_sidebar(
  title = "Mi primer app",
  sidebar = sidebar(
    sliderInput("nrand", "Simulaciones", min = 50,
                max = 100, value = 70),
    selectInput("col", "Color", c("red", "blue", "black")),
    checkboxInput("punto", "Puntos:", value = FALSE)
    ),
  navset_card_underline(
    title = "Resultados",
    nav_panel("Lineas", plotOutput("grafico")),
    nav_panel("Histograma", plotOutput("grafico2")),
    nav_panel("Tabla", tableOutput("tabla"))
    )
  )
```

```
# Server
server <- function(input, output) {
  output$grafico <- renderPlot({
    plot(
      head(datos, input$nrand),
      type =  ifelse(input$punto, "b", "l"),
      col = input$col
      )
  })

  output$grafico2 <- renderPlot({
    hist(head(datos, input$nrand), col = input$col)
  })

  output$tabla <- renderTable({
    x <- head(datos, input$nrand)
    data.frame(mean(x), sd(x), length(x))
  })

}

shinyApp(ui, server)
```

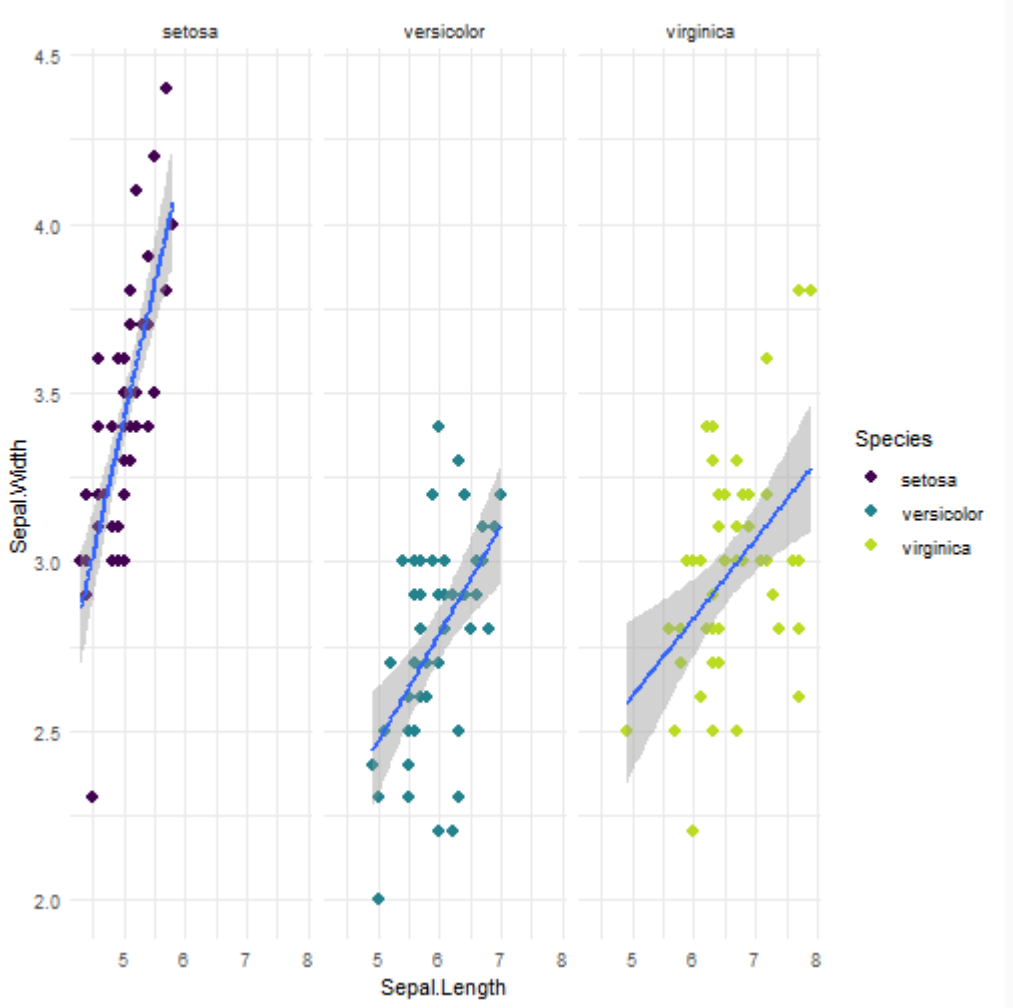
HTMLWidgets

Antes, un poco de {ggplot2}

```
library(ggplot2)

data(iris)

ggplot(iris, aes(Sepal.Length, Sepal.Width)) +
  geom_point(aes(color = Species), size = 2.5) +
  scale_color_viridis_d(end = .9) +
  geom_smooth(method = "lm") +
  facet_wrap(vars(Species)) +
  theme_minimal()
```



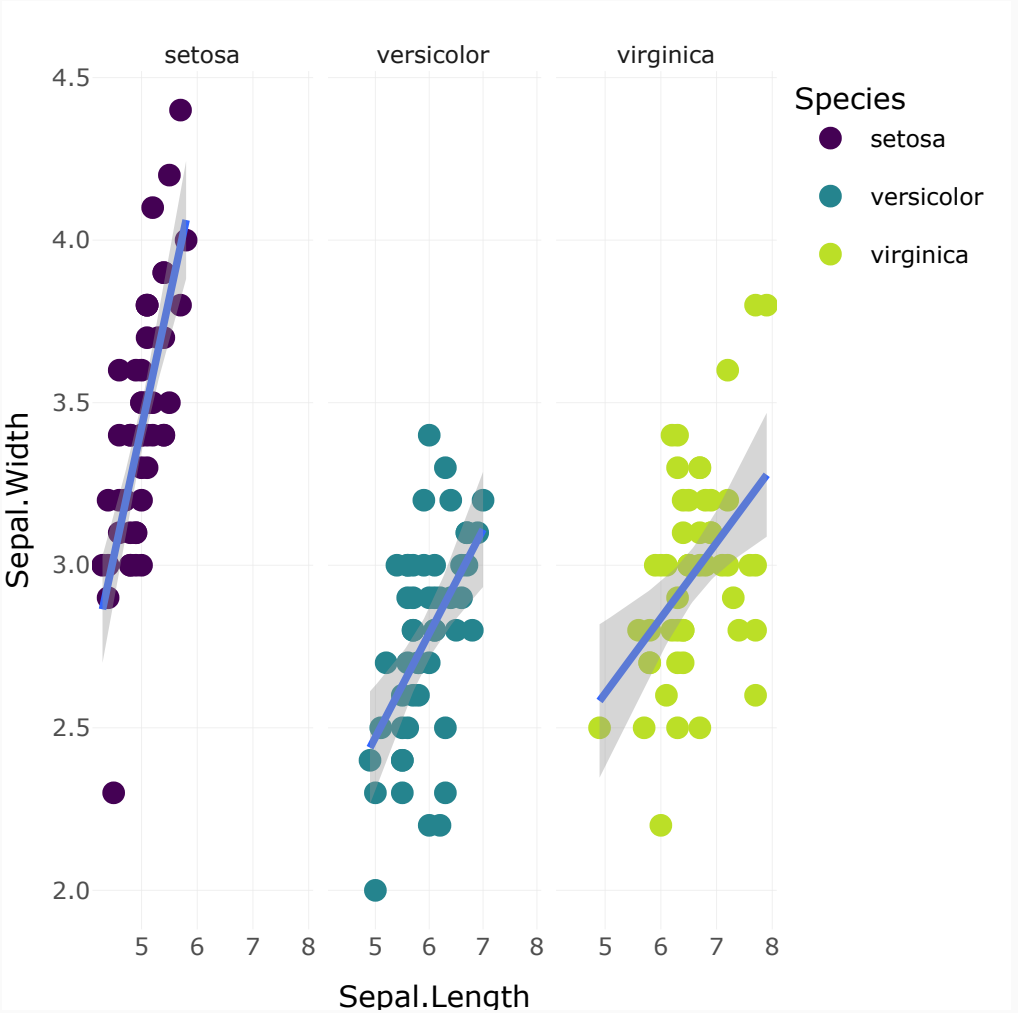
{plotly}

```
library(ggplot2)
library(plotly)

data(iris)

p <- ggplot(iris, aes(Sepal.Length, Sepal.Width)) +
  geom_point(aes(color = Species), size = 2.5) +
  scale_color_viridis_d(end = .9) +
  geom_smooth(method = "lm") +
  facet_wrap(vars(Species)) +
  theme_minimal()

ggplotly(p)
```



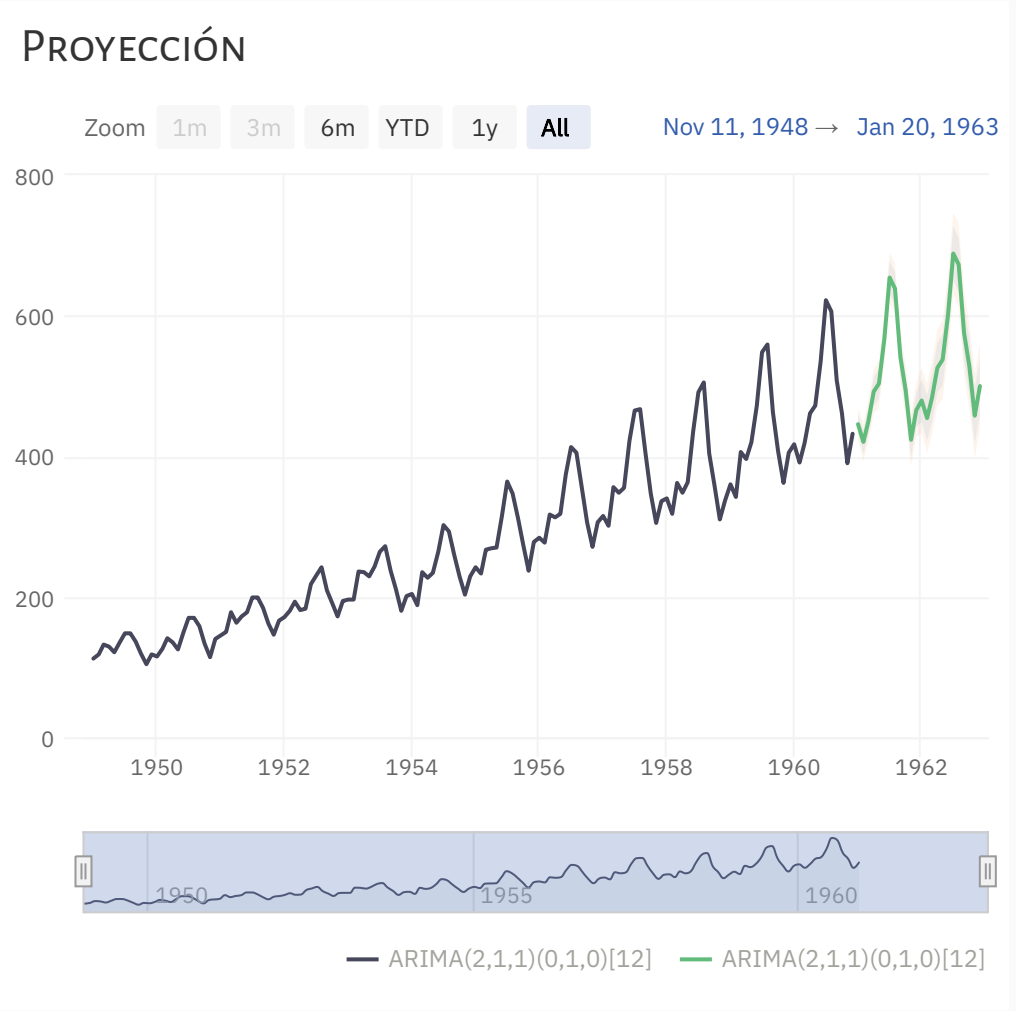
{highcharter}

```
library(highcharter)
library(forecast)

data("AirPassengers")

modelo <- forecast(auto.arima(AirPassengers))

hchart(modelo) |>
  hc_add_theme(hc_theme_hcrt()) |>
  hc_navigator(enabled = TRUE) |>
  hc_rangeSelector(enabled = TRUE) |>
  hc_title(text = "Proyección")
```



Unos datos

```
library(rvest)    # descargar datos de paginas web

url <- "https://www.sismologia.cl/sismicidad/catalogo/
```

```
datos <- read_html(url) |>
  html_table() |>
  dplyr::nth(2) |>
  janitor::clean_names() |>
  tidyr::separate(
    latitud_longitud,
    into = c("latitud", "longitud"),
    sep = " ", convert = TRUE
  )

datos
```

```
## # A tibble: 7 × 6
##   fecha_local_lugar            fecha_utc latitud longitud
##   <chr>                        <chr>       <dbl>    <dbl>
## 1 2025-11-17 19:24:1129 km al… 2025-11-…   -22.6    -68.
## 2 2025-11-17 11:59:0813 km al… 2025-11-…   -20.5    -69.
## 3 2025-11-17 11:02:4941 km al… 2025-11-…   -28.9    -71.
## 4 2025-11-17 10:29:0628 km al… 2025-11-…   -20.9    -70.
## 5 2025-11-17 09:10:5134 km al… 2025-11-…   -20.2    -69.
## 6 2025-11-17 07:24:2015 km al… 2025-11-…   -20.4    -69.
## 7 2025-11-16 21:37:1446 km al… 2025-11-…   -28.8    -71.
```

{DT}

```
library(DT)

datatable(datos)
```

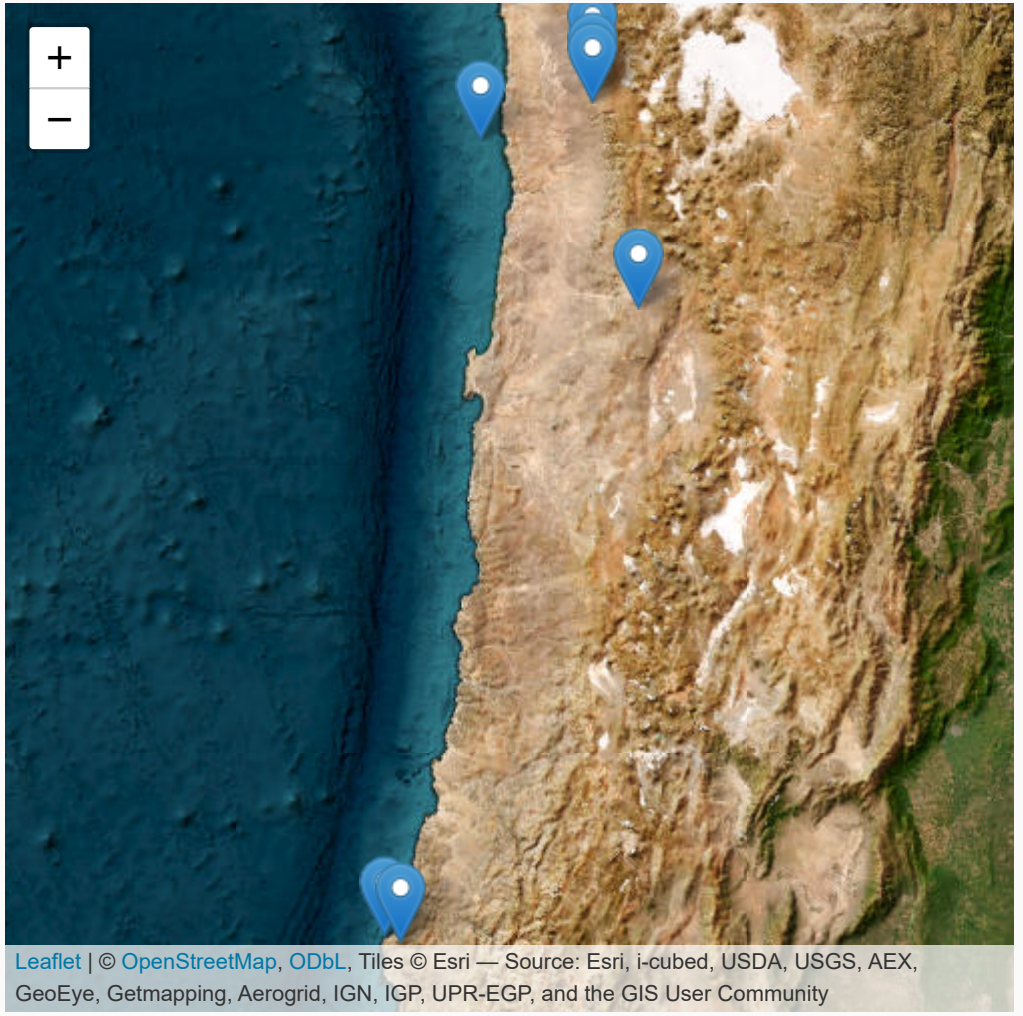
Show 10 entries Search:

	fecha_local_lugar	fecha_utc	latitud	longitud
1	2025-11-17 19:24:1129 km al SE de Calama	2025-11-17 22:24:11	-22.644	-68.715
2	2025-11-17 11:59:0813 km al SE de Pica	2025-11-17 14:59:08	-20.539	-69.215
3	2025-11-17 11:02:4941 km al N de Punta de Choros	2025-11-17 14:02:49	-28.901	-71.327
4	2025-11-17 10:29:0628 km al SO de Patache	2025-11-17 13:29:06	-20.922	-70.447
5	2025-11-17 09:10:5134 km al N de Pica	2025-11-17 12:10:51	-20.203	-69.217
6	2025-11-17 07:24:2015	2025-11-17	-20.423	-69.205

{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 un HTMLWidget a la app

A la aplicación de ejemplo del sidebar que se muestra a continuación, modificar para que sea un gráfico *interactivo* usando el plaque {plotly} y el ejemplo https://plotly.com/r/histograms/.

```
library(shiny)
library(bslib)

ui <- page_sidebar(
  title = "Hello Shiny!",
  sidebar = sidebar(
    sliderInput(
      "bins", label = "Number of bins:",
      min = 1, value = 30, max = 50
    )
  ),
  plotOutput("distPlot")
)
```

```
server <- function(input, output) {

  output$distPlot <- renderPlot({

    x    <- faithful[, 2]
    bins <- seq(min(x), max(x), length.out = input$bins

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

    })
}

shinyApp(ui, server)
```

Ejercicio: Agregar un HTMLWidget a la app (solución)

A la aplicación de ejemplo del sidebar que se muestra a continuación, modificar para que sea un gráfico *interactivo* usando el plaque {plotly}.

```
library(shiny)
library(bslib)
library(plotly)

ui <- page_sidebar(
  title = "Hello Shiny!",
  sidebar = sidebar(
    sliderInput(
      "bins", label = "Number of bins:",
      min = 1, value = 30, max = 50
    )
  ),
  plotlyOutput("distPlot")
)
```

```
server <- function(input, output) {

  output$distPlot <- renderPlotly({

    x    <- faithful[, 2]
    bins <- list(start = min(x),  end = max(x),  size =

    plot_ly(
      x = ~ x, type = "histogram",
      nbinsx = bins,
      marker = list(color = 'darkgray', line = list(co
    ) |>
      layout(
        xaxis = list(title = 'Waiting time to next eru
        title = 'Histogram of waiting times'
        )
}

shinyApp(ui, server)
```

Ejercicio: Agregar + HTMLWidgets a la app

A la siguiente aplicación implementar HTMLWidgets

```
# UI
library(shiny)
library(bslib)

datos <- rnorm(100)

ui <- page_sidebar(
  title = "Mi primer app",
  sidebar = sidebar(
    sliderInput("nrand", "Simulaciones", min = 50,
                max = 100, value = 70),
    selectInput("col", "Color", c("red", "blue", "black")),
    checkboxInput("punto", "Puntos:", value = FALSE)
    ),
  navset_card_underline(
    title = "Resultados",
    nav_panel("Lineas", plotOutput("grafico")),
    nav_panel("Histograma", plotOutput("grafico2")),
    nav_panel("Tabla", tableOutput("tabla"))
    )
  )
```

```
# Server
server <- function(input, output) {
  output$grafico <- renderPlot({
    plot(
      head(datos, input$nrand),
      type =  ifelse(input$punto, "b", "l"),
      col = input$col
      )
  })

  output$grafico2 <- renderPlot({
    hist(head(datos, input$nrand), col = input$col)
  })

  output$tabla <- renderTable({
    x <- head(datos, input$nrand)
    data.frame(mean(x), sd(x), length(x))
  })

}

shinyApp(ui, server)
```

Ejercicio: Agregar + HTMLWidgets a la app (solución)

Revisando https://jkunst.com/highcharter/articles/hchart.html:

```
# UI
library(shiny)
library(bslib)
library(highcharter)
library(DT)

datos <- rnorm(100)

ui <- page_sidebar(
  title = "Mi primer app",
  sidebar = sidebar(
    sliderInput("nrand", "Simulaciones", min = 50,
                max = 100, value = 70),
    selectInput("col", "Color", c("red", "blue", "black")),
    # checkboxInput("punto", "Puntos:", value = FALSE)
  ),
  navset_card_underline(
    title = "Resultados",
    nav_panel("Lineas", highchartOutput("grafico")),
    nav_panel("Histograma", highchartOutput("grafico2")),
    nav_panel("Tabla", DTOutput("tabla"))
  )
)
```

```
# Server
server <- function(input, output) {
  output$grafico <- renderHighchart({
    hchart(
      ts(head(datos, input$nrand)),
      name = "Datos",
      color = input$col
      )
  })

  output$grafico2 <- renderHighchart({
    hchart(head(datos, input$nrand), color = input$col)
  })

  output$tabla <- renderDT({
    x <- head(datos, input$nrand)
    data.frame(mean(x), sd(x), length(x))
  })

}

shinyApp(ui, server)
```