



# USO DE R SHINY

Madin Rivera, Alberto.

Mar 30, 2024

# Índice

<b>INTERFAZ DE USUARIO</b>	<b>3</b>
1. LOS TRES COMPONENTES DE Shiny . . . . .	5
2. COMPONENTES DEL <code>ui = fluidPage()</code> . . . . .	6
3. EL SERVER . . . . .	7
4. AYUDA DE <code>sidebarLayout()</code> . . . . .	8
5. DOCUMENTACIÓN SOBRE Shiny . . . . .	12
6. HTML DENTRO DE Shiny . . . . .	14
7. PRUEBAS CON LOS TITULARES ESTILO HTML . . . . .	17
8. PRUEBA DE PÁRRAFOS Y ECUACIONES MATEMÁTICAS . . . . .	18
9. PRUEBA DE IMÁGEN EN Shiny . . . . .	20
<b>PRINCIPALES WIDGETS DE Shiny</b>	<b>22</b>
1. WIDGET BÁSICOS: EXPLORANDO LA ESTRUCTURA DE UNA APLICACIÓN Shiny . . . . .	24

# INTERFAZ DE USUARIO

Una interfaz de usuario en R **Shiny** es una forma poderosa y flexible de construir aplicaciones web interactivas directamente desde R. Con **Shiny**, puedes transformar un código R en aplicaciones web que los usuarios pueden explorar, interactuar y compartir fácilmente a través de un navegador web. La interfaz de usuario **Shiny** se basa en dos componentes fundamentales: **ui** y **server**.

### Componentes de una Aplicación Shiny:

#### 1. **ui (User Interface):**

- La parte **ui** de una aplicación **Shiny** es donde se define la interfaz de usuario. Aquí es donde especificas qué elementos visuales estarán presentes en la aplicación web y cómo se organizarán.
- Se puede incluir una variedad de elementos como títulos, paneles laterales, paneles principales, gráficos, tablas, controles de entrada (como botones, deslizadores, campos de texto, etc.) y más.
- La **ui** se define utilizando funciones de construcción de interfaz específicas de **Shiny**, que permiten crear una variedad de elementos tipo **HTML** y **CSS** directamente desde R.

#### 2. **server:**

- El componente **server** de una aplicación **Shiny** maneja la lógica detrás de la aplicación. Aquí es donde el código R que procesa los datos y responde a las acciones del usuario.
- Se puede definir las reacciones a los eventos del usuario, como hacer clic en un botón o cambiar un valor en un control de entrada.
- El código del **server** puede realizar cálculos, filtrar datos, generar gráficos dinámicos y realizar cualquier otra tarea necesaria para proporcionar una experiencia interactiva y dinámica al usuario.

#### 3. **ShinyApp():**

- **ShinyApp()** es una función que combina el componente **ui** y el componente **server** para crear una aplicación **Shiny** completa.
- Toma el objeto **ui** y la función **server** como argumentos y los une en una aplicación web interactiva lista para ser lanzada a producción.
- Una vez que se haya definido **ui** y **server**, se llama a **ShinyApp()** para crear una aplicación web y se puede ejecutar localmente o desplegar en un servidor para que otros puedan acceder a ella a través de internet.

Una interfaz de usuario en R **Shiny** permite crear aplicaciones web interactivas y de manera fácil y rápida, combinando la definición de la interfaz de usuario con la lógica del servidor para ofrecer una experiencia de usuario dinámica y personalizada.

## 1. LOS TRES COMPONENTES DE Shiny

Se aprenderá un ejemplo de los que proporciona Shiny. Creando las tres partes del componente de un Shiny:

1. `UI = ui()`
2. `Server = server()`
3. `APP = shinyApp()`

En este ejemplo, se explora los tres componentes fundamentales que conforman una aplicación en Shiny. Estos componentes son esenciales para cualquier aplicación Shiny y se utilizan para definir la interfaz de usuario, la lógica del servidor y para combinar ambos aspectos en una aplicación web interactiva. El componente UI, abreviatura de User Interface (Interfaz de Usuario), se encarga de definir cómo se verá y se organizará la aplicación desde el punto de vista del usuario. El componente Server, por otro lado, se ocupa de la lógica y el comportamiento de la aplicación, procesando los datos y respondiendo a las acciones del usuario. Finalmente, el componente App, abreviatura de Application (Aplicación), se utiliza para unir UI y Server, creando así la aplicación web completa. A través de este ejemplo, veremos cómo se crean y se combinan estos tres componentes básicos en una aplicación Shiny funcional.

```
library(shiny)

# Parte 1
ui = fluidPage()

# Parte 2
server = function(input, output){}

# Parte 3
shinyApp(ui = ui, server = server)
```

## 2. COMPONENTES DEL `ui = fluidPage()`

Empezamos con la interfaz de usuario, poniendo todos los datos. Empezando por el título (`titlePanel()`). Separando todos los parámetros por comas, porque se trata de una lista de parámetros, y las **listas** en R se separan por coma.

Después del título pondremos un `SidebarLayout()`<sup>1</sup>.

```
library(shiny)

# Parte 1
ui = fluidPage(
  # Título de la página
  titlePanel("Título de la Aplicación"),

  # Se agregar un sidebarLayout
  sidebarLayout(
    sidebarPanel("Panel Lateral"),
    mainPanel("Panel Principal")
  )
)

# Parte 2
server = function(input, output){
  # Aquí va el cuerpo de la función del server
}

# Parte 3
shinyApp(ui = ui, server = server)
```

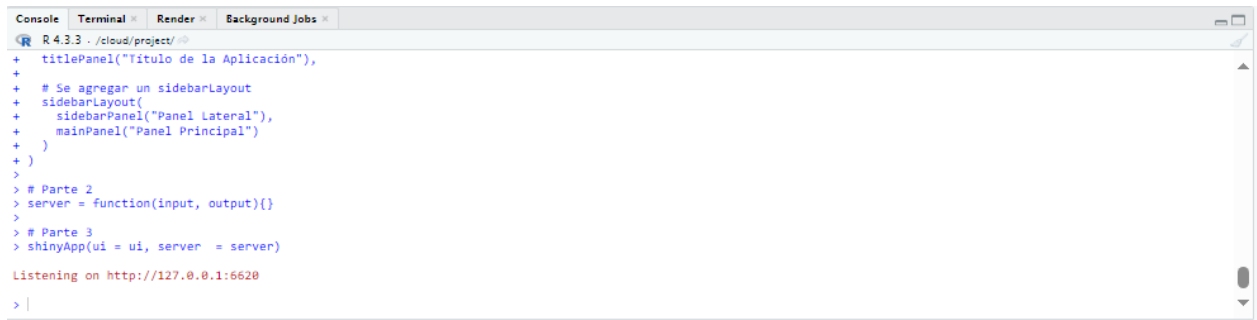
---

<sup>1</sup>Dentro de Shiny, un `SidebarLayout()` es un tipo de diseño de interfaz de usuario que permite organizar elementos en dos secciones principales: una barra lateral (sidebar) y un panel principal. Este diseño es comúnmente utilizado en aplicaciones web para presentar controles de entrada, opciones de configuración o cualquier otro tipo de contenido que se desee mostrar de forma separada del contenido principal.

### 3. EL SERVER

Cuando se utiliza la aplicación **Shiny**, se observa que en la consola lanza un mensaje diciendo **Listening on <http://127.0.0.1:3913>**.<sup>2</sup> Aquí hay una explicación más detallada:

1. **127.0.0.1**: Esta dirección IP es conocida como “localhost”. Se refiere a tu propia máquina. Cuando se ejecuta un servidor web en tu máquina y accedes a través de esta dirección, estás accediendo a tu propia máquina.
2. **Puerto**: Los servidores web, incluidos los servidores **Shiny**, utilizan puertos para comunicarse. Los números de puerto son como “**puertas**” que permiten que los datos entren y salgan de tu máquina. El número de puerto asignado puede variar y es seleccionado automáticamente por el sistema para evitar conflictos con otros servicios que se estén ejecutando en tu máquina.
3. **Listening on**: Esta es simplemente una notificación de que el servidor Shiny está listo para recibir solicitudes en la dirección y puerto especificados. Mientras veas este mensaje, significa que el servidor Shiny está en funcionamiento y listo para recibir conexiones.



```
R 4.3.3 > ./cloud/project/
+ titlePanel("Título de la Aplicación"),
+
+ # Se agregar un sidebarLayout
+ sidebarLayout(
+   sidebarPanel("Panel Lateral"),
+   mainPanel("Panel Principal")
+ )
+
>
> # Parte 2
> server = function(input, output){}
>
> # Parte 3
> shinyApp(ui = ui, server = server)

Listening on http://127.0.0.1:6628
> |
```

Figura 1: Salida de la consola

---

<sup>2</sup>Cuando ejecutas una aplicación Shiny en R, verás un mensaje que indica “Listening on <http://127.0.0.1:XXXX>”, donde “XXXX” es un número de puerto específico. Esto significa que la aplicación Shiny está corriendo localmente en tu máquina en una dirección IP local (127.0.0.1) en el puerto indicado.

## 4. AYUDA DE `sidebarLayout()`

La función `sidebarLayout()` en Shiny nos proporciona una manera conveniente de organizar la disposición de los elementos en nuestra aplicación. Al llamar a `help("sidebarLayout")`, podemos acceder a la **documentación** que informa sobre los distintos elementos que podemos utilizar en esta función. Además de los elementos que ingresamos directamente, como `sidebarPanel()` y `mainPanel()`, también podemos utilizar otros parámetros para personalizar la distribución de la maqueta. Por ejemplo, al establecer `position = "right"`, estamos indicando que queremos que el panel lateral se posicione a la derecha en lugar de su ubicación predeterminada a la izquierda. Esto nos ofrece flexibilidad para adaptar la disposición de nuestra aplicación a nuestras necesidades específicas y proporciona una experiencia de usuario más intuitiva y eficiente.

```
library(shiny)

# Parte 1
ui = fluidPage(
  # Título de la página
  titlePanel("Título de la Aplicación"),

  # Se agregar un sidebarLayout
  sidebarLayout(
    sidebarPanel("Panel Lateral"),
    mainPanel("Panel Principal"),

    # Panel lateral a la derecha
    position = "right"
  )
)

# Parte 2
server = function(input, output){
  # Aquí va el cuerpo de la función del server
}

# Parte 3
shinyApp(ui = ui, server = server)
```



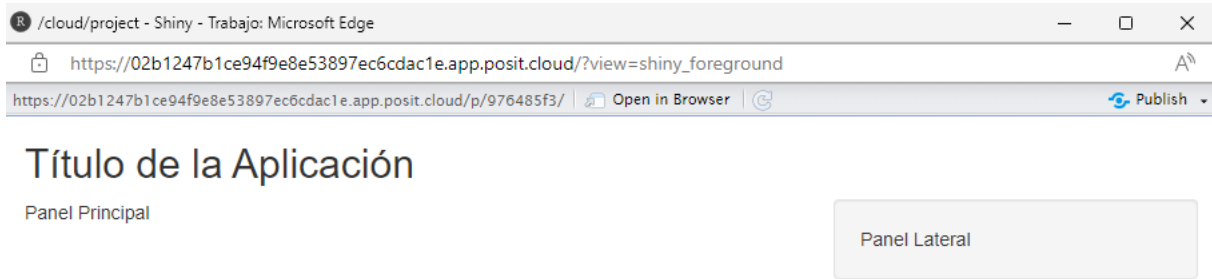


Figura 2: Aplicación con el panel lateral derecho

El parámetro `fluid` dentro de `sidebarLayout()` determina si se utilizará un diseño de ancho fijo o fluido en nuestra aplicación Shiny. Cuando `fluid = TRUE` (que es el valor predeterminado), el diseño se ajustará automáticamente al tamaño de la ventana del navegador, lo que significa que ocupará todo el ancho disponible. Por otro lado, si establecemos `fluid = FALSE`, el diseño será de ancho fijo y no se ajustará dinámicamente al tamaño de la ventana del navegador. En este ejemplo, hemos establecido `fluid = FALSE`, lo que significa que el diseño será de ancho fijo y mantendrá su tamaño independientemente del tamaño de la ventana del navegador.

```
library(shiny)

# Parte 1
ui = fluidPage(
  # Título de la página
  titlePanel("Título de la Aplicación"),

  # Se agregar un sidebarLayout
  sidebarLayout(
    sidebarPanel("Panel Lateral"),
    mainPanel("Panel Principal"),

    # Panel lateral a la derecha
    position = "right",

    # Fluid (por defecto está TRUE)
    fluid = FALSE
  )
)
```

```
# Parte 2
server = function(input, output){
  # Aquí va el cuerpo de la función del server
}

# Parte 3
shinyApp(ui = ui, server = server)
```

Además del parámetro `fluid`, también podemos especificar la anchura de los paneles lateral y principal (`sidebarPanel` y `mainPanel`) dentro de `sidebarLayout()`. Utilizando el parámetro `width`, podemos definir el ancho de cada panel en una escala de **1** a **12**, donde **12** representa el ancho máximo disponible. En este caso, hemos asignado un ancho de **5** unidades tanto al panel lateral como al panel principal, lo que significa que cada uno ocupará aproximadamente el **42%** del ancho total disponible. Esta especificación nos permite ajustar la distribución del espacio en nuestra aplicación Shiny y optimizar la disposición de los elementos según nuestras necesidades de diseño y visualización.

```
library(shiny)

# Parte 1
ui = fluidPage(
  # Título de la página
  titlePanel("Título de la Aplicación"),

  # Se agregar un sidebarLayout
  sidebarLayout(
    sidebarPanel("Panel Lateral", width = 5),
    mainPanel("Panel Principal", width = 5),

    # Panel lateral a la derecha
    position = "left",
  )
)

# Parte 2
server = function(input, output){
  # Aquí va el cuerpo de la función del server
}

# Parte 3
shinyApp(ui = ui, server = server)
```

## 5. DOCUMENTACIÓN SOBRE Shiny

Si nos dirigimos al enlace proporcionado <https://shiny.posit.co/r/reference/shiny/latest/>, podemos acceder a una documentación completa sobre los comandos disponibles en Shiny. Esta documentación enumera y explica detalladamente cada comando, lo que resulta útil si tenemos dudas sobre cómo funcionan o si necesitamos ejemplos de uso. Además, si deseamos personalizar la apariencia de nuestra aplicación utilizando estilos HTML, podemos acceder al siguiente [enlace](#) para construir formas más avanzadas dentro de Shiny utilizando HTML. En el código proporcionado, se muestra un ejemplo de cómo construir elementos HTML dentro de Shiny, como encabezados (`h1()`), párrafos (`p()`), y divisiones (`div()`). Además, se ilustra cómo incorporar elementos multimedia como archivos de audio utilizando etiquetas HTML5 y cómo suprimir los espacios en blanco entre las etiquetas utilizando la función `tags$span`.

```
tags$html(  
  tags$head(  
    tags$title('My first page')  
  ),  
  tags$body(  
    h1('My first heading'),  
    p('My first paragraph, with some ', strong('bold'), ' text.'),  
    div(  
      id = 'myDiv', class = 'simpleDiv',  
      'Here is a div with some attributes.'  
    )  
  )  
)  
  
# html5 <audio> with boolean control attribute  
# https://www.w3.org/TR/html5/infrastructure.html#sec-boolean-attributes  
tags$audio(  
  controls = NA,  
  tags$source(  
    src = "myfile.wav",  
    type = "audio/wav"  
  )  
)  
  
# suppress the whitespace between tags  
tags$span(  
  tags$strong("I'm strong", .noWS="outside")  
)
```

Dentro de la interfaz de usuario de este fragmento de código de R Shiny, estamos explorando el uso del formato **Markdown** para la creación de contenido dinámico. En la función `ui()`, se ha incorporado un ejemplo de **Markdown** que incluye un título principal, un párrafo de texto, una lista desordenada y un enlace. Este script **Markdown** se renderiza directamente en la aplicación **Shiny**, lo que permite la inclusión de contenido formateado de manera más flexible y legible. Además, hemos agregado una imagen que muestra el resultado de este **Markdown** dentro de la aplicación Shiny. A continuación, se proporciona un título para la imagen:

```
library(shiny)

# Parte 1
ui = fluidPage(
  markdown("
    # Markdown Example

    This is a markdown paragraph, and will be contained within a `

` tag
    in the UI.

    The following is an unordered list, which will be
    represented in the UI as
    a `

` with `- ` children:

    * a bullet
    * another

    [Links] (https://developer.mozilla.org/en-US/docs/Web/HTML/Element/a)
    work;
    so does emphasis.

    To see more of what's possible, check out
    [commonmark.org/help] (https://commonmark.org/help).
  ")
)

# Parte 2
server = function(input, output){}

# Parte 3
shinyApp(ui = ui, server = server)


```



Figura 3: Markdown dentro de Shiny App R

## 6. HTML DENTRO DE Shiny

Antes de entrar en el código, se está llevando a cabo una exploración sobre la integración de HTML dentro de Shiny. Dentro de la función `ui()`, se está configurando la apariencia y estructura de una aplicación Shiny. Se ha definido una página fluida (`fluidPage()`) que incluye un título principal para la aplicación. Además, se ha establecido un diseño de panel lateral y principal mediante `sidebarLayout()`, donde se especifica el contenido del panel lateral y principal. Posteriormente, se intenta integrar etiquetas HTML dentro de Shiny, utilizando titulares (`h1()` a `h6()`) como ejemplo. Estos titulares HTML se agregan directamente en el cuerpo de la función `ui()` para mostrar cómo se visualizarían en la aplicación Shiny. Este ejercicio busca demostrar la compatibilidad y flexibilidad de Shiny para trabajar con elementos HTML y cómo pueden ser utilizados para personalizar la apariencia y el formato de las aplicaciones.

```
library(shiny)

# Parte 1
ui = fluidPage(
  # Título de la página
  titlePanel("Título de la Aplicación"),

  # Se agregar un sidebarLayout
  sidebarLayout(
    sidebarPanel("Panel Lateral", width = 5),
    mainPanel("Panel Principal", width = 5),

    # Panel lateral a la derecha
  )
)
```

```

    position = "left",
  ),

# Etiqueta de HTML (Se puede hasta h6)
h1("Hola h1"),
h2("Hola h2"),
h3("Hola h3"),
h4("Hola h4"),
h5("Hola h5"),
h6("Hola h6")
)

# Parte 2
server = function(input, output){
  # Aquí va el cuerpo de la función del server
}

# Parte 3
shinyApp(ui = ui, server = server)

```

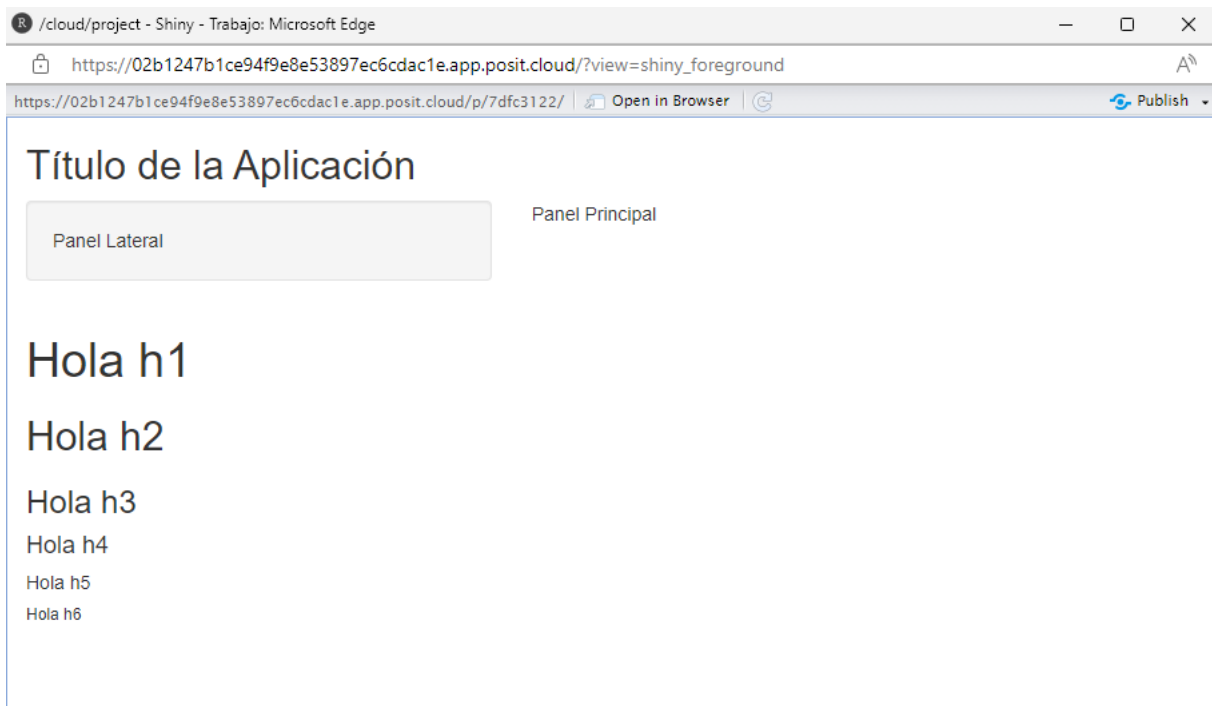


Figura 4: HTML Dentro de Shiny

Teniendo estos principios de HTML, podemos sobrescribir los títulos en los diferentes paneles de nuestra aplicación Shiny utilizando las etiquetas de título HTML. Por ejemplo, hemos utilizado la etiqueta `h1()` para reemplazar el título en el `titlePanel()`, lo que nos permite

personalizar aún pas la presentación del título principal de la aplicación. Además, también hemos también aplicado esta técnica al `sidebarPanel()` y al `mainPanel()`, donde hemos reemplazado los títulos predeterminados con etiquetas `h1()` de HTML. Esto nos ofrece una mayor flexibilidad en la presentación y personalización del aspecto y la estructura de nuestra aplicación Shiny.

```
library(shiny)

# Parte 1
ui = fluidPage(
  # Título de la página con h1 HTML
  titlePanel(h1("Título de la Aplicación (h1 de HTML)")),

  # Se agregar un sidebarLayout
  sidebarLayout(
    sidebarPanel("Panel Lateral", h1("Con HTML"), width = 5), # Con HTML
    mainPanel("Panel Principal", h1("Con HTML"), width = 5), # Con HTML

    # Panel lateral a la derecha
    position = "left",
  ),
)

# Parte 2
server = function(input, output){
  # Aquí va el cuerpo de la función del server
}

# Parte 3
shinyApp(ui = ui, server = server)
```



Figura 5: Entrada de títulos con estilo HTML



## 7. PRUEBAS CON LOS TITULARES ESTILO HTML

En esta parte, llevamos a cabo pruebas relacionadas con los titulares HTML dentro de la interfaz de usuario de Shiny. Dentro de la función `ui()`, se ha configurado un diseño fluido que consta de dos paneles: un panel lateral y un panel principal. En el panel lateral, hemos incluido ejemplos de titulares HTML que van desde nivel 1 hasta nivel 6 (`h1()` a `h6()`). Estos titulares se utilizan para estructurar y dar énfasis a diferentes secciones de contenido en una página web. En el panel principal, también hemos incluido ejemplos de titulares HTML para demostrar cómo se ven en diferentes niveles de jerarquía. Exploraremos cómo estos titulares contribuyen a organizar y mejorar la legibilidad del contenido en nuestras aplicaciones de Shiny.

```
library(shiny)

# Parte 1
ui = fluidPage(
  titlePanel("Pruebas con los titulares HTML"),

  sidebarLayout(
    sidebarPanel(
      h1("Título nivel 1"),
      h2("Título nivel 1"),
      h3("Título nivel 1"),
      h4("Título nivel 1"),
      h5("Título nivel 1"),
      h6("Título nivel 1")
    ),
    mainPanel(
      h1("Título nivel 1"),
      h2("Título nivel 1"),
      h3("Título nivel 1"),
      h4("Título nivel 1"),
      h5("Título nivel 1"),
      h6("Título nivel 1")
    )
  )
)

# Parte 2
server = function(input, output){
  # Aquí va el cuerpo de la función del server
}

# Parte 3
shinyApp(ui = ui, server = server)
```

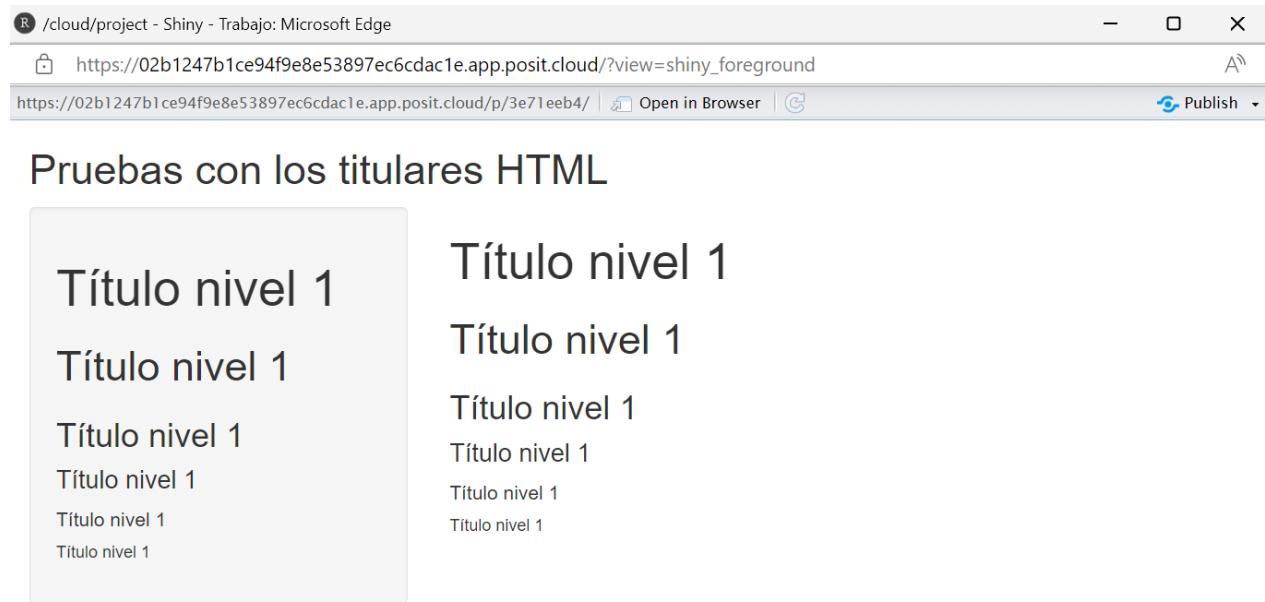


Figura 6: Pruebas con los titulares de diferentes tamaños de HTML

## 8. PRUEBA DE PÁRRAFOS Y ECUACIONES MATEMÁTICAS

En esta parte vamos a probar distintas opciones de texto basado en HTML que nos permite usar Shiny. Dentro del panel principal (`mainPanel()`) de la aplicación, se ha incorporado diversas funciones HTML para formatear el texto de manera interesante y dinámica. Por ejemplo, podemos incluir ecuaciones matemáticas utilizando LaTeX, como se muestra a continuación. Además, se destacan funciones como `br()`, que insertan saltos de línea, `p()`, que crea párrafos de texto, `strong()` para hacer el texto en negrita, `em()` para cursiva, `code()` para mostrar texto como código, y `div()` para segmentos de texto con estilos específicos. Además, se utilizan `span()` para aplicar estilos a grupos específicos de palabras. Observa cómo estas funciones se utilizan para mejorar la legibilidad y presentación del contexto textual en la interfaz de usuario de Shiny:

```
library(shiny)

# Parte 1
ui = fluidPage(
  titlePanel("Pruebas de párrafo con HTML y ecuaciones matemáticas"),

  sidebarLayout(
    sidebarPanel(),
    mainPanel(
      p("Aquí está una ecuación matemática  
utilizando LaTeX:"),
      withMathJax("$$E=mc^2$$"),
    )
  )
)
```

```
p("También puedes escribir ecuaciones en línea,  
  como esta: ",  
  withMathJax("$\\sum_{i=1}^n x_i$")),  
p("La función br() se utiliza para insertar un  
  salto de línea en el texto."),  
p("p crea un párrafo de texto"),  
p("un nuevo párrafo de texto con p()",  
  style = "font-family: 'times'; font-size: 16pt;"),  
strong("strong() pone el texto en negrita"),  
em("con em() ponemos el texto en cursiva"),  
br(),  
code("code() <- muestra el texto en código"),  
div("div() crea segmentos de texto con estilos similares",  
  style = "color:blue"),  
br(),  
p("span() hace lo mismo que un div,",  
  " pero funciona con ", span("Grupos de palabras",  
                               style = "color:red"))  
)  
)  
)  
  
# Parte 2  
server = function(input, output){  
  # Aquí va el cuerpo de la función del server  
}  
  
# Parte 3  
shinyApp(ui = ui, server = server)
```

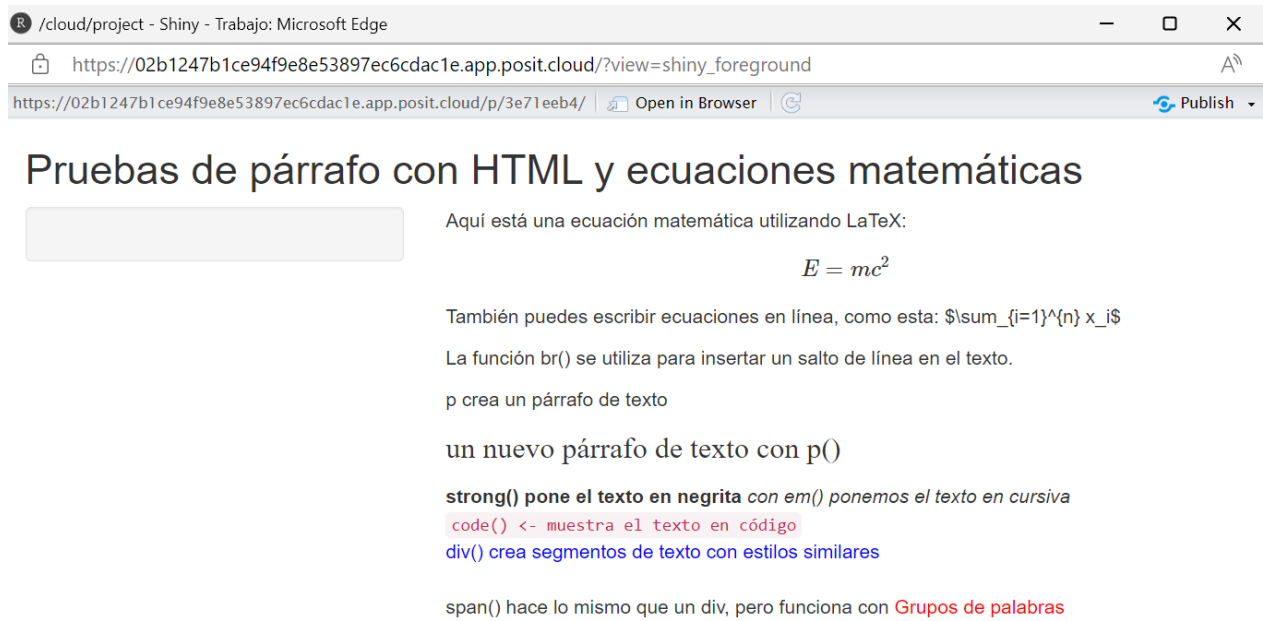


Figura 7: Prueba de párrafos y ecuaciones matemáticas

## 9. PRUEBA DE IMÁGEN EN Shiny

```
library(shiny)

# Parte 1
ui = fluidPage(
  titlePanel("Prueba de imagen"),

  sidebarLayout(
    sidebarPanel(

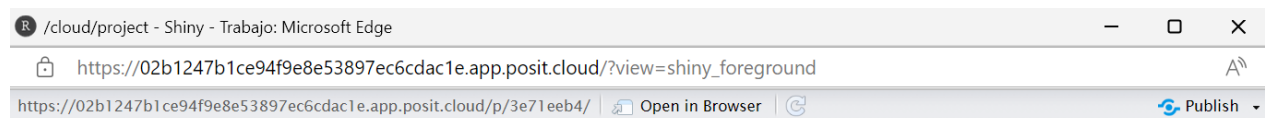
    ),
    mainPanel(
      # Aquí se maneja una imagen a través de un enlace
      # (el link debe ir junto)
      img(src = "https://upload.wikimedia.org/
wikipedia/en/4/4c/
Porter_Robinson_-_Cheerleader.jpg",

      # Estilo de la imagen
      style = "width: 300px; height: auto;
```

```
      display: block; margin-left: auto;
      margin-right: auto;")
    )
  )
)

# Parte 2
server = function(input, output){
  # Aquí va el cuerpo de la función del server
}

# Parte 3
shinyApp(ui = ui, server = server)
```



## Prueba de imagen



Figura 8: Prueba de Imagen dentro de Shiny

## PRINCIPALES WIDGETS DE Shiny

En **Shiny R**, un **widget** es un elemento interactivo de la interfaz de usuario que permite a los usuarios interactuar con la aplicación. Los widgets en **Shiny** son controles de entrada o salida que permiten a los usuarios ingresar datos, seleccionar opciones, ver resultados o realizar otras acciones dentro de la aplicación. Estos widgets pueden incluir botones, casillas de verificación, campos de texto, menús desplegables, deslizadores, gráficos, tablas y más.

Los widgets son esenciales para crear aplicaciones web interactivas y dinámicas en **Shiny**, ya que proporcionan la forma principal para que los usuarios interactúen con los datos y los resultados generados por la aplicación. Los usuarios pueden utilizar los widgets para enviar información al servidor de Shiny, lo que desencadena la ejecución de código R en el servidor para procesar los datos y actualizar la salida visible en la interfaz de usuario.

Los widgets en **Shiny R** son elementos de la interfaz de usuario que permiten la interacción entre el usuario y la aplicación, facilitando la entrada de datos, la selección de opciones y la visualización de resultados en tiempo real.

## 1. WIDGET BÁSICOS: EXPLORANDO LA ESTRUCTURA DE UNA APLICACIÓN Shiny

En este ejemplo, se está explorando la estructura básica de una aplicación Shiny, centrándonos en la disposición de los elementos en la interfaz de usuario. Utilizando la función `fluidPage()`, se crea una página fluida que se ajusta dinámicamente al tamaño de la ventana del navegador. Dentro de esta página, utilizando la función `titlePanel()` para establecer un título principal para nuestra aplicación Shiny. Luego, generamos una fila (`fluidRow()`) que contiene tres columnas de igual ancho (`column()`) dentro de nuestra aplicación. Cada columna contiene un segmento de texto que representa una parte de la interfaz de usuario. Este ejemplo nos proporciona una introducción básica a la estructura de una aplicación Shiny y cómo organizar elementos dentro de ella.

```
library(shiny)

ui = fluidPage(
  titlePanel("Widget básicos"),

  # Genera una fila dentro de nuestra aplicación
  fluidRow(
    column(4, "Segmento 1 de 3"),
    column(4, "Segmento 2 de 3"),
    column(4, "Segmento 3 de 3")
  )
)

server = function(input, output){}

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



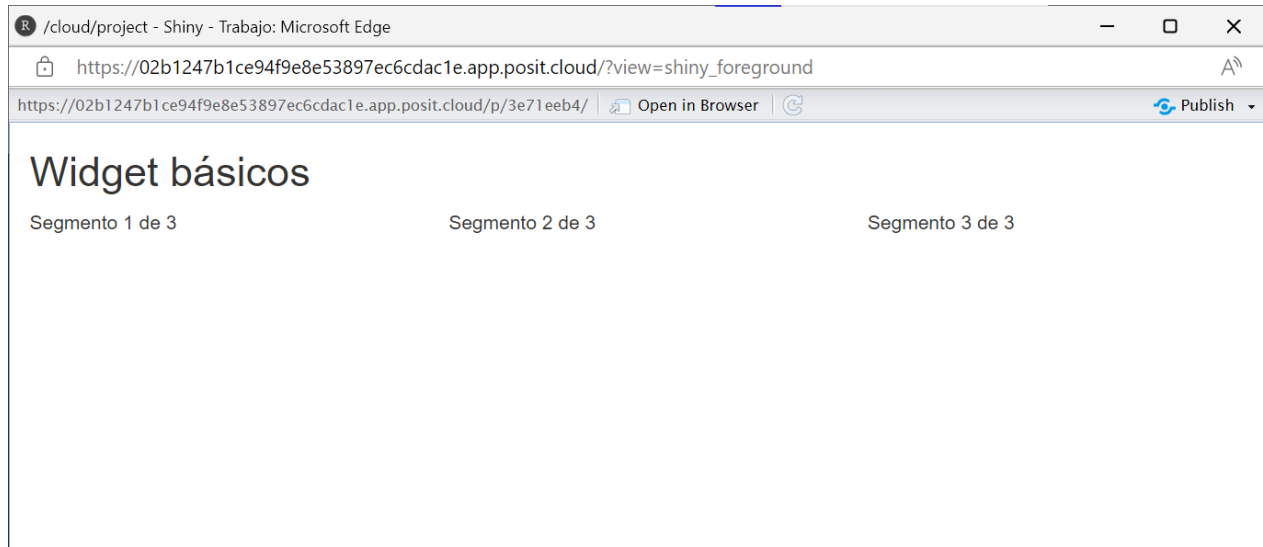


Figura 9: División por segmentos

En este código, estamos utilizando comandos HTML dentro de una aplicación Shiny para definir la estructura y el contenido de la interfaz de usuario. Dentro de la función `fluidPage()`, estamos utilizando la función `titlePanel()` para establecer un título principal para nuestra aplicación. Luego, creamos una fila (`fluidRow()`) que contiene tres columnas (`column()`) de igual ancho. En cada columna, utilizamos la función `h3()` para crear encabezados HTML de tercer nivel (`<h3>`) que representan distintas secciones de la aplicación. Específicamente, hemos creado encabezados para representar secciones de “Botones”, “Test” y “Resultados”. Al utilizar comandos HTML dentro de una aplicación Shiny, podemos personalizar la apariencia y la estructura de nuestra interfaz de usuario de manera más detallada y flexible.

```
library(shiny)

ui = fluidPage(
  titlePanel("Widget básicos"),

  # Genera una fila dentro de nuestra aplicación
  fluidRow(
    column(4,
      h3("Botones")),
    column(4,
      h3("Test")),
    column(4,
      h3("Resultados"))
  )
)

server = function(input, output){}
```

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

Dentro de este bloque de código de R se está construyendo una aplicación **Shiny** que presenta una variedad de **widgets** básicos para interactuar con el usuario. La aplicación consta de una página fluida que contiene una fila con cuatro columnas. Cada columna está diseñada para mostrar un tipo diferente de widget:

- **Botones:** Utilizando `actionButton()` y `submitButton()` para crear botones que los usuarios pueden hacer clic para realizar acciones o enviar formularios.
- **Grupos de casillas de verificación:** Usando la función `checkboxGroupInput()` para permitir a los usuarios seleccionar múltiples opciones de un conjunto de casillas de verificación.
- **Casilla de Verificación Individual:** Utilizando `checkboxInput()` para proporcionar a los usuarios una casilla de verificación individual que pueden marcar o desmarcar.
- **Fecha y Hora:** Utilizando la función `dateInput()` para permitir a los usuarios seleccionar una fecha y hora, con el valor inicial establecido en la fecha y hora actual.

Esta aplicación muestra cómo utilizar estos widgets básicos en **Shiny** para crear una interfaz de usuario interactiva y funcional.

```
library(shiny)

ui = fluidPage(
  titlePanel("Widget básicos"),
  fluidRow(
    column(3,
      h3("Botones"),
      actionButton("accion", "Acción"),
      br(),
      br(),
      submitButton("¡Ir!",
                    icon = icon("calendar"))),
    column(3,
      h3("checkGroup"),
      checkboxGroupInput("checkGroup", "Opciones:",
                         choices = c("Opción 1", "Opción 2", "Opción 3"),
                         selected = "Opción 1")),
    column(3,
      h3("checkBox"),
      checkboxInput("checkBox", "Opcion A",
                    value = TRUE)
    ),
    column(3,
      dateInput("fecha",
                h3("Fecha y Hora"),
                value = format(Sys.time(),
                              "%Y-%m-%d %H:%M:%S")))
  )
)
```

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

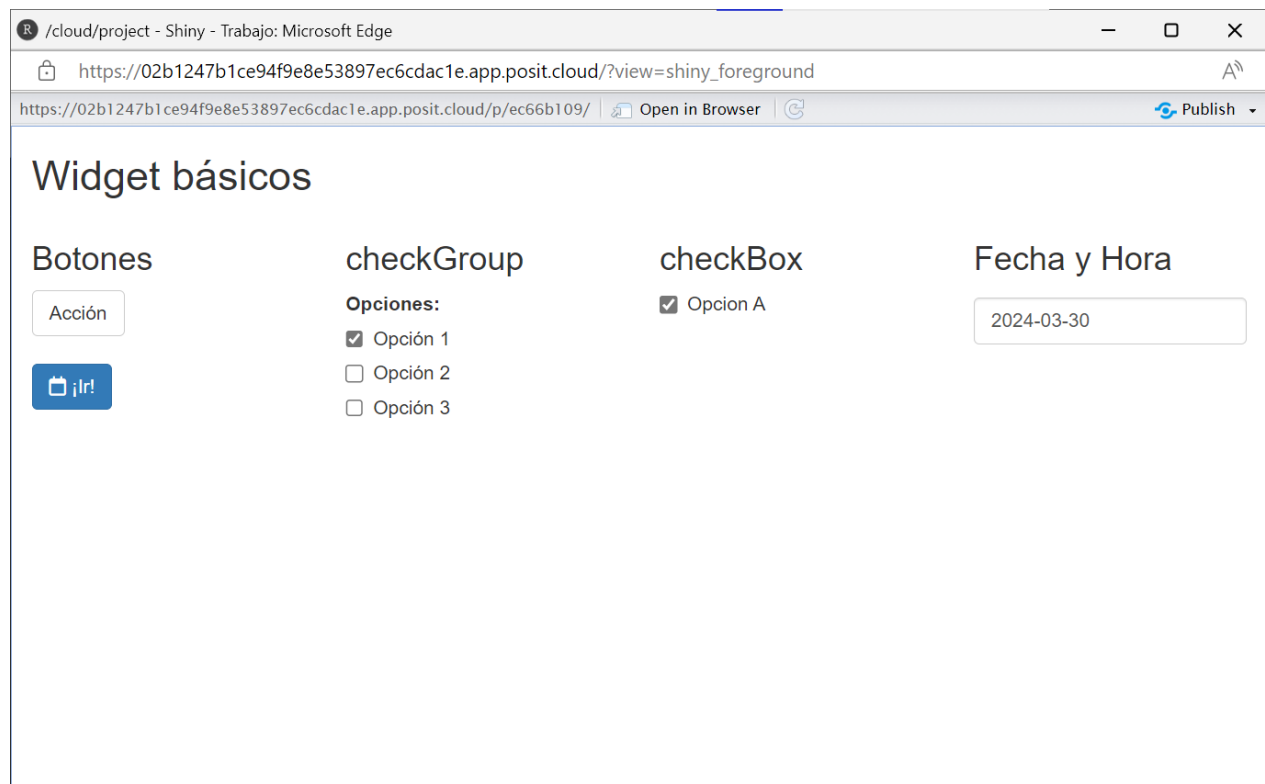


Figura 10: Widgets básicos para explorar en Shiny