



# MANUAL DE R SHINY

Madin Rivera, Alberto.

Mar 31, 2024



# Índice

<b>FUNCIONES BÁSICAS</b>	<b>3</b>
1. FUNCIONES BÁSICAS . . . . .	5
2. FUNCIONES AVANZADAS . . . . .	6
<b>INTERFAZ DE USUARIO CON Shiny</b>	<b>9</b>
1. LOS TRES COMPONENTES DE Shiny . . . . .	11
2. COMPONENTES DEL <code>ui = fluidPage()</code> . . . . .	13
3. EL SERVER . . . . .	15
4. PERSONALIZACIÓN DE LA DISPOSICIÓN CON <code>sidebarLayout()</code> . . . . .	16
5. EXPLORACIÓN AVANZADA DE DOCUMENTACIÓN Y PERSONALIZACIÓN HTML EN Shiny . . . . .	23
6. EXPLORACIÓN Y USO DE ETIQUETAS HTML EN Shiny . . . . .	28
7. OPTIMIZACIÓN DE LA JERARQUÍA VISUAL CON TITULARES HTML . . . . .	33
8. EXPLORANDO TEXTO DINÁMICO Y ECUACIONES MATEMÁTICAS EN Shiny . . . . .	36
9. INCORPORANDO IMÁGENES EN APLICACIONES Shiny . . . . .	39
<b>EXPLORANDO LOS WIDGETS PRINCIPALES EN Shiny</b>	<b>43</b>
1. EXPLORACIÓN INICIAL DE LA ESTRUCTURA BÁSICA EN Shiny: INTRODUCCIÓN A LOS WIDGETS . . . . .	45
2. EXPANSIÓN DE LA INTERACTIVIDAD: INCORPORACIÓN DE NUEVOS WIDGETS . . . . .	53
3. EXPANSIÓN INTERACTIVA: MEJORA DE LA INTERFAZ DE USUARIO CON WIDGETS AVANZADOS . . . . .	58



# FUNCIONES BÁSICAS

R, como lenguaje de programación, ha emergido como una herramienta fundamental en el ámbito de la ciencia de datos y el análisis estadístico. Su popularidad y adopción generalizada se deben a su flexibilidad, potencia y la amplia gama de paquetes disponibles que facilitan tareas específicas en el procesamiento y análisis de datos. Una de las características más destacadas de R es su extensa comunidad de usuarios y desarrolladores, que contribuyen constantemente con nuevos paquetes, funciones y recursos que enriquecen su ecosistema. Esta comunidad activa y colaborativa ha permitido que R se convierta en un estándar de facto en el campo de la ciencia de datos, siendo utilizado por investigadores, profesionales de la industria y académicos en una variedad de disciplinas, desde la biología hasta la economía.

Para aquellos involucrados en el análisis de datos, comprender y dominar las funciones de R es esencial para lograr resultados precisos y significativos. Las funciones en R son herramientas poderosas que permiten automatizar tareas repetitivas, encapsular algoritmos complejos y modularizar el código para una fácil reutilización. Desde funciones básicas para manipulación de datos hasta algoritmos avanzados de aprendizaje automático, R ofrece una amplia variedad de funciones predefinidas en su base, mientras que la comunidad continúa desarrollando y compartiendo nuevas funciones a través de paquetes. Esto proporciona a los usuarios de R una ventaja significativa al abordar problemas complejos de análisis de datos y modelado estadístico, permitiéndoles explorar, visualizar y comprender mejor los datos en búsqueda de insights valiosos.

A continuación, se verán algunas de las principales funciones de R que son fundamentales para comprender su uso básico dentro del contexto de **Shiny**, un entorno de desarrollo de aplicaciones web interactivas.

# 1. FUNCIONES BÁSICAS

## 1. print()

La función `print()` se utiliza para imprimir en la consola el valor o los objetos especificados. Es una de las funciones más básicas en R y se utilizan comunmente para mostrar resultados.

```
# Ejemplo de uso de la función print()  
print("¡Hola, mundo!")
```

## 2. sum()

La función `sum()` se utiliza para calcular la suma de los elementos de un vector numérico.

```
# Ejemplo de uso de la función sum()  
vector = c(1, 2, 3, 4, 5)  
total = sum(vector)  
print(total)
```

## 3. mean()

La función `mean()` se utiliza para calcular la media (promedio) de los elementos en un vector numérico.

```
# Ejemplo de uso de la función mean()  
vector = c(1, 2, 3, 4, 5)  
promedio = mean(vector)  
print(promedio)
```

## 4. length()

La función `length()` se utiliza para obtener la longitud (número de elementos de un vector).

```
# Ejemplo de uso de la función length()  
vector = c(1, 2, 3, 4, 5)  
longitud = length(vector)  
print(longitud)
```

## 5. seq()

La función `seq()` se utiliza para generar secuencias de números.

```
# Ejemplo de uso de la función seq()  
secuencia = seq(1, 10, by = 2)  
# Genera una secuencia del 1 al 10 con incrementos de 2  
print(secuencia)
```

Estas son solo algunas de las funciones básicas en R que serán útiles al comenzar a trabajar con este lenguaje. A medida que uno se familiariza más con R, se puede explorar y utilizar una amplia variedad de funciones para manipular datos, realizar análisis estadísticos y crear visualizaciones.

## 2. FUNCIONES AVANZADAS

A continuación se presenta otra lista de funciones más avanzadas dentro de lo básico en R base.

### 6. `data.frame()`

La función `data.frame()` es utilizada para crear un data frame, que es una estructura de datos tabular similar a la hoja de cálculo, donde las columnas pueden ser diferentes tipos de datos (numéricos, caracteres, factores, etc.).

```
# Ejemplo de uso de la función data.frame()
datos = data.frame(
  Nombre = c("Juan", "María", "Pedro"),
  Edad = c(25, 30, 28),
  Altura = c(170, 165, 180)
)
print(datos)
```

### 7. `list()`

La función `list()` se utiliza para crear listas en R, que son estructuras de datos flexibles que pueden contener elementos de diferentes tipos.

```
# Ejemplo de uso de la función list()
mi_lista = list(nombre = "Juan", edad = 25, casado = FALSE)
print(mi_lista)
```

### 8. Bucles for

El bucle `for` se utiliza para iterar sobre una secuencia de valores y ejecutar un bloque de código repetidamente.

```
# Ejemplo de uso de un bucle for
for (i in 1:5) {
  print(paste("Iteración", i))
}
```

### 9. Funciones definidas por el usuario

Las funciones definidas por el usuario permiten escribir las propias funciones para realizar tareas específicas.

```
# Ejemplo de definición de una función
mi_funcion = function(x, y) {
  resultado = x^2 + y^2
  return(resultado)
}
print(mi_funcion(3, 4))
```

### 10. Shiny `fluidPage()`



En Shiny, la función `fluidPage()` se utiliza para crear una página fluida en la interfaz de usuario, donde los elementos se ajustan automáticamente al tamaño de la ventana del navegador.

```
# Ejemplo de uso de la función fluidPage() en Shiny
library(shiny)

ui = fluidPage(
  titlePanel("Mi primera aplicación Shiny"),
  sidebarLayout(
    sidebarPanel(
      # Aquí van los controles de entrada, si los hubiera
    ),
    mainPanel(
      # Aquí van los resultados o gráficos generados
    )
  )
)

server = function(input, output) {
  # Aquí va la lógica de la aplicación
}

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

Estas funciones representan solo una fracción del amplio abanico de herramientas que R pone a disposición de los analistas de datos y desarrolladores de aplicaciones interactivas. Además de las funciones mencionadas, R cuenta con una gran cantidad de paquetes y bibliotecas especializadas que abarcan desde técnicas estadísticas avanzadas hasta técnicas de visualización de datos de última generación. Por ejemplo, paquetes como `ggplot2` ofrecen capacidades gráficas de alta calidad para crear visualizaciones atractivas y comprensibles, mientras que paquetes como `dplyr` y `tidyr` proporcionan funciones robustas para el procesamiento y manipulación eficiente de datos.

Además, la comunidad de usuarios de R es muy activa y colaborativa, lo que significa que siempre hay una gran cantidad de recursos disponibles, como blogs, tutoriales y foros de discusiones que pueden ayudar a los usuarios a resolver problemas específicos y aprender nuevas técnicas. Esto hace que R sea una herramienta poderosa y versátil no solo para el análisis de datos, sino también para la creación de aplicaciones interactivas y el desarrollo de soluciones complejas en el ámbito de la ciencia de datos y la estadística.

Las funciones básicas de R son solo el comienzo de un viaje fascinante hacia la comprensión y el dominio de una de las herramientas más poderosas en el arsenal de cualquier profesional de datos.



# INTERFAZ DE USUARIO CON Shiny

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()`

Dentro de esta parte, 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)
```

El desarrollo de aplicaciones interactivas con **Shiny** implica la comprensión y la manipulación de sus tres componentes principales. En este contexto. A través del código de ejemplo proporcionado, se explorará cómo diseñar y combinar estos componentes para crear una aplicación **Shiny** desde cero, lo que proporcionará una base sólida para futuros desarrollos y exploraciones en el ámbito de la creación de aplicaciones web interactivas con **R** y **Shiny**.

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

Los componentes del UI en **Shiny**, representados por la función `fluidPage()`<sup>1</sup>, son fundamentales para el diseño y la organización de la interfaz de usuario en una aplicación. Comenzando definiendo la estructura general de la página, donde se incorporan todos los elementos visuales y funcionales. Entre estos componentes, el título, definido con `titlePanel()`, juega un papel crucial al proporcionar una etiqueta descriptiva que contextualiza el propósito de la aplicación. En la sintaxis de R, los parámetros dentro de las funciones se separan por comas, ya que representan una lista de configuraciones que se aplicarán al elemento. Después de establecer el título, podemos estructurar la disposición de la interfaz utilizando el `SidebarLayout()`. Este diseño particular es ampliamente utilizado en aplicaciones web **Shiny**, ya que permite una organización eficiente de los elementos en dos secciones principales: la barra lateral (sidebar) y el panel principal. Esta separación facilita la presentación de controles de entrada, opciones de configuración y otros tipos de contenido de manera clara y organizada, lo que mejora la experiencia del usuario y la accesibilidad de la aplicación.

---

<sup>1</sup>Si bien `fluidPage()` es una de las funciones más comunes para construir la interfaz de usuario en aplicaciones **Shiny**, no es la única opción disponible. Dependiendo de las necesidades específicas del proyecto, los desarrolladores pueden optar por utilizar otras funciones como `navbarPage()`, `sidebarLayout()`, `dashboardPage()`, entre otras. Estas funciones proporcionan diferentes diseños y estructuras para la interfaz de usuario, permitiendo una mayor flexibilidad en el diseño y la organización de la aplicación. Al explorar y experimentar con estas alternativas, los desarrolladores pueden personalizar aún más la apariencia y la funcionalidad de sus aplicaciones **Shiny**, adaptándolas a las necesidades y preferencias de los usuarios finales.

```
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
}

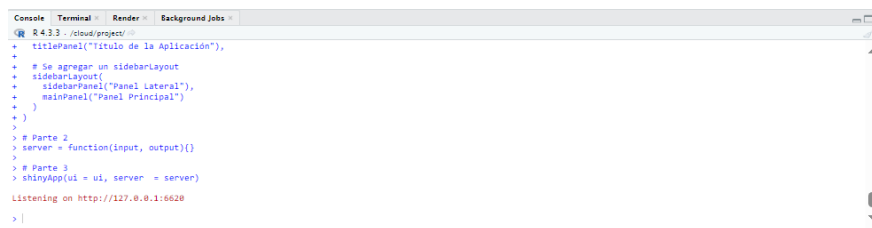
# Parte 3
shinyApp(ui = ui, server = server)
```



### 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.



```
Console | Terminal | Render | Background Jobs
R 4.3.3 . /cloud/project/
+ titlePanel("Título de la Aplicación"),
+
+ # Se agregan 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://127.0.0.1:6620
> |
```

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. PERSONALIZACIÓN DE LA DISPOSICIÓN CON `sidebarLayout()`

La función `sidebarLayout()` en Shiny ofrece una herramienta sólida para estructurar la distribución de los elementos dentro de una aplicación. Al consultar la documentación a través de la función `help("sidebarLayout")`, se puede explorar los diversos parámetros y elementos que se pueden emplear en esta función. Además de los componentes principales como `sidebarPanel()` y `mainPanel()`, se encuentran disponibles otros parámetros que permiten personalizar la disposición de la interfaz. Por ejemplo, al especificar `position = "right"`, se indica que se prefiere posicionar el panel lateral en el lado derecho en lugar de su posición predeterminada en el lado izquierdo. Esta capacidad de personalización proporciona una versatilidad que llega a permitir adaptar la disposición de la aplicación de acuerdo con los requisitos específicos, lo que se traduce en 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)
```

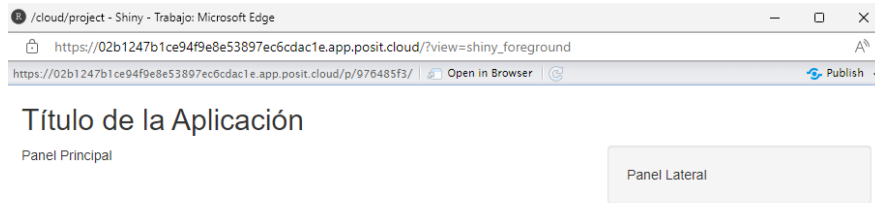


Figura 2: Aplicación con el panel lateral derecho

El código suministrado ejemplifica cómo implementar un `sidebarLayout()` con el panel lateral ubicado en la posición derecha. Este ejemplo ilustra la aplicación práctica de este parámetro para ajustar la disposición estándar y alcanzar un diseño más adecuado para ciertos escenarios de uso. Además, es importante destacar que 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 se establece `fluid = FALSE`, el diseño será de ancho fijo y no se ajustará dinámicamente al tamaño de la ventana del navegador. En este ejemplo, se establece `fluid = FALSE`, lo que significa que el diseño será de ancho fijo y mantendrá su tamaño independientemente del tamaño de la ventana del navegador<sup>3</sup>.

---

<sup>3</sup>Para obtener más información sobre las opciones y parámetros disponibles de `fluidPage()` en Shiny, se recomienda visitar la documentación oficial de Shiny en: <https://www.rdocumentation.org/packages/shiny/versions/1.8.1/topics/fluidPage>

```
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 =`, que determina si el diseño será de ancho fijo o fluido, se tiene la capacidad de especificar la anchura de los paneles lateral y principal (`sidebarPanel()`, `mainPanel()`) dentro de `sidebarLayout()`. Al utilizar el parámetro `width`, se puede definir el ancho de cada panel en una escala de 1 a 12, donde 12 representa el ancho máximo disponible. En este caso, se ha asignado un ancho de 5 unidades tanto al panel lateral como al panel principal, lo que significa que cada uno ocupará aproximadamente 42 por ciento del ancho total disponible. Esta capacidad de especificación de ancho permite ajustar la distribución del espacio de nuestra aplicación **Shiny** y optimizar la disposición de los elementos según las necesidades de diseño y visualización<sup>4</sup>.

---

<sup>4</sup>Por ejemplo, si estamos desarrollando una aplicación que requiere una visualización intensiva de datos en el panel principal y controles de entrada en el panel lateral, asignar un ancho mayor al panel principal puede ser beneficioso para permitir una visualización más amplia de los datos. Por otro lado, si el panel lateral contiene controles o información secundaria que no requiere tanto espacio, asignarle un ancho menor puede ayudar a optimizar el uso del espacio en la interfaz de usuario. La capacidad de especificar el ancho de los paneles lateral y principal nos proporciona un mayor control sobre la disposición y la apariencia de nuestra aplicación **Shiny**, lo que contribuye a una experiencia de usuario más efectiva y atractiva.

```
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. EXPLARACIÓN AVANZADA DE DOCUMENTACIÓN Y PERSONALIZACIÓN HTML<sup>5</sup> EN Shiny

Al acceder al siguiente [enlace](#), se puede adentrar a una extensa documentación que abarca todos los comandos disponibles en Shiny. Esta fuente exhaustiva enumera y desglosa meticulosamente cada comando, brindando claridad y orientación sobre su funcionamiento y aplicaciones. Esta documentación se convierte en una herramienta invaluable cuando surge la necesidad de comprender a fondo un comando en particular o cuando se busca ejemplos prácticos de uso en aplicaciones reales. Además, si se aspira a personalizar la apariencia de la aplicación Shiny mediante estilos HTML, se puede recurrir al siguiente [enlace](#) para explorar técnicas avanzadas de construcción utilizando HTML dentro de Shiny. El código proporcionado en este contexto ejemplifica cómo integrar elementos HTML, tales como encabezados (`h1()`), párrafos (`p()`), y divisiones (`div()`), dentro de nuestra aplicación Shiny, ofreciendo así una visión práctica de cómo aprovechar al máximo las capacidades de personalización que ofrece HTML. Además, se demuestra cómo incorporar elementos multimedia, como archivos de audio, mediante etiquetas HTML5 y cómo controlar los espacios en blanco entre las etiquetas utilizando la función `tags$span`, lo que brinda un nivel adicional de refinamiento estético y funcionalidad a nuestras aplicaciones Shiny<sup>6</sup>.

---

<sup>5</sup>Es importante destacar que HTML (HyperText Markup Language) no es considerado un lenguaje de programación en el sentido tradicional. En cambio, HTML se clasifica como un lenguaje de marcado, diseñado para estructurar y presentar contenido en la web mediante etiquetas y elementos. A diferencia de los lenguajes de programación, como Python o R, HTML no cuenta con capacidades para realizar cálculos o ejecutar lógica de programación.

<sup>6</sup>En contraste, los lenguajes de programación como Python, R, Java, C++, entre otros, son sistemas formales diseñados para expresar algoritmos y realizar cálculos. Estos lenguajes permiten la creación de programas que pueden interactuar con el usuario, procesar datos, realizar operaciones matemáticas y mucho más. Por lo tanto, mientras que HTML se centra en la presentación y estructuración de contenido web, los lenguajes de programación tienen un alcance más amplio y están destinados a la creación de aplicaciones y sistemas más complejos.

```
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. **Markdown** es un lenguaje de marcado ligero que permite formatear el texto de manera sencilla y legible, utilizando una sintaxis intuitiva. 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 en comparación con el **HTML** tradicional. Esta capacidad de integrar **Markdown** en **Shiny** abre nuevas posibilidades para crear aplicaciones más atractivas y comprensibles para los usuarios finales.

Además de la inclusión de contenido **Markdown**, este fragmento de código también ilustra la capacidad de **Shiny** para incorporar imágenes dentro de la interfaz de usuario. La imagen proporcionada muestra el resultado del **Markdown** dentro de la aplicación **Shiny**, lo que permite a los desarrolladores visualizar cómo se presentará el contenido formateado a los usuarios finales. Al combinar la capacidad de **Markdown** con la capacidad de **Shiny** para renderizar contenido dinámico, los desarrolladores pueden crear aplicaciones web interactivas que no solo sean funcionales, sino también estéticamente agradables y fáciles de entender para los usuarios.

```
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. EXPLORACIÓN Y USO DE ETIQUETAS HTML EN Shiny

Antes de adentrarnos en el código, es importante destacar la exploración que estamos realizando sobre la integración de etiquetas HTML dentro de **Shiny**. En este contexto, estamos analizando cómo incorporar elementos HTML en la interfaz de usuario de una aplicación **Shiny** para personalizar su apariencia y estructura. Dentro de la función `ui()`, que se utiliza para definir la interfaz de usuario en **Shiny**, estamos configurando la apariencia y estructura de la aplicación. Se ha optado por una página fluida (`fluidPage()`) como base, la cual permite un diseño flexible y adaptable al tamaño de la ventana del navegador. Además, se ha utilizado la función `sidebarLayout()` para establecer un diseño de panel lateral y principal, donde se especifica el contenido de cada uno.

En el contexto de este ejercicio, estamos explorando la posibilidad de integrar etiquetas HTML dentro de **Shiny**, utilizando titulares (`h1()` a `h6()`) como ejemplo. Estas etiquetas HTML se agregan directamente en el cuerpo de la función `ui()` para mostrar cómo se visualizarían en la aplicación **Shiny**. Esta práctica demuestra la compatibilidad y flexibilidad de **Shiny** para trabajar con elementos HTML, lo que permite a los desarrolladores personalizar la apariencia y el formato de sus aplicaciones de manera más avanzada y detallada. Al incorporar etiquetas HTML en **Shiny**, los desarrolladores tienen la capacidad de crear aplicaciones web más sofisticadas y atractivas visualmente, lo que mejora la experiencia del usuario final y amplía las posibilidades de diseño y presentación de datos.

```
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)
```

La integración de principios básicos de HTML dentro de las aplicaciones Shiny abre un abanico de posibilidades para la personalización y mejora del diseño visual. Un aspecto fundamental radica en la capacidad de sobrescribir los títulos en los diferentes paneles de la aplicación utilizando etiquetas de título HTML, como `h1()`, `h2()`, `h3()`, entre otras. Esta técnica permite una adaptación más precisa y detallada del contenido, otorgando al desarrollador un mayor control sobre la presentación del contenido y la jerarquía visual de la información. Al reemplazar los títulos predeterminados de los paneles con etiquetas HTML, los desarrolladores pueden ajustar el tamaño, el estilo y otros atributos visuales de los títulos, lo que contribuye a una presentación más efectiva y estéticamente agradable de la aplicación Shiny.

Además de la personalización de los títulos, la integración de etiquetas HTML ofrece la posibilidad de enriquecer el contenido con otros elementos visuales y de estructuración. Por ejemplo,

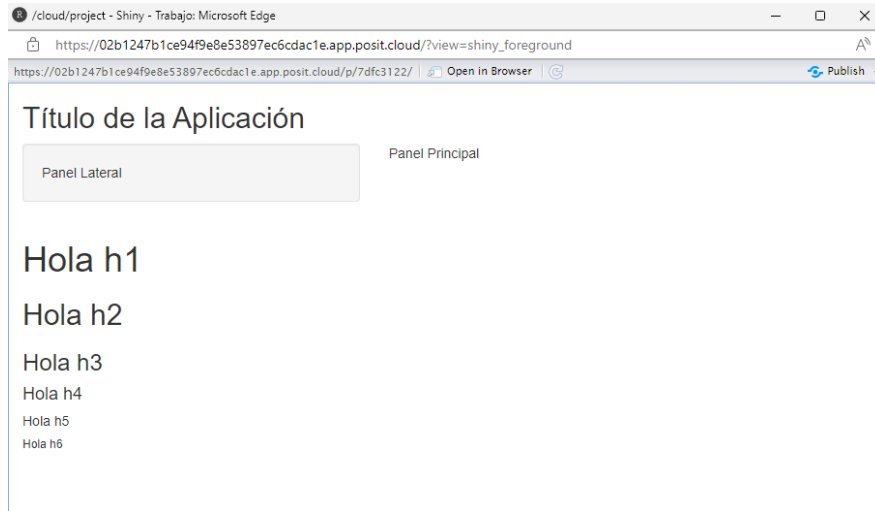


Figura 4: HTML Dentro de Shiny

es posible incorporar elementos como listas ordenadas y no ordenadas, tablas, imágenes y más, utilizando las etiquetas **HTML** correspondientes. Esta capacidad de integración de contenido **HTML** enriquece la experiencia del usuario al proporcionar una presentación más dinámica y atractiva de los datos y la información. Además, permite una mejor organización y presentación de la información, lo que facilita la comprensión y la navegación por parte del usuario. En este sentido, la combinación de Shiny con **HTML** amplía las posibilidades de diseño y presentación de aplicaciones web interactivas en el ámbito de la ciencia de datos y el análisis de datos.

La integración de etiquetas **HTML** dentro de las aplicaciones **Shiny** representa un avance significativo en la personalización y mejora del diseño visual. Al utilizar etiquetas **HTML** para sobrescribir los títulos y enriquecer el contenido, los desarrolladores pueden crear aplicaciones Shiny más atractivas, dinámicas y funcionales. Esta integración ofrece una mayor flexibilidad y control sobre la presentación del contenido, lo que contribuye a una experiencia del usuario más satisfactoria y efectiva. En última instancia, esta combinación de **Shiny** y **HTML** potencia el desarrollo de aplicaciones web interactivas en el ámbito de la ciencia de datos, facilitando la visualización y el análisis de datos de manera más efectiva y atractiva.



```
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. OPTIMIZACIÓN DE LA JERARQUÍA VISUAL CON TITULARES HTML

En esta sección, explora la influencia de los titulares HTML en la jerarquía visual de las aplicaciones Shiny. Dentro de la función `ui()`, hemos configurado un diseño fluido que consta de un panel lateral y un panel principal. En el panel lateral, hemos incorporado ejemplos de titulares HTML que abarcan desde el nivel 1 hasta el nivel 6 (`h1()` a `h6()`), cada uno representando un grado diferente de importancia y énfasis. Estos titulares se utilizan para estructurar y organizar el contenido, ayudando a los usuarios a identificar y comprender mejor la información presentada. En el panel principal, también hemos incluido ejemplos de titulares HTML para demostrar cómo se ven en diferentes niveles de jerarquía. A través de estas pruebas, investigamos cómo los titulares HTML pueden optimizar la jerarquía visual y mejorar la legibilidad del contenido en nuestras aplicaciones Shiny, proporcionando una experiencia de usuario más intuitiva y agradable.

Además de su capacidad para estructurar el contenido, los titulares HTML ofrecen la ventaja adicional de mejorar la accesibilidad y la indexación del contenido por parte de los motores de búsqueda. Al utilizar etiquetas de título HTML adecuadas y jerarquizadas correctamente, se mejora la capacidad de los motores de búsqueda para entender la estructura y el contexto del contenido, lo que puede resultar en una mejor clasificación en los resultados de búsqueda. Esto es especialmente relevante en aplicaciones web de ciencia de datos, donde la visibilidad y accesibilidad del contenido son aspectos críticos para su difusión y utilidad. Al emplear titulares HTML en nuestras aplicaciones Shiny, no solo mejoramos la organización y legibilidad del contenido, sino que también contribuimos a su accesibilidad y visibilidad en línea<sup>7</sup>.

---

<sup>7</sup>Es importante tener en cuenta que los titulares HTML son solo una de las muchas herramientas disponibles para optimizar la presentación y estructuración del contenido en las aplicaciones Shiny. Junto con otros elementos de diseño, como tablas, gráficos y estilos visuales, los titulares HTML pueden formar parte de una estrategia integral para mejorar la experiencia del usuario y maximizar el impacto comunicativo de las aplicaciones Shiny. Al combinar estas técnicas de diseño y presentación, los desarrolladores pueden crear aplicaciones Shiny más efectivas y atractivas, que cumplan con los estándares de usabilidad, accesibilidad y estética en el ámbito de la ciencia de datos y el análisis de datos.

```
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)
```

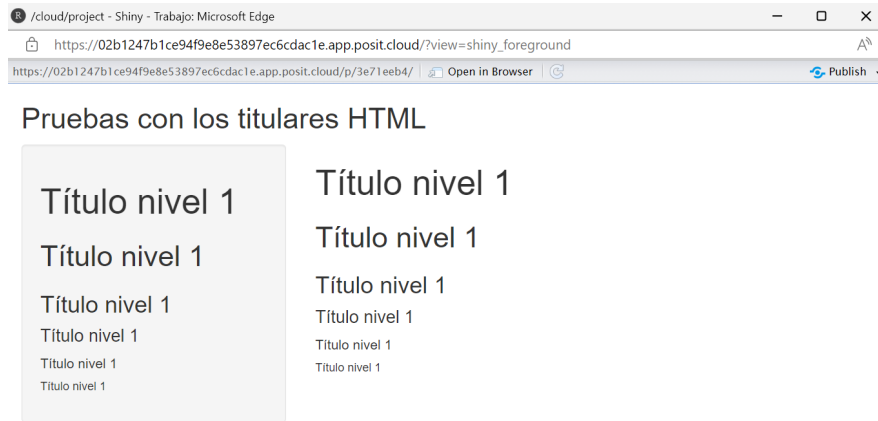


Figura 6: Pruebas con los titulares de diferentes tamaños de HTML

## 8. EXPLORANDO TEXTO DINÁMICO Y ECUACIONES MATEMÁTICAS EN Shiny

En esta sección, se explora más detallado las opciones de texto dinámico y ecuaciones matemáticas disponibles en Shiny a través de HTML. Dentro del panel principal (`mainPanel()`) de la aplicación, hemos incorporado diversas funciones HTML para formatear el texto de manera interesante y dinámica. Por ejemplo, podemos incluir ecuaciones matemáticas utilizando **LaTeX**<sup>8</sup>, como se muestra en los ejemplos. Además, destacamos el uso de funciones como `br()`, que inserta saltos de línea, `p()`, que crea párrafos de texto, `strong()` para resaltar texto en negrita, `em()` para aplicar cursiva, `code()` para mostrar texto como código, y `div()` para segmentos de texto con estilos específicos.

Además de estas funciones, se explora el uso de `span()` para aplicar estilos a grupos específicos de palabras. Observamos cómo estas funciones se utilizan para mejorar la legibilidad y presentación del contenido textual en la interfaz de usuario de Shiny, brindando a los desarrolladores una gama de opciones para personalizar la apariencia y el estilo del texto en sus aplicaciones.

Las opciones de texto dinámico y ecuaciones matemáticas son especialmente relevantes en aplicaciones de ciencia de datos, donde la claridad y la presentación adecuada de la información son cruciales para la comprensión y la toma de decisiones informadas. Al utilizar estas funciones en Shiny, los desarrolladores pueden crear aplicaciones más interactivas y atractivas, que mejoren la experiencia del usuario y faciliten la comunicación efectiva de resultados y hallazgos científicos.

```
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: "),
```

<sup>8</sup>En un contexto de ciencia de datos, la capacidad de presentar de manera efectiva resultados y análisis es fundamental para comunicar hallazgos y facilitar la comprensión de datos complejos. Por lo tanto, explorar opciones avanzadas de formato de texto, como las ofrecidas por HTML en Shiny, puede enriquecer significativamente la experiencia del usuario y mejorar la utilidad de las aplicaciones desarrolladas. Además, la inclusión de ecuaciones matemáticas utilizando LaTeX permite a los usuarios expresar conceptos y relaciones matemáticas de manera clara y precisa, lo que es crucial en áreas como la modelización estadística y el análisis cuantitativo.

```
    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)
```

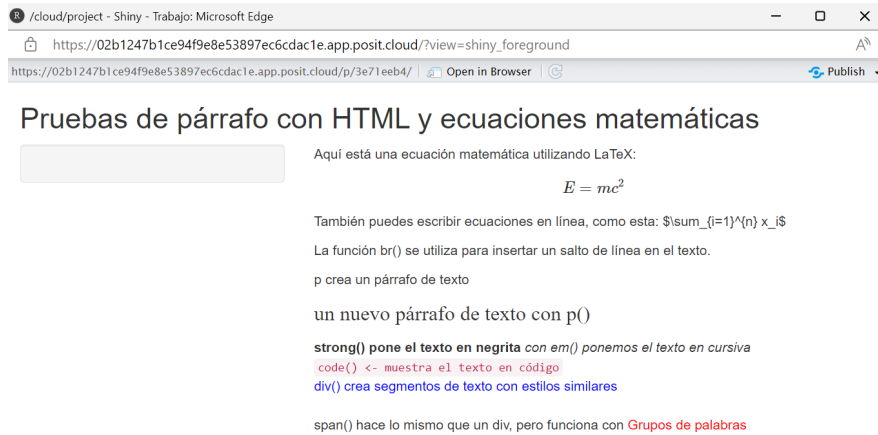


Figura 7: Prueba de párrafos y ecuaciones matemáticas



## 9. INCORPORANDO IMÁGENES EN APLICACIONES Shiny

En esta sección, explora cómo incorporar imágenes dentro de las aplicaciones Shiny. El código de R proporcionado demuestra cómo crear una aplicación Shiny que muestra una imagen en la interfaz de usuario. En la función `ui()`, se define la interfaz de usuario utilizando `fluidPage()`, donde se establece un panel lateral (`sidebarPanel()`) vacío y un panel principal (`mainPanel()`). Dentro del panel principal, se utiliza la función `img()`<sup>9</sup> para insertar una imagen en la aplicación. La imagen se carga desde un enlace proporcionado en el atributo `src =`. Además, se establece un estilo para la imagen utilizando el atributo `style =`, que ajusta el ancho, la altura y la alineación de la imagen.

```
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;",

      # Título de la imagen
      title = "Porter Robinson - Cheerleader")
    )
  )
)
```

---

<sup>9</sup>La función `img()` de Shiny ofrece una serie de parámetros adicionales que permiten un mayor control sobre la presentación y el comportamiento de las imágenes en las aplicaciones Shiny. Algunos de estos parámetros incluyen `alt`, que especifica un texto alternativo para la imagen que se muestra cuando la imagen no puede cargarse; `title`, que proporciona un título emergente cuando el usuario pasa el cursor sobre la imagen; y `class`, que permite asignar clases CSS a la imagen para aplicar estilos personalizados. Además, la función `img()` puede utilizarse en combinación con otras funciones y paquetes de Shiny para lograr funcionalidades avanzadas, como la generación dinámica de imágenes basadas en datos o la manipulación de imágenes cargadas por el usuario. Al explorar y experimentar con estos parámetros y posibilidades adicionales, los desarrolladores pueden crear aplicaciones Shiny aún más dinámicas y visualmente atractivas que satisfagan las necesidades específicas de sus usuarios y casos de uso.

```
)  
)  
  
# Parte 2  
server = function(input, output){  
  # Aquí va el cuerpo de la función del server  
}  
  
# Parte 3  
shinyApp(ui = ui, server = server)
```

La inclusión de imágenes en aplicaciones Shiny puede ser útil para presentar gráficos, diagramas, logotipos o cualquier otro elemento visual que complemente los datos o la información presentada en la aplicación. Esto puede mejorar significativamente la experiencia del usuario al proporcionar una representación visual de los datos o conceptos discutidos en la aplicación. Además, la capacidad de ajustar el estilo de la imagen permite a los desarrolladores personalizar su apariencia para que se integren perfectamente con el diseño general de la aplicación y mejoren su estética visual.

El ejemplo proporcionado ilustra cómo una imagen puede ser fácilmente integrada en una aplicación Shiny y cómo se puede controlar su estilo para lograr el aspecto deseado. Esta funcionalidad puede ser especialmente útil en aplicaciones que requieren la visualización de contenido visual o gráficos para comunicar eficazmente información a los usuarios. Con la capacidad de incorporar imágenes, Shiny ofrece a los desarrolladores una herramienta poderosa para crear aplicaciones interactivas y visualmente atractivas que mejoren la experiencia del usuario.

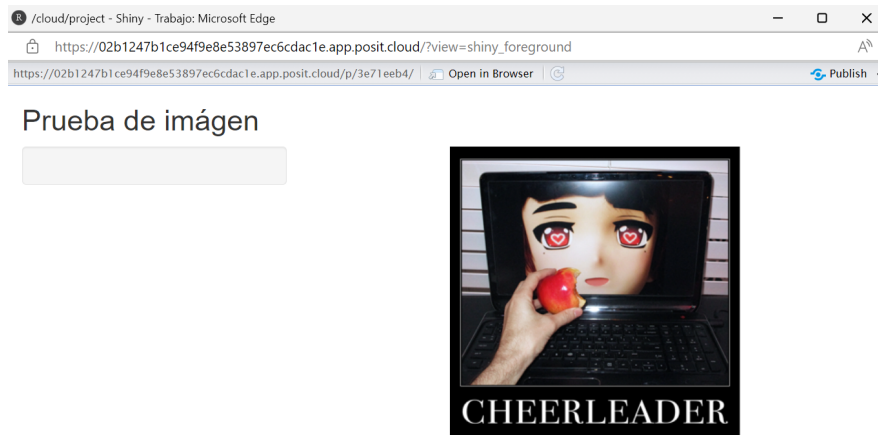


Figura 8: Prueba de Imagen dentro de Shiny

Durante esta sección dedicada a la interfaz de usuario con **Shiny**, exploramos minuciosamente los componentes esenciales y las estrategias avanzadas para optimizar la experiencia del usuario en aplicaciones web desarrolladas con este framework en R. En primer lugar, nos sumergimos en la comprensión de los tres pilares fundamentales de Shiny: la interfaz de usuario (**ui**), el servidor (**server**), y la función **shinyApp()** que facilita la interacción entre ellos. Este entendimiento profundo nos proporcionó una base sólida para abordar de manera efectiva la personalización y optimización de nuestras aplicaciones.

Posteriormente, nos adentramos en la configuración de la interfaz de usuario utilizando las funciones **fluidPage()** y **sidebarLayout()**, explorando las distintas posibilidades para organizar y presentar los elementos de la aplicación de manera coherente y eficiente. A través de esta exploración, aprendimos a distribuir los componentes de nuestra aplicación de manera estratégica, mejorando así la navegabilidad y la experiencia del usuario final.

Continuamos nuestra exploración con un enfoque en la documentación avanzada y la integración de **HTML** en **Shiny**, lo que nos permitió alcanzar un mayor nivel de personalización y control sobre el aspecto y la funcionalidad de nuestras aplicaciones. Esta capacidad de integración nos brindó la libertad de diseñar interfaces de usuario únicas y adaptadas a las necesidades específicas de cada proyecto, abriendo un amplio abanico de posibilidades creativas.

Finalmente, profundizamos en la dinámica del texto y las ecuaciones matemáticas dentro de **Shiny**, así como en la incorporación de imágenes para enriquecer la experiencia visual del usuario. Estas herramientas nos permitieron crear aplicaciones interactivas y atractivas que no solo presentan datos de manera efectiva, sino que también invitan a la exploración y la participación del usuario, estableciendo así un puente sólido entre los análisis de datos y la audiencia final.

En la siguiente sección, nos embarcaremos en una exploración más detallada sobre los principales widgets en **Shiny**. Estos elementos interactivos son fundamentales para la creación de aplicaciones web dinámicas en **Shiny**, ya que permiten a los usuarios interactuar directamente con los datos y los resultados presentados. Exploraremos una amplia variedad de widgets, desde simples botones hasta sliders, descubriendo cómo utilizar cada uno de ellos para mejorar la experiencia del usuario y agregar funcionalidades avanzadas a nuestras aplicaciones.

# EXPLORANDO LOS WIDGETS PRINCIPALES EN `Shiny`

En esta sección, se adentra en el apasionante mundo de los widgets<sup>10</sup> en **Shiny**, elementos fundamentales que potencian la interactividad y la usabilidad en las aplicaciones web desarrolladas con este potente framework en R. Los widgets, que funcionan como controles de entrada o salida, constituyen la esencia misma de la experiencia de usuario en **Shiny**, permitiendo a los usuarios interactuar de forma dinámica con los datos y resultados presentados. Desde simples botones hasta complejos gráficos interactivos, los widgets ofrecen una amplia variedad de opciones para personalizar y enriquecer la interacción del usuario en las aplicaciones **Shiny**, adaptándose a diversas necesidades y escenarios.

Constituyendo la columna vertebral de la creación de aplicaciones web interactivas en **Shiny**, los widgets desempeñan un papel crucial al facilitar la comunicación bidireccional entre los usuarios y la aplicación. Al proporcionar una interfaz intuitiva y amigable, los widgets permiten a los usuarios ingresar datos, seleccionar opciones y visualizar resultados de manera dinámica y fluida. La capacidad de respuesta y dinamismo de los widgets en **Shiny** radica en su capacidad para enviar información al servidor de **Shiny**, donde se ejecuta código R para procesar los datos y actualizar la interfaz de usuario en tiempo real. Esta sinergia entre el lado del cliente y el lado del servidor convierte a los widgets en componentes esenciales para crear aplicaciones web efectivas y atractivas en **Shiny**, que satisfacen las necesidades de los usuarios de manera eficiente y elegante.

---

<sup>10</sup>Un widget, en el contexto de desarrollo de aplicaciones web, es un componente de la interfaz de usuario que permite a los usuarios interactuar con la aplicación de diversas formas. En el contexto de **Shiny**, un widget es un control de entrada o salida que facilita la interacción entre el usuario y la aplicación. Los widgets pueden incluir botones, casillas de verificación, campos de texto, menús desplegables, deslizadores, gráficos interactivos y más. Estos elementos permiten a los usuarios ingresar datos, seleccionar opciones y visualizar resultados de manera dinámica, mejorando así la experiencia general del usuario en la aplicación.

# 1. EXPLORACIÓN INICIAL DE LA ESTRUCTURA BÁSICA EN Shiny: INTRODUCCIÓN A LOS WIDGETS

En este análisis inicial, se adentra en la estructura fundamental de una aplicación **Shiny**, centrada en la organización y disposición de los elementos en la interfaz de usuario. Comenzamos utilizando la función `fluidPage()` para crear una página dinámica que se ajusta automáticamente al tamaño del navegador. Al agregar un `titlePanel()`, se establece un título principal para nuestra aplicación, proporcionando una identidad clara desde el principio. Luego, creando una fila (`fluidRow()`) que contiene tres columnas de ancho igual (`column()`), cada una destinada a representar un segmento específico de la interfaz de usuario. En estas columnas, insertamos segmentos de texto para ilustrar las distintas partes de la aplicación. Este ejemplo ofrece una introducción esencial a la estructura básica de una aplicación **Shiny**, sentando las bases para exploraciones más profundas de su funcionalidad y personalización.

```
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)
```



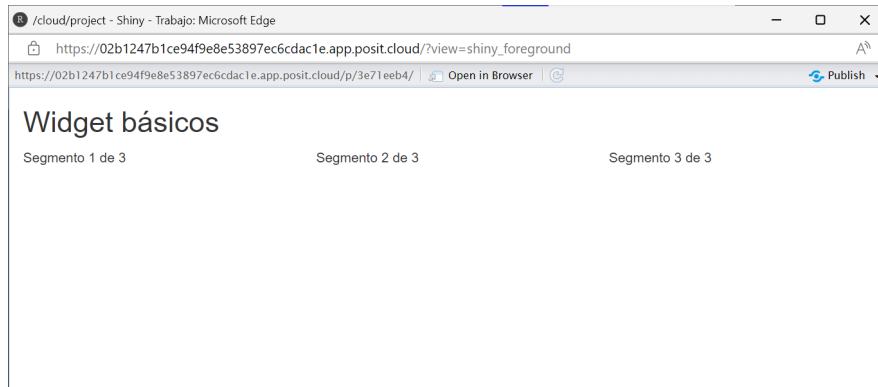


Figura 9: División por segmentos

En este bloque de código, se está explorando la integración de comandos HTML dentro de una aplicación **Shiny** para definir tanto la estructura como el contenido de la interfaz de usuario de manera más precisa y flexible. Dentro de la función `fluidPage()`, se utiliza la función `titlePanel()` para establecer un título principal para nuestra aplicación, lo que proporciona una identificación clara y distintiva para los usuarios. Posteriormente, se genera una fila (`fluidRow()`) que se divide en tres columnas (`column()`) de igual ancho. En cada columna, aprovechamos la función `h3()` para crear encabezados HTML de tercer nivel (`<h3>`) que representan diferentes secciones de la aplicación, como “**Botones**”, “**Test**” y “**Resultados**”. Estos encabezados jerarquizan y organizan visualmente la información, facilitando la navegación y comprensión para los usuarios finales.

```
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 `checkboxInput()` 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)
```

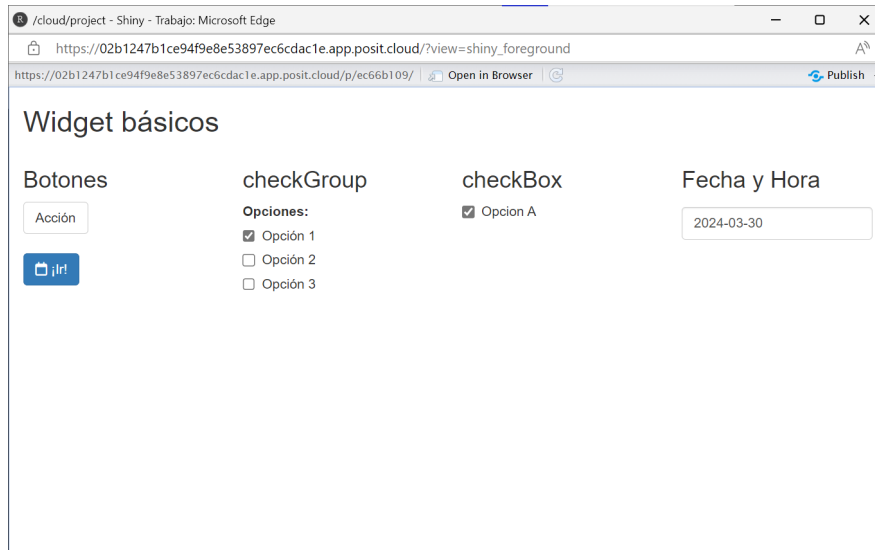


Figura 10: Widgets básicos para explorar en Shiny

## 2. EXPANSIÓN DE LA INTERACTIVIDAD: INCORPORACIÓN DE NUEVOS WIDGETS

Dentro de la interfaz de usuario (UI) de la aplicación **Shiny**, se están agregando cuatro nuevos widgets dentro de un nuevo `fluidRow()`. Estos widgets son esenciales para interactuar con los usuarios y capturar su entrada de datos de manera efectiva.

El primer widget es un `dateRangeInput()`, que proporciona a los usuarios la capacidad de seleccionar un rango de fechas. Este widget está configurado para mostrar el rango de fechas en formato día/mes/año ("`dd/mm/yyyy`") y se ha establecido un separador (`separator =`) para visualizar claramente el rango seleccionado.

El segundo widget añadido es un `fileInput()`. Este widget permite a los usuarios cargar archivos en la aplicación. Se ha configurado con una etiqueta de botón personalizada ("Archivo") y un marcador de posición para guiar a los usuarios sobre qué tipo de archivo deben seleccionar. Además, se ha habilitado la opción `multiple = TRUE`, lo que permite que los usuarios carguen varios archivos simultáneamente.

El tercer widget agregado es un `helpText()`, que no es técnicamente un widget interactivo, pero proporciona información útil y explicativa a los usuarios. Este caso se utiliza para ofrecer aclaraciones sobre el funcionamiento de otros widgets en la aplicación, así como sobre variables y funciones relacionadas.

Por último, se incluye un `numericInput()`, que permite a los usuarios introducir valores numéricos dentro de un rango específico. Este widget se ha configurado con un valor inicial de 80 y permite a los usuarios aumentar o disminuir el valor en incrementos de 5 unidades, dentro de un rango de 50 a 120.

```

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")))
  ),

  # Se agrega una función fluidRow()
  fluidRow(
    column(3,
      dateRangeInput("Fechas",
        h3("Rango de Fechas"),
        language = "es",
        format = "dd/mm/yyyy",
        # start = "2018-12-01"
        # end = "2050-12-30"
        separator = " a ")
    ),
    column(3,
      fileInput("Fichero",
        # Carga un archivo en una carpeta temporal
        h3("Fichero"),

```



```
        buttonLabel = "Archivo",
        placeholder = "Selecciona un Fichero",
        # Para subir más de un fichero
        multiple = TRUE)
    ),
    column(3,
      h3("Texto de Ayuda"),
      helpText("Nota: el texto de ayuda no es
                un widget realmente,",
                "Pero nos permite aclarar el funcionamiento de otros
                widgets. Incluso variables y otras funciones.")
    ),
    column(3,
      numericInput("Número",
        h3("Input numérico"),
        min = 50,
        max = 120,
        step = 5, # Suba y baje de 5 en 5
        value = 80
      )
    )
  )
)

server = function(input, output){}

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

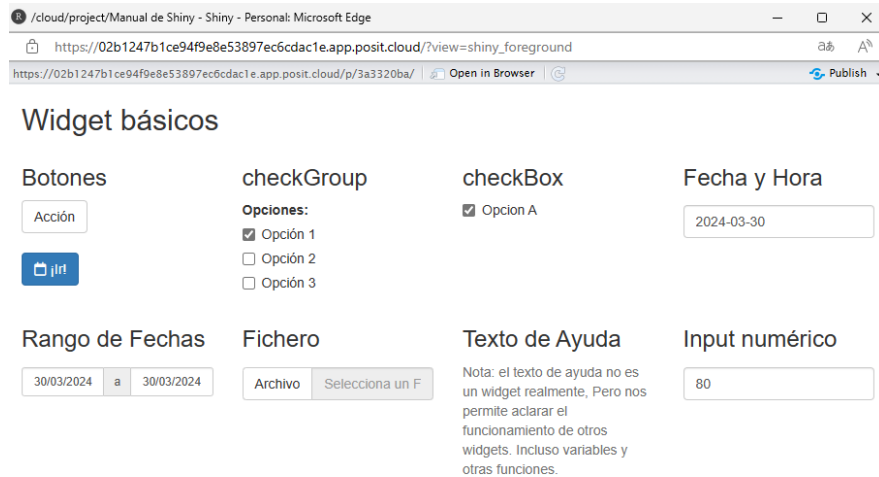


Figura 11: Ampliación de widgets

Estos nuevos widgets amplían la funcionalidad de la aplicación Shiny al proporcionar a los usuarios una gama más amplia de opciones de interacción y entrada de datos, desde la selección de fechas hasta la carga de archivos y la introducción de valores numéricos. Esta diversidad de widgets enriquece la experiencia del usuario y hace que la aplicación sea más versátil y útil para sus propósitos previstos.

### 3. EXPANSIÓN INTERACTIVA: MEJORA DE LA INTERFAZ DE USUARIO CON WIDGETS AVANZADOS

Se agregan unos nuevos cuatro elementos dentro de la interfaz de usuario (UI) dentro de nuestra aplicación **Shiny**. Aquí hay un resumen de lo que se está haciendo dentro de esta nueva parte:

Se agrega una nueva fila a partir de la función `fluidRow()`. Se está creando un nuevo `fluidRow()` para agregar más elementos a la interfaz de usuario. Dentro de esta función, se definirán las columnas adicionales para colocar los demás widgets.

Dentro de esta nueva fila, se agrega el nuevo widget `radioButtons()`, que permite al usuario seleccionar una opción de un conjunto de opciones mutuamente excluyentes. Las opciones se definen utilizando la lista `choices =`.

El siguiente widget es un `SliderInput`, en el cual se crea con la función `sliderInput()`, que ofrece al usuario un control deslizante para seleccionar un valor dentro de un rango dado. En este caso, se están definiendo dos controles deslizantes: el primero con un solo punto de inicio y fin (`Slider_1`), y otro con dos puntos (`Slider_2`), lo que permite la selección de un rango.

Por último, se añaden dos `textInput()`, que son campos de entrada de texto donde el usuario puede escribir texto. Uno de los `textInput()` tiene un valor predeterminado establecido (`Texto_1`), mientras que el otro tiene un marcador de posición (`Texto_2`), que proporciona una sugerencia sobre qué tipo de información debe ingresarse.

```
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")))
  ),
  fluidRow(
    column(3,
      dateRangeInput("Fechas",
        h3("Rango de Fechas"),
        language = "es",
        format = "dd/mm/yyyy",
        # start = "2018-12-01"
        # end = "2050-12-30"
        separator = " a ")
    ),
    column(3,
      fileInput("Fichero",
        # Carga un archivo en una carpeta temporal
        h3("Fichero"),
        buttonLabel = "Archivo",
```

```

        placeholder = "Selecciona un Fichero",
        # Para subir más de un fichero
        multiple = TRUE)
    ),
    column(3,
        h3("Texto de Ayuda"),
        helpText("Nota: el texto de ayuda no es
            un widget realmente,",
            "Pero nos permite aclarar el funcionamiento de otros
            widgets. Incluso variables y otras funciones.")
    ),
    column(3,
        numericInput("Número",
            h3("Input numérico"),
            min = 50,
            max = 120,
            step = 5, # Suba y baje de 5 en 5
            value = 80
        )
    )
),
# Se agrega una segunda función fluidRow()
fluidRow(
    column(3,
        radioButtons("radio",
            h3("Botón Radio"),
            choices = list("Opción 1" = 1,
                "Opción 2" = 2,
                "Opción 3" = 3),
            selected = 2)
    ),
    column(3,
        selectInput("Selección",
            h3("Selección"),
            choices = list("Opción A" = 1,
                "Opción B" = 2,
                "Opción C" = 3),
            selected = 3
        )
    ),
    column(3,
        sliderInput("Slider_1",
            h3("Sliders"),
            min = 0, # Barra que inicia desde 0

```

```
        max = 200, # Barra que termina en 200
        value = 25 # Se queda desde 50
      ),
    br(), # br() es un espacio
    # Slider con 2 elementos
    sliderInput("Slider_2",
      h3("Slider con 2 elementos"),
      min = 0,
      max = 200,
      step = 10, # Va de 10 en 10
      # Con dos despliegues en 30 y 70
      value = c(30 , 70),
      # Para redondear decimales
      round = TRUE)
  ),
  column(3,
    textInput("Texto_1",
      h3("Input Texto"),
      value = "Texto por defecto"),
    textInput("Texto_2",
      h3("Input Texto"),
      placeholder = "Introduce tu Nombre")
  )
)

server = function(input, output){}

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

Esta parte de código está agregando más opciones interactivas a la interfaz de usuario de la aplicación Shiny, incluyendo botones de radio, menús desplegables, controles deslizantes y campos de entrada de texto para mejorar la experiencia del usuario.

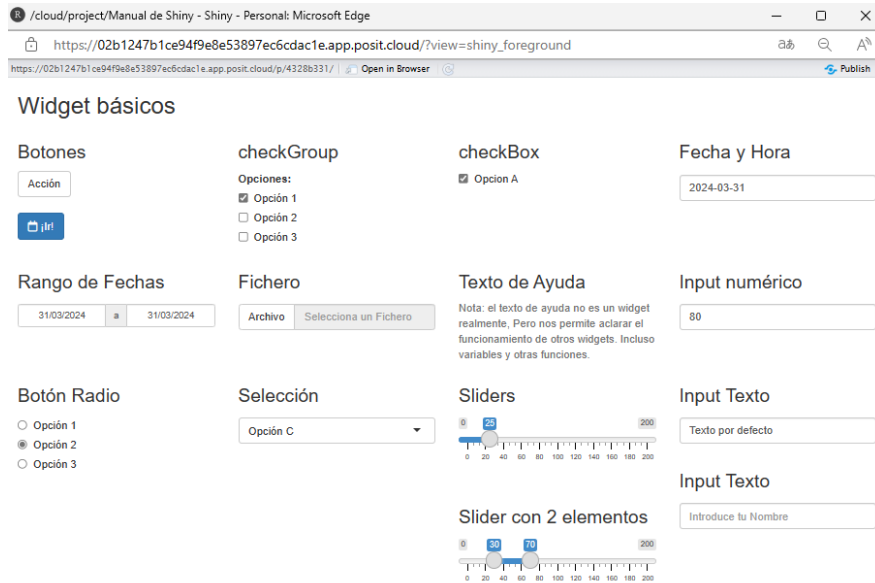


Figura 12: Extensión de los últimos Widgets

En esta parte dedicada al desarrollo de aplicaciones **Shiny**, se ha abordado un proceso gradual de familiarización con la estructura y funcionalidades básicas de esta plataforma para la creación de aplicaciones web interactivas en **R**. Comenzando con una exploración inicial, se ha delineado la organización fundamental de una aplicación **Shiny**, destacando la importancia de funciones esenciales como `fluidPage()` y `titlePanel()` para establecer una base estructural sólida. A través de una disposición en columnas y filas, se ha ilustrado cómo estructurar la interfaz de usuario, proporcionando un esquema claro y accesible para futuras iteraciones y desarrollos<sup>11</sup>.

En el avance hacia una mayor interactividad, se ha introducido una variedad de widgets básicos dentro de la aplicación, ampliando así las capacidades de entrada de datos y acción por parte de los usuarios. Desde botones de acción hasta la selección de fechas y la carga

<sup>11</sup>Para continuar ampliando la funcionalidad de tu aplicación **Shiny** y explorar aún más el potencial de los widgets, considera las siguientes opciones: **1) Explorar Widgets Avanzados:** Investiga widgets más avanzados disponibles en **Shiny**, como `selectizeInput()` para selectores de búsqueda, `verbatimTextOutput()` para mostrar texto sin procesar, o `plotOutput()` para visualizaciones dinámicas. Estos widgets pueden agregar una capa adicional de interactividad y personalización a tu aplicación. **2) Integrar Widgets Específicos de Dominio:** Si tu aplicación se centra en un área específica, como finanzas, biología o geoespacial, busca widgets diseñados para ese dominio. Por ejemplo, puedes utilizar `leaflet()` para integrar mapas interactivos o `plotly::plot_ly()` para visualizaciones **3D**. **3) Personalización Avanzada de Widgets:** Aprende cómo personalizar aún más los widgets utilizando parámetros adicionales y opciones de estilo. Por ejemplo, puedes ajustar colores, tamaños, estilos de fuente y comportamientos de los widgets para adaptarlos a las necesidades de tu aplicación y mejorar la experiencia del usuario. **4) Explorar Extensiones de Widgets:** Investiga las extensiones de widgets disponibles en paquetes complementarios de **Shiny** o desarrolla tus propias extensiones personalizadas. Estas extensiones pueden proporcionar funcionalidades adicionales o widgets especializados que no están disponibles en la biblioteca estándar de **Shiny**.



de archivos, cada widget ha sido seleccionado estratégicamente para ofrecer una experiencia de usuario rica y dinámica. Esta incorporación de widgets básicos sienta las bases para una interacción más compleja y diversa, mientras se mantiene una interfaz intuitiva y fácil de usar.

La expansión de la interactividad se ha manifestado aún más con la introducción de widgets avanzados, tales como botones de radio, menús desplegados y controles deslizantes. Estos elementos proporcionan una mayor flexibilidad y precisión en la captura de la entrada del usuario, permitiendo una personalización más detallada de la experiencia de la aplicación. Además, la inclusión de elementos como el `helpText()` demuestra un enfoque centrado en el usuario, brindando información contextual y orientación dentro de la interfaz para mejorar la usabilidad y comprensión.

En conjunto, este capítulo ha representado un progreso significativo en el desarrollo de aplicaciones **Shiny**, desde la comprensión de su estructura básica hasta la implementación de widgets avanzados. A través de una combinación de diseño cuidadoso, selección de widgets y atención a la experiencia del usuario, se ha sentado una base sólida para futuras iteraciones y refinamientos, con el objetivo final de crear aplicaciones web interactivas y eficaces en **R**.

