

# **TALLER SHINY**

**Diplomado Data Science**  
**Felipe Peña Graf**





# **INTERFAZ GRÁFICA**



# AGREGANDO CONTENIDO

---

- ¿Cómo le agregamos contenido a una página web?
- Cuando escribimos HTML, el contenido queda marcado con etiquetas (tags)

`<h1></h1>`

`<div></div>`

`<a> </a>`

# HTML

---

- La interfaz gráfica de las aplicaciones en Shiny se basan en HTML

```
ui <- fluidPage()
```

```
fluidPage()
```

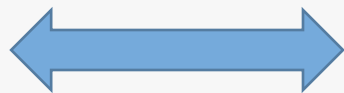
```
<div class= 'container-fluid></div>
```

# HTML EN R

---

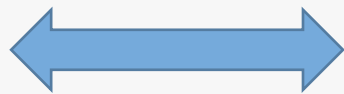
- En R usaremos la función de etiquetas TAG para poder generar HTML de forma sencilla:

`tags$h1()`



`<h1></h1>`

`tags$a()`



`<a1></a1>`

# TAGS

- Cada elemento en la lista de tags es una función que recrea una etiqueta de HTML:
- Las más usadas son:
  - a
  - b
  - div
  - area
  - code
  - button
  - h1,h2,h3
  - img

```
tags$h1
```

```
function (...)  
tag("h1", list(...))  
<environment: namespace:htmltools>
```

```
tags$h1()
```

```
<h1></h1>
```

# TAGS

---

```
tags$a(href="www.rstudio.com", "Rstudio")
```



Etiqueta

Argumentos  
con nombre

Argumentos sin nombre  
quedan dentro de la  
etiqueta

```
<a href="www.rstudio.com"> "Rstudio"</a>
```

# TEXT

---

- El texto se puede escribir directamente y no necesita etiquetas
- No se podrá formatear bien

```
fluidPage(  
  "This is a Shiny app.",  
  "It is also a web page."  
)
```



# IMÁGENES

---

## img()

Agrega una imagen. Use el argumento src para dar el URL de la imagen.

Para usar una imagen local se debe guardar en la carpeta www.

```
tag$img(height = 100,  
        width = 100,  
        src = "dirección web imagen")
```

# HTML()

---

- También si sabemos la estructura del html que queremos agregar podemos hacerlo directamente usando la función:

**HTML(text,..., .noWS = NULL)**



Texto HTML a concatenar

Parámetro para  
gestionar el espacio  
en blanco

# STYLE!

---

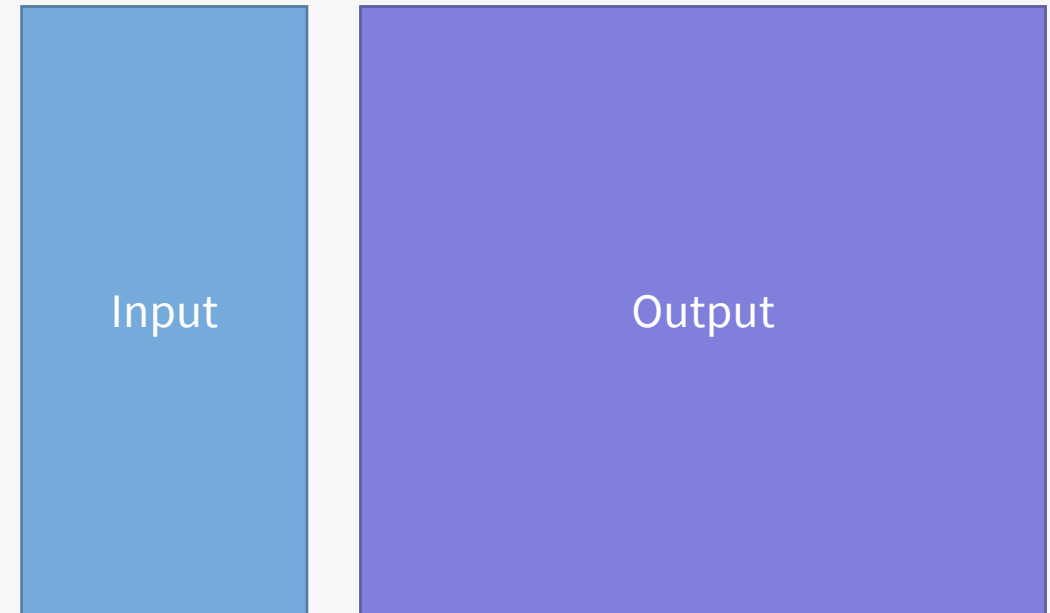
- Le podemos dar estilo a las etiquetas aprovechando el atributo style de HTML. La estructura es la siguiente:

`<tagname style="property:value;">`

# LAYOUT

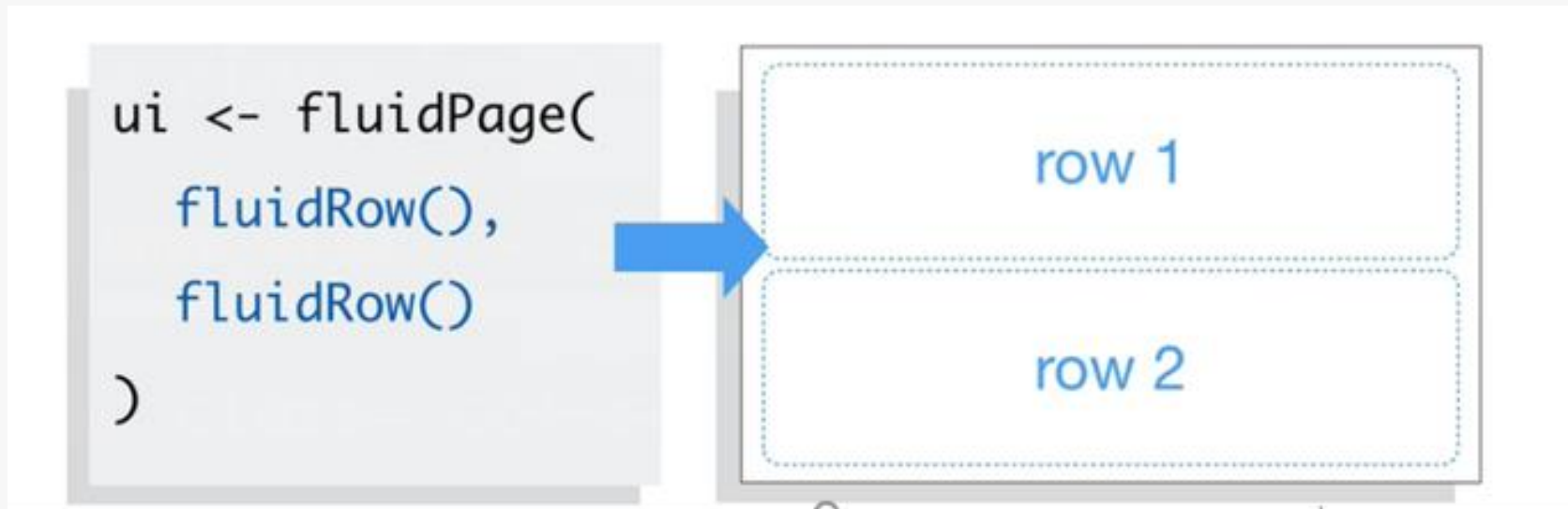
---

- Las funciones de layout nos permiten dividir la interfaz de usuario en una grilla:
- Ya hemos visto `fluidPage` (lo hemos usado al menos)
- Ahora veremos como distribuir los elementos como queramos



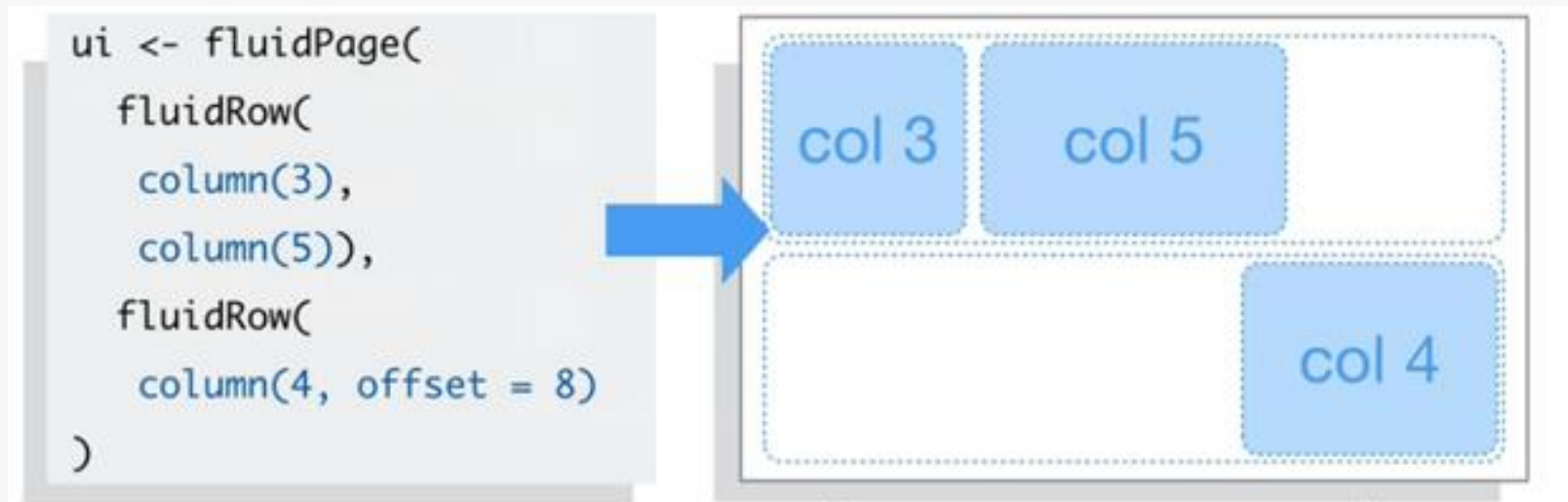
# FLUID ROW

- `fluidRow()` agrega filas flexibles a la grilla, cada nueva fila se agrega debajo de la anterior.



# COLUMNS

- **column()** agrega columnas dentro de una fila, cada nueva columna queda a la derecha de la columna anterior.
- Se le puede dar el ancho y el offset de cada columna con valores que van entre 1 y 12.
- Estos valores provienen de Bootstrap.



# AGREGANDO ELEMENTOS A LA ESTRUCTURA

- Para agregar un elemento a la grilla, este debe quedar como argumento en una función layout.

```
fluidRow("In the row")
```

```
<div class="row">In the row</div>
```

```
column(2, plotOutput("hist"))
```

```
<div class="col-sm-2">
```

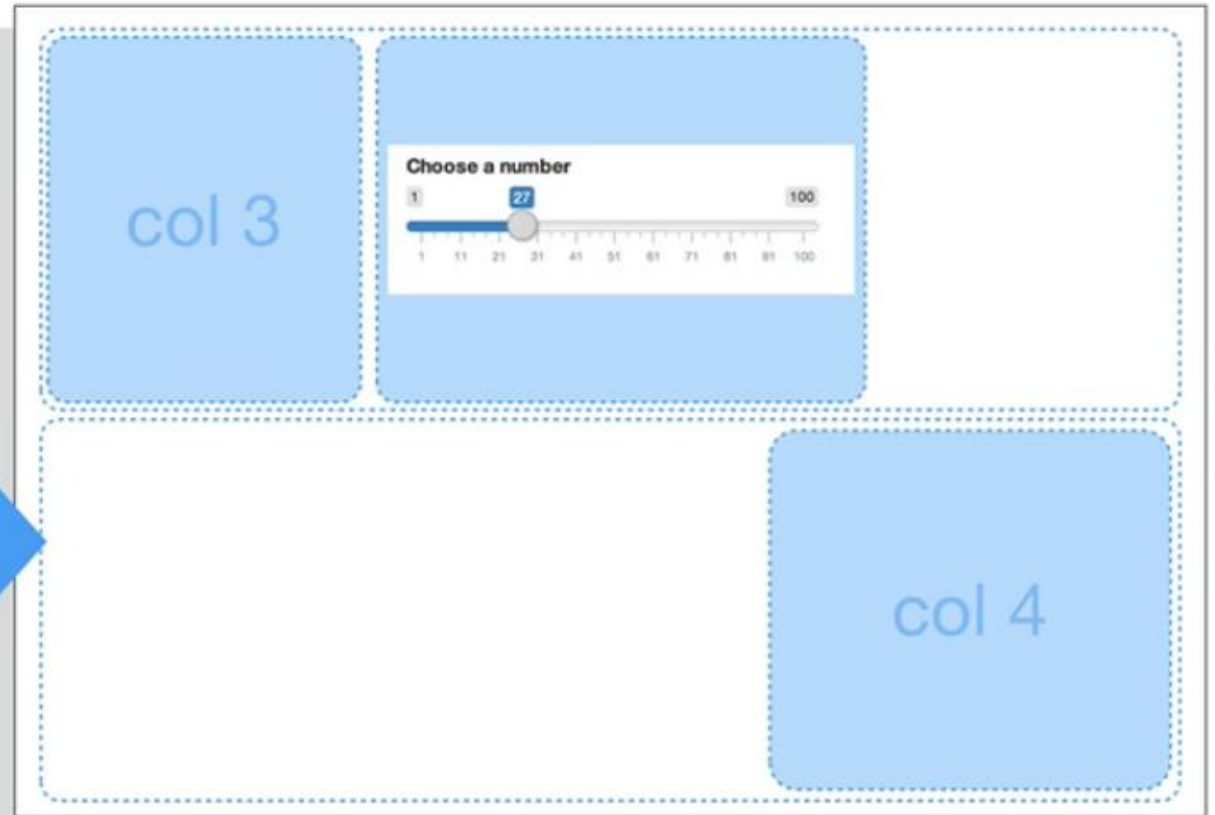
```
  <div id="hist" class="shiny-plot-output"
```

```
    style="width: 100% ; height: 400px"></div>
```

```
</div>
```

# OTRO EJEMPLO

```
fluidPage(  
  fluidRow(  
    column(3),  
    column(5, sliderInput(...))  
  ),  
  fluidRow(  
    column(4, offset = 8)  
  )  
)
```





# PANELES

- Los paneles nos permiten agrupar elementos en grupos que mantienen ciertas características comunes.
- Ejemplo: la galería de widgets de shiny.

The screenshot shows the Shiny Widgets Gallery interface. At the top, there's a navigation bar with "Shiny from R Studio" and links for "Back to Gallery" and "Get Code". Below this is a blue header with the title "Shiny Widgets Gallery". A paragraph explains that the "Current Value(s)" window displays the value provided to shinyServer, which changes as users interact with the widgets.

Three widget examples are shown in a grid:

- Action button:** Features a button labeled "Action". The "Current Value:" window shows a list with two elements: `[1] 0` and `attr(,"class")`, and another list with `[1] "integer"` and the attribute `"shinyActionButtonValue"`. A "See Code" button is at the bottom.
- Single checkbox:** Features a checkbox labeled "Choice A" which is checked. The "Current Value:" window shows `[1] TRUE`. A "See Code" button is at the bottom.
- Checkbox group:** Features three checkboxes labeled "Choice 1", "Choice 2", and "Choice 3". "Choice 1" is checked. The "Current Values:" window shows `[1] "1"`. A "See Code" button is at the bottom.

The "Action button" widget is highlighted with a blue border.

# WELLPANEL

## wellPanel()

Agrupar elementos en la caja gris que usamos generalmente de entrada.

### Action button

Action

---

Current Value:

```
[1] 0  
attr(,"class")  
[1] "integer" "shinyActionButtonValue"
```

See Code

# TIPOS DE PANELES

## **absolutePanel()**

Panel position set rigidly (absolutely), not fluidly

## **conditionalPanel()**

A JavaScript expression determines whether panel is visible or not.

## **fixedPanel()**

Panel is fixed to browser window and does not scroll with the page

## **headerPanel()**

Panel for the app's title, used with `pageWithSidebar()`

## **inputPanel()**

Panel with grey background, suitable for grouping inputs

## **mainPanel()**

Panel for displaying output, used with `pageWithSidebar()`

## **navlistPanel()**

Panel for displaying multiple stacked `tabPanels()`. Uses sidebar navigation

## **sidebarPanel()**

Panel for displaying a sidebar of inputs, used with `pageWithSidebar()`

## **tabPanel()**

Stackable panel. Used with `navlistPanel()` and `tabsetPanel()`

## **tabsetPanel()**

Panel for displaying multiple stacked `tabPanels()`. Uses tab navigation

## **titlePanel()**

Panel for the app's title, used with `pageWithSidebar()`

## **wellPanel()**

Panel with grey background.

# TABPANEL

---

## tabPanel()

Crea elementos en capas que se convierten en pestañas. Cada pestaña es una interfaz de usuario por si sola

**tabPanel("Tab 1",...)**

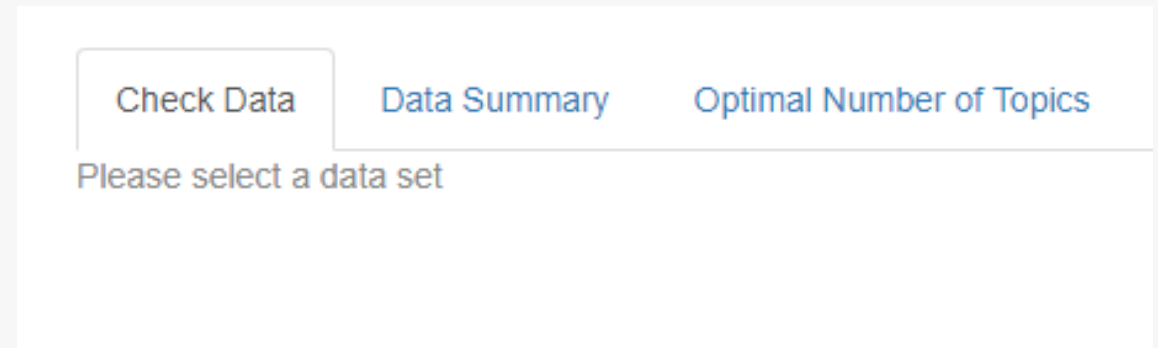
# TABSETPANEL

## tabsetPanel()

Combina las pestañas en un solo panel interactivo.

Las pestañas son navegables y llamamos a tabPanels para movernos o agregar nuevos.

```
tabsetPanel(type = "tabs",  
  tabPanel("Check Data", tableOutput("table1")),  
  tabPanel("Data Summary",  
    h2(textOutput("tSummaryTitle")),  
    tableOutput("summaryStatistics"),  
    plotOutput("summaryPlot1")  
  ),  
  tabPanel("Optimal Number of Topics",  
    h2(textOutput("t2Text1")),  
    tableOutput("table2"),  
    h2(textOutput("t2Text2")),  
    tableOutput("table3"))
```



# NAVLISTPANEL

## navlistPanel()

Combina las pestañas en un solo panel interactivo.

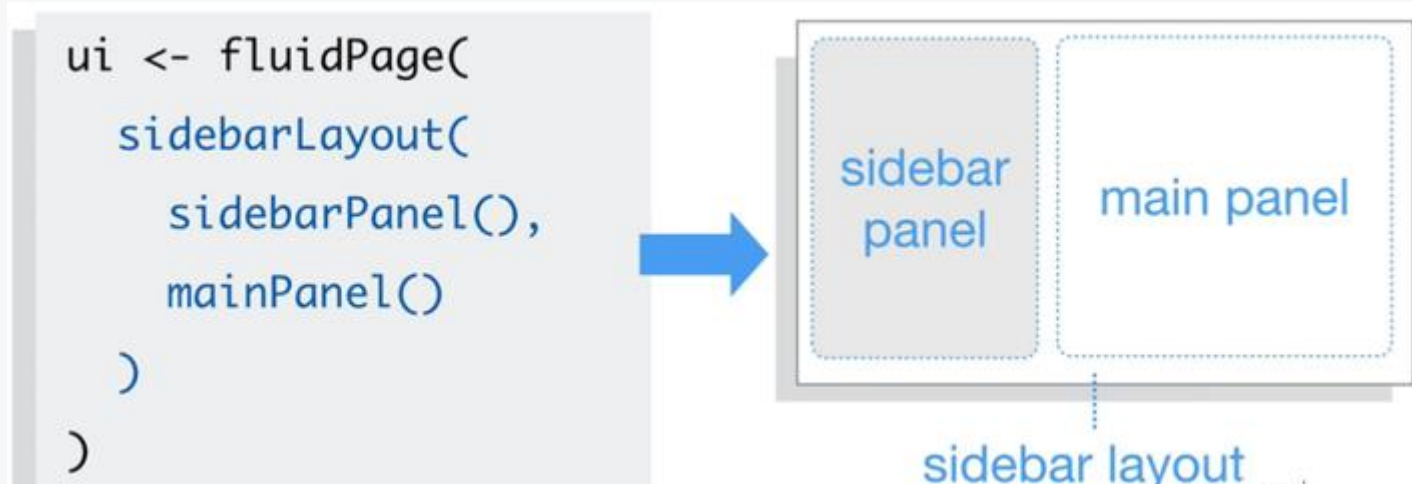
La diferencia es que ahora crea enlaces para movernos en la navegación.




# SIDEBARLAYOUT


## sidebarLayout()

La versión casi por defecto actualmente en shiny. Utiliza sidebarPanel() y mainPanel() para dividir la aplicación en dos secciones



# BOOTSTRAP 3

 [Home](#) [Docs](#) [Examples](#) [Icons](#) [Themes](#) [Blog](#)


 [Download](#)

## Build fast, responsive sites with Bootstrap


Quickly design and customize responsive mobile-first sites with Bootstrap, the world's most popular front-end open source toolkit, featuring Sass variables and mixins, responsive grid system, extensive prebuilt components, and powerful JavaScript plugins.

[Get started](#) [Download](#)

Currently **v5.0.2** · [v4.6.x docs](#) · [All releases](#)

  
Use tools,  
not snake oil.  
[Listen to Podcast](#)

Let software do the work of driving culture change.  
ads via Carbon

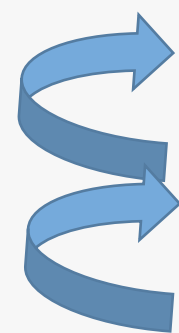




# CSS + BOOTSRAP 3

---

Se le puede dar estilo a la aplicación de 3 maneras

- 
1. Link a un archivo CSS
  2. Escribir CSS en header
  3. Escribir CSS en cada tag con el estilo apropiado

El estilo se puede aplicar:

1. Tag
2. Clase
3. Id

# ESTRATEGIA RECOMENDADA

---

```
ui <- fluidPage(  
  includeCSS("file.css")  
)
```



```
<head>  
  <style>  
    p {  
      color:red;  
    }  
  </style>  
</head>  
<body>  
  <div class="container-fluid">  
  </div>
```

# ARCHIVOS DE ENTRADA

---

- Para consumir archivos usaremos fileInput.

```
fileInput(  
    inputId,  
    label,  
    multiple = FALSE,  
    accept = NULL,  
    width = NULL,  
    buttonLabel = "Browse...",  
    placeholder = "No file selected"  
)
```

# EJERCICIO PRÁCTICO 3

---

- Realice un Clustering utilizando k-Means
- Agregue una distribución de panel sencilla
- Muestre una tabla con los resultados en una pestaña
- Muestre un gráfico en otra
- Agregue HTML y una introducción apropiada al inicio
- Agregue una imagen de su elección
- Publique su aplicación en [shinyapps.io](https://shinyapps.io)

# EJEMPLO PRÁCTICO 3

## Ejercicio Práctico 3 con Iris



### Resumen

En base a muestras aleatorias mostramos información del dataset

Número de grupos 1:

1 10 20

Título de Gráfico

Escriba texto aquí

Número de elementos en dataset :

10 84 150

Calcular/Actualizar

Distribución Scatterplot Clustering

Número de grupos

2





# INCOPORANDO OTRAS BIBLIOTECAS

# SHINYJS

---

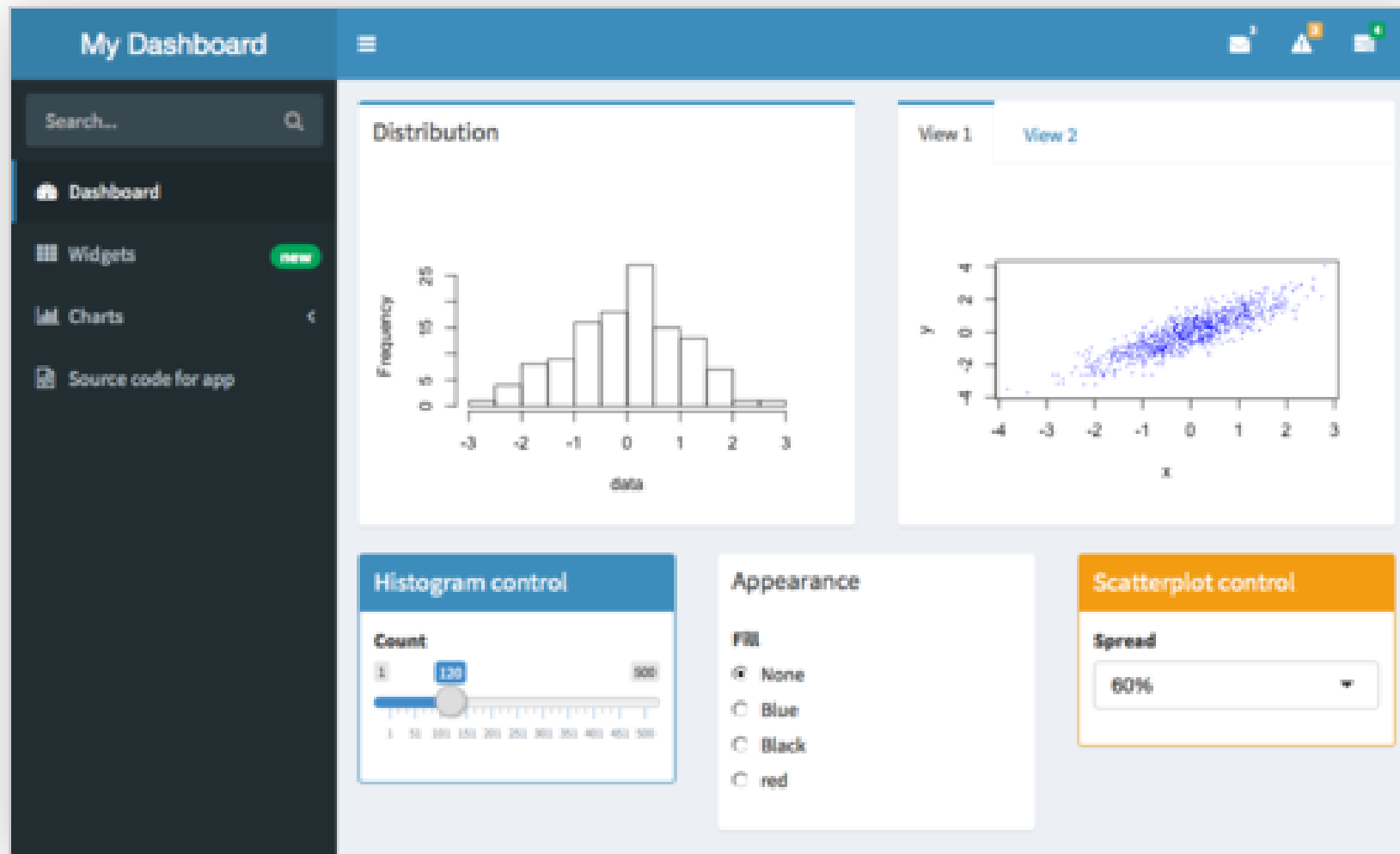


Easily improve the user experience of your Shiny apps in seconds

[DEMO](#)

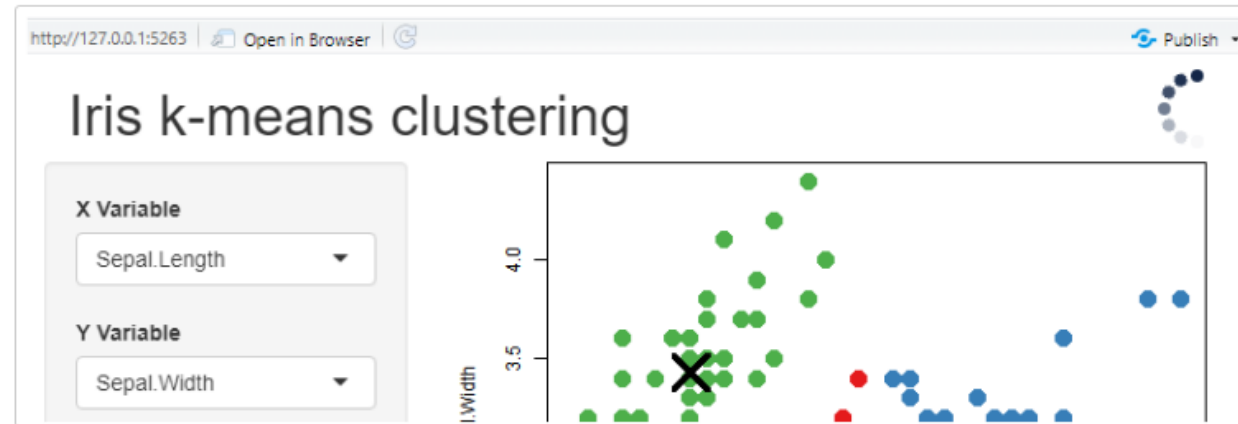
[GET STARTED](#)

# SHINY DASHBOARD



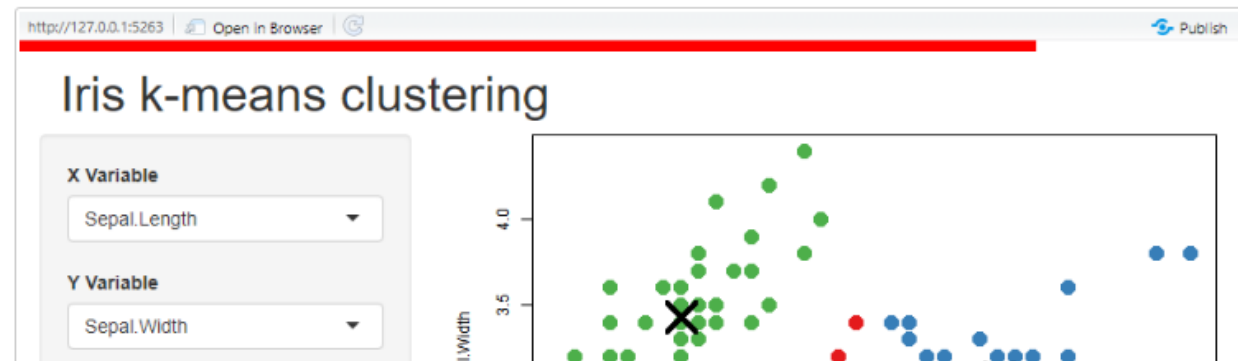


# SHINY BUSY



- a top progress bar : with `add_busy_bar()`.

```
add_busy_bar(color = "red", height = "8px")
```



- a GIF that will animate when app is busy : with `add_busy_gif()`.

# CONCEPTOS NO CUBIERTOS

---

Validación


Módulos

Embedding

Bookmarking

Testing

# R-BLOGGERS



**R-BLOGGERS**  
R news and tutorials contributed by hundreds of R bloggers

HOMEABOUTRSSADD YOUR BLOG!LEARN RR JOBSCONTACT US

## Top 3 Coding Best Practices from the Shiny Contest

Posted on July 21, 2021 by Nick Strayer in R bloggers | 0 Comments

[This article was first published on [RStudio Blog](#), and kindly contributed to [R-bloggers](#)]. (You can report issue about the content on this page [here](#))

Want to share your content on R-bloggers? [click here](#) if you have a blog, or [here](#) if you don't.

Share

Tweet

Recently we wrapped up another round of the Shiny Contest and, as always, the entries


Go

Subscribe

52793 readers

Follow @rbloggers

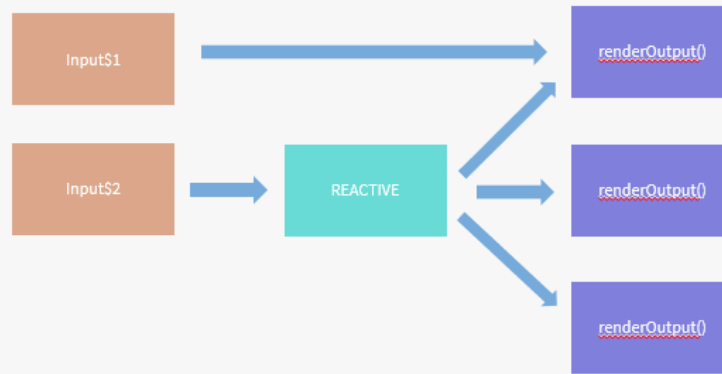
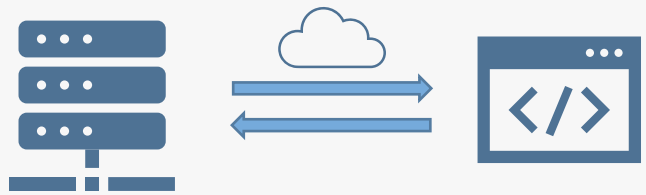
93.2K followers



**R bloggers**  
Like Page79K likes

<https://www.r-bloggers.com/2021/07/top-3-coding-best-practices-from-the-shiny-contest/>

# RESUMIENDO



**Construir y  
Compartir**

**Controlar  
Reacciones**

**Personalizar  
Apariencia**

# **TALLER SHINY**

**Diplomado Data Science**  
**Felipe Peña Graf**

