

Reportes Interactivos

Juan C. Correa

Material de uso exclusivo para
INGENIO PANTALEON, S.A.
Diagonal 6, 10-31, Zona 10

Ciudad de Guatemala

1 Breve Introducción

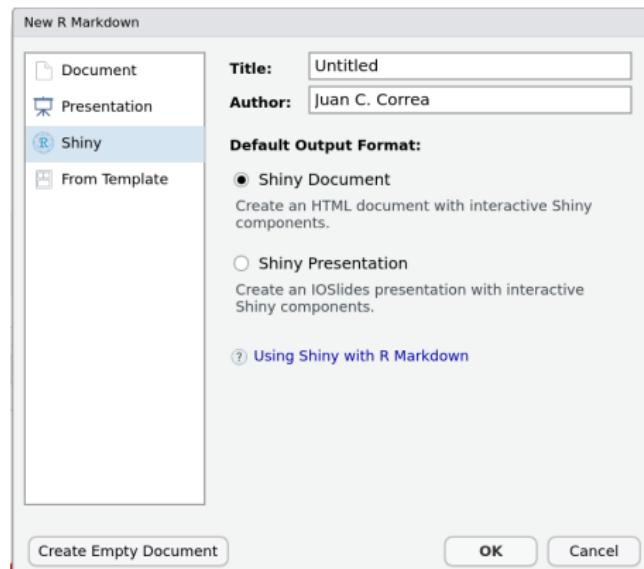
2 Inputs & Outputs

3 Server

4 Más Recursos

Breve Introducción

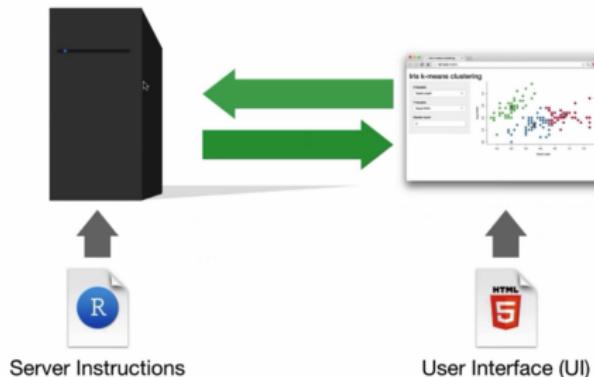
En RStudio, el usuario puede generar reportes ejecutivos dinámicos a través de la herramienta **Shiny**



<https://vimeo.com/rstudioinc/review/131218530/212d8a5a7a/>

Breve Introducción

Lo primero que debemos entender para hacer un reporte interactivo son los componentes que se requieren para desarrollarlo.



Componente 1: Interfaz de Usuario (User Interface)

Componente 2: Instrucciones del Servidor (Server Instructions).

Breve Introducción

Los componentes se integran a través de una plantilla shiny que siempre tiene la misma estructura.

App template

The shortest viable shiny app

```
library(shiny)
ui <- fluidPage()

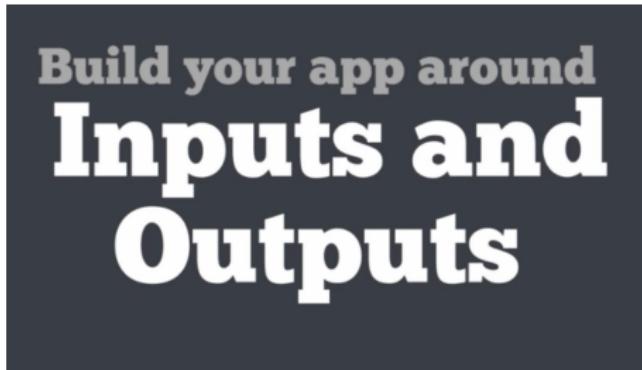
server <- function(input, output) {}

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

© 2015 RStudio, Inc.

Breve Introducción

Los reportes interactivos se diseñan alrededor de Inputs y Outputs



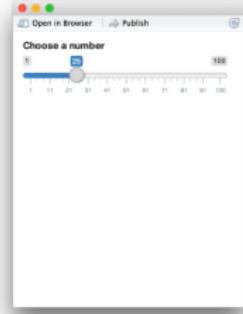
Inputs & Outputs

Los Inputs son elementos del reporte que el usuario final puede manipular a su antojo para mirar cómo cambian los outputs en función de estas simples manipulaciones

```
library(shiny)
ui <- fluidPage(
  sliderInput(inputId = "num",
    label = "Choose a number",
    value = 25, min = 1, max = 100)
)

server <- function(input, output) {}

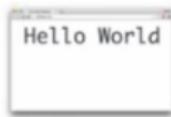
shinyApp(server = server, ui = ui)
```



Recap

```
library(shiny)
ui <- fluidPage()
server <- function(input, output) {}
shinyApp(ui = ui, server = server)
```

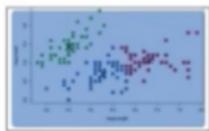
Begin each app with the template



Add elements as arguments to **fluidPage()**



Create reactive inputs with an ***Input()** function



Display reactive results with an ***Output()** function



Use the server function to assemble inputs into outputs

Inputs & Outputs

Estas son las sintaxis para los inputs.

Buttons

Action

Submit

```
actionButton()  
submitButton()
```

Single checkbox

Choice A

Checkbox group

- Choice 1
- Choice 2
- Choice 3

Date input

2014-01-01

Date range

2014-01-24 to 2014-01-24

```
dateRangeInput()
```

File input

Choose File No file chosen

```
fileInput()
```

Numeric input

1

```
numericInput()
```

Password Input

```
passwordInput()
```

Radio buttons

- Choice 1
- Choice 2
- Choice 3

```
radioButtons()
```

Select box

Choice 1

Sliders



```
selectInput()
```

Text input

Enter text...

```
textInput()
```

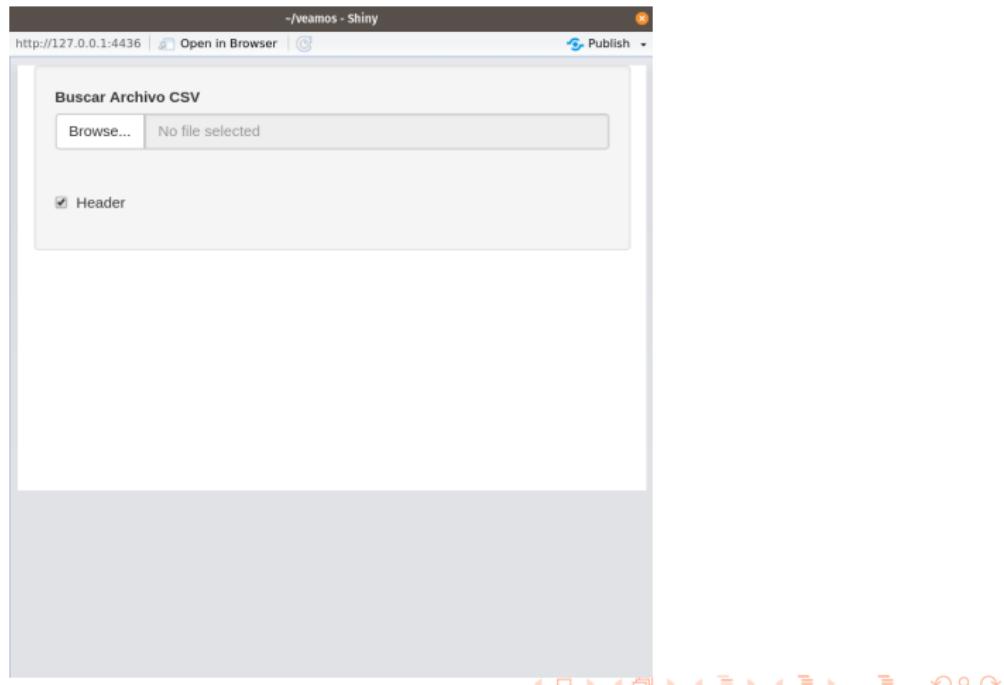
Inputs & Outputs

Veamos un ejemplo

```
1 library(shiny)
2 ui <- fluidPage(sideBarLayout(sideBarPanel(
3     fileInput("file1", "Buscar Archivo CSV", accept = ".csv"),
4     checkboxInput("header", "Header", TRUE)),
5     mainPanel(tableOutput("contents"))))
6
7 server <- function(input, output) {
8     output$contents <- renderTable({
9         file <- input$file1
10        ext <- tools::file_ext(file$datapath)
11        req(file)
12        validate(need(ext == "csv", "Please upload a csv file"))
13        read.csv(file$datapath, header = input$header)
14    })
15}
16
17 shinyApp(ui, server)|
```

Inputs & Outputs

Al hacer clic en el botón Browse, se nos va a abrir una ventana para que busquemos el archivo específico que queremos mostrar en la aplicación Shiny.



Inputs & Outputs

Este es el resultado que deberíamos ver

The screenshot shows a Shiny application window titled "veamos - Shiny" at the URL "http://127.0.0.1:4436". The top navigation bar includes "Open in Browser" and "Publish" buttons. A modal dialog titled "Buscar Archivo CSV" is open, showing a file selection input with "Ejemplo1.csv" selected and a blue progress bar indicating "Upload complete". A checkbox labeled "Header" is checked. Below the modal, a table displays the uploaded data:

TCH	Rendimiento	Edad_Rango
85	Inicial	<10
80	Inicial	<10
81	Inicial	<10
80	Inicial	<10
85	Inicial	<10
82	Inicial	<10

Inputs & Outputs

También existe una familia de funciones para Outputs

Function	Inserts
dataTableOutput()	an interactive table
htmlOutput()	raw HTML
imageOutput()	image
plotOutput()	plot
tableOutput()	table
textOutput()	text
uiOutput()	a Shiny UI element
verbatimTextOutput()	text

*Output()

To display output, add it to `fluidPage()` with an
`*Output()` function

`plotOutput(outputId = "hist")`

the type of output
to display

name to give to the
output object

© 2015 RStudio, Inc.

Tell the
server
how to assemble
inputs into outputs

Server

Hay tres reglas para escribir las funciones que el servidor usará para desarrollar la aplicación Shiny

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

- **Regla 1:** El output debe guardarse con `output$`
- **Regla 2:** Para generar el output debe especificarse la forma cómo va a mostrarse y esto se hace con las funciones `render*`()
- **Regla 3:** Use valores de Input con `input$`

Regla 1

```
output$hist  
↓  
plotOutput("hist")
```

Regla 2

```
server <- function(input, output) {  
  output$hist <- renderPlot({  
    })  
}
```

Regla 2

Use the **render***() function that creates the type of output you wish to make.

function	creates
renderDataTable()	An interactive table <small>(from a data frame, matrix, or other table-like structure)</small>
renderImage()	An image (saved as a link to a source file)
renderPlot()	A plot
renderPrint()	A code block of printed output
renderTable()	A table <small>(from a data frame, matrix, or other table-like structure)</small>
renderText()	A character string
renderUI()	a Shiny UI element

Regla 2

`render*`()

Builds reactive output to display in UI

```
renderPlot({ hist(rnorm(100)) })
```

type of object to build

code block that builds the object

Regla 2

```
server <- function(input, output) {  
  output$hist <- renderPlot({  
    hist(rnorm(100))  
  })  
}
```

Regla 3

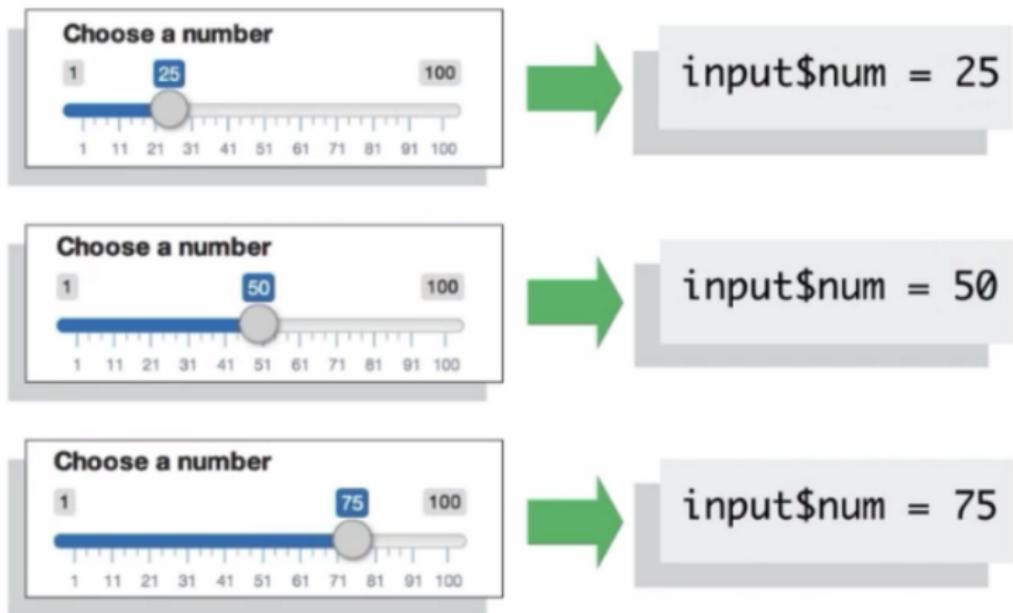
```
sliderInput(inputId = "num",...)
```



```
input$num
```

Server

Regla 3



Regla 3

```
server <- function(input, output) {  
  output$hist <- renderPlot({  
    hist(rnorm(input$num))  
  })  
}
```

Recap: Server



Use the server function to assemble inputs into outputs. Follow 3 rules:

`output$hist <-`

1. Save the output that you build to `output$`

```
renderPlot({  
  hist(rnorm(input$num))  
})
```

2. Build the output with a `render*()` function

`input$num`

3. Access input values with `input$`



Create reactivity by using `Inputs` to build `rendered Outputs`

Más Recursos

shinyapps.io by RStudio

Home Features Pricing Support Log In

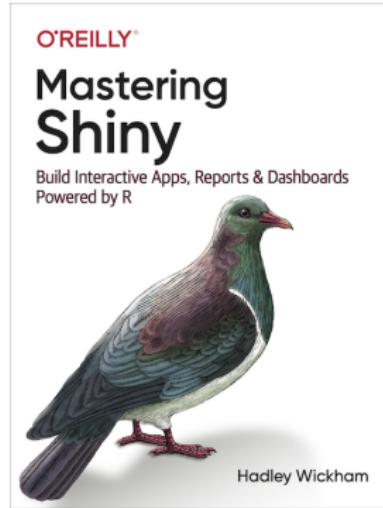
The screenshot shows the shinyapps.io pricing page with five plans:

- FREE**: \$0 /month. Description: New to Shiny? Deploy your applications for FREE. Includes: 5 Applications, 25 Active Hours, Community Support, RStudio Dashboard.
- STARTER**: \$9 /month (or \$100/year). Description: More applications, More active hours! Includes: 25 Applications, 100 Active Hours, Premium Email Support.
- BASIC**: \$39 /month (or \$440/year). Description: Take your users to the next level. Includes: Unlimited Applications, 500 Active Hours, Performance Boost, Premium Email Support.
- STANDARD**: \$99 /month (or \$1,100/year). Description: Password protection! Authenticate your users! Includes: Unlimited Applications, 2,000 Active Hours, Authentication, Performance Boost, Premium Email Support.
- PROFESSIONAL**: \$299 /month (or \$3,300/year). Description: Professional has it all! Personalize your domain. Includes: Unlimited Applications, 10,000 Active Hours, Authentication, Account Sharding, Performance Boost, Premium Email Support.

Para desarrollar aplicaciones Shiny accesibles por Internet, hay distintos planes de afiliación.

Más Recursos

Probablemente la mejor referencia sobre Shiny es el libro de Hadley Wickham.



<https://mastering-shiny.org/index.html>