

¿CÓMO CONSEGUIR
AYUDA?

LO BÁSICO

- LA DOCUMENTACIÓN DE R
 - ESCRIBIR “?” ANTES DEL COMANDO
 - UTILIZAR LA PESTAÑA **HELP** DE RSTUDIO

MÁS ALLÁ

- TANTO R COMO SHINY HA SIDO DESARROLLADO POR PROFESIONALES DE FORMA ORGÁNICA
- PERO TAMBIÉN GRACIAS AL TRABAJO DE MUCHOS VOLUNTARIOS
- PREGUNTANDO LAS DUDAS EN LOS CANALES CORRECTOS CONTRIBUYES A MEJORAR TODO
 - BUSCA EL MEJOR LUGAR PARA REALIZAR TU PREGUNTA
 - FORMÚLALA DE UNA FORMA CLARA Y CONCISA
 - UTILIZA EJEMPLOS REPRODUCIBLES
 - AGRADECE LAS CONTRIBUCIONES Y ESCRIBE UN FEEDBACK

CANALES

- EXISTEN DIFERENTES CANALES, SEGÚN EL TIPO DE DUDA:
 - GENERALES
 - R
 - SHINY
 - SHINY SERVER
 - SHINYAPPS.IO
 - RSTUDIO IDE

DUDAS GENERALES

- EL FORO DE ESTE CURSO
- MENSAJE PRIVADO
- www.HEROESDELDATO.COM
- GOOGLEA (PREFERIBLEMENTE EN INGLÉS)
 - PROGRAMMING R
 - SHINY R
- R-BLOGGERS:
 - [HTTPS://WWW.R-BLOGGERS.COM/](https://WWW.R-BLOGGERS.COM/)
- DATA CAMP:
 - [HTTPS://WWW.DATA CAMP.COM/](https://WWW.DATA CAMP.COM/)
- KAGGLE:
 - [HTTPS://WWW.KAGGLE.COM/](https://WWW.KAGGLE.COM/)

R

- COMUNIDAD DE RSTUDIO:
 - [HTTPS://COMMUNITY.RSTUDIO.COM/](https://community.rstudio.com/)
- R-BLOGGERS:
 - [HTTPS://WWW.R-BLOGGERS.COM/](https://www.r-bloggers.com/)
- DATA CAMP:
 - [HTTPS://WWW.DATA CAMP.COM/](https://www.datacamp.com/)

- STACKOVERFLOW:
 - [HTTPS://STACKOVERFLOW.COM/QUESTIONS/TAGGED/R](https://stackoverflow.com/questions/tagged/r)

SHINY

- COMUNIDAD DE RSTUDIO:
 - <https://community.rstudio.com/c/shiny>
- GRUPO DE DISCUSIÓN DE SHINY EN GOOGLE:
 - <https://groups.google.com/forum/#!forum/shiny-discuss>
- STACKOVERFLOW:
 - <https://stackoverflow.com/questions/tagged/shiny>
- REFERENCIA DE SHINY:
 - <https://shiny.rstudio.com/reference/shiny/>

OTRAS APLICACIONES SHINY

- SHINY SERVER
 - [HTTPS://SUPPORT.RSTUDIO.COM/HC/EN-US/COMMUNITY/TOPICS/200092706-SHINY-SERVER](https://support.rstudio.com/hc/en-us/community/topics/200092706-Shiny-Server)
- SHINYAPPS.IO:
 - [HTTPS://GROUPS.GOOGLE.COM/FORUM/#!FORUM/SHINYAPPS-USERS](https://groups.google.com/forum/#!forum/shinyapps-users)
- RSTUDIO IDE:
 - [HTTPS://SUPPORT.RSTUDIO.COM/HC/EN-US/COMMUNITY/TOPICS/200022748-RSTUDIO-IDE](https://support.rstudio.com/hc/en-us/community/topics/200022748-RStudio-IDE)
 - [HTTPS://GITHUB.COM/RSTUDIO/RSTUDIO/ISSUES](https://github.com/rstudio/rstudio/issues)

BUENAS PRÁCTICAS PARA PEDIR AYUDA

- PARA OPTIMIZAR TUS BÚSQUEDAS:
 - BUSCA EN EL ARCHIVO SI TU PREGUNTA YA HA SIDO RESPONDIDA
 - ESCRIBE UN EJEMPLO REPRODUCIBLE QUE MUESTRE TU PROBLEMA
 - SE PRECISO
 - INCLUYE UN EXTRACTO DEL ERROR QUE TE OCURRE
 - SESSIONINFO()
 - SE RESPETUOSO Y AGRADECIDO

EJEMPLO REPRODUCIBLE

- UN EJEMPLO REPRODUCIBLE ES UN PROGRAMA O CÓDIGO EN R QUE:
 - PUEDE SER EJECUTADO POR CUALQUIERA
 - RECREA TU PROBLEMA
- UN BUEN EJEMPLO REPRODUCIBLE ES:
 - MÍNIMO: NO PONGAS CÓDIGO EXTRA. CÍÑETE AL PROBLEMA.
 - COMPLETO: TIENE QUE CONTENER TODO LO QUE CUALQUIERA PUEDA NECESITAR
- CONSEJOS:
 - NO TUS COMPARTAS DATOS A MENOS QUE SEA NECESARIO
 - UTILIZA GIST (DE GITHUB) PARA ALOJAR TUS EJEMPLOS
 - NO OLVIDES EXPLICAR QUÉ ES LO QUE FALLA

CHULETA DE SHINY

Shiny

Hoja de referencia
lee mas en shiny.rstudio.com

Shiny 0.10.0 Actualizado: 6/14



2. server.R Instrucciones que constituyen los componentes R de tu app. Para escribir server.R:

- A Provee server.R con el mínimo de código necesario, `shinyServer(function(input, output) {})`.
- B Define los componentes en R para tu app entre las llaves {} después de `function(input, output)`.
- C Guarda cada componente R destinados para tu interfaz (UI) como `output$<nombre componente>`.
- D Crea cada componente de salida con una función `render*`.
- E Dale a cada función render* el código R que el servidor necesita para construir el componente. El servidor notará valores reactivos que aparecen en el código y reconstruirá el componente cada vez que estos valores cambian.
- F Has referencia a valores en "widgets" con `input$<nombre del widget>`.

4. Reactividad Cuando una entrada (input) cambia, el servidor reconstruye cada salida (output) que depende de ella (también si la dependencia es indirecta). Puedes controlar este comportamiento a través de la cadena de dependencias.

1. Estructura Cada app es una carpeta que contiene un archivo `server.R` y comúnmente un archivo `ui.R` (opcionalmente contiene archivos extra)

funciones render*

function	espera	crea
renderDataTable	objetos como tablas	tabla DataTables.js
renderImage	lista atributos imágenes	imagen HTML
renderPlot	gráfica	gráfica
renderPrint	salida impresa	texto
renderTable	objetos como tablas	tabla simple
renderText	cadena de caracteres	texto
renderUI	objeto "tag" o HTML	elemento UI (HTML)

valores de entrada (input) son reactivos.
Deben estar rodeados por uno de:

- render* - crea un componente shiny UI (interfaz)
- reactive - crea una expresión reactiva
- observe - crea un observador reactivivo
- isolate - crea una copia no-reactiva de un objeto reactiv

3. Ejecución Coloca código en el lugar donde correrá la menor cantidad de veces

```
# carga paquetes, scripts, datos
❶ shinyServer(function(input, output) {❶
  # crea variables específicos para usuario
  output$texto <- renderText({
    input$title
  })
  ❷ output$gráfica <- renderPlot(
    x <- mtcars[, input$x], ❷
    ❸ y <- mtcars[, input$y]
    plot(x, y, pch = 16)
  )
})
❸
```

Corre una vez - código puesto fuera de `shinyServer` solo corre una vez cuando inicias tu app. Usalo para instrucciones generales. Crea una sola copia en memoria.

Corre una vez por usuario - código puesto dentro de `shinyServer` corre una vez por cada usuario que visita tu app (o refresca su navegador). Usalo para instrucciones que necesitas dar por cada usuario del app. Crea una copia para cada usuario.

Corre a menudo - código puesto dentro de una función `render*`, `reactive`, o `observe` correrá muchas veces. Usalo solo para código que el servidor necesita para reconstruir un componente UI después de que un widget cambia.

reactive - usa `reactive` para crear objetos que se usarán en múltiples salidas.

isolate - usa `isolate` para usar una entrada sin dependencia. Shiny no reconstruirá la salida cuando una entrada aislada cambia

observe - usa `observe` para crear código que corre cuando una entrada cambia, pero que no crea un objeto de salida.

`input$a` ↔ `output$z`

```
output$z <- renderText({
  input$a
})
```

`input$a` ↔ `x` ↔ `output$y` ↔ `output$z`

```
x <- reactive({
  input$a
})
output$y <- renderText({
  x()
})
output$z <- renderText({
  x()
})
```

`input$a` ↔ `input$b` ↔ `output$z`

```
output$z <- renderText({
  paste(
    isolate(input$a),
    input$b
  )
})
```

`input$a` ↔ `observer`

```
observe({
  input$a
  # código para correr
})
```