Shiny: Visualizacion de datos con R

Diplomado en Data Science 22, MatPUC

Joshua Kunst Fuentes jbkunst@gmail.com

Shiny: Visualizacion de datos con R

Version 2022

Johnson

Jue 21/7 - Sáb 23/7 - Jue 28/7 - Jue 4/8

Touring

Mie 27/7 - Mie 02/8 - Mie 10/8 - Mie 17/8

Programa

- Clase 1
  - Aplicación (web), ejemplos.
  - Introducción a shiny: Interfaz usuario y servidor
  - Reactividad
- Clase 2
  - Layouts
  - Integración HTMLWidgets
- Clase 3
  - Temas, templates y diseño
  - Compartir una app
- Clase 4
  - Expresiones reactivas
  - Orden de ejecución
  - Extensiones shiny

Ejercicio: Juntando inputs, laytous, htmlwidgets, {bslib}

Recuerde el ejercicio de la clase pasada: Generar una aplicación que considere/tenga:

- El app debe ser generada con un `navbarPage` (con una única sección `tabPanel`)
- En la única sección considere un `sidebarPanel`.
- La aplicación debe poseer un selector de fecha.
- Dada una fecha, la aplicación muestre una tabla con la información de sismos de la fecha asociada y un mapa. Ayúdese generando una función para descargar la información dada una fecha.
- Adicionalmente un *leaflet* con la ubicación de los sismos.
- Incorpore texto con información de la fecha seleccionada.

Adicionalmente:

- Incorpore la tabla de sismos a la aplicación con {DT}.
- Seleccione un retail, banco u otra marca de su preferencia y realice el ejercicio de incorporar los colores y tipografías. Utilizando `bs_theme( ... , primary = color, "navbar-light-bg" = otrocolor)`

Solución

Expresiones reactivas

Expresiones reactivas (*reactive expressions*)

La idea de expresiones reactiva es que podemos limitar que es lo que se (re)ejecuta al cambiar un input.

Una expresión reactiva es código R que usa un input y retorna un valor, la expresion se actualizará cuando el valor del (de los) inputs de los cuales dependen cambien.

Estos elementos se definen en el `server`. Se crea una expresion con la función `reactive` la que toma una expresión/código R entre `{}`, de la misma forma que las funciones `render` (`renderPlot`, `renderTable`):

```
server ← function(input, output) {
  ...
  elementoReactivo ← reactive({
    codigo R ...
  })
  ...

}
```

```
server ← function(input, output) {

  dataSismos ← reactive({
    fecha ← ymd(input$fecha)
    ...
    datos ← read_html(url) ▷
      html_table() ▷ ...
    datos

  })

  output$mapa ← renderLeaflet({

    datos ← dataSismos()

    leaflet(datos) ▷ ...
  })

  output$tabla ← renderDT({
    datatable(dataSismos())
  })

})
```

Expresiones reactivas: Ejemplo

Expresiones reactivas: Ejemplo *mejorado*

Ejercicio: Creando una expresión reactiva

Para el ejercicio anterior:

- Cree la expresión reactiva necesaria para reducir la cantidad de descargas.

- Incluya un `sliderInput` para filtrar los registros de la tabla (no así los registros/markers del mapa).

Solución

Orden de ejecución

Una vez

```
# A place to put code

ui <- fluidpage(

)

server <- function(input, output) {

  # Another place to put code

  output$map <- renderPlot({

    # A third place to put code

  })
}

shinyApp(ui, server)
```

Run once when app is launched

Una vez por usuario

```
# A place to put code

ui <- fluidpage(

)

server <- function(input, output) {

  # Another place to put code

  output$map <- renderPlot({

    # A third place to put code

  })
}

shinyApp(ui, server)
```

Run once each time a user visits the app

Muchas veces

```
# A place to put code

ui <- fluidpage(

)

server <- function(input, output) {

  # Another place to put code

  output$map <- renderPlot({

    # A third place to put code

  })
}

shinyApp(ui, server)
```

Run once each time a user changes a widget that output$map depends on

Extensiones para Shiny

Extensiones para Shiny

Podemos decir que ya existe un *shinyverso* dada la cantidad de paquetes que extienden shiny, agregando tanto diseños, nuevas funcionalidades, etc:

https://github.com/nanxstats/awesome-shiny-extensions

Menciones honrosas (*en mi opinión*):

- https://rinterface.github.io/shinydashboardPlus/
- https://fullpage.rinterface.com/
- https://waiter.john-coene.com/
- https://cicerone.john-coene.com/

Ejercicio: Incorporando extensiones

Para el ejercicio incluya las características de {waiter} y {ciceroene}

Solución

Cosas que no vimos...

... Que ni tan necesarias, pero que en futuro podrían ayudar!

- `bindCache`: Se puede asociar un resultado de un `render` a ciertos inputs, para guardar automáticament el resultado sin tener que volver a ejecutar el código dentro del $render`$. https://shiny.rstudio.com/articles/caching.html

- Bookmarking: La posibilidad de registrar la aplicación con ciertos inputs seleccionados. Como cuando comparter una aplicación de un retail con ciertos filtros. https://mastering-shiny.org/action-bookmark.html

- HTMLWidgets Proxys: Características de algunos HTMLWidgets para no *recontruir* el gráfico sino que actualizarlo.