Introducción a shiny y su uso como monitor para datos educativos

Natalia da Silva, @STAT_NT

Instituto de Estadística-UDELAR

Sobre mi

- Instituto de Estadística- Universidad de la República (IESTA-UDELAR), Montevideo Uruguay.
- PhD y Msc. en Estadística en Iowa State University, USA
- Intereses: aprendizaje supervisado, estadística computacional, visualización estadística y ciencia de datos.
- Co-fundadora of R-Ladies Ames, R-Ladies Montevideo, GURU::MVD y LatinR Conference tiempo para mandar trabajos hasta el 17 de Julio.

Contact info:

email: natalia.dasilva@fcea.edu.uy

twitter: @pacocuak

webpage: http://natydasilva.com

Sobre la charla

- Slides: https://github.com/natydasilva/ANALITICA_fundacion
- Introducción a shiny
- Problema aplicado a datos educativos
- Shiny app como monitor de uso de plataformas educativas

¿Qué es `shiny`?

- `shiny` es un paquete de R que nos permite crear aplicaciones interactivas web.
- Uno puede crear aplicaciones web complicadas sin saber, HTML, JavaScript o CSS.
- El código puede ser realizado completamente en R (o personalizado en HTML/JavaScript)
- Galería de RStudio sobre `shiny` http://shiny.rstudio.com/gallery/
- Versión más actualizada: https://github.com/rstudio/shiny

Documentación y ayuda para `shiny`

- Tutoriales de RStudio http://shiny.rstudio.com/tutorial/
- Libro fines 2020 https://mastering-shiny.org
- Lista de correos de `shiny` https://groups.google.com/forum/#!forum/shiny-discuss
- Si querés tener tu `shiny` en un servidor web: http://shiny.rstudio.com

Componentes claves de una shiny

En `shiny` trabajamos en dos códigos separados, el `ui` y el `server`.

- `ui`: "user interface" (interface de usuario), documento web en html, no necesito saber html lo hago desde R. Define como se ve la app, diseño de la misma (the front end) .
- `server`: Define como la app funciona. Conjunto de instrucciones seguidas por el servidor cuando cambia el `input` (the back end).
- Las expresiones reactivas son el tercer componente clave.

Reactividad

Cuando construyo una `shiny`, siempre debemos pensar en `inputs` y `outputs`

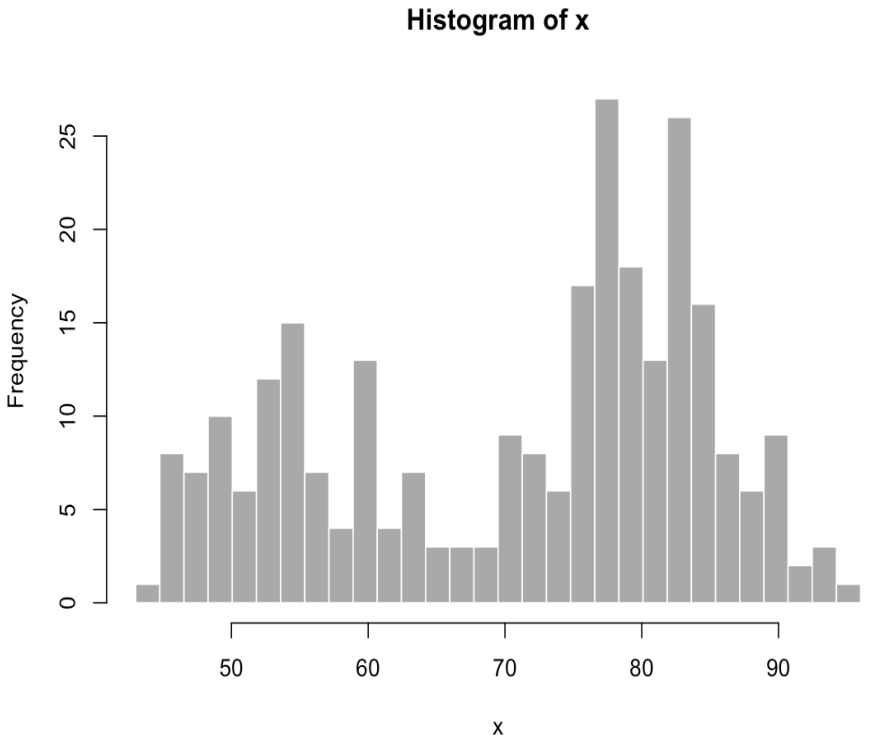
- Las aplicaciones shiny usan **programación reactiva**

  programación reactiva: permite automáticamente actualizar los `outputs` cuando cambian los `inputs`

- Valor del `input` => código R => valor del `output`

Ejemplo inicial

Datos del géiser old faithful)



Ejemplo shiny app

Necesitamos el paquete `shiny`, si no lo tienen:

```
install.packages("shiny")
library(shiny)
```

Clonen el repo https://github.com/natydasilva/ANALITICA_fundacion o bajen el zip.

Hay un archivo app.R que contiene la app inicial.

```
library(shiny)
runApp("docs/app.R")
```

Esta aplicación es la que surge por defecto cuando selecciono

File | New Project y selecciono New Directory y Shiny Web Application.

Ejemplo shiny app

```
ui <- fluidPage(

   # Título de la aplicación
   titlePanel("Datos del géiser old faithful"),

   # Barra lateral con control deslizante para el número de bins
   sidebarLayout(
      sidebarPanel(
         sliderInput(inputId = "bins",
                     label = "Número de bins:",
                     min = 1,
                     max = 50,
                     value = 30)
      ),

      # Muestro el gráfico de la distribución generada
      mainPanel(
         plotOutput(outputId = "distPlot")
      )
   )
)
```

Ejemplo shiny app

```
# Definimos en el server la lógica requerida para dibujar
# el histograma
server <- function(input, output) {

   output$distPlot <- renderPlot({
      # generamos los bins basados en input$bins de ui.R
      x    <- faithful[, 2]
      bins <- seq(min(x), max(x), length.out = input$bins + 1)

      # dibujamos el histograma con el número especificado de bins
      hist(x, breaks = bins, col = 'darkgray', border = 'white')
   })
}

# Corremos la aplicación
shinyApp(ui = ui, server = server)
```

Baby shiny app

Fuera de docs generá un nuevo archivo app.R y escribí el siguiente código

```
library(shiny)
ui <- fluidPage(
  "Mi primera shiny app!"
)
server <- function(input, output){

}

shinyApp(ui, server)
```

- Corré el código

Baby shiny app

Qué hace nuestro código?

1. Cargamos la librería `shiny`
2. `ui` define la interface de usuario, la página web en HTML donde las personas interactuan. En este caso en nuestra web pusimos "Mi primera shiny app!".
3. Definimos lo que hace la app en la función `server`. En este caso no hace nada porque está vacía.
4. Ejecutamos `shinyApp(ui, server)` que construye e inicializa la app desde el UI y el server

Baby shiny app

Correr la app y frenarla

Para correr:

- usando Run app en RStudio
- Ctrl + Shift + Enter
- `shiny::runApp()`

Para frenar:

- Presiono el signo de stop en la consola
- Presiono Esc en la consola
- Cierran la pestaña de la app

Shiny input y outputs

- `input`: recolecta los valores que el usuario decide
- `output`: las respuestas que observo cuando cambio un input, ejemplo un gráfico

Se agregan elementos a la app como argumentos de `fluidPage()`

```
ui <- fluidPage(
  #* Input() functions,
  #*  Output() functions

)
```

`fluidPage()` es una función de diseño (layout function) que determina la estructura básica de la página

Shiny input y outputs

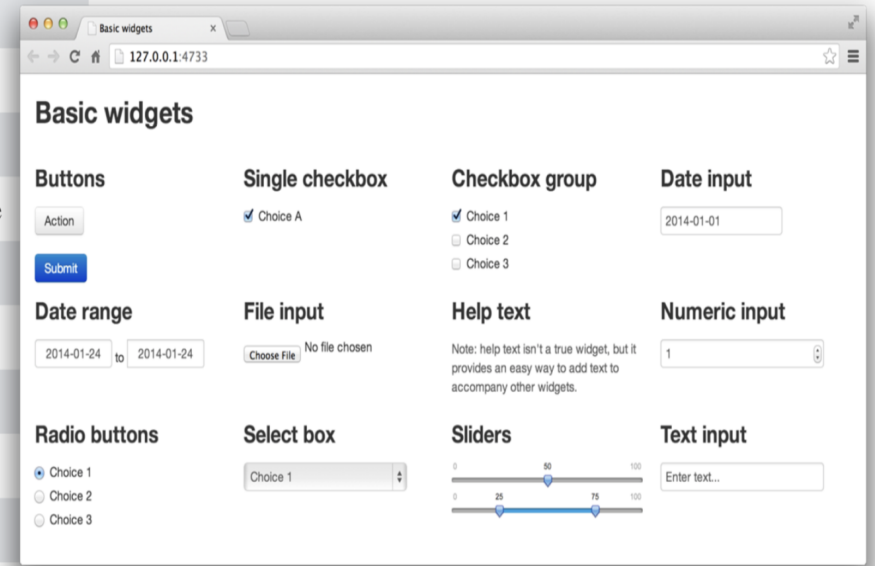
```
ui <- fluidPage(
sliderInput(inputId = "cant",
                      label = "Tamaño muestral:",
                      min = 1,
                      max = 50,
                      value = 30)

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

`sliderInput` es un input de control que permite al usuario interactuar con la app dando un valor definido en un rango.

Shiny input

Shiny input

function	widget
`actionButton`	Action Button
`checkboxGroupInput`	A group of check boxes
`checkboxInput`	A single check box
`dateInput`	A calendar to aid date selection
`dateRangeInput`	A pair of calendars for selecting a date range
`fileInput`	A file upload control wizard
`helpTextInput`	Help text that can be added to an input form
`numericInput`	A field to enter numbers
`radioButton`	A set of radio buttons
`selectInput`	A box with choices to select from
`sliderInput`	A slider bar
`submitButton`	A submit button
`textInput`	A field to enter text



Sintaxis para input

- `sliderInput( inputId = “num”, label = “Number of observations”, ...)`
- `inputId` es el nombre interno, tiene que ser una etiqueta única
- `label` se muestra como la etiqueta de la ventana

`...` parámetros específicos para distintos `inputs`, por ejemplo `sliderInput` tiene `min` y `max`

Shiny output

Output function	creates
htmlOutput	raw HTML
imageOutput	image
plotOutput	plot
tableOutput	table
textOutput	text
uiOutput	raw HTML
verbatimTextOutput	text

Sintaxis para output

- plotOutput(outputId = “hist”)
- `outputId` es un nombre interno que debe ser único

```
ui <- fluidPage(
sliderInput(inputId = "cant",
                      label = "tamaño muestral:",
                      min = 1,
                      max = 500,
                      value = 30),
plotOutput(outputId = "hist")

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

Sintaxis para output

- Parece no haber diferencia pero detrás en el html que se genera hay un espacio reservado para un gráfico.
- Solamente agregamos el nombre del gráfico pero aún no especificamos que tipo de gráfico quiero.
- Para especificar el tipo de gráfico tenemos que especificarlo en la función `server`.

Recapitulando

- Iniciar cada shiny con el mismo código minimal del inicio (baby shiny)
- Agregá nuevos elementos como argumentos de `fluidPage()`
- Creá `inputs` reactivos con una función `*Input()`
- Mostrá resultados reactivos con la función `*Output()`
- Usá la función `server` para conectar `inputs` con `outpus`

Función `server`

La funcion `server` conecta los `inputs` con los `outputs`

Hay tres reglas básicas para escribir la función `server`

server regla 1

Si constuímos un objeto `output` debemos guardarlo en `output$hist`, se llama `hist` porque así se llama el gráfico que quiero mostrar
`plotOutput(outputId = "hist")`

server regla 2

Construir objetos que se muestran con `render{tipo}`

Las funciones `render` se crean para producir `outputs` de diferente tipo (ej: texto,tablas y figuras) que son traducidos a html.

Estas funciones son a menudo pareadas con funciones `{tipo}Output` Ejemplo `renderPrint()` con `verbatimTextOutput()` para mostrar resumenes estadísticos con ancho fijo. `renderTable()` con `tableOutput()` para mostrar los datos de `input` en una tabla.

`renderPlot` hace un gráfico reactivo que es adecuado para asignar a un `output`.

```
server <- function(input, output) {
output$hist <- renderPlot({(hist(rnorm(500)))
  })
}
```

`server` regla 3

Usar el valor `input` con `input$`

Si queremos que algo cambie cuando se selecciona un `input`. Acá está la reactividad en acción

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

La dependencia es creada impicitamente porque usamos `input$cant` al interior de una función `output`. `input$cant` es usada con el valor por defecto en UI y se actualizará automaticamente cuando el valor cambie. Esta es la clave de la reactividad, los `outputs` se recalculan cuando cambian los `inputs`.

Funciones `render*()`

render function	creates
`renderImage`	images (saved as a link to a source file)
`renderPlot`	plots
`renderPrint`	any printed output
`renderTable`	data frame, matrix, other table like structures
`renderText`	character strings
`renderUI`	a Shiny tag object or HTML

SHINY

```
ui <- fluidPage(
sliderInput(inputId = "cant",
                      label = "tamaño muestral:",
                      min = 1,
                      max = 500,
                      value = 30),
plotOutput(outputId = "hist")

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

Recapitulando server

- Usá el server para conectar inputs y outputs
- 3 reglas, guardar los outputs que se construyen en `output$`, construí el objeto `output` con una función `render*()`, accedé a los valores generados en input con `input$`

Si seguimos estas reglas la shiny va a usar los Inputs para crear Outputs reactivos

Ejemplo de aplicación con datos educativos

Plan Ceibal en Uruguay:

- Se implementa en Uruguay 2010 como política pública de carácter universal el Plan Ceibal que forma parte de la iniciativa mundial One Laptop per Child (OLPC, 2005), (computadoras para el sitema educativo).
- Plan Ceibal ha implementado el “modelo uno a uno” que consiste en otorgar un dispositivo (laptop o tablet) a cada alumno y docente de la enseñanza pública básica (Educación Inicial y Primaria, y Educación Media Básica).
- Ha logrado generar igualdad de acceso a la tecnología, así como se asegura el acceso a internet en todos los centros educativos públicos.
- Dentro del Plan Ceibal hay distintas plataformas de aprendizaje donde los niños desarrollan distintas actividades.

Monitor de uso de plataformas educativas con shiny

- Utilizando información de la plataforma educativa CREA2 el objetivo es desarrollar herramientas estadísticas para evaluar y monitorear el uso de la misma. Trabajamos con datos de primaria (4to, 5toy 6to).
- Calcular algunos resumenes útiles para monitorear la plataforma educativa a diferentes niveles de análisis (clase, grado, escuela, departamento) en distintas ventanas temporales.
- Datos, actividad de los estudiantes en CREA2 (2015 y 2017) diarios.

Monitor de uso de plataformas educativas con shiny

- Se recibe un conjunto de datos con formato longitudinal donde cada observación (cada fila) contiene la información del total de actividad realizada por el alumno en cada día que tiene alguna actividad.
- Son 1.355.972 registros correspondientes a 120276 alumnos de los cuales 87915 tienen alguna actividad.
- 77 variables, interacción del estudiante con la plataforma CREA, nivel de actividad (ejemplo, días de ingreso), actividad en los foros (comentarios posteados), envío de tareas (envíos calificables) y medidas generales como el total de ingresos a la plataforma.
- Otras variables de características del alumno, (sexo, edad, grado, o, contexto socioeconómico del centro, departamento, localidad).
- A su vez se cuenta con información del dispositivo y el tipo de conexión a internet.

Monitor de uso de plataformas educativas con shiny

- Indice de Engagement, medida de la intensidad y regularidad en el uso de la plataforma.
  .
- Acotado $[0, 1]$.
- Valores positivos indican uso de la plataforma.

Shiny app:

http://164.73.240.157:3838/App-Ceibal/