R + Interactividad + Web Shiny Server

Creando Aplicaciones web en R con Shiny

Eduardo San Miguel Martín

Quien soy / Sobre mi:

- Licenciado en Psicología UCM
- Investigador Diplomado en Ciencias Cognitivas UCM
- Autor del paquete de visualización interactiva fisheyeR
  http://cran.r-project.org/web/packages/fisheyeR/index.html
- Consultor de Analítica de Negocio en Indra Sistemas

Indice

- Motivación
- Un poco de historia
- Qué es Shiny
- Características
- Reactive Programming
- Shiny Input - Output
- Servidor Shiny
- Enlaces

Motivación

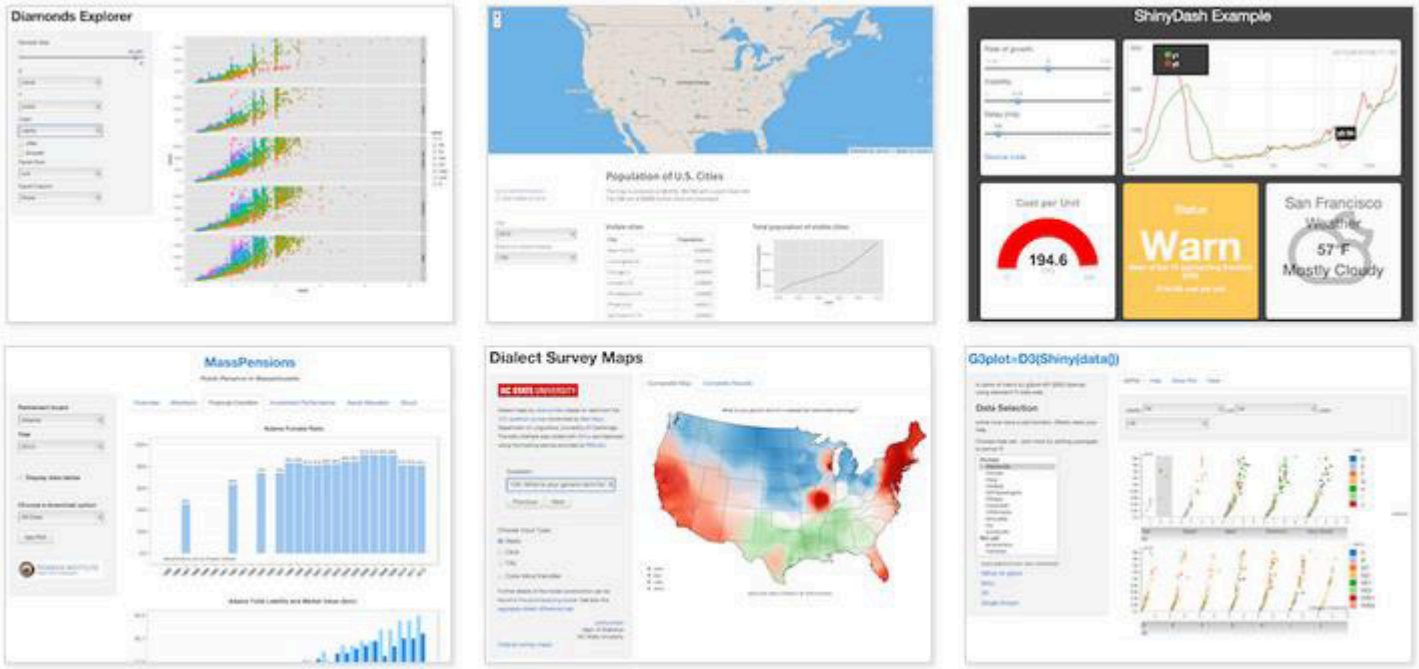
- R ofrece excelentes características para manipular, analizar y representar datos.
- R se utiliza esencialmente como aplicación de escritorio (local).
- ¿Cómo compartir los desarrollos realizados en R de una manera rápida y flexible?
- ¡Llevar R al navegador!

Un poco de historia

- Soluciones anteriores a Shiny:
  - rApache
  - Rserve (Java, C++, C#, Python, Ruby, .NET)
  - Rpad
  - gWidgetsWWW
  - deployR
  - Rook
  - ...
  - Custom hacks

Qué es Shiny

- Shiny es un paquete en R creado en 2012 por Rstudio (GPLv3) para desarrollar aplicaciones Web utilizando R.



Hello Shiny

ui.R

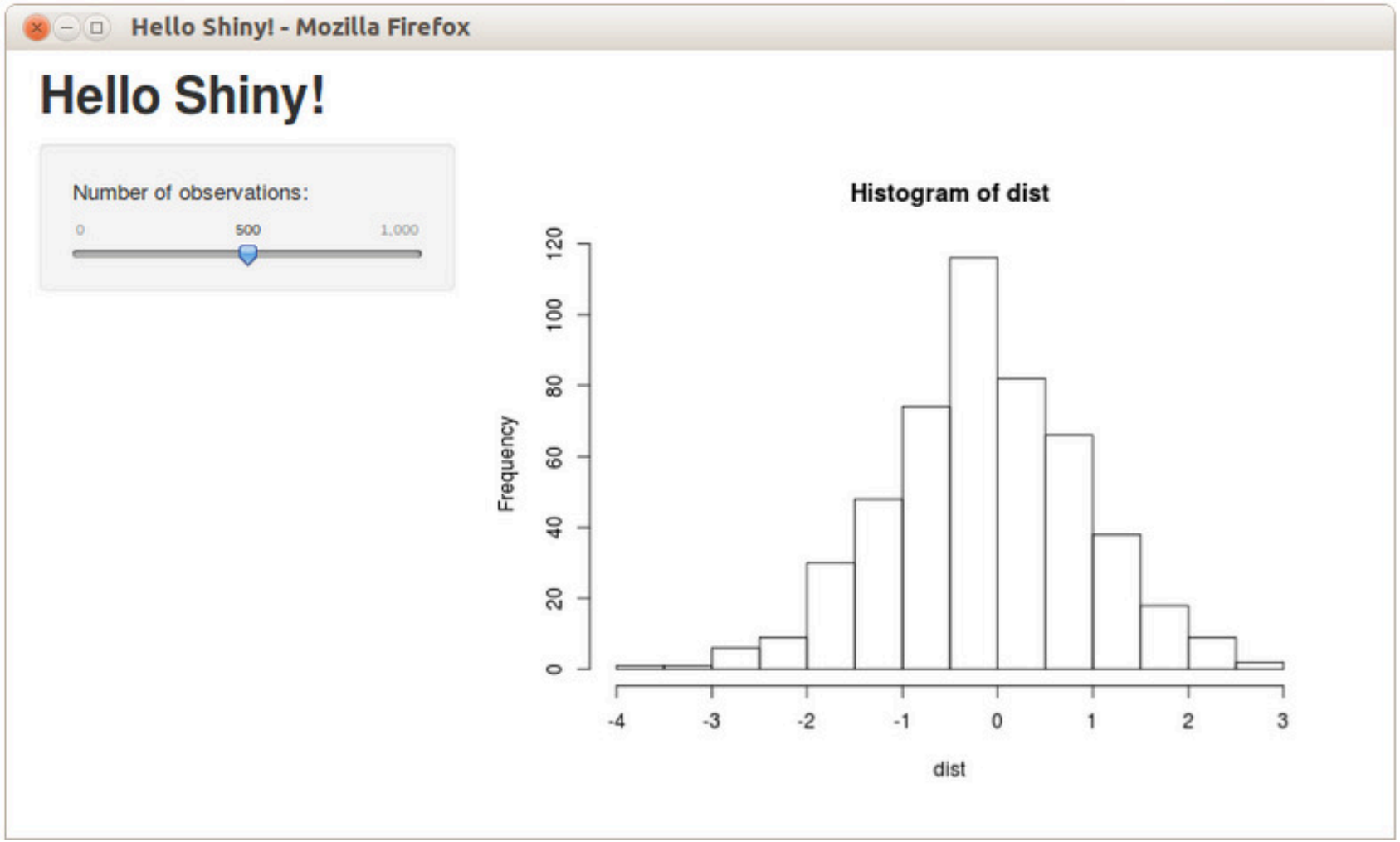
```
library(shiny)

# Define UI for application that plots random distributions
shinyUI(pageWithSidebar(

  # Application title
  headerPanel("Hello Shiny!"),

  # Sidebar with a slider input for number of observations
  sidebarPanel(
    sliderInput("obs",
                "Number of observations:",
                min = 1,
                max = 1000,
                value = 500)
  ),

  # Show a plot of the generated distribution
  mainPanel(
    plotOutput("distPlot")
  )
))
```

Hello Shiny

server.R

```
library(shiny)

# Define server logic required to generate and plot a random distribution
shinyServer(function(input, output) {

  # Expression that generates a plot of the distribution. The expression
  # is wrapped in a call to renderPlot to indicate that:
  #
  #  1) It is "reactive" and therefore should be automatically
  #     re-executed when inputs change
  #  2) Its output type is a plot
  #
  output$distPlot <- renderPlot({

    # generate an rnorm distribution and plot it
    dist <- rnorm(input$obs)
    hist(dist)
  })
})
```

Hello Shiny


Hello Shiny! - Mozilla Firefox
Hello Shiny!
Number of observations:
0
500
1,000
Histogram of dist
Frequency
0
20
40
60
80
100
120
-4
-3
-2
-1
0
1
2
3
dist

Características

- Programación Reactiva (Reactive Programming).
- HTML5/CSS3 y Javascript + node.js y R.
- Basado en tecnología websockets (websockets package).
- Permite mostrar salida gráfica de R.
- Soporte navegadores IE8/IE9.
- Incluye CSS3 UI basado en Twitter Bootstrap.
- Disponible en CRAN: install.packages("shiny")

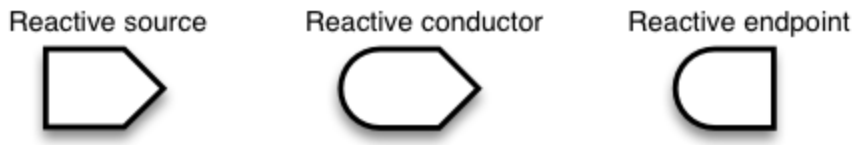
Reactive Programming

```
a <- 3

b <- a + 2

a <- 7

b == ?
```

Imperative: b = 5
Reactive: b = 9

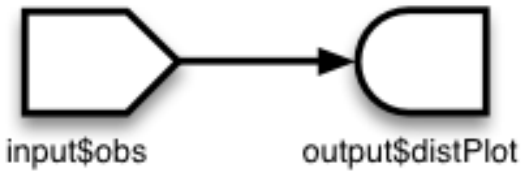
Reactive Programming

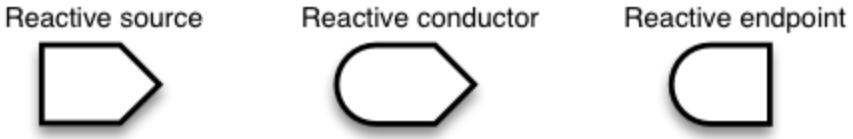
- La programación Reactiva enfatiza el uso de:
  - Valores que cambian en el tiempo
  - Expresiones que registran esos cambios



Reactive Programming

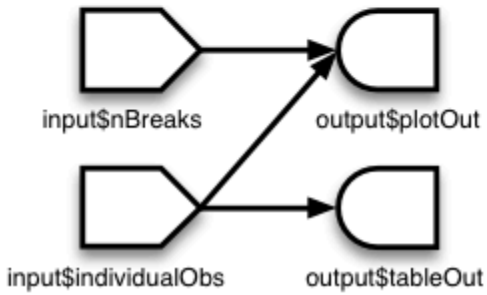
```
shinyServer(function(input, output) {
  output$distPlot <- renderPlot({
    hist(rnorm(input$obs))
  })
})
```

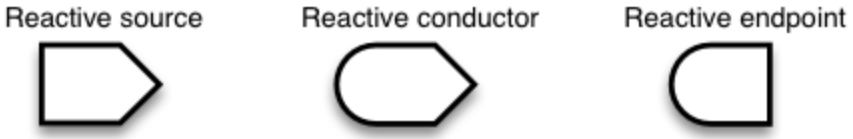




Reactive Programming

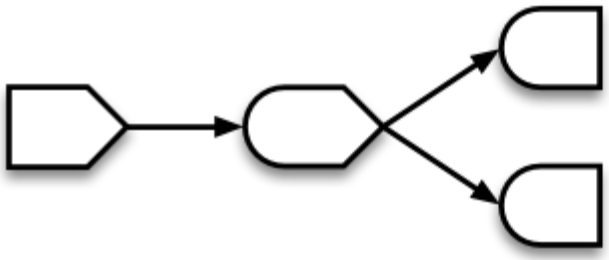
```
shinyServer(function(input, output) {
  output$plotOut <- renderPlot({
    hist(faithful$eruptions, breaks = as.numeric(input$nBreaks))
    if (input$individualObs)
      rug(faithful$eruptions)
  })

  output$tableOut <- renderTable({
    if (input$individualObs)
      faithful
    else
      NULL
  })
})
```

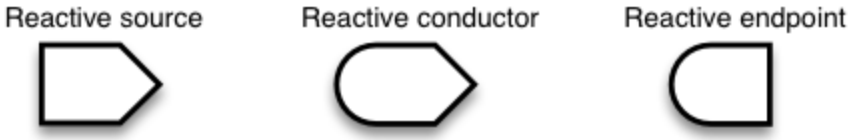




Reactive Programming

```
fib <- function(n) ifelse(n<3, 1, fib(n-1)+fib(n-2))

shinyServer(function(input, output) {
  currentFib         <- reactive({ fib(as.numeric(input$n)) })

  output$nthValue    <- renderText({ currentFib() })
  output$nthValueInv <- renderText({ 1 / currentFib() })
})
```





Shiny Input & Output

- Shiny implementa los principales INPUTs en HTML (Text, RadioButtons, Sliders, SelectButtons, etc…).
- Implementa predefinidos los principales elementos de salida de R: Tablas (data.frames), Gráficas (plot) y texto.
- La definición de elementos de entrada/salida es extensible por medio de HTML, CSS , Javascript y R.

Shiny Input

checkboxInput(inputId, ...)
dateInput(inputId, ...)
dateRangeInput(inputId, ...)
numericInput(inputId, ...)
radioButtons(inputId, ...)
selectInput(inputId, ...)
submitButton(text = "Apply Changes")
textInput(inputId, ...)

Shiny Output

htmlOutput(outputId)
imageOutput(outputId, ...)
plotOutput(outputId, ...)
tableOutput(outputId)
textOutput(outputId)
uiOutput(outputId)

Servidor Shiny

- Servidor Local implementado en el paquete.
- RStudio ofrece “Glimmer” y “Spark”
  - Free managed hosting platforms for Shiny (Beta)
- RStudio‘s Shiny-Server
  - Open source
  - Node.js
  - Linux
- Amazon Machine Image on EC2
  - AMI: ami-e2a3358b “ShinyServer.”

Enlaces de Interés

- Shiny en CRAN

http://cran.r-project.org/web/packages/shiny/index.html

- Shiny Home

http://www.rstudio.com/shiny/

- Tutorial Shiny (en inglés)

http://rstudio.github.io/shiny/tutorial/

- Shiny Server (GitHub)

https://github.com/rstudio/shiny-server

- Spark and Glimmer

https://rstudio.wufoo.com/forms/shiny-server-beta-program/

Muchas Gracias