ANALYTICS TAKEOVER

GROWTH ACCELERATION PARTNERS

Building Interactive Dashboards with Shiny

Sergio Morales E.

Wait! Before we begin:

Find this presentation at:
fireblend.com/shiny_talk.pdf

...and all code samples at:
github.com/fireblend/shiny_talk

What is Shiny?

“R package that makes it easy to build **interactive web apps** based on **data.**”

ANALYTICS TAKEOVER

A Super-Quick R Primer

- **R:** Download at *https://cran.r-project.org/*
- **RStudio:** Download at *https://rstudio.com/*
- **Functional programming**
- `<-` for **variable assignment**
- **1-indexed** data structures
- Wanna learn? ***https://r4ds.had.co.nz/***

Shiny Quick Start

Install, load and run:

```
install.packages("shiny")

library(shiny)

runExample("01_hello")
```

(There are 11 of these!)

Let’s see what one of these looks like!

fireblend.shinyapps.io/ejemplo

GROWTH ACCELERATION PARTNERS

The Structure of a Shiny App

The UI Object

Controls the **layout** and **appearance** of your app

The Server Function

Defines the **logic** and interactivity **mappings**

Code Skeleton

```
library(shiny)

ui <- ...

server <- ...

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

Building a User Interface

- Start by invoking the **fluidPage** function, a generic responsive layout.
- Use this as a **container** for other components.
- The function’s **nesting structure** mirrors the **visual hierarchy** in the resulting UI.

What will this look like?

```
ui <- fluidPage(
  titlePanel("Hello World!"),
  sidebarLayout(position = "right",
                sidebarPanel("This is a side panel"),
                mainPanel("This is a main panel!")
  )
)
```

Some Layout and Higher-Level Hierarchy Components

- **sidebarLayout( )** for side + main layout.
- **fluidRow( )** + **column( )** for grid-based layouts.
- **tabsetPanel( )** + **tabPanel( )** for tab-based UI.
- **navlistPanel( )** for navigation lists.
- Plenty others!

Adding some *style*

- Most **HTML** tags have an analogous Shiny function you can wrap text with (**p( )**, **hX( )**, **strong( )**, **img( )**, etc).

- Shiny's visual style is entirely based on Bootstrap, you can specify alternate themes (css files) using the **theme** parameter for **fluidPage( )**.

Example time!

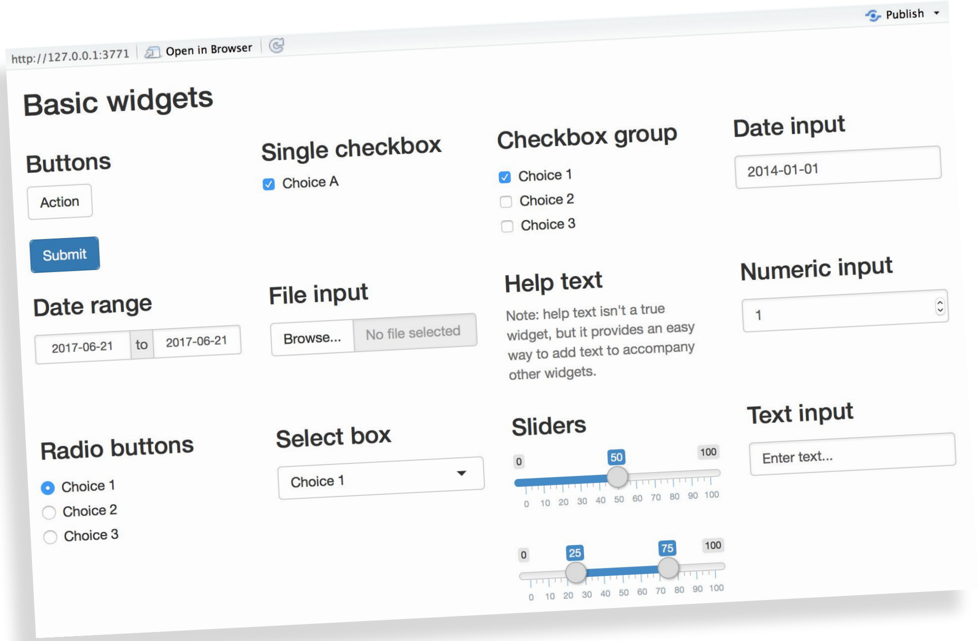
fireblend.shinyapps.io/ejemplo2

GROWTH ACCELERATION PARTNERS

Interactive Components/Widgets

There's a whole lot of 'em!

- actionButton
- radioButtons
- checkbox**Input**
- date**Input**
- file**Input**
- numeric**Input**
- slider**Input**
- select**Input**
- etc…



Note: help text isn't a true widget, but it provides an easy way to add text to accompany other widgets.

Check out ***http://shiny.rstudio.com/gallery/widget-gallery.html***

Adding Reactive Output

2 Simple steps:

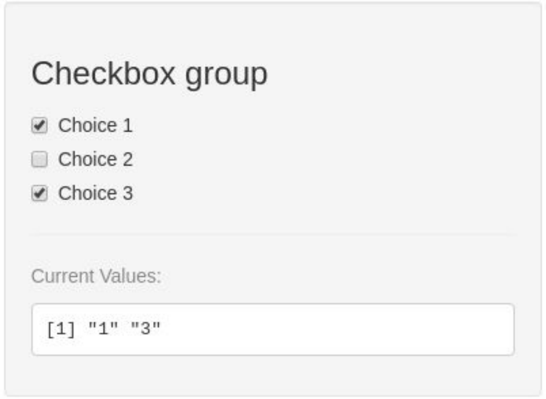
- Declare an **input object** and an **output object** in the layout. This can be text, images, tables, dataframes, raw HTML, etc…

- Specify **how to display** the output in the server function, and **map it to an interactive widget**.

Retrieving a widget’s value

All widgets follow the **same behavior** for value retrieval:

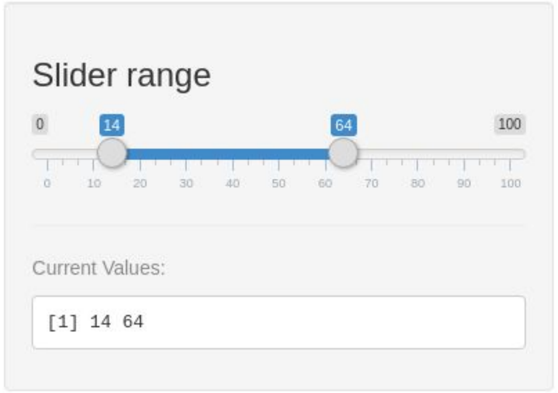
- Must have an **id** to be referenced on server function
- id is used to retrieve a **value array**
- Remember, **arrays are 1-indexed**!


Checkbox group

- [x] Choice 1
- [ ] Choice 2
- [x] Choice 3

Current Values:

```
[1] "1" "3"
```


Slider range

Current Values:

```
[1] 14 64
```

A Basic Interactive App

Layout Function:

```
ui <- fluidPage(

  titlePanel("Example"),
  sidebarLayout(

    sidebarPanel(
      selectInput("var",
        label = "Choose an option",
        choices = c("Option A", "Option B")
      )
    )

    mainPanel(
      textOutput("selected_var")
    )
  )
)
```

Server Function:

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

  output$selected_var <- renderText({
    paste("You chose: ", input$var)
  })

}
```

Code Execution Behavior: What executes when?

When application is first executed

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

Everytime a user visits the application

```
    output$selected_var <- renderText({
```

Everytime a widget triggers an output update

```
        paste("You chose: ", input$var)
    })
}
```

Adding Visualizations

Most R visualization packages are compatible with Shiny: **ggplot2**, **lattice**, **leaflet**, etc.

Just plug the **generation call** into the server function!

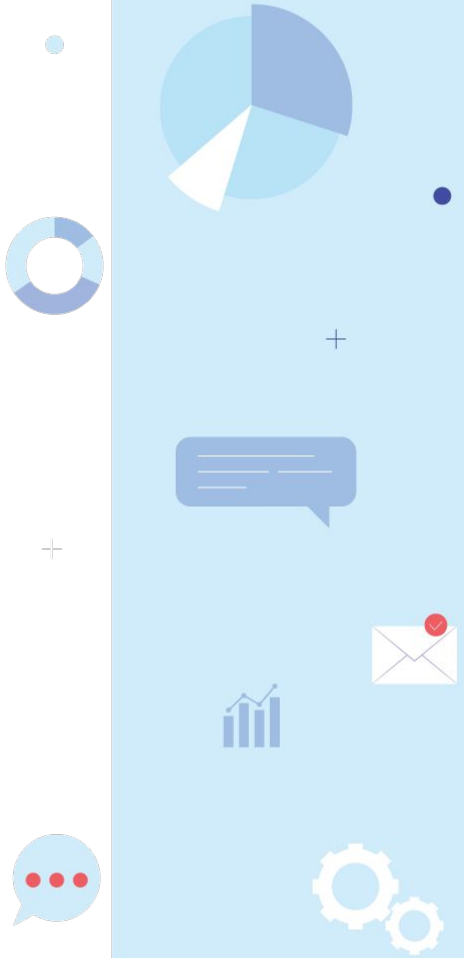
```
server <- function(input, output) {
  output$plot_points <- renderPlot({
    ggplot(data, aes(x = input$var_1, y = input$var_2)) +
      geom_point(colour = "red")
  },
  height = 400, width = 600)
}
```

Reactive Expressions: Caching Data

When working with **non-static data**, we should **limit the amount of times** it is loaded.

We can establish **reactive expressions** that cache data until their contents become **outdated** due to widget interaction.

For this, we declare a **reactive** block within our server.



Reactive Expressions

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

  data <- reactive({
    begin = input$begin_date
    end = input$end_date
    <...retrieve data...>
  })

  output$plot_points <- renderPlot({
    ggplot(data(), aes(x=input$v1, y=input$v2))+
    geom_point(colour = "red")
  },
  height = 400, width = 600)

}
```

Reactive block only called when the cached data has become **outdated** due to inputs it depends on.

Putting it all together!

`fireblend.shinyapps.io/pokemon`

GROWTH ACCELERATION PARTNERS

Preparing a Shiny App

In order to easily **publish** a Shiny app, its directory structure must be formatted in the following way:

```
<app name>
    /app.R
    /DESCRIPTION
```

Or

```
<app name>
    /ui.R
    /server.R
    /DESCRIPTION
```

Sharing & Publishing Applications

Depending on your purpose, there are several ways to share your Shiny apps online.

If the recipient is also running Shiny on RStudio, they can pull your app directly from a hosted zip file, a Github repo or a Github gist with the **runUrl(...)**, **runGithub(...)** and **runGist(...)** functions.

```
library(“shiny”)
runGitHub("shiny_talk", "fireblend", subdir = "pokemon/")
```

Sharing & Publishing Applications

Alternatively, you can **embedded your apps into a webpage** using an iframe, however they **must be running on a Shiny server**.

You can:

- **Setup your own:** *github.com/rstudio/shiny-server*

- **Use a free/paid service:** *shinyapps.io*
  (Free account includes hosting for **5** shiny apps)

Thank you!

Questions?

GROWTH ACCELERATION PARTNERS

ANALYTICS TAKEOVER