Introduction to Shiny

16TH Annual UT-KBRIN Bioinformatics Summit 2017
Presenter: Joshua Lambert
joshua.lambert@uky.edu
twitter.com/joshuawlambert
github.com/joshuawlambert

Friday April 21, 2017

First Let's Install Shiny in RStudio!

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

- Also, you can click on the packages tab in RStudio, then Install Button, then type "Shiny", then "Install"
- 10 people, CRAN, Steve Jobs
- More on shinyFSA.org, R package: ***install.packages("rFSA",type="source")***

What is Shiny?

> ***Shiny*** *is an open source* ***R*** *package that provides an elegant and powerful web framework for building web applications using R. Shiny helps you turn your analyses into interactive web applications without requiring HTML, CSS, or JavaScript knowledge. -- RStudio.com*

- These *apps* can be deployed locally, on the web, or on your own server.
- Users with a basic familiarity with ***R*** can create applications within ***RStudio*** easily and deploy them for free at `http://www.shinyapps.io/`.
- Visit Shiny Gallery, ShowMeShiny.com, or Dean Attali to see some neat examples and generate ideas about how Shiny can help you present your data and research questions in unique ways.
- RStudio offers many tutorials and there is a very active Shiny Google Group where you can post questions to Rstudio staff and other users.

Shiny App Basics

- Every* Shiny Application has two files: ui.R and server.R.

1. ***ui.R*** is an *R* file where you specify the user interface. Here you get to choose what your application will look like. Do you want a dropdown box? Or maybe a list of checkboxs for the user to select? What do you want to display? Histograms? Maps?
2. ***server.R*** is an *R* file where server side R calculations are done. Complex Algoirthm? Very specific Plot? That stuff goes here.

- What about a Bar Optimizer that recommends drinks based on your own bar's inventory? The choice is yours!

A Simple Shiny Example

```
library(shiny)
runExample("01_Hello")
```

- Or view source code on Github

Hello Shiny! ui.R File

```
library(shiny)

# Define UI for application that draws a histogram
shinyUI(fluidPage(

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

  # Sidebar with a slider input for the number of bins
  sidebarLayout(
    sidebarPanel(
      sliderInput("bins",
                  "Number of bins:",
                  min = 1,
                  max = 50,
                  value = 30)
    ),

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

Hello Shiny! server.R File

```
library(shiny)

# Define server logic required to draw a histogram
shinyServer(function(input, output) {

  # Expression that generates a histogram. 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$distPlot <- renderPlot({
    x    <- faithful[, 2]  # Old Faithful Geyser data
    bins <- seq(min(x), max(x), length.out = input$bins + 1)

    # draw the histogram with the specified number of bins
    hist(x, breaks = bins, col = 'darkgray', border = 'white')
  })

})
```

Inputs Types

- There are many types of inputs that you can add to your shiny app.
- To add them to your application you would simply add one of these functions to your ui.R file:
  - 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)
  - helpText (Help text that can be added to an input form)
  - numericInput (A field to enter numbers)
  - radioButtons (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)
- Most these functions has a required inputId field where you specify the name of the input. On the server side, when you want to use that input you simply use input$NAME_OF_INPUT to render the value the user provided.

Printing R Output to the Shiny App

- This is usually done with a function in the ui.R file assigning the outputs variable name (and other UI parameters) and an assignment of R proceedures to that will produce the output desired in the server.R file.
- Examples of Outputs are
  - Scatterplots, histograms, piecharts, bar graphs, and other things that typically go in the plots window. Use plotOutput(ui), and renderPlot(server).
  - Static and dynamic text, summary, and other R text statements. Use textOutput(ui), and renderText/renderPrint(server).
  - Images. Use imageOuput(ui), and renderImage(server)
  - Datatables and Tables. Use dataTableOuput/tableOutput(ui), and renderDataTable/renderTable(server).
  - Dynamic user interface. Use uiOuput(ui), and renderUI(server).

Reactive Values

- If you wish to use Shiny your ultimate goal is to take inputs from the user and use those in your R code and display the results on the screen.

  input values => R code => output values

- Reactive Values: change when user changes the input field or interact with the app.
- Reactive Expressions: change when any of the input fields contained in the expression change.

Creating your own application

- Allow RStudio to create the necessary folder and files to get started. You can do this by going to RSTUDIO:“File>New Project>New Directory>Shiny Web Application>” and select what you would like your app to be called, and where it should be located.
- RStudio will create a folder, with a server.R, ui.R, and .Rproj file in it.
- From here you can edit out the generic ingredients and put in new modules.

An Example

```
library(shiny)
runGitHub(repo = "KBRIN17", username = "joshuawlambert",subdir = "Dataset Viewe
```

- Or View code on Github