PharmaSUG 2015 - Paper DV12

R you ready to show me Shiny for PharmaSUG 2015

Amulya R Bista, Pharmacyclics Inc, Sunnyvale, CA, USA
Jeff Cai, Pharmacyclics Inc, Sunnyvale, CA, USA

ABSTRACT

This paper demonstrates the power of open source applications, R together with Shiny, and illustrates an efficient way to review source data and perform exploratory analysis without the need for extensive programming. Shiny is a web application framework for R developed by RStudio that allows creating simple-to-use web application with dynamic data filters and real-time exploratory analysis (RStudio, Inc). Shiny allows users to integrate additional R packages, JavaScript and CSS into the backend program for any customization. In addition, Shiny also includes enhanced data security and authentication systems such as LDAP, Active Directory and Google Account. The Shiny web app is robust enough to adjust its layout to any platforms and also can be used by beginner as the app only requires point and click for data virtualization.

INTRODUCTION

With increasing numbers of clinical trials, comes a vast amount of source data that needs to be reviewed and cleaned. Currently, we use SAS generated “data dump” listings and patient profiles to review data which is time-consuming for both reviewers and programmers. Interactive web application built using Shiny and R can help achieve greater efficiency for both the programmer and subject matter expert reviewer during data review and exploratory analysis. The Shiny application automatically updates the data tables and graphs in real-time, which reduces the time spent to generate listings and patient profiles. In addition, the backend requires minimal maintenance support. Addition of real-time data filters helps reviewers explore subject level details individually or in any chosen subset of subjects. Several examples of the Shiny web applications that are presented in this paper are built using R, Shiny, JavaScript and CSS.

SHINY APPLICATION WORKFLOW

Shiny web app can be created using only R, use of CSS and other packages are optional. There are two key aspects of Shiny web application development.

1. SERVER DEPLOYMENT

In order for a Shiny app to run, we have to create the accompanying Shiny server. Currently Shiny server can only be installed on a Linux based machine such as Ubuntu and Centos. RStudio provides details in the administrator guide to install and maintain the server (RStudio, Inc). We can host multiple Shiny apps per server therefore eliminating server cost. In addition, Shiny server allows allocating resources based on specific applications, which can be helpful if we have multiple apps with different priorities.

2. CREATING BACKEND PROGRAM USING R AND SHINY

Once we have the latest version of R and the Shiny package installed within our system, we can start developing Shiny apps. Shiny follows a pre-defined R script filename Server.R and UI.R for any specific application, which need to be in same directory location. In addition, we can also create any additional programs to perform intermediate variable derivation or data set creation. RStudio has implemented various types of widgets such as Input Box, Slider, Data Range, and so on, that can be used for filtering data and producing analysis. Static graphics can be easy to create, but difficult to visualize if we have lots of data points. Therefore, to better assist reviewers with data visualization, we can integrate tooltip functionality into Shiny graph.

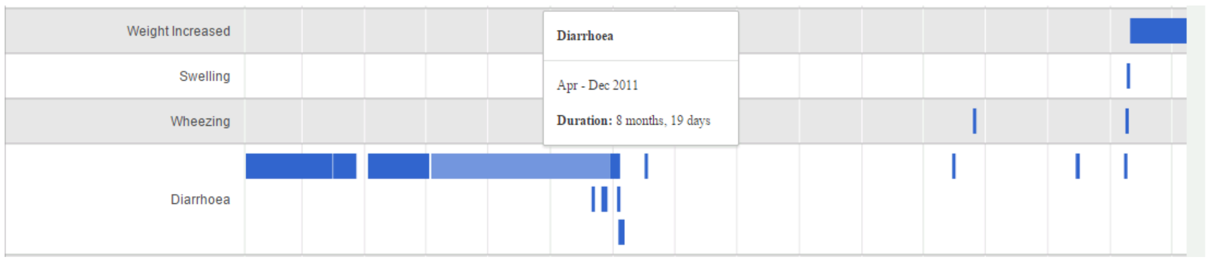

Figure 1: Timeline Graph with Tooltip Functionality

The most intuitive part of using Shiny app for data review and exploratory analysis is availability of dynamic UI and real time data rendering to present graphics and tables. Once we select a desired filter, analysis or data changes instantly with respected to the filter. We can even create dynamic filters that can adjust in real time without any changes to Shiny backend. In term of data format compatibility, we can import any source data format into R and Shiny with the help of various available packages event for the proprietary sas7bdat format of SAS datasets. However, this does require some intermediate processing to adjust different data attributes.

a. SERVER.R AND UI.R

When we initiate a Shiny app, the Shiny server retrieves the Server.R which then pulls the User interface from UI.R and any other additional R program that is needed for the app to process the data. Shiny is also fully equipped to integrate any additional CSS and JavaScript within the web application if a user wants to customize the layout of the UI. In addition, Shiny can also address the issue with big data computation which takes time to process. We can define reactive expression within Shiny app that lets us control every part of the app, preventing any unnecessary backend processing (RStudio Inc). Example of reactive expression is shown below.

Server.R

```
dataInput <- reactive({
getSymbols(input$symb, src = "yahoo",
from = input$dates[1],
to = input$dates[2],
auto.assign = FALSE)
})
```

Shiny utilizes the reactive programming model to concurrently change graphs or tables when a user changes any widget (Gonzalez, 2013). Implementation of reactive programming model helps user visualize the data more effectively and efficient without the need for additional processing. Figure 2 shows the workflow chart of a Shiny app in relation to clinical data. Every time a user requests information, Server.R fetches the information from source data sets. Source datasets are assigned as read only which prevents any accidental change.

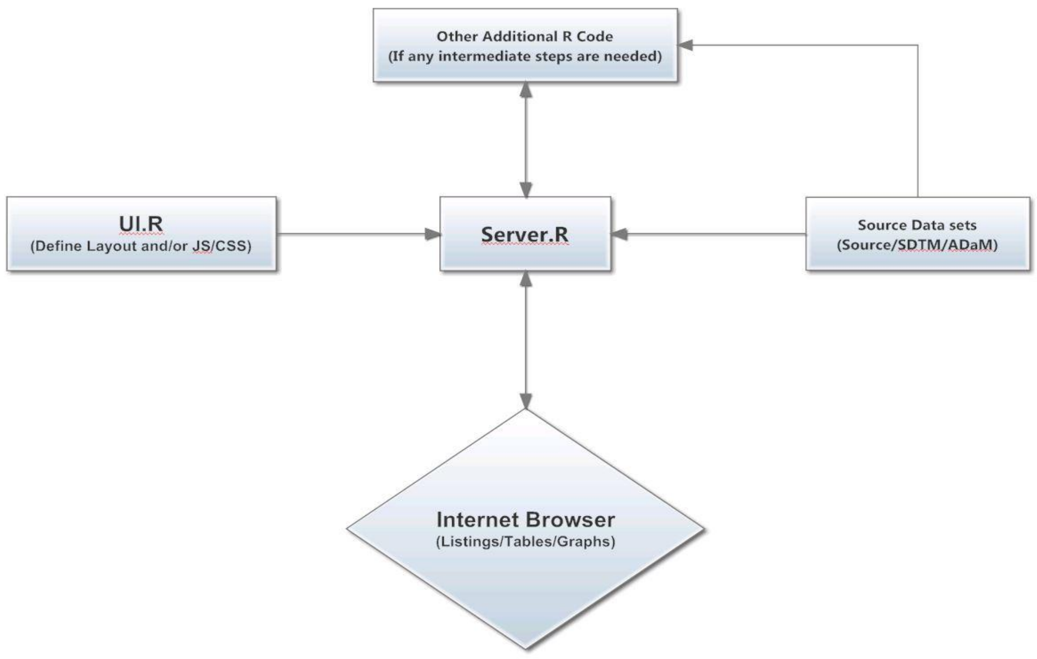

Figure 2: Work Flow Chart of a Shiny App

Below is an example of Simply Shiny App that only requires Server.R and UI.R [(RStudio Inc)

Server.R

```
# This is the server logic for a Shiny web application.
  library( Shiny )
  #Define any other library that is required.

  #Define Intermediate R Program if needed.
  #source( "[filename].R" )

  ShinyServer(function(input, output) {

    output$main_plot <- renderPlot({

      hist(faithful$eruptions,
        probability = TRUE,
        breaks = as.numeric(input$n_breaks),
        xlab = "Duration (minutes)",
        main = "Geyser eruption duration")

      if (input$individual_obs) {
        rug(faithful$eruptions)
      }

      if (input$density) {
        dens <- density(faithful$eruptions,
            adjust = input$bw_adjust)
        lines(dens, col = "blue")
      }

    })
})
```

UI.R

```
ShinyUI(bootstrapPage(

  selectInput(inputId = "n_breaks",
      label = "Number of bins in histogram (approximate):",
      choices = c(10, 20, 35, 50),
      selected = 20),

  checkboxInput(inputId = "individual_obs",
      label = strong("Show individual observations"),
      value = FALSE),

  checkboxInput(inputId = "density",
      label = strong("Show density estimate"),
      value = FALSE),

  plotOutput(outputId = "main_plot", height = "300px"),

  # Display this only if the density is shown
  conditionalPanel(condition = "input.density == true",
    sliderInput(inputId = "bw_adjust",
        label = "Bandwidth adjustment:",
        min = 0.2, max = 2, value = 1, step = 0.2)
  )

))
```

WHY SHINY?

The primary reason to use custom built Shiny app for clinical data review and exploratory analysis is its need-based scalability and resiliency. Every clinical trial has its own requirements and analysis that may not be same. Therefore, developing custom apps using the R /Shiny model can provide great flexibility and versatility. And also, any technical issues can be solved in-house since it is all open-source software making it even more efficient tool. Furthermore, community support for R and Shiny is rapidly growing, with many public discussion boards and groups, that together with complete administration guide and user manual available to support different aspects of the development process.

CONCLUSION

Shiny web application can help to automate the data review process and increase efficiency for any exploratory analysis. Shiny is evolving and extremely promising tool that can be implemented for various tasks. Additional examples can be found at http://shiny.rstudio.com/gallery/.

REFERENCES

Gonzalez, S. (2013, 04 12). *Data Visualization: Reactive Functions in Shiny*. Retrieved 03 12, 2013, from Data Community DC: http://www.datacommunitydc.org/blog/2013/04/data-visualization-reactive-functions-in-shiny

RStudio Inc. (n.d.). *Shiny by RStudio*. Retrieved 02 20, 2015, from RStudio: http://shiny.rstudio.com/gallery/

RStudio, Inc. (n.d.). *Shiny by RStudio*. Retrieved 02 20, 2015, from RStudio: http://shiny.rstudio.com/tutorial/lesson1/

ACKNOWLEDGMENTS

We would like to thank our colleagues Brian Munneke and Linda Gau for their comments on this paper and also provide valuable input on the Shiny web application project.

RECOMMENDED READING

- Shiny Tutorial at http://shiny.rstudio.com/tutorial/

CONTACT INFORMATION

Your comments and questions are valued and encouraged. Contact the author at:

Name: Amulya R Bista
Enterprise: Pharmacyclics Inc
Address: 999 E Arques Ave.
City, State ZIP: 94085
Work Phone: 408-215-3760
E-mail: amulyabista@yahoo.com