Exploratory Data Analysis with R

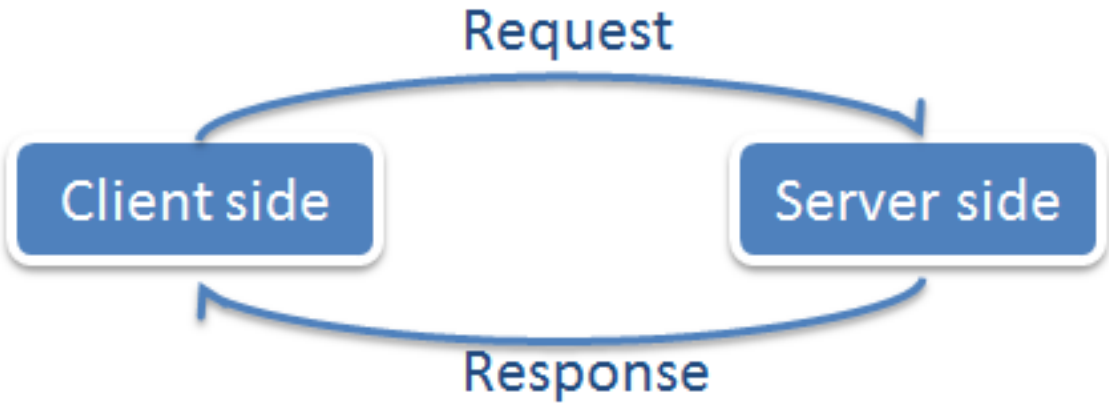
Building a shiny app

Xuemao Zhang
East Stroudsburg University

November 16, 2022

Overview

- Reactive Web Framework



http://littleactuary.github.io/blog/Web-application-framework-with-Shiny/

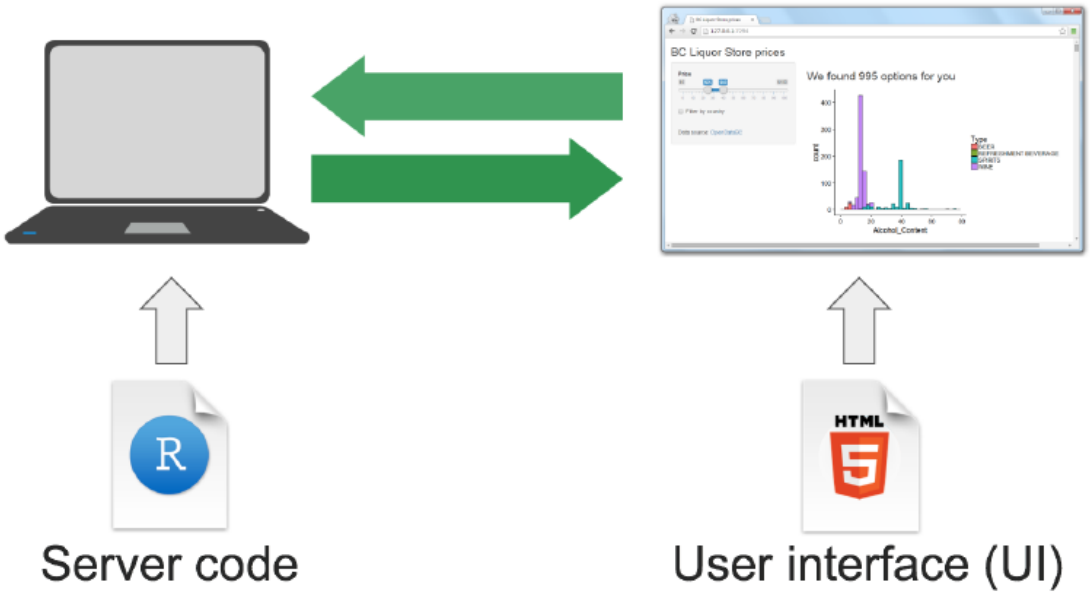
> *"Reactive Systems are highly responsive, giving users effective interactive feedback"*

Reactive Framework and Data Science

"The impact of data scientists' work depends on how well others can understand their insights to take further actions"

- Benefit 1: Interactive display and manipulation of data
- Benefit 2: No installation required
- Benefit 3: Easy to develop and share with clients and project teams
- Benefit 4: Open source library

R shiny reactive architecture

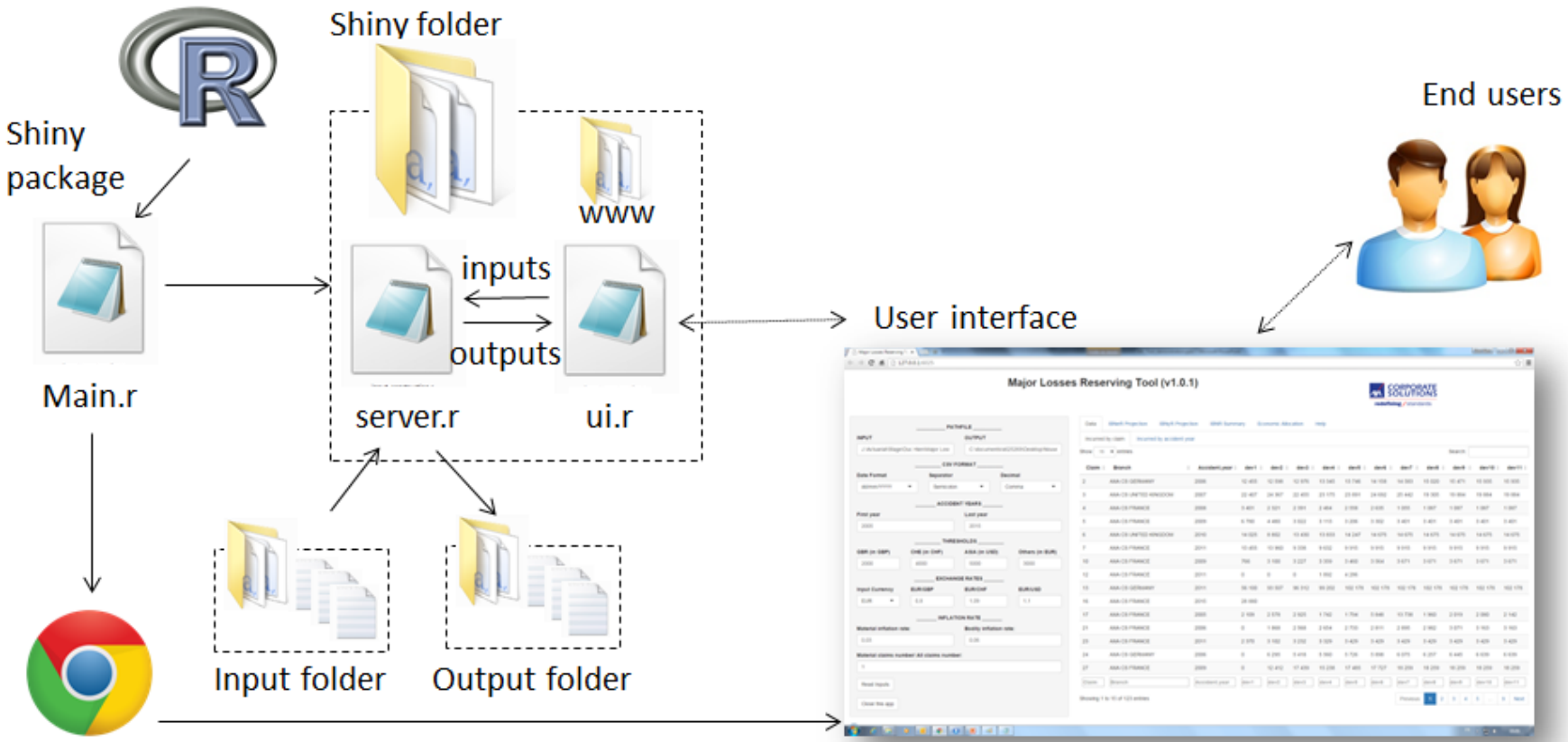


http://cl.indiana.edu/

R shiny reactive architecture

- 1. Shiny is an R package for building interactive web applications
- 2. Open-Sourced by RStudio
- 3. Uses web sockets (new HTTP):
  - Interactive communication sessions between the user's browser and a server without having to poll the server for a reply
- 4. Entirely extensible - custom input/output

Shiny Library



http://littleactuary.github.io/blog/Web-application-framework-with-Shiny/

Shiny Gallery - Get Inspired

https://www.rstudio.com/products/shiny/shiny-user-showcase/

A Shiny Example

- Create a "Shiny Web App. . . " from Rstudio
  - **Note.** Apps need to be in their own directory
- The file `app.R` will be created if you choose to use a single file.
- Or you can create two files: `server.R` and `ui.R`
- You should now see a button labelled `Run App`. Clicking on this will run it. Try it!

A Shiny Example

```
# Define UI for application that draws a histogram
ui <- fluidPage(

    # Application title
    titlePanel("Old Faithful Geyser Data"),

    # Sidebar with a slider input for 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")
        )
    )
)
```

A Shiny Example

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

    output$distPlot <- renderPlot({
        # generate bins based on input$bins from ui.R
        x    <- faithful[, 2]
        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')
    })
}
```

Structure

The structure of the app is very simple. It has a slider to control the number of bins of the histogram. Let's take a look at how it is constructed.

- There are two functions: `ui()`, `server()`
  - `ui()`: sets up the slider and the canvas to draw on
  - `server()`: does the computations
- Global code can be added at the top of the file (before ui()) such as reading and processing data, loads libraries, etc.

User Interface

- `fluidPage` allows the size of components to be driven by the size of the browser window. That is, fluidPage creates a display that automatically adjusts to the dimensions of your user's browser window.
- `titlePanel` puts a header on the page
- `sidebarLayout` sets up the container for the user interface. It always takes two arguments:
  - `sidebarPanel` puts in the side panel:
    - `sliderInput` defines the slide input, asking for the variable, we call it `bins` used in server() to control the number of bins;
    - "Number of bins:" is the label shown to the user, and
    - the initial number of bins here is in the `value` vector
  - `mainPanel` makes the drawing canvas: `plotOutput` specifies we want to make a plot in this panel, and the code instructions to use are called `distPlot`. You need to look in the server function to find the code in `output$distPlot`

Server

- The function `output$distPlot` contains code to create the plot, based on the user input.
- `renderPlot` indicates the output is a plot
- The code is the same as the plotting code see thus far, *except for* `breaks = bins`, where `bins` is given by the ui() input.

Shiny Inputs

Shiny has many different input options: please google input options

https://shiny.rstudio.com/tutorial/written-tutorial/lesson3/

- `actionButton()` - creates a clickable button
- `checkboxInput()` and `checkboxGroupInput()`
- `dateInput()` - calendar to select a date
- `dateRangeInput()` - select a range of dates
- `fileInput()` - upload a file
- `numericInput()` - input a numeric value
- `radioButtons()` - select one or more items
- `sliderInput()` - slide along a range of values
- `textInput()` - input a string

Shiny Outputs

Shiny also has many output options:

- `renderDataTable()` - outputs an interactive, sortable data table
- `htmlOutput()` - output html elements
- `renderPlot()` - output an R plot
- `renderPlotly()` - output a plotly plot
- `renderPrint()` - output text from print() in R
- `renderTable()` - output an HTML table
- `renderText()` - output text from R
- `renderUI()` - output a custom part of the user interface
- `renderImage()` - print an image to the page

Other User Interface Options

- `tabsetPanel()` - make multiple different output views (i.e. a plot in one tab, a data table in another)
- `helpText()` - create additional text to help users navigate your applet
- `submitButton()` - only update outputs when this button is clicked
- `conditionalPanel()` - only show certain UI options when conditions are met (i.e. if a certain tab is open, or a certain input is selected)

Deploy an app

- Sign up for an account on https://www.shinyapps.io/
- Authenticate your account
- You may need to do some setup in your session, e.g. install the library `rsconnect`

Your turn: Gapminder Example

- Create a Shiny app for the Gapminder data:
  - Include a menu, so that the user can choose a continent to display.
  - Make each plot interactive by year using `ggplotly()`.

Hint:

- Inside the `sidebarPanel()`, use `selectInput()`.
- In the ui function, change `plotOutput` to `plotlyOutput`.
- In the server function, change `renderPlot` to `renderPlotly`.
- To minimize the code writting, you can modify the code from the flexdashboard lecture.

R Shiny lab: Create your own Shiny App

- Join forces with your neighbours or work alone
- You need to make your own app, or interactive document, on a topic of your choice
- Some ideas are data from TidyTuesday(https://github.com/rfordatascience/tidytuesday) or fivethirtyeight(https://github.com/fivethirtyeight/data)
- Your app needs to have
  - at least one interactive plot
  - some GUI element like a menu or checkboxes

Resources

- RStudio Tutorial: http://shiny.rstudio.com/tutorial/
- RStudio Written Tutorial: https://shiny.rstudio.com/tutorial/written-tutorial/lesson1/
- Deploy your app for others to use: https://www.shinyapps.io/
- Shiny Setup, Showcase, and Server: http://shiny.rstudio.com
- Community discussion: https://community.rstudio.com

License