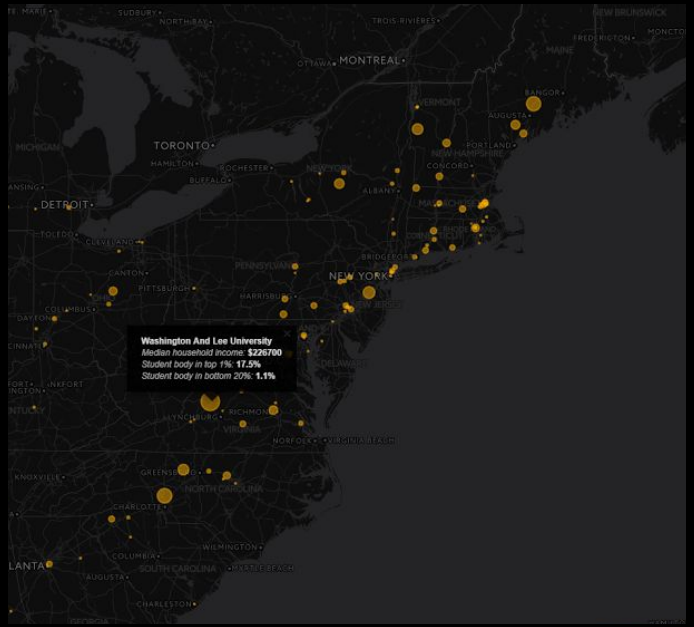
Building Map-based Dashboards with Shiny and Leaflet

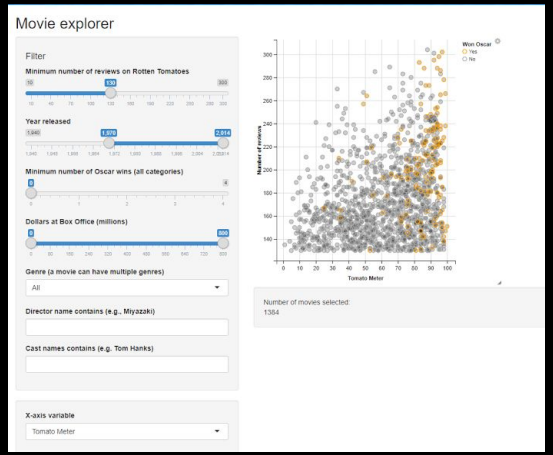
LEAFLET

- JavaScript library for interactive maps
- Open-source
- Light-weight, simple, user friendly
- Additional libraries/plugins to use leaflet in QGIS, Python and R



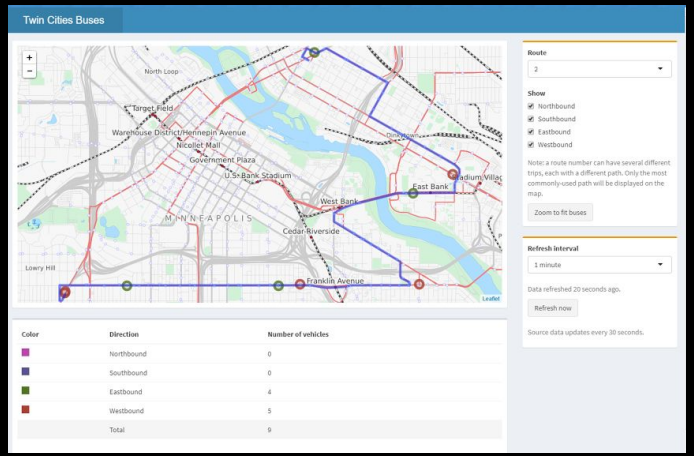
SHINY

- R package providing a framework to build web applications
- Build apps with nothing but R code
- Incorporate R's data analysis functionality on the back-end
- Deploy applications on the web for free with Shiny Server



SHINY

- Create interactive visualizations using R bindings to JS libraries (e.g. Leaflet, DataTables, D3, Plotly)
- UI built with bootstrap framework
- Optionally customize styling by writing your own CSS



WHY USE SHINY?

- Incorporate all of the analytical capacity of R into user-friendly graphical interfaces
- You don’t need to be a web developer
- Shiny apps are very quick
- It is easy to get from 0 to something in very little time and even less code

WHAT ARE THE LIMITATIONS?

- Doesn’t have nearly the functionality of a more robust web framework
- No selective access and permissions
- User Interface layout can be clunky

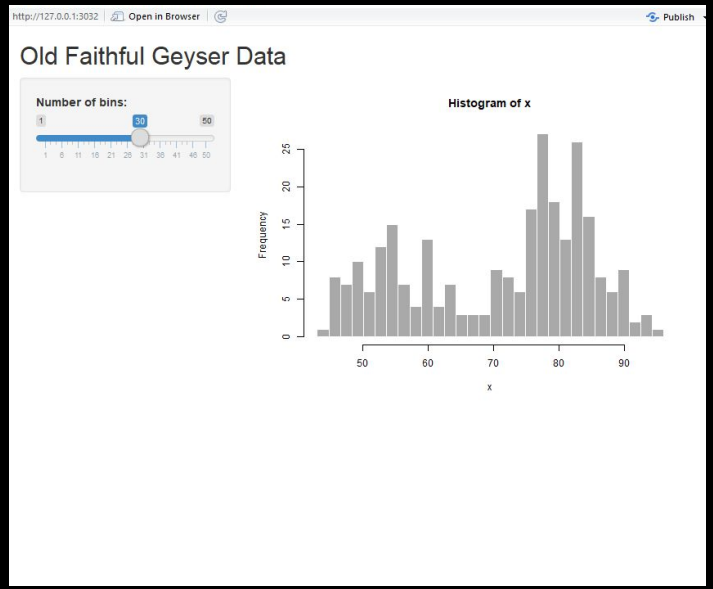
OVERALL

- Not for full-service websites
- Good for interactive visualizations and dashboards
- Focus on analysis

UI + SERVER

```r
#
# This is the user-interface definition of a Shiny web application.
# run the application by clicking 'Run App' above.
#
# Find out more about building applications with Shiny here:
#
#    http://shiny.rstudio.com/
#

library(shiny)

# Define UI for application that draws a histogram
shinyUI(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")
    )
  )
))
```

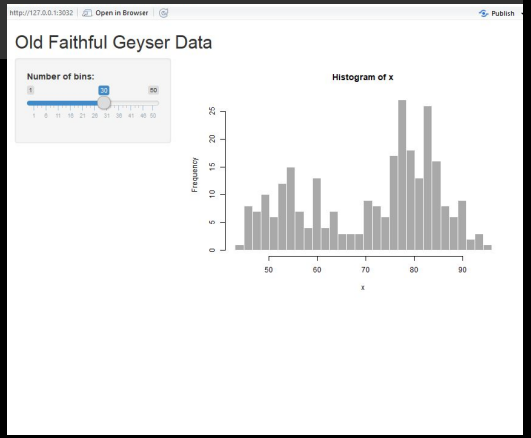
```r
#
# This is the server logic of a Shiny web application. You can ru
# application by clicking 'Run App' above.
#
# Find out more about building applications with Shiny here:
#
#    http://shiny.rstudio.com/
#

library(shiny)

# Define server logic required to draw a histogram
shinyServer(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')

  })

})
```



UI

- Fluidpage adjusts to different devices with different resolutions
- Predefined ‘sidebarLayout’ template
- Choose from a number of input widgets

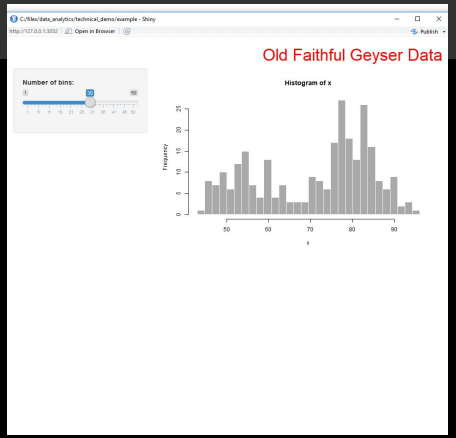
```
9
10 library(shiny)
11
12 # Define UI for application that draws a histogram
13 shinyUI(fluidPage(
14
15   # Application title
16   titlePanel("Old Faithful Geyser Data"),
17
18   # Sidebar with a slider input for number of bins
19   sidebarLayout(
20     sidebarPanel(
21       sliderInput("bins",
22                   "Number of bins:",
23                   min = 1,
24                   max = 50,
25                   value = 30)
26     ),
27
28     # Show a plot of the generated distribution
29     mainPanel(
30       plotOutput("distPlot")
31     )
32   )
33 ))
34
```



UI

- Customize styling with html/CSS
- Specify types of UI elements with output functions
- Label UI elements to reference them in Server Script

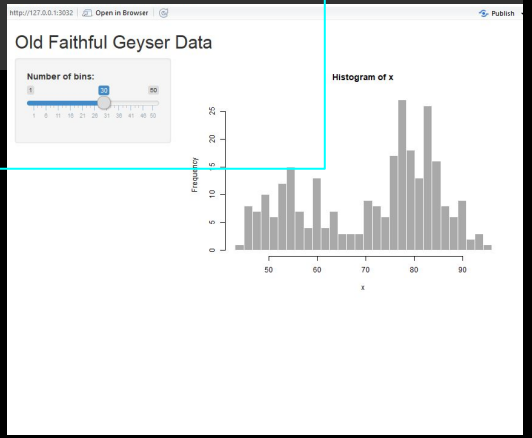
```
library(shiny)

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

  # Application title
  h1("Old Faithful Geyser Data", style = 'text-align:right; color: red'),

  # 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")
    )
  )
))
```



SERVER

- Function of two lists of inputs and outputs
- Additional functions output to specific UI elements by label
- Reference inputs by UI label as well

```
 9
10 library(shiny)
11
12 # Define server logic required to draw a histogram
13 shinyServer(function(input, output) {
14
15   output$distPlot <- renderPlot({
16
17     # generate bins based on input$bins from ui.R
18     x    <- faithful[, 2]
19     bins <- seq(min(x), max(x), length.out = input$bins + 1)
20
21     # draw the histogram with the specified number of bins
22     hist(x, breaks = bins, col = 'darkgray', border = 'white')
23
24   })
25
26 })
27
```

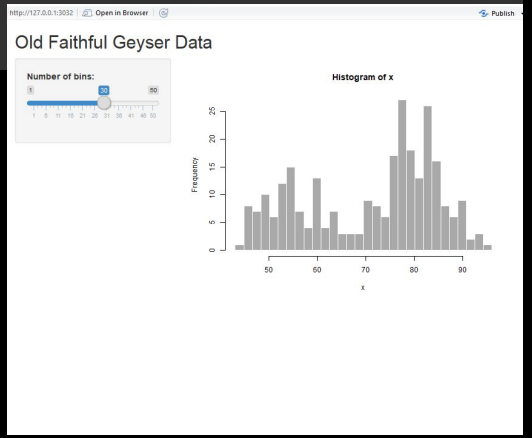


SERVER

- Render functions run when an input parameter is changed

- Trigger server code on different click events with additional reactive expressions

```
 9
10 library(shiny)
11
12 # Define server logic required to draw a histogram
13 shinyServer(function(input, output) {
14
15   output$distPlot <- renderPlot({
16
17     # generate bins based on input$bins from ui.R
18     x    <- faithful[, 2]
19     bins <- seq(min(x), max(x), length.out = input$bins + 1)
20
21     # draw the histogram with the specified number of bins
22     hist(x, breaks = bins, col = 'darkgray', border = 'white')
23
24   })
25
26 })
27
```



AN EXAMPLE

App: https://simonkassel.shinyapps.io/azavea_technical_demo_shiny/

Code: https://github.com/simonkassel/azavea-technical-demo-Shiny

Or run directly from an R console:

```
install.packages("shiny") # if not already installed

shiny::runGitHub("simonkassel/azavea-technical-demo-Shiny")
```