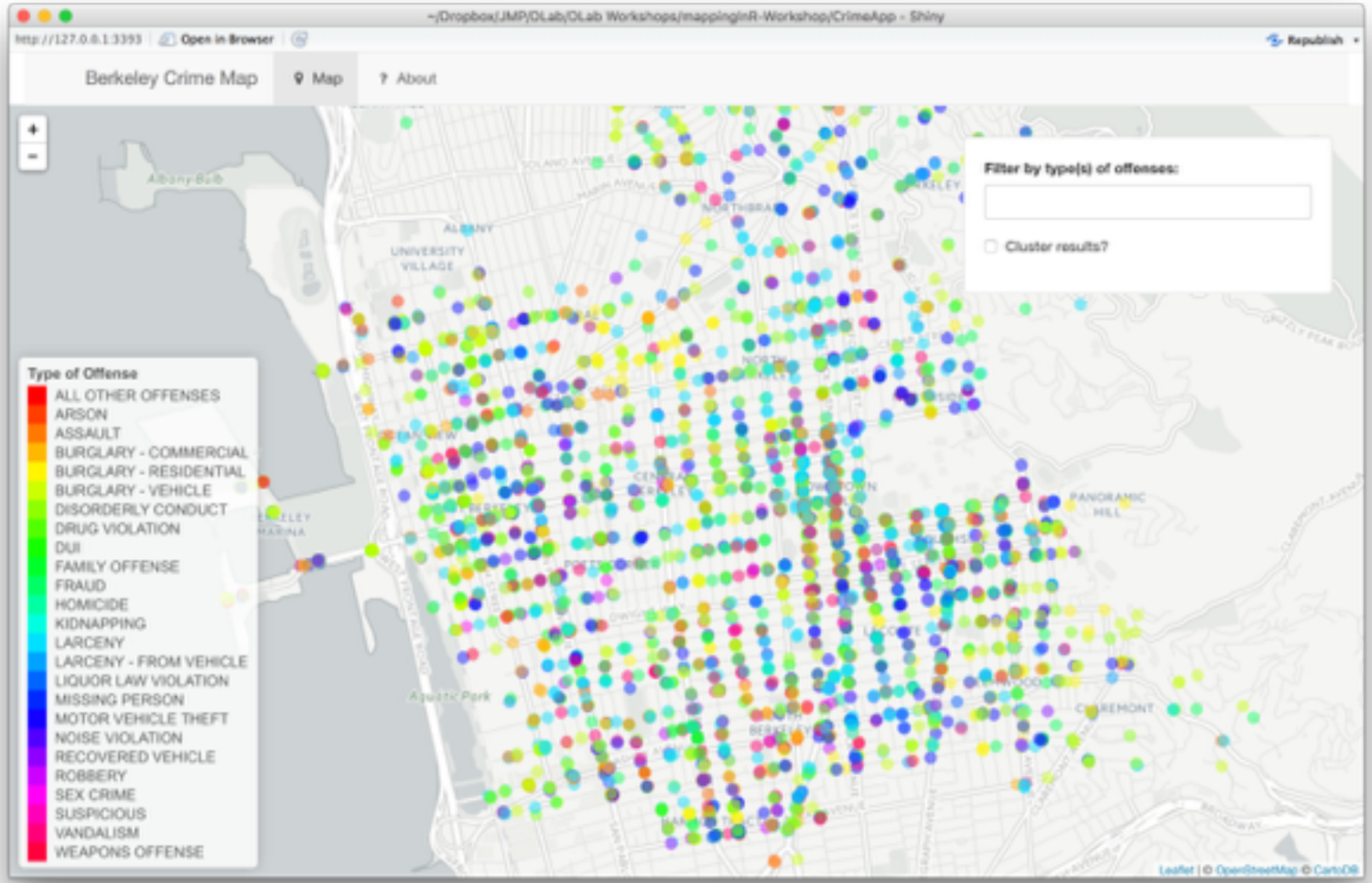
Leaflet + R + Shiny



Download slides and sample code files from: http://tinyurl.com/shiny-leaflet

Also make sure you have R and RStudio installed please!

R Install: https://cran.cnr.berkeley.edu/

RStudio download: https://www.rstudio.com/

Class Survey

1. Who has never used R?

2. Who has used…

   - R a little, but is a beginner?
   - Shiny?
   - Leaflet?
   - Shiny + Leaflet?!

Outline

1. What is Leaflet?
2. Brief R Basics
3. Let's make some maps!
    - In ggplot
    - In Leaflet
4. What is Shiny?
    - make a shiny app Shiny + Leaflet

What is Leaflet?

- Leaflet.js is a JavaScript library for making interactive maps in a web browser

- you can plug your data into Leaflet.js from R with the package **leaflet**

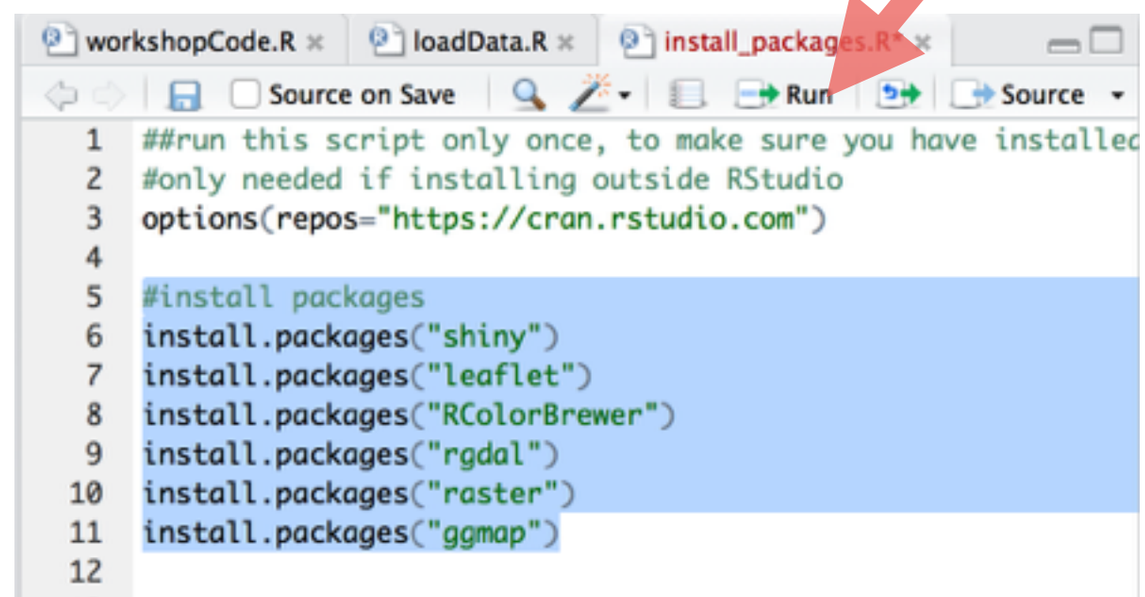
- Add data with addMarkers(), addPolygons(), etc…

```
library(leaflet)

m <- leaflet() %>%
  addTiles() %>%  # Add default OpenStreetMap map tiles
  addMarkers(lng=174.768, lat=-36.852, popup="The birthplace of R")
m  # Print the map
```

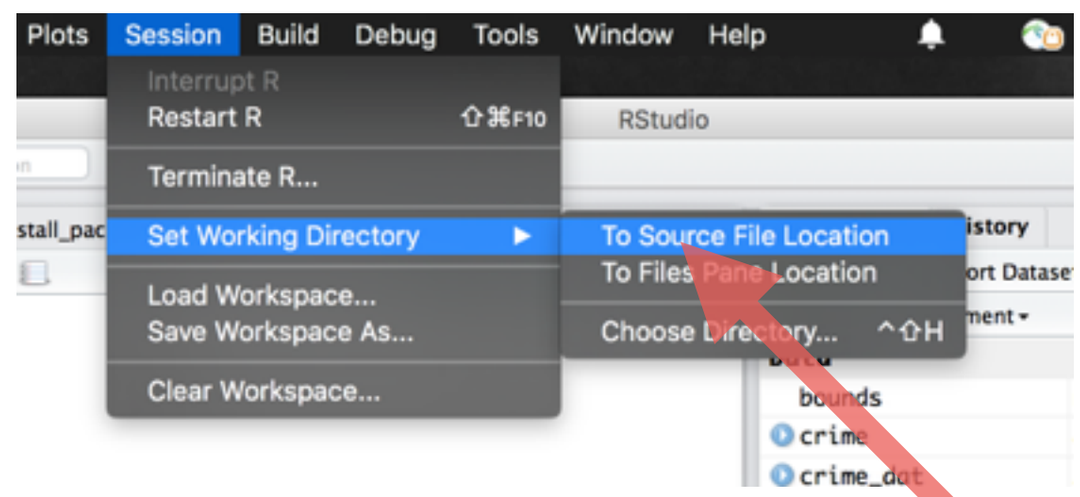
https://rstudio.github.io/leaflet/

R Basics

1. How to select and run code: select text and click ‘Run’

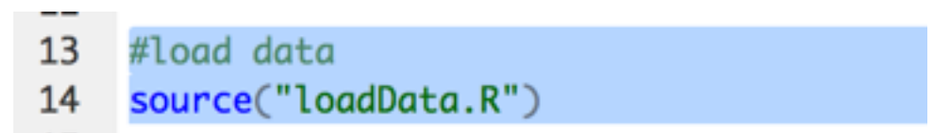


2. How to install packages. Open ‘install_packages.R’



3. Set working directory

4. Load data

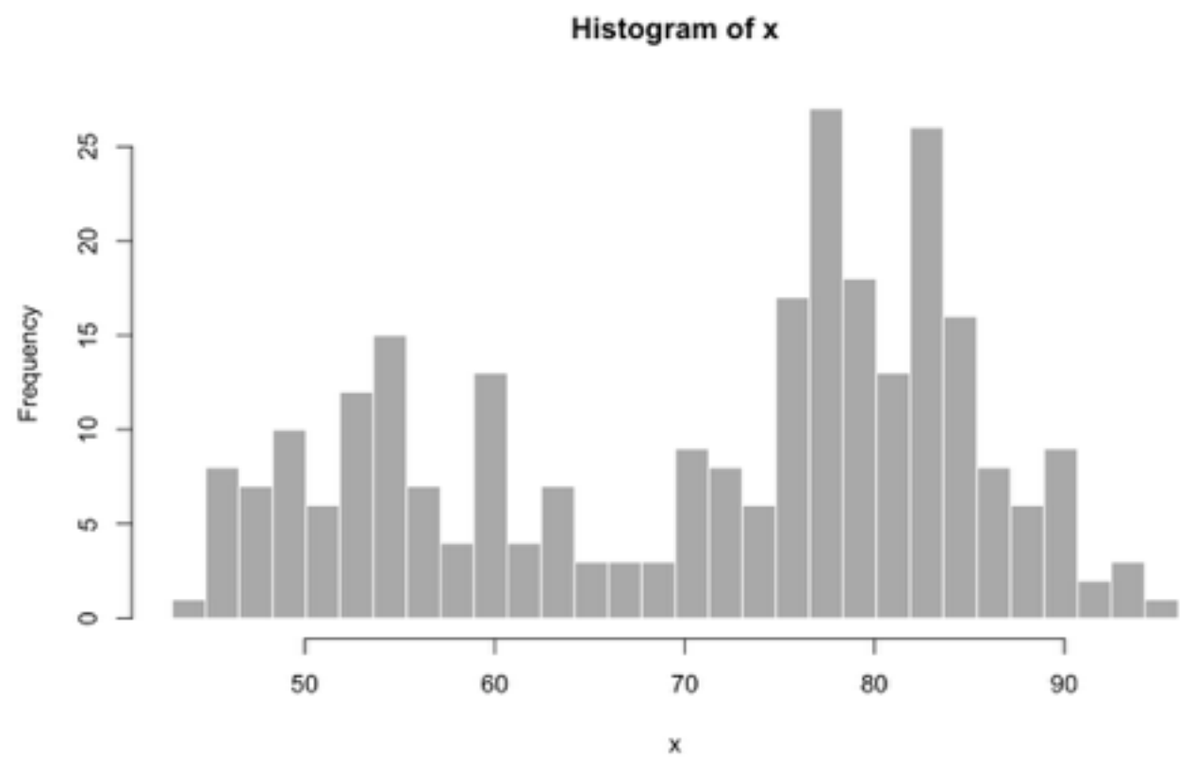


Open workshopCode.R

- Be on the lookout for comments in my code
    - #EXPLORE: …
    - #EXPLORE MORE: …

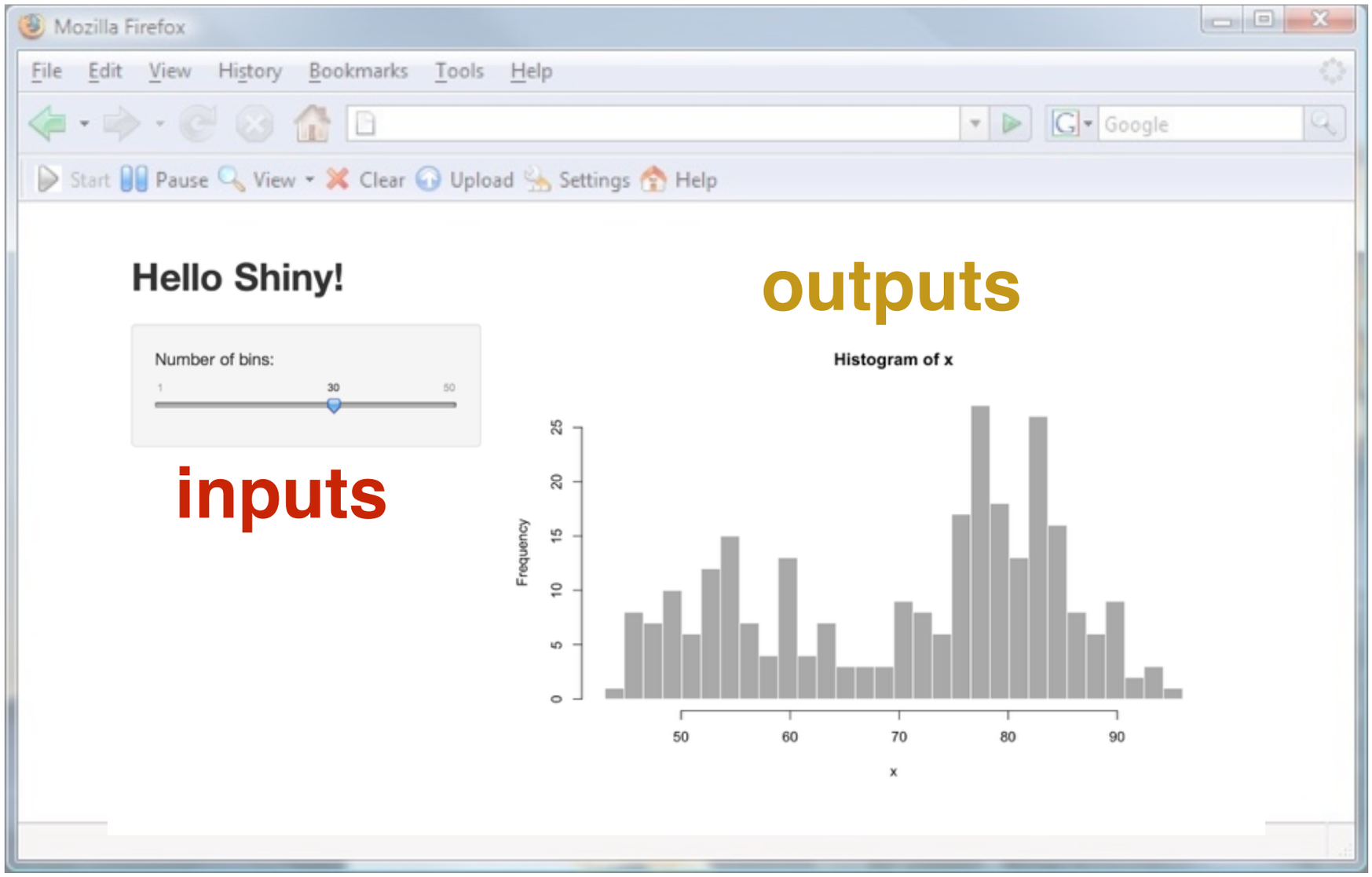
Shiny + Leaflet!

This is a histogram.



http://shiny.rstudio.com/tutorial/lesson1/

This is a ShinyApp!



http://shiny.rstudio.com/tutorial/lesson1/

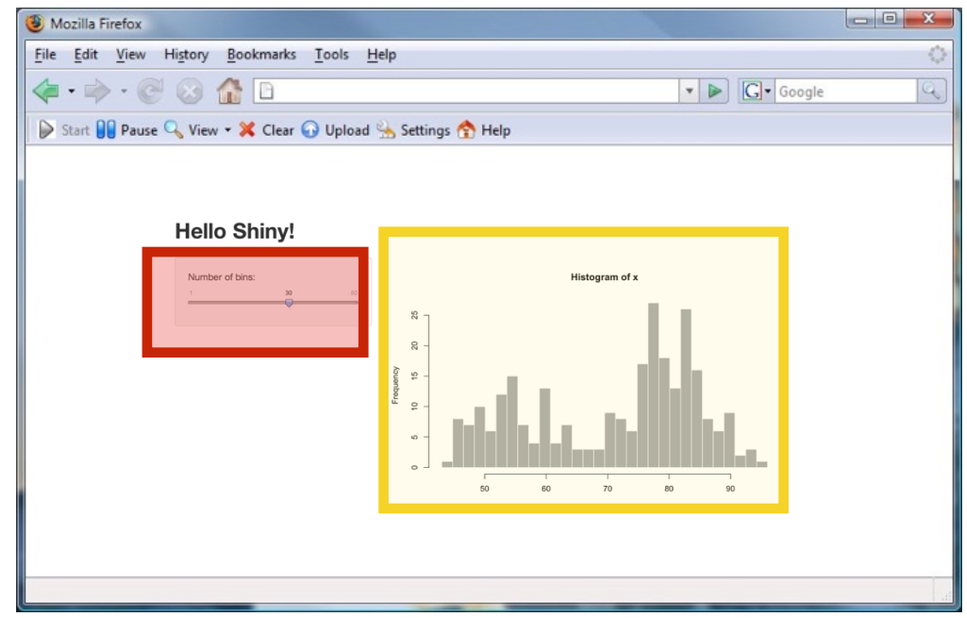
Shiny

outputs

"make a histogram!"

"make changes to the map!"

inputs



server.R

"back-end"

ui.R

"front-end"

Shiny

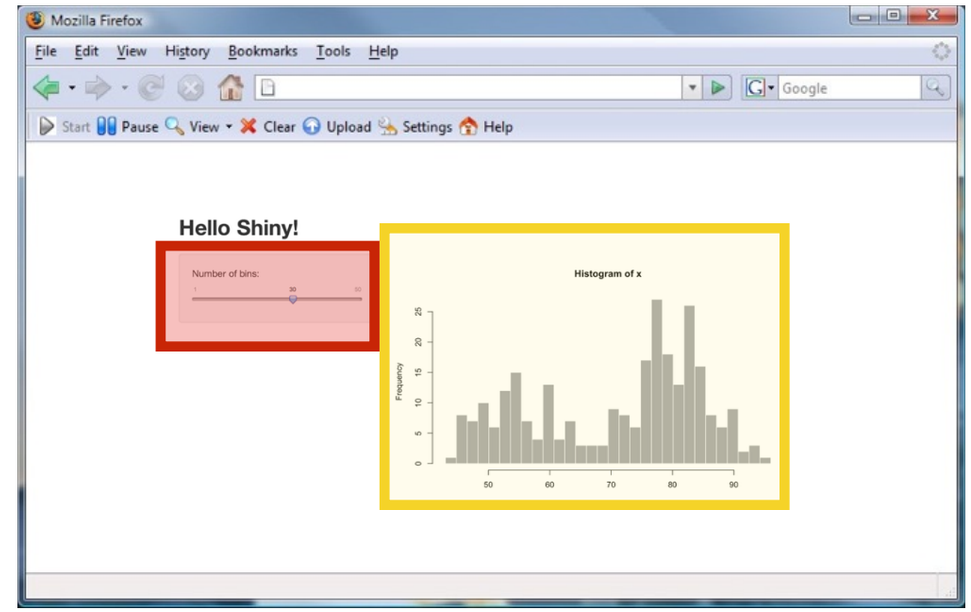
server.R

outputs

"make a histogram!"

"make changes to the map!"

inputs



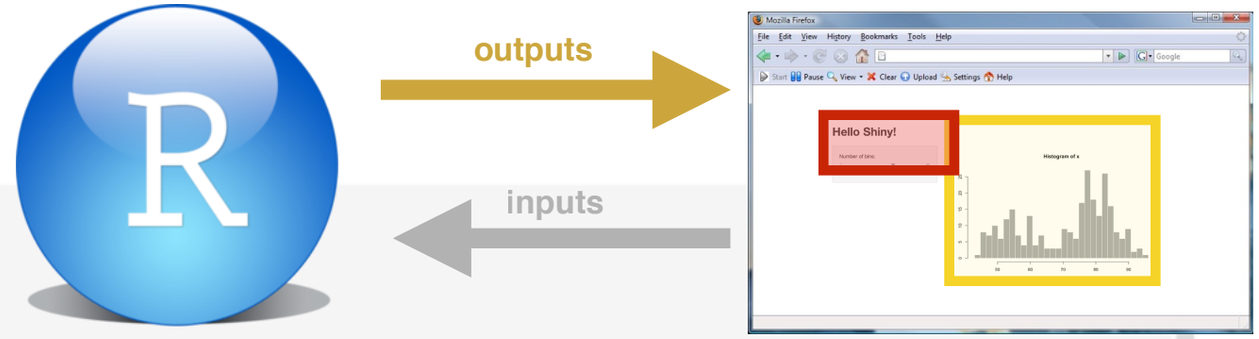
ui.R

variables and functions

variables and functions

global.R

server.R



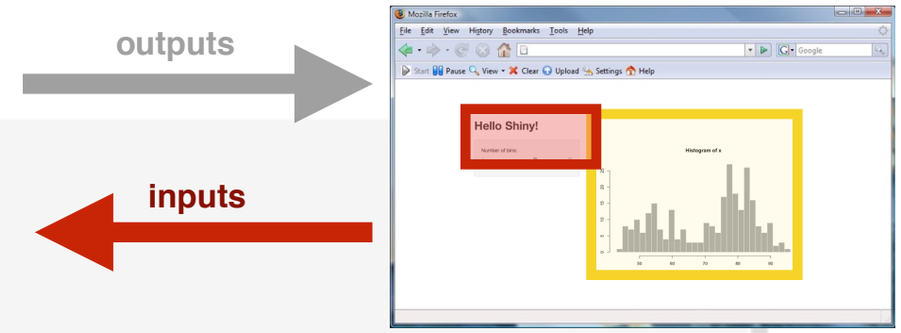

```r
library(shiny)

# 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 re-execute automatically
  #     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')
  })
})
```

Create an output titled 'distPlot' that returns a histogram.

Use an input titled 'bins'

http://shiny.rstudio.com/tutorial/lesson1/

ui.R




```r
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")
    )
  )
))
```

Create a slider input called 'bins' and send it from ui.R to server.R

Plot the output from server.R titled "distPlot"

http://shiny.rstudio.com/tutorial/lesson1/

More Shiny

- These functions pass inputs from ui.R to server.R
    - sliderInput(…
    - selectizeInput(…
    - checkboxInput(…
    - ….(and many more types of inputs!)
- These tell server.R to be on the watch for inputs
    - myVariable <- **reactive** ({…
    - **observe**({…

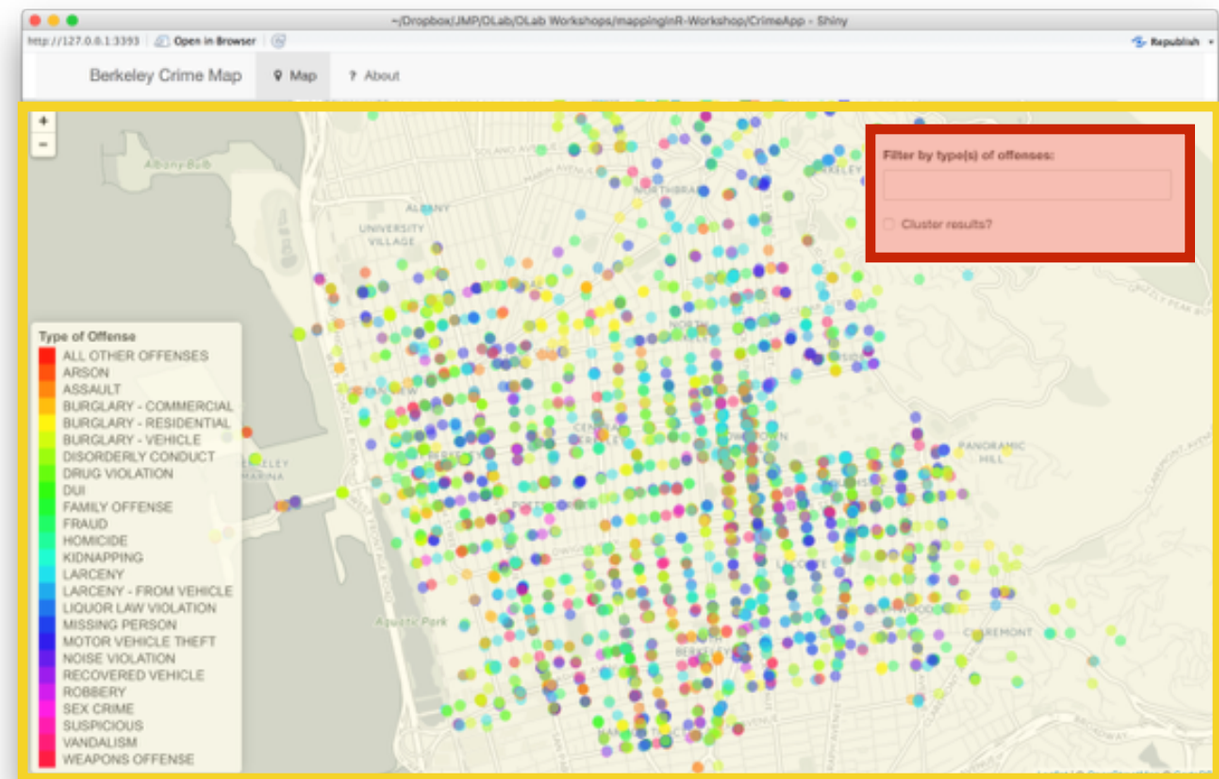
Shiny + Leaflet

outputs

"make a **Leaflet map**!"

"make changes to the map!"

inputs



server.R

"back-end"

ui.R

"front-end"

Open '**ui.R**' and '**server.R**' in folder '**CrimeApp**'

workshopCode.R × | loadData.R × | install_packages.R × | ui.R ×

```
fluidPage(
  navbarPage("Berkeley Crime Map", id="nav",
             tabPanel("Map", icon = icon("map-marker"),
                      div(class="outer",
                          tags$style(type = "text/css",
                                     ".outer {position: fixed; top: 50px; left: 0; right: 0;
                                     bottom: 0; overflow: hidden; padding: 0}"),
                          leafletOutput("map", width="100%", height="100%"),

                          absolutePanel(top = 30, right = 30, draggable=TRUE,
                                        wellPanel(style = "background-color: #ffffff; width: 350px",
                                                  selectizeInput('offenseFilter', 'Filter by type(s) of offenses:',
                                                                 choices = c(offenseList), multiple=TRUE),
                                                  checkboxInput('mapCluster', 'Cluster results?')
                                        )
                          )
                      )
             ),
             tabPanel("About",
                      icon = icon("question"),
```

show the **leaflet output** titled '**map**'

Create **inputs** for
1) filtering **offenses** and
2) **clustering** results

```r
#set the initial color palette
offenseColor <- colorFactor(rainbow(25), berkeleyCrime$CVLEGEND)

#set options for filtering based on type(s) of offense
filteredData <- reactive({
  if (is.null(input$offenseFilter)) {data <- berkeleyCrime}
  else {data <- subset(berkeleyCrime, CVLEGEND %in% input$offenseFilter)}
})
```

Use input called '**offenseFilter**' to filter data, if this input is not NULL

```r
output$map <- renderLeaflet({
  # Use leaflet() here, and only include aspects of the map that
  # won't need to change dynamically (at least, not unless the
  # entire map is being torn down and recreated).
  leaflet(filteredData()) %>%
    addProviderTiles("CartoDB.Positron") %>%
    setView(-122.28, 37.87, zoom = 14)
})
```

Create output called '**map**' that renders a Leaflet map

```r
#set mapCluster variable based on input checkbox
mapClusterResult <- reactive({
  if(input$mapCluster){TRUE}
  else {NULL}
})
```

Use input called '**mapCluster**' to set clustering options, if this input is not NULL

```r
#update map based on changed inputs
observe({
  leafletProxy("map", data = filteredData()) %>%
    clearMarkers() %>%
    clearControls() %>%
    clearMarkerClusters %>%
    addCircleMarkers(
      stroke = FALSE, fillOpacity = 0.5, radius=6, color = ~offenseColor(CVLEGEND),
      clusterOptions = mapClusterResult(),
      popup = ~paste("<strong>Offense:</strong>",CVLEGEND,
                     "<br>",
                     "<strong>Date:</strong>",EVENTDT,
                     "<br>",
                     "<strong>Time:</strong>",EVENTTM)
    ) %>%
    addLegend(title = "Type of Offense", position = "bottomleft",
              pal = offenseColor, values = ~CVLEGEND, opacity = 1)
```

Observe for changes in 'map' and update the map as necessary

How to run a shiny app

workshopCode.R | loadData.R | install_packages.R* | ui.R

Run App

```
1  fluidPage(
2    navbarPage("Berkeley Crime Map", id="nav",
3               tabPanel("Map", icon = icon("map-marker"),
4                        div(class="outer",
5                            tags$style(type = "text/css", ".outer {position:
6                            leafletOutput("map", width="100%", height="100%")
7
8                            absolutePanel(top = 30, right = 30, draggable=TRU
9                                          wellPanel(style = "background-color
10                                                   selectizeInput('offenseFi
11                                                   checkboxInput('mapCluster
12                                         )
13                           )
14                       )
15              ),
16              tabPanel("About",
17                       icon = icon("question"),
18
19                       #content on left hand side of the page
20                             h1("About"),
21
```

1:1 (Top Level) R Script

ShinyApps.io

- You can publish your ShinyApps on http://www.shinyapps.io/
- ShinyApps hosts your App and you can embed it in a blog or other website.
- Without this (or your own Shiny Server), you'll only be able to view your map on your local machine.