Analytics of Business Intelligence

Chapter # 6 - Web Dashboards with Shiny

Dr. Wajahat Gilani

Rutgers Business School

April 24, 2020

Introduction

As a data analyst, you attempt to transform analysis into actionable insights that solve business problems and drive change. **Poorly communicated insights diminish the value of your analysis.** In corporations, and even smaller firms, information and analysis is increasing delivered via the web. R isn't a real programming language like python which has web frameworks to integrate with server-side systems to deliver interactive web application/dashboards. But R provides its own web application framework called Shiny, that allows you to build web applications/dashboards using R code. This section we will discuss the basic building blocks of a Shiny web app:

- Creating a basic Shiny app
- Creating a marketing-campaign Shiny app
- Deploying your Shiny app

We will be using the text file Ch8_marketing.csv.

Create a Basic Shiny App

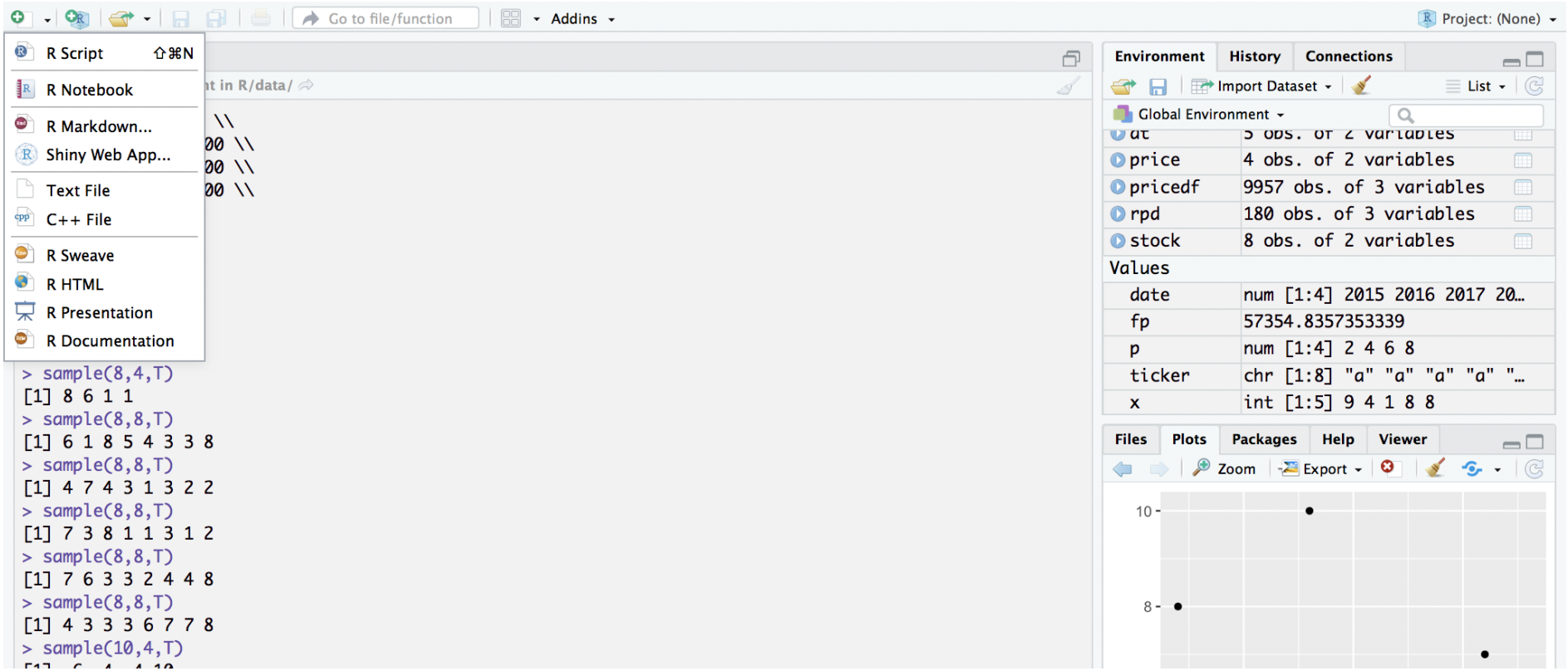
Shiny apps have specific folder and file structures. At a minimum, a Shiny app has a user interface (client-side logic) and server-side logic.

The **client-side logic** is what the user sees in their web browser. The **server-side logic** is what gets executed on your computer or server that has all the data. The files essential for a standard Shiny app are:

- A **ui.R** file containing all the client-side logic (**u**ser **i**nterface)
- A **server.R** file containing all the server-side logic

RStudio will automatically create the **ui.R** and **server.R** files when you create a new project and choose **New Directory** and **Shiny Web Applicaton**.

Create a Basic Shiny App - Click on Plus



Create a Basic Shiny App - Click on Shiny Web App

Go to file/function | Addins | Project: (None)

Console | Terminal

~/Documents/Investment in R/data/

```
1 & 2015.00 & 2.00 \\
  2 & 2016.00 & 4.00 \\
  3 & 2017.00 & 6.00 \\
  4 & 2018.00 & 8.00 \\
   \hline
\end{tabular}
\end{table}
> sample(8,4,T)
[1] 8 6 2 6
> sample(8,4,T)
[1] 2 7 4 2
> sample(8,4,T)
[1] 8 6 1 1
> sample(8,8,T)
[1] 6 1 8 5 4 3 3 8
> sample(8,8,T)
[1] 4 7 4 3 1 3 2 2
> sample(8,8,T)
[1] 7 3 8 1 1 3 1 2
> sample(8,8,T)
[1] 7 6 3 3 2 4 4 8
> sample(8,8,T)
[1] 4 3 3 3 6 7 7 8
> sample(10,4,T)
[1]  6  4  4 10
> x=sample(10,5,T)
> y=sample(10,5,T)
> dt = data.table(x,y)
> ggplot(dt,aes(x=x,y=y)) + geom_point()
> dt
   x  y
1: 9  2
2: 4 10
3: 1  8
4: 8  5
5: 8  7
>
```

Environment | History | Connections

Import Dataset | List

Global Environment

dt	5 obs. of 2 variables
price	4 obs. of 2 variables
pricedf	9957 obs. of 3 variables
rpd	180 obs. of 3 variables
stock	8 obs. of 2 variables
Values	
date	num [1:4] 2015 2016 2017 20…
fp	57354.8357353339
p	num [1:4] 2 4 6 8
ticker	chr [1:8] "a" "a" "a" "a" "…
x	int [1:5] 9 4 1 8 8

New Shiny Web Application

Application name: lab6

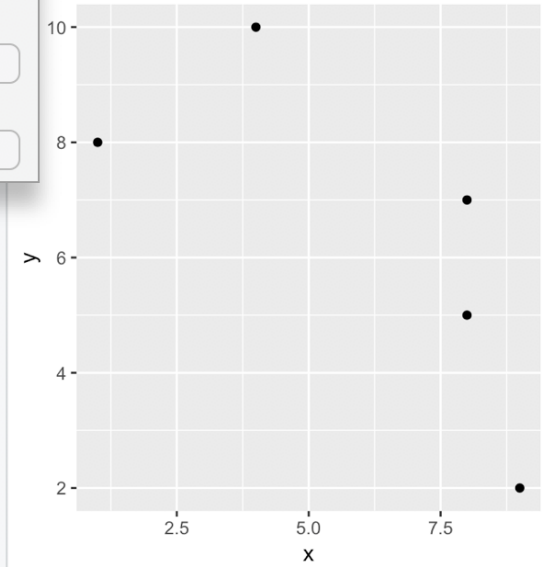
Application type: Single File (app.R) / Multiple File (ui.R/server.R)

Create within directory: ~/Documents/ABI/ABI Fall 2019/lab6 Browse...

Shiny Web Applications | Create | Cancel

Plots | Packages | Help | Viewer

Zoom | Export



Create a Basic Shiny App - Click on Shiny Web App

Go to file/function | Addins | Project: (None)

ui.R | server.R | Run App

```
#
# This is the user-interface definition of a Shiny web application. You can
# 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"),

```

1:1 (Top Level) R Script

Console | Terminal

~/Documents/Investment in R/data/

```
> sample(8,8,T)
[1] 4 3 3 3 6 7 7 8
> sample(10,4,T)
[1]  6  4  4 10
> x=sample(10,5,T)
> y=sample(10,5,T)
> dt = data.table(x,y)
> ggplot(dt,aes(x=x,y=y)) + geom_point()
> dt
   x  y
1: 9  2
2: 4 10
3: 1  8
4: 8  5
5: 8  7
>
```

Environment | History | Connections

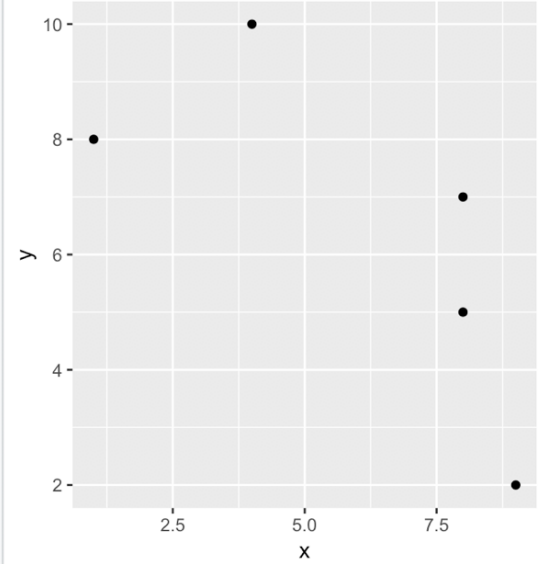
Import Dataset | List

Global Environment

price	4 obs. of 2 variables
pricedf	9957 obs. of 3 variables
rpd	180 obs. of 3 variables
stock	8 obs. of 2 variables
Values	
date	num [1:4] 2015 2016 2017 20…
fp	57354.8357353339
p	num [1:4] 2 4 6 8
ticker	chr [1:8] "a" "a" "a" "a" "…
x	int [1:5] 9 4 1 8 8

Files | Plots | Packages | Help | Viewer

Zoom | Export

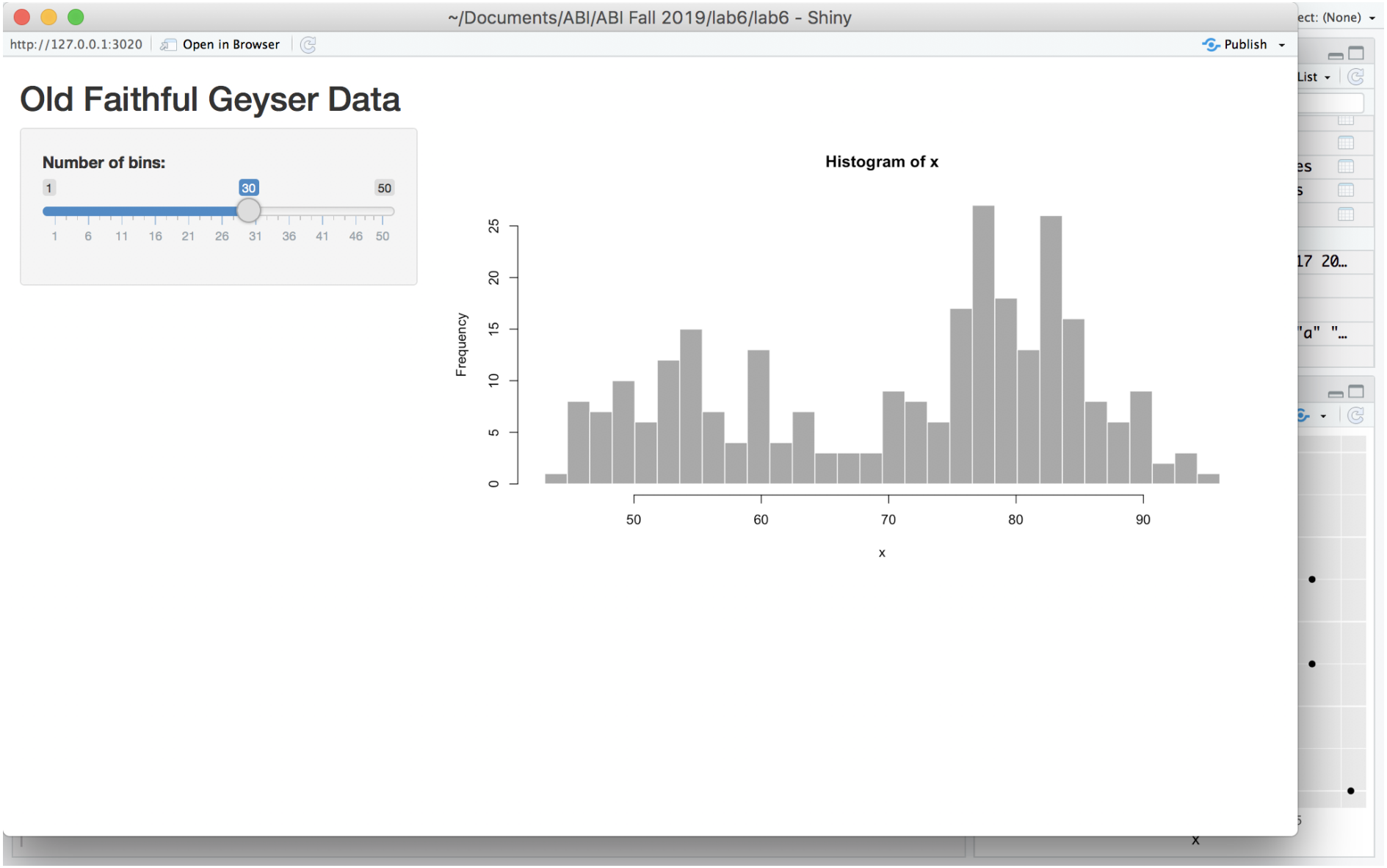


The ui.R File

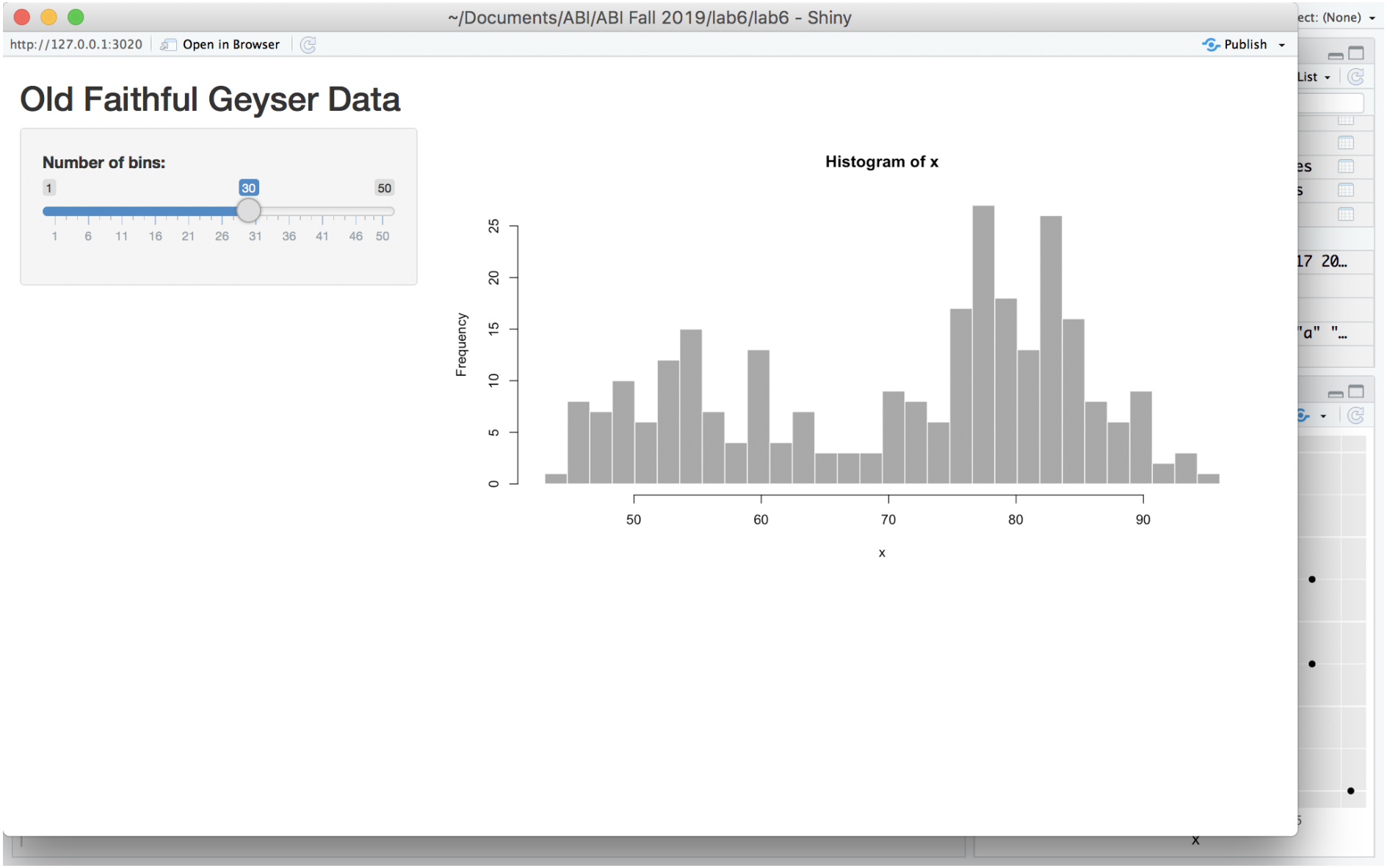
When you create the new Shiny app, there is already a sample web app defined in the ui.R file. Go ahead and click on Run App

```
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   )
```

The ui.R File - Sample Web Application



The ui.R File - Sample Web Application



The ui.R File - shinyUI

```
library(shiny)
shinyUI(fluidPage(

  titlePanel("Old Faithful Geyser Data"),

  sidebarLayout(
    sidebarPanel(
      sliderInput("bins","Number of bins:",min = 1,max
          = 50,value = 30)
  ),

  mainPanel(plotOutput("distPlot"))
  )
))
```

The shiny package has all the functions. The shinyUI() function should be at the beginning of every ui.R file.

The ui.R File - shinyUI

```
library(shiny)
shinyUI(fluidPage(

  titlePanel("Old Faithful Geyser Data"),

  sidebarLayout(
    sidebarPanel(
      sliderInput("bins","Number of bins:",min = 1,max
          = 50,value = 30)
  ),
```

The fluidPage() function allows the web page to stretch or constrict the app, so that it can fit the computer and mobile web browsers. The titlePanel() gives the app a title on the top of the page.

The ui.R File - shinyUI

```
llibrary(shiny)
shinyUI(fluidPage(

  titlePanel("Old Faithful Geyser Data"),

  sidebarLayout(
    sidebarPanel(
    sliderInput("bins","Number of bins:",min = 1,max =
        50,value = 30)
  ),

  mainPanel(plotOutput("distPlot")))

))
```

The design elements go between the titlePanel() function and the last 2 parenthesis.

The ui.R File - Widgets

The functions sliderInput() and sidebarPanel() are called widgets. R has many of them to allow users to interact the data. For example, radio buttons, drop-down lists, checkboxes, and text inputs. To see a list of widgets, you can go to the URL:
https://shiny.rstudio.com/gallery/widget-gallery.html

Lets made a web dashboard for our linear regression model that we created in lecture # 4, and lets use the sliderInput() to input a marketing expenditure to predict and output revenue.

```
sliderInput("bins","Number of bins:",min = 1,max = 50,
    value = 30)
```

The sliderInput() has 5 parameters.

The ui.R File - Widgets

```
sliderInput("bins","Number of bins:",min = 1,max = 50,
    value = 30)
```

The sliderInput() has 5 parameters.

- inputId - reference for that particular slider
- label - text label to inform the user about the slider
- min - floor for the slider, for our regression model it is our minimum data point
- max - the highest value allowed, for regression model, highest data point
- value - the default value, can be anything

The ui.R File - sliderInput

```
sidebarLayout(
  sidebarPanel(
    sliderInput("spend","Expenditure Level in $K",min = 54,max = 481,value = 250)
    ),

    mainPanel(plotOutput("prediction_plot"))
  ) #closes sidebarLayout
```

The sidebarLayout() is a design theme that includes a sidebarPanel(), for widgets on the left side, and a mainPanel() that is a larger area on the right side for output. The plotOutput() function tells the browser to render a plot called **prediction_plot**, this will come from the server.R file. We are finished with our simple user interface.

The server.R File

```
library(shiny)

shinyServer(function(input, output) {

  output$distPlot <- renderPlot({

  x    <- faithful[, 2]
  bins <- seq(min(x), max(x), length.out = input$bins
       + 1)

  hist(x, breaks = bins, col = 'darkgray', border = '
      white')

  })

})
```

The server is defined by shinyServer().

The server.R File

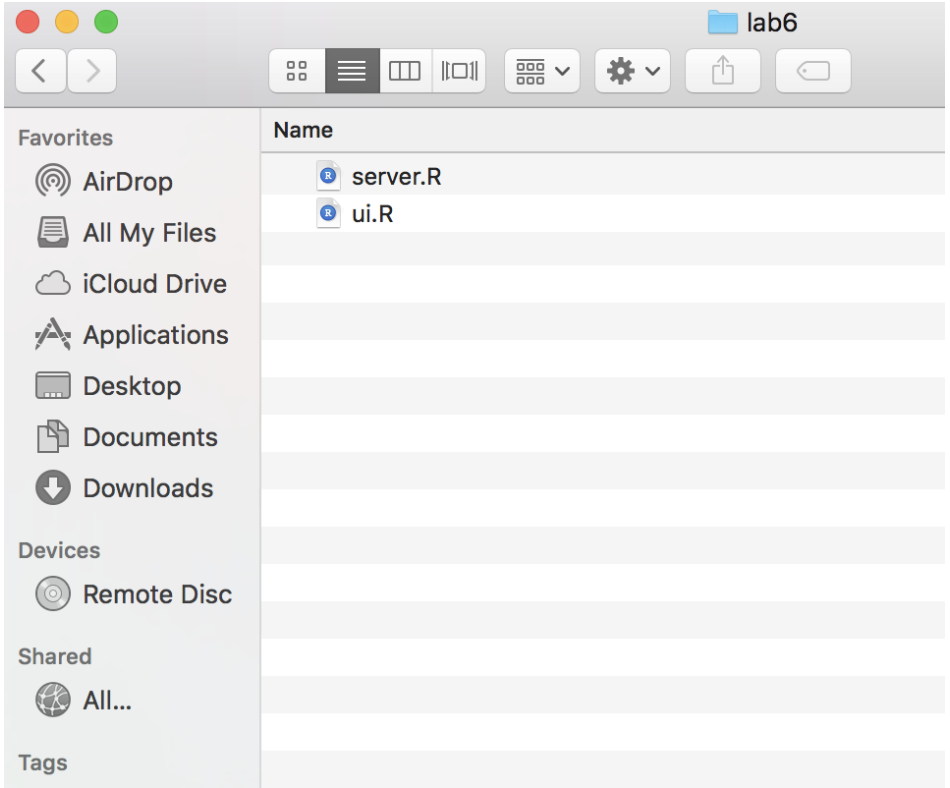
```
shinyServer(function(input, output, session){. . .})
```

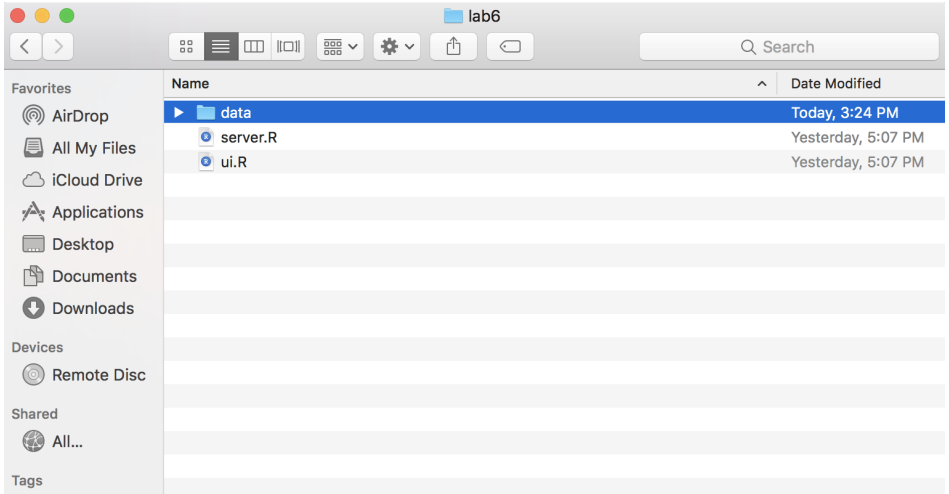
The basic syntax for shinyServer() is that it pulls in all the **inputs** from ui.R into the first parameter, the second parameter **output** is what gets **pushed** back to the user interface, and the third parameter allows you to create variables that are specific for each user **session**. Remember, you might have multiple people using your web app in a company at the same time.

```
library(shiny)
library(data.table)
library(ggplot2)
```

The shiny library is already called and we will add the data.table and ggplot2() libraries as well, the next step is to load up the data, but lets store it in a place that the program can find it.

The server.R File - Load up Data





The data folder is created and inside that folder we will place our data file Ch8_marketing.csv. Add the line to read in the data file **from the point of view of the code**.

```
revenue = read.csv('./data/Ch8_marketing.csv')
```

The server.R File - Scatter Plot

Add in the other lines of code we used to run the regression.

```
revenue = read.csv('./data/Ch8_marketing.csv')
setDT(revenue)
model = lm(revenues~marketing_total,data = revenue)
```

Now change the shinyServer() to do a simple scatter plot.

```
shinyServer(function(input, output) {

  output$prediction_plot <- renderPlot({
    ggplot(revenue,aes(x=revenues,y=marketing_total))
        + geom_point(color='purple') + geom_smooth(
        method = 'lm')

  })

})
```

Predicting Outputs

```
newrev = data.table(marketing_total=seq(460,470,5))
predict.lm(model1,newrev,interval = 'predict')
```

```
     fit      lwr      upr
55.89403 49.75781 62.03025
56.15368 50.01331 62.29404
56.41332 50.26873 62.55791
```

For the value of $460,000 you get an estimate of revenue of $55,894. Now that can't be an exact number because it is an estimate. That is why you are given a 95% confidence interval. What the confidence interval is saying is that if you were to make 100 predictions with marketing_total being 460, then 95 out of the 100 results would fall in between 55.89403 and 62.03025. If you wanted a 99% interval, then you would use the parameter:

```
predict.lm(model1,newrev,level=.99,interval = 'predict
    ')
```

```
     fit      lwr      upr
55.89403 47.79622 63.99184
56.15368 48.05040 64.25695
```

The server.R File - Reading in Inputs

We want to get the value from the slider and now predict what the revenue will be and display it, both on the graph and as text.

```
shinyServer(function(input, output) {

  output$prediction_plot <- renderPlot({
    newrev=data.table(marketing_total=input$spend)
    pred=predict.lm(model,newrev,interval = 'predict')
    ggplot(revenue,aes(x=revenues,y=marketing_total))
       + geom_point(color='purple') + geom_smooth(
       method = 'lm')+labs(title=paste('Predicted
       Range of Revenue for',input$spend,'is',round(
       pred[2])*1000,'to',round(pred[3])*1000))

  })

})
```

Your Turn

Now change the slider to allow input from 2 to 10. Create a web dashboard for the bike stations cluster project, with different colors for each region and a specialized dot, pointing to where each kiosk will go.