

Introduction to R Shiny

Aaron Geller





What is Shiny and why should I use it?

Shiny is an R package for creating interactive web applications without needing to know HTML, CSS or JavaScript.

Why use Shiny?

- Exploratory data analysis
- Contextual learning and exploration (e.g., data on a map)
- Dashboards
- Tools for your colleagues
- Teaching
- And more...see the [Shiny gallery](#)!



Before starting in Shiny

- Think conceptually about what you want to create. Maybe make a sketch or outline.
- Consider both
 - **functionality** (e.g., how do I want to interactively manipulate the data) and
 - **form** (e.g., what layout is going to be the most straightforward to the user).
- Create a working static version of your figure (or table) first (without Shiny).



Basic Shiny App Structure

app.R

```
library(shiny)

ui <- fluidPage()

server <- function(input, output){}

shinyApp(ui, server)
```



Basic Shiny App Structure

app.R

```
library(shiny)

ui <- fluidPage()

server <- function(input, output){}

shinyApp(ui, server)
```

- Load the Shiny library so the functions we need are available.



Basic Shiny App Structure

app.R

```
library(shiny)

ui <- fluidPage()

server <- function(input, output){}

shinyApp(ui, server)
```

- Load the Shiny library so the functions we need are available.
- The user interface (UI) displays widgets for manipulating data (input) and figures, tables, etc. (output).



Basic Shiny App Structure

app.R

```
library(shiny)

ui <- fluidPage()

server <- function(input, output){}

shinyApp(ui, server)
```

- Load the Shiny library so the functions we need are available.
- The user interface (UI) displays widgets for manipulating data (input) and figures, tables, etc. (output).
- The server contains a series of R statements to define what the app does.



Basic Shiny App Structure

app.R

```
library(shiny)

ui <- fluidPage()

server <- function(input, output){}

shinyApp(ui, server)
```

- Load the Shiny library so the functions we need are available.
- The user interface (UI) displays widgets for manipulating data (input) and figures, tables, etc. (output).
- The server contains a series of R statements to define what the app does.
- Create the Shiny App.



How to run a Shiny app

Option 1: from a terminal

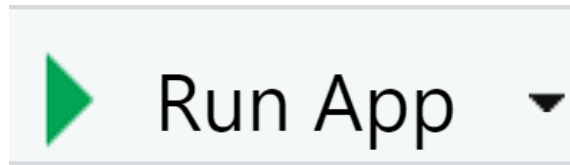
- Within a directory that has your app.R file, run the following within an R session:

```
library(shiny)
runApp("app.R")
```

- This should automatically open a browser to the URL where your local version of the app is hosted

Option 2: from R Studio

- Launch R Studio and then open your application file ("app.R")
- Click the Run App button.



- This should automatically open a window showing your local version of the app.



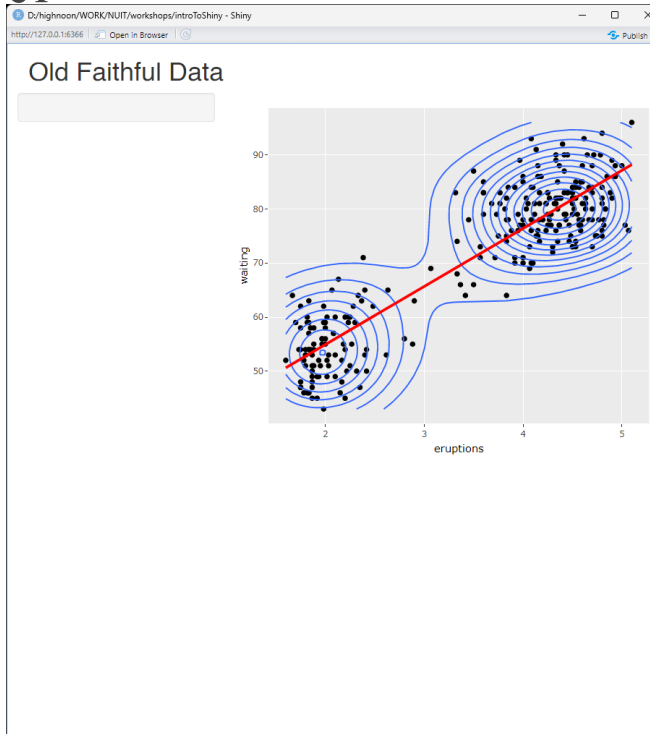
Examples

1. Just a plot, to show how to start the server
2. Add inputs that control the plot content
3. Add a text box showing the fit details
4. Add dropdown to choose which marginal plots to show
5. Add `conditionalPanel` to control displaying `marginalFormat`
6. Add slider to control `binwidth` for histograms



Example 1

Just a plot, to show how to start the server





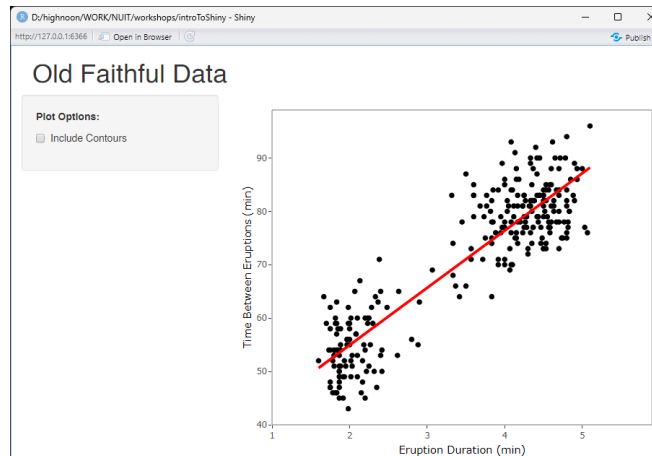
Example 2

Add inputs that control the plot content

```
21 # checkboxes to turn on/off plot elements
22 # https://shiny.rstudio.com/reference/shiny/latest/checkboxInput.html
23 strong("Plot Options:"), # a simple html element to provide a title for this section of the UI
24 checkboxInput(
25   "showContours", "Include Contours",
26   value = FALSE),
27 ),
28
```

```
40
41 # when we have input values that we want to use for generating output,
42 # we need to wrap that portion of the code in observe({}), or another reactive container
43 observe({
44   # create the scatter plot
45   main_plot <- ggplot(faithful, aes(eruptions, waiting)) +
```

```
52
53 # add the contours if requested by the user
54 if (input$showContours) main_plot <- main_plot + geom_density2d()
55
```



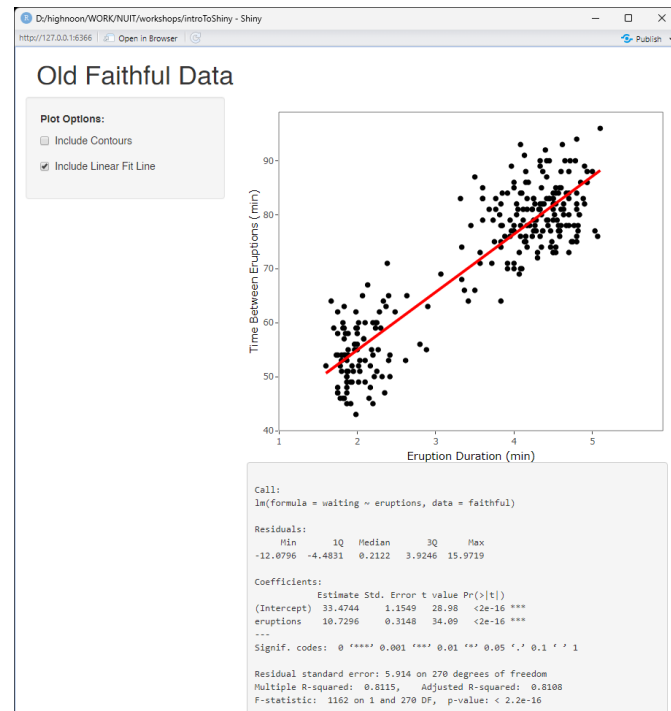
- Exercise 2.1: add input to turn line on/off (default on)
- Exercise 2.2: change sidebarLayout to verticalLayout
- Exercise 2.3: change sidebarPanel to wellPanel



Example 3

Add a text box showing the fit details

```
33 # Main panel for displaying outputs
34 mainPanel(
35   # https://shiny.rstudio.com/reference/shiny/latest/plotOutput.html
36   plotlyOutput("finalPlot", height = "500px"),
37
38   # https://shiny.rstudio.com/reference/shiny/1.0.3/verbatimTextOutput.html
39   verbatimTextOutput("modelSummary")
40 )
41
71
72 output$modelSummary <- renderPrint(
73   summary(lm(waiting ~ eruptions, data = faithful))
74 )
```

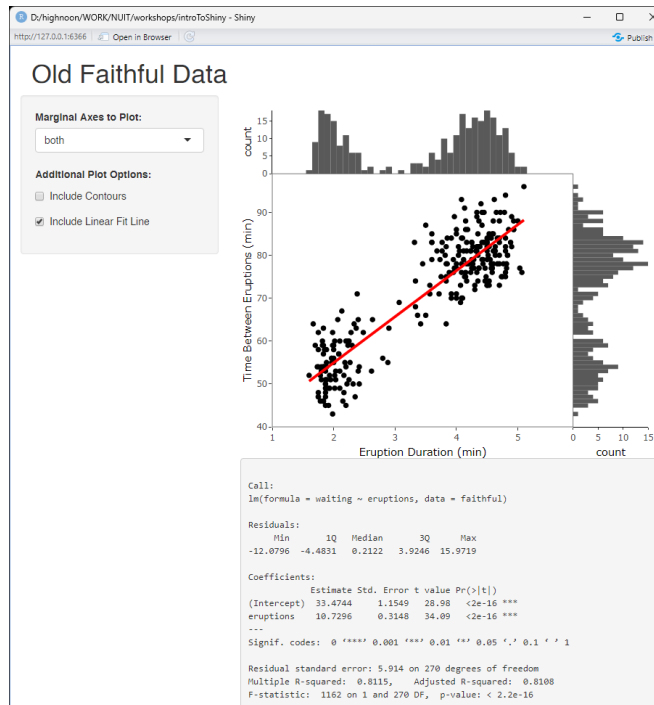


Example 4

Add dropdown to choose marginal plots

```
21 # dropdown to choose which marginal plots to show
22 # https://shiny.rstudio.com/reference/shiny/latest/selectinput.html
23 selectinput(
24   "marginstoshow", "Marginal Axes to Plot:",
25   c("x", "y", "both"),
26   selected = "both"
27 ),
28
29 / 3
74 # set up empty plots that will hold the marginal distributions
75 top_plot <- ggplot(faithful, aes(eruptions)) +
76   scale_x_continuous(limits = c(1, 5.9), expand = c(0, 0)) +
77   scale_y_continuous(limits = c(0, NA), expand = c(0, 0)) +
78   theme_classic() +
79   theme(panel.grid.major = element_blank(), panel.grid.minor = element_blank())
80
81 right_plot <- ggplot(faithful, aes(waiting)) + coord_flip() +
82   scale_x_continuous(limits = c(40, 99), expand = c(0, 0)) +
83   scale_y_continuous(limits = c(0, NA), expand = c(0, 0)) +
84   theme_classic() +
85   theme(panel.grid.major = element_blank(), panel.grid.minor = element_blank())
86
87 # add the histograms
88 top_plot <- top_plot + geom_histogram(binwidth = 0.1)
89 right_plot <- right_plot + geom_histogram(binwidth = 1)
90
91 if (input$marginstoshow == "y") top_plot <- plotly_empty()
92 if (input$marginstoshow == "x") right_plot <- plotly_empty()
93
94 f <- subplot(top_plot, plotly_empty(), main_plot, right_plot,
95   nrow = 2, heights = c(0.2, 0.8), widths = c(0.8, 0.2), margin = 0,
96   sharex = TRUE, sharey = TRUE)
97
98
99 # store the figure in the "finalPlot" key of the output variable which can be seen by the UI
100 output$finalPlot <- renderPlotly(f)
101
```

- Exercise 4.1: add dropdown to choose histogram vs. density
- Exercise 4.2: add option to choose none for marginsToShow

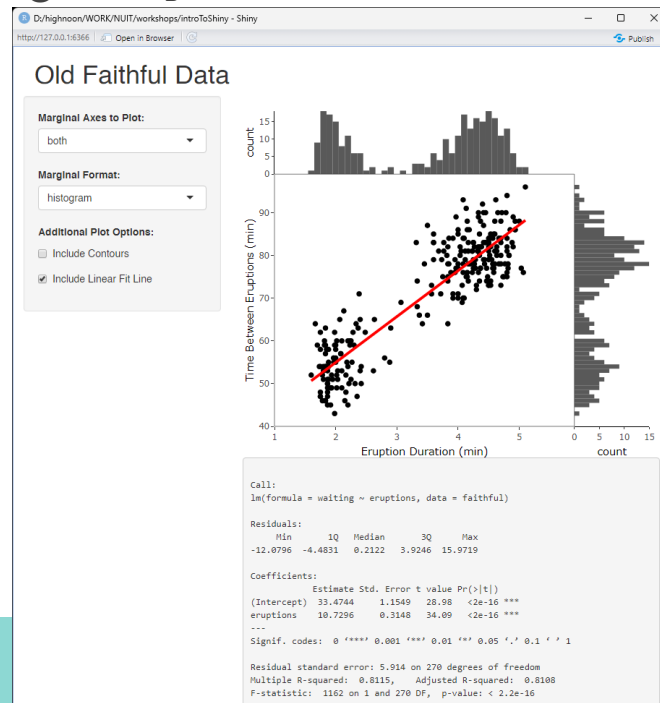


Example 5

Add conditionalPanel to control displaying marginalFormat

```
29 # dropdown to choose the type of marginal distribution
30 # https://shiny.rstudio.com/reference/shiny/1.6.0/conditionalPanel.html
31 conditionalPanel(
32   condition = "input.marginsToShow != 'none'",
33   selectInput(
34     "marginalFormat", "Marginal Format:",
35     c("histogram", "density"),
36     selected = "histogram"
37   )
38 ),
39 )
```

- Exercise 5.1: use conditionalPanel to show fit output only when showing the fit line



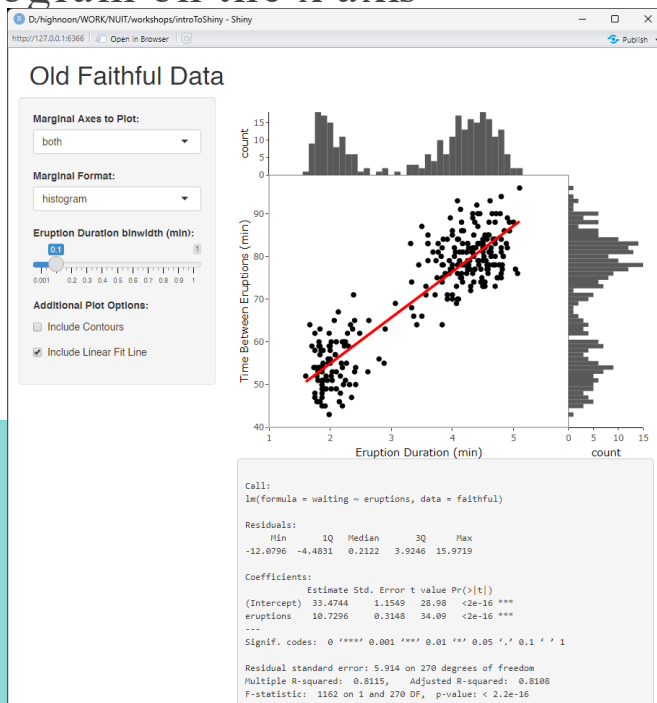
Example 6

Add slider to control `binwidth` for the histogram on the x axis

```
# slider to choose the bin size for the x axis (only relevant for histogram margin type)
# https://shiny.rstudio.com/reference/shiny/latest/sliderInput.html
sliderInput("xbinwidth", "Eruption Duration binwidth (min):",
  min = 1e-3, max = 1, value = 0.1),
```

```
top_plot <- top_plot + geom_histogram(binwidth = input$xbinwidth)
```

- Exercise 6.1: add slider to control the binwidth on the y axis
- Exercise 6.2: use `conditionalPanel` to only show these sliders for the histogram marginal format
 - *Bonus*: use multiple conditions within the `conditionalPanel` to only show these sliders for histogram format and when the correct marginal panel is displayed





Hosting on shinyapps.io

First, sign up for an account on shinyapps.io

Option 1: from a terminal

- Within a directory above the one that has your app.R file, run the following within an R session:

```
library(rsconnect)
deployApp("dirName")
```

- replace “dirName” with the name of the directory that contains app.R.
- This will create a url similar to:
`https://[you].shinyapps.io/[dirName]`

Option 2: from R Studio

- Launch R Studio and then open your application file (“app.R”)
- Click the publish button.



- This will open a GUI window to select the file(s) to publish and other options, and will create the URL