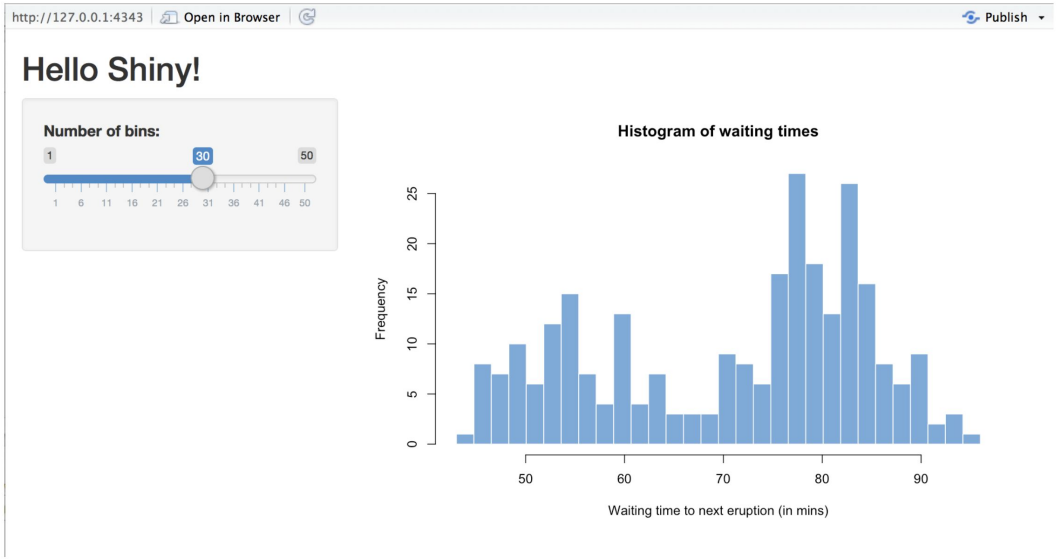
R-Ladies Philly Introduction to Shiny dashboards

Anastasia Lucas

Before we get to Shiny dashboards, what is a Shiny app?

https://shiny.rstudio.com/

- interactive web app
- built on HTML

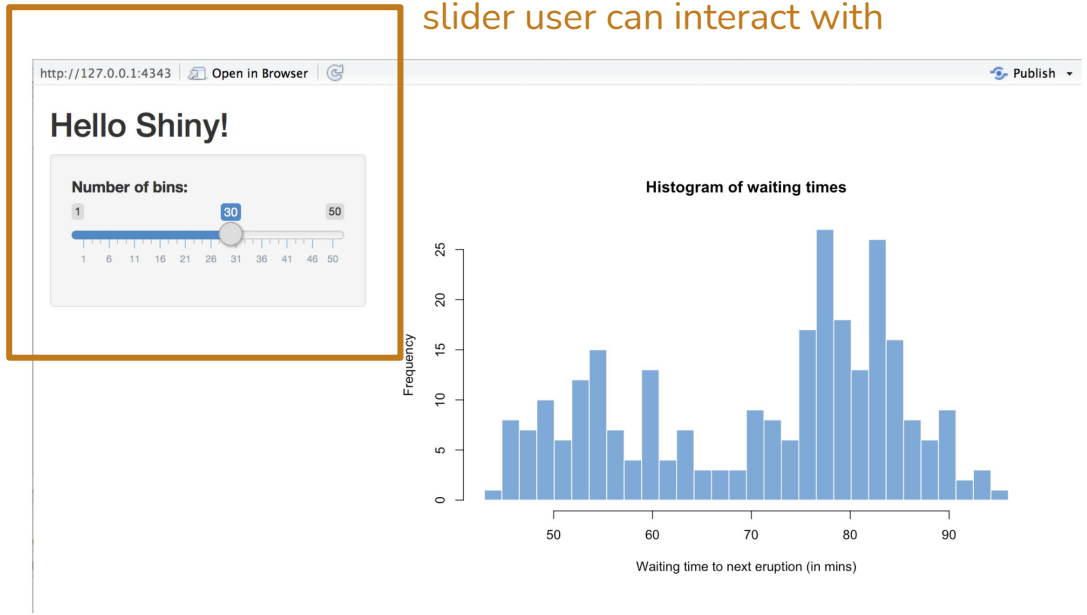


Before we get to Shiny dashboards, what is a Shiny app?

https://shiny.rstudio.com/

interactive web app

slider user can interact with

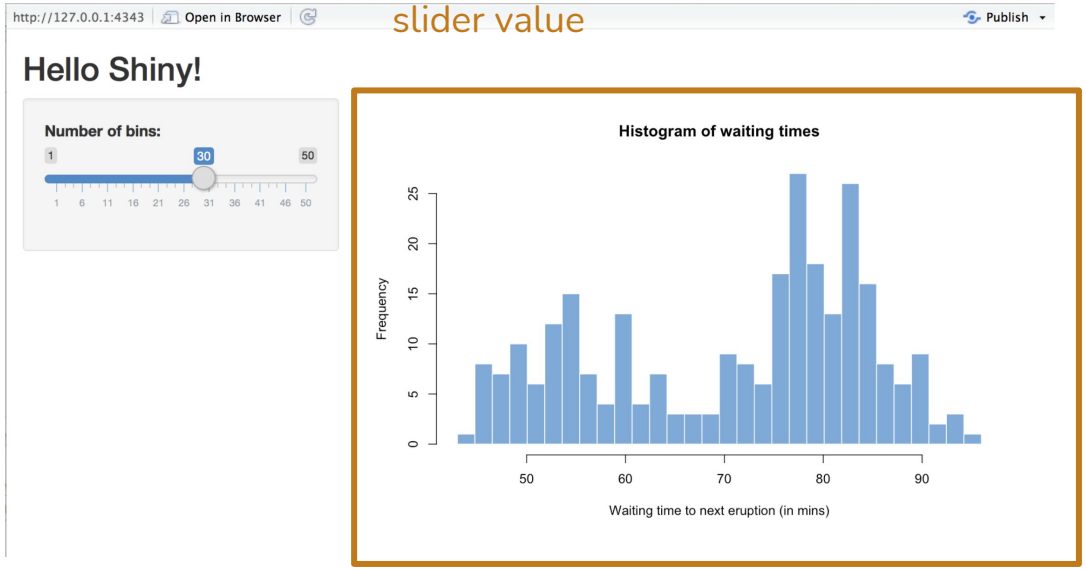


Before we get to Shiny dashboards, what is a Shiny app?

https://shiny.rstudio.com/

interactive web app

histogram which changes based on slider value

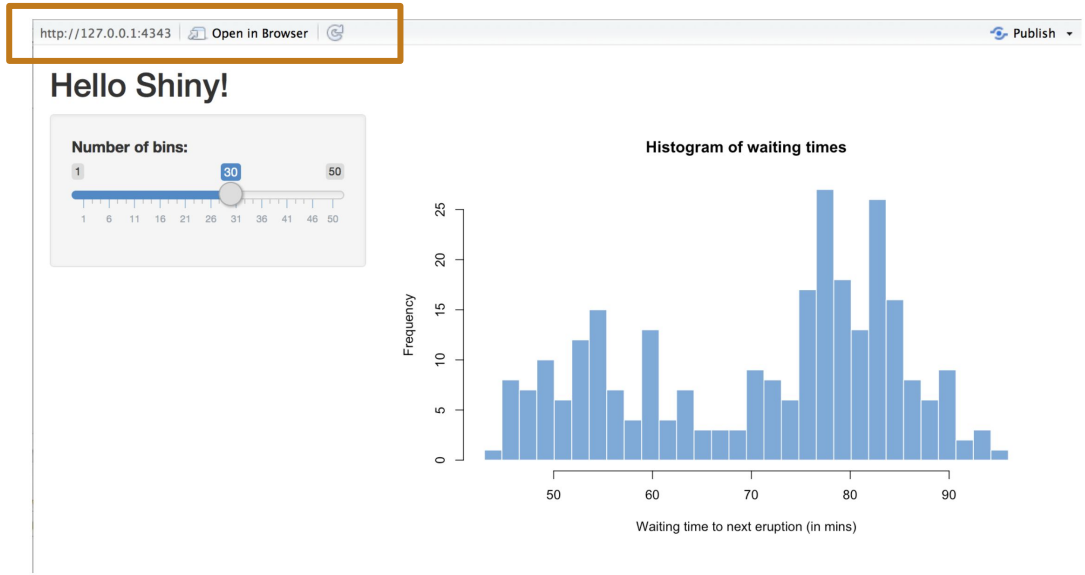


Before we get to Shiny dashboards, what is a Shiny app?

runs on a server

https://shiny.rstudio.com/

interactive web app



Shiny apps allow for interactivity and reactivity

- Interactivity: an element reacts to user actions
  - ex: hover text, on-click actions, highlight on hover
  - R packages: plotly, ggiraph, Shiny
- Reactivity: data is updated from the server without refreshing the site
  - ex: changing binwidth, filtering, changing parameters in models
  - R packages: Shiny, shinydashboard, flexdashboard

How would we make a Shiny app?

Shiny apps have three main components:

How would we make a Shiny app?

Shiny apps have three main components:

1. a user interface object

```
library(shiny)

# Define UI for app that draws a histogram
ui <- fluidPage(
  # App title
  titlePanel("Hello Shiny!"),
  # Sidebar layout with input and output definitions
  sidebarLayout(
    # Sidebar panel for inputs
    sidebarPanel(
      # Input: Slider for the number of bins
      sliderInput(inputId = "bins",
                  label = "Number of bins:",
                  min = 1,
                  max = 50,
                  value = 30)

    ),
    # Main panel for displaying outputs
    mainPanel(
      # Output: Histogram
      plotOutput(outputId = "distPlot")

    )
  )
)
```

https://shiny.rstudio.com/articles/basics.html

How would we make a Shiny app?

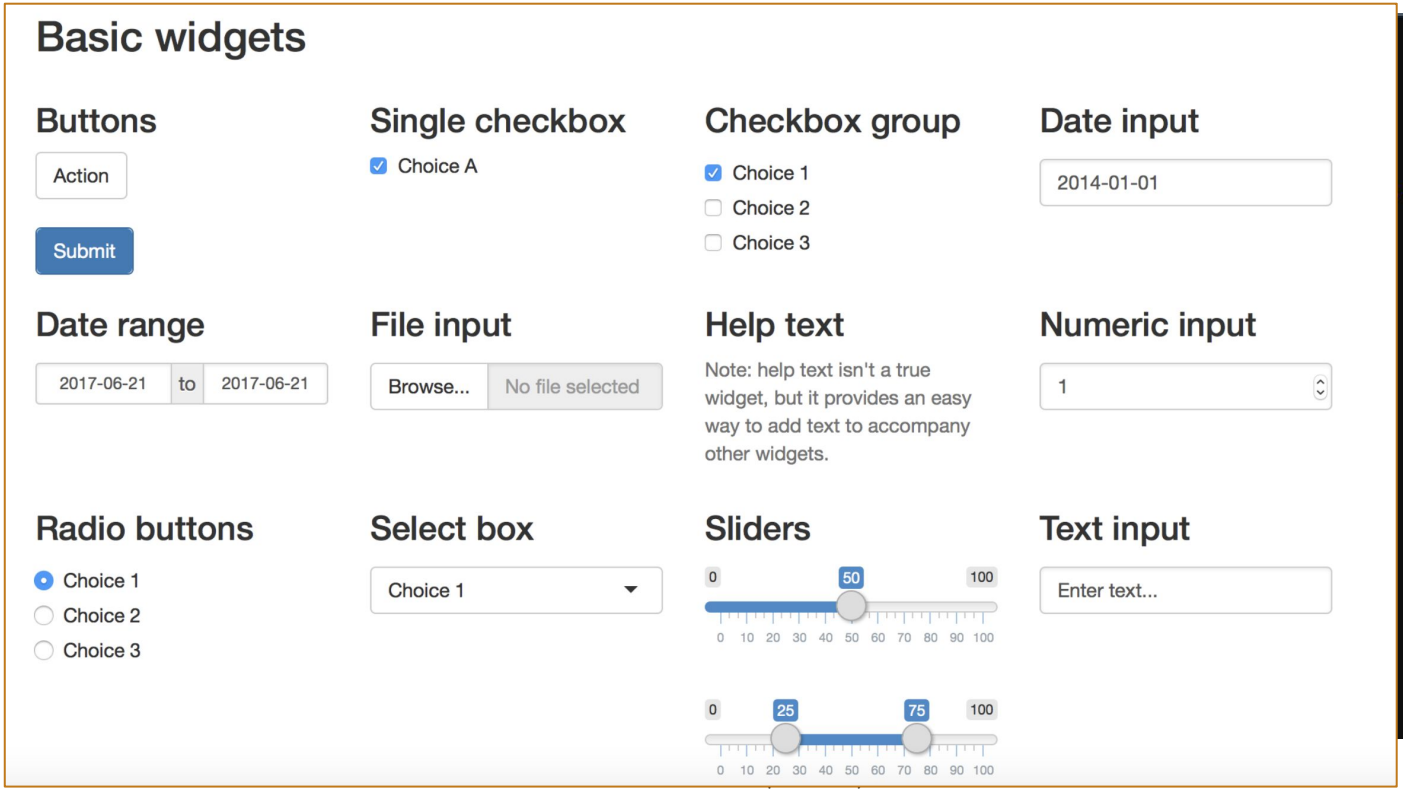
Shiny apps have three main components:

1. a user interface object

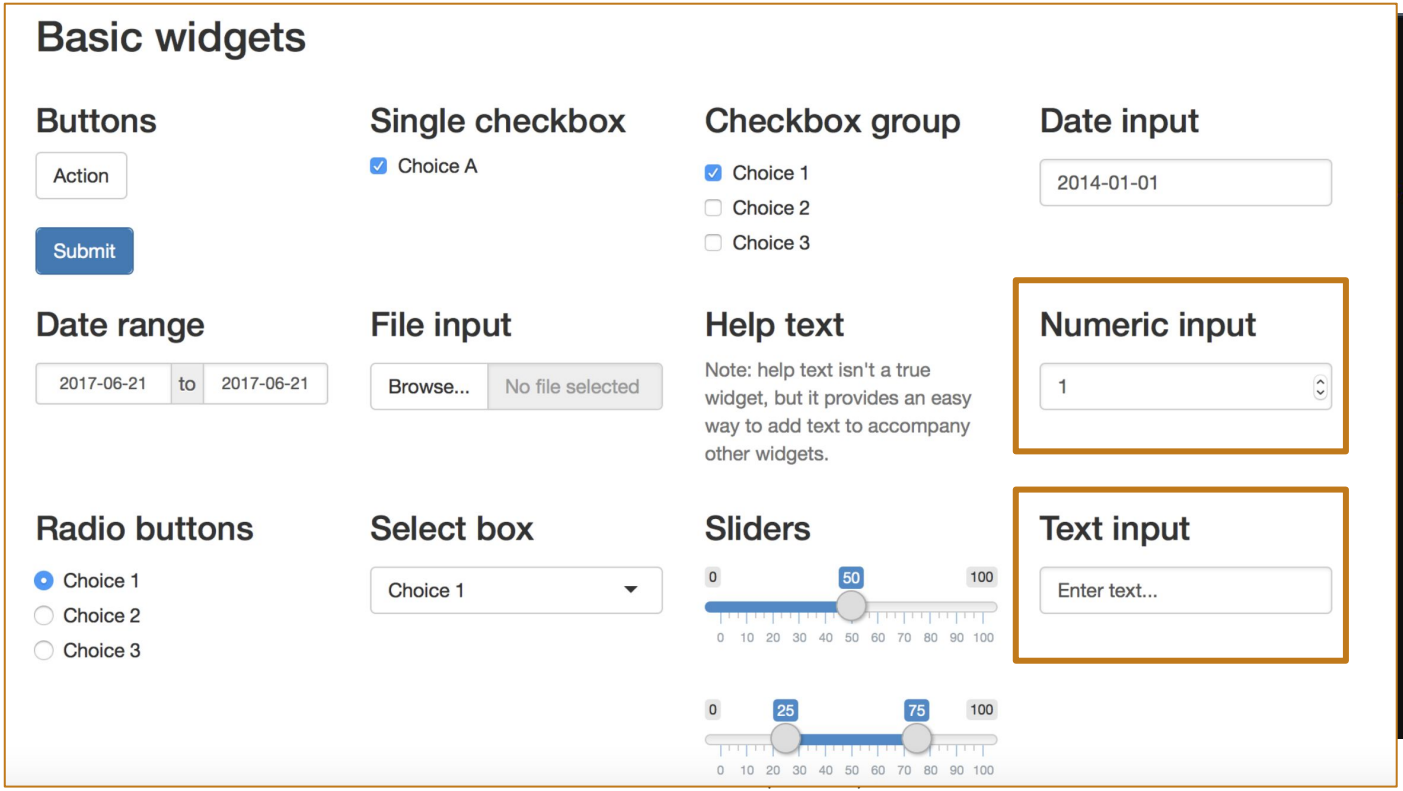
```
library(shiny)

# Define UI for app that draws a histogram
ui <- fluidPage(
  # App title
  titlePanel("Hello Shiny!"),
  # Sidebar layout with input and output definitions
  sidebarLayout(
    # Sidebar panel for inputs
    sidebarPanel(
      # Input: Slider for the number of bins
      sliderInput(inputId = "bins",
                  label = "Number of bins:",
                  min = 1,
                  max = 50,
                  value = 30)

    ),
    # Main panel for displaying outputs
    mainPanel(
      # Output: Histogram
      plotOutput(outputId = "distPlot")

    )
  )
)
```

https://shiny.rstudio.com/articles/basics.html

How would we make a Shiny app?



How would we make a Shiny app?



How would we make a Shiny app?

Shiny apps have three main components:

1. a user interface object
2. a server function

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

  # Histogram of the Old Faithful Geyser Data
  # with requested number of bins
  # This expression that generates a histogram is wrapped in a call
  # to renderPlot to indicate that:
  #
  # 1. It is "reactive" and therefore should be automatically
  #    re-executed when inputs (input$bins) change
  # 2. Its output type is a plot
  output$distPlot <- renderPlot({

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

    hist(x, breaks = bins, col = "#75AADB", border = "white",
         xlab = "Waiting time to next eruption (in mins)",
         main = "Histogram of waiting times")

  })

}
```

https://shiny.rstudio.com/articles/basics.html

How would we make a Shiny app?

Shiny apps have three main components:

1. a user interface object
2. a server function

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

  # Histogram of the Old Faithful Geyser Data
  # with requested number of bins
  # This expression that generates a histogram is wrapped in a call
  # to renderPlot to indicate that:
  #
  # 1. It is "reactive" and therefore should be automatically
  #    re-executed when inputs (input$bins) change
  # 2. Its output type is a plot
  output$distPlot <- renderPlot({

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

    hist(x, breaks = bins, col = "#75AADB", border = "white",
         xlab = "Waiting time to next eruption (in mins)",
         main = "Histogram of waiting times")

  })

}
```

https://shiny.rstudio.com/articles/basics.html

How would we make a Shiny app?

Shiny apps have three main components:

1. a user interface object
2. a server function

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

  # Histogram of the Old Faithful Geyser Data
  # with requested number of bins
  # This expression that generates a histogram is wrapped in a call
  # to renderPlot to indicate that:
  #
  # 1. It is "reactive" and therefore should be automatically
  #    re-executed when inputs (input$bins) change
  # 2. Its output type is a plot
  output$distPlot <- renderPlot({

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

    hist(x, breaks = bins, col = "#75AADB", border = "white",
         xlab = "Waiting time to next eruption (in mins)",
         main = "Histogram of waiting times")

  })

}
```

https://shiny.rstudio.com/articles/basics.html

How would we make a Shiny app?

Shiny apps have three main components:

1. a user interface object
2. a server function
3. a call to `ShinyApp()`

```
ui <- {
  ### UI
  }

server <- {
  ### Server
  }

shinyApp(ui = ui, server = server)

runApp(shinyApp(ui = ui, server = server))
```

https://shiny.rstudio.com/articles/basics.html

So what is a Shiny dashboard?

- Simply put, a collection of Shiny apps
- More generally, a graphical interface for users to quickly visualize important metrics

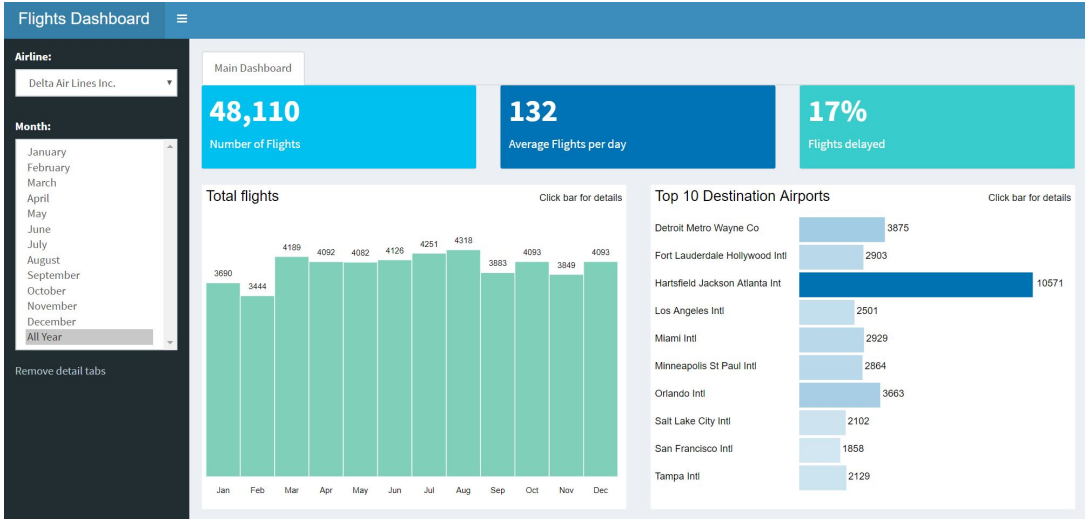


Image: https://db.rstudio.com/best-practices/dashboards/dashboard.png

Reasons to use Shiny dashboard

- If you know Shiny, it provides a nice framework for making a professional looking product
- Even if you don't know Shiny, it is pretty easy to get a professional looking product without wrangling with layouts
- You can do a lot without needing to know HTML and CSS
- If you already know flexdashboard, this may not provide a ton of advantages
    - However, there is a new package called shinydashboardPlus that provides even more features and is built off of shinydashboard

Things to think about before we start

Before we start coding, I like to ask myself a few questions

1. Who will my users be?
2. What insights do we want our users to be able to gain?
3. Is this the best way to present the data given #1 & #2?

and one more...

How will users access the dashboard?

R provides a couple of hosting options:

1. Deploy to the cloud using Shinyapps.io
    a. Different pricing levels offer different features, limits on active users, etc.
    b. Free tier does not include authentication
2. Host on Shiny server
    a. Open source
3. Deploy with RStudio Connect
    a. Commercial software
    b. Not free

* We needed authentication, so we set up the free Shiny server and app on an AWS server, then set up our website which had authentication to be a proxy for the server.

Now let’s do some exercises!

(In case you missed it) https://github.com/anastasia-lucas/rladies_shinydashboards