Creating Interactive Data Visualizations with

Shiny

●●●

GW Libraries Workshop
Dan Kerchner ~ Fall 2023

go.gwu.edu/rshinyworkshop

Shiny is a web application framework for R

Why R Shiny?

- You can add interactivity to R data visualizations you've built using *any* R libraries

  Example: shiny.library.gwu.edu/aciar/StatApps

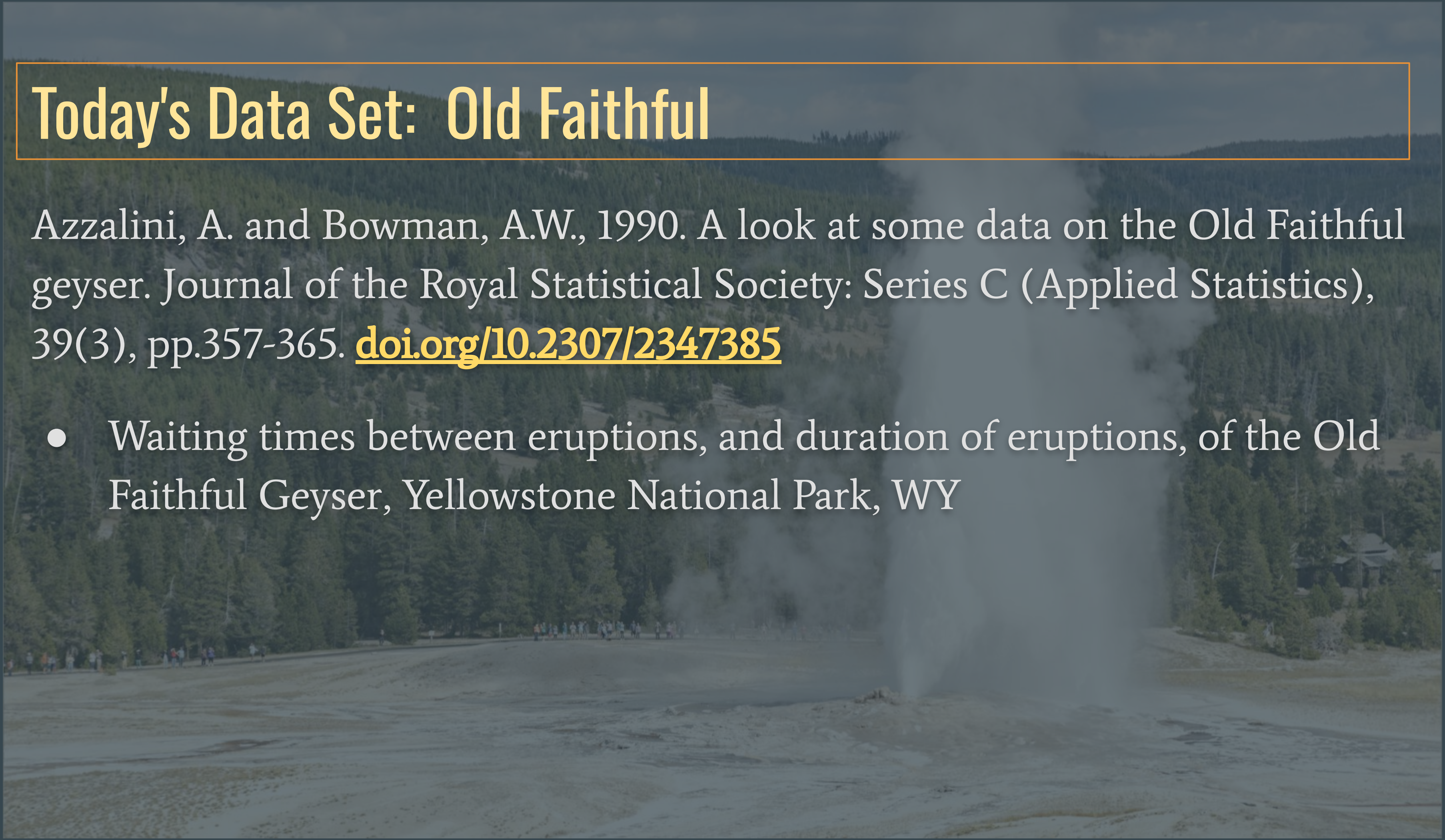
- Other types of interactivity are also possible with some R libraries when deployed on R Shiny
  - learnr - interactive tutorials
  - flexdashboard - "dashboard" style interactive pages

Goals

Today's Goal

- Create 2 Shiny apps:
  - Old Faithful
  - Framingham study data
- Publish a Shiny app in shinyapps.io

Today's Data Set: Old Faithful



Azzalini, A. and Bowman, A.W., 1990. A look at some data on the Old Faithful geyser. Journal of the Royal Statistical Society: Series C (Applied Statistics), 39(3), pp.357-365. doi.org/10.2307/2347385

- Waiting times between eruptions, and duration of eruptions, of the Old Faithful Geyser, Yellowstone National Park, WY

Today's Data Set: Framingham Heart Study

- framinghamheartstudy.org
- Long-term prospective study of the etiology of cardiovascular disease among a population of subjects in Framingham, MA
- Began in 1948 with 5,209 subjects
- Is the source of the term "risk factor"
- Over 3,000 peer-reviewed papers published based on this study
- Participants were each followed for a total of 24 years for cardiovascular events (heart attack, stroke, death, etc.)

Structure of an R Shiny app

```
library(shiny)
ui <- fluidPage(
  # Comma-separated list of components
  # collecting input from the user
  # using output in rendering
)
```

← FORM

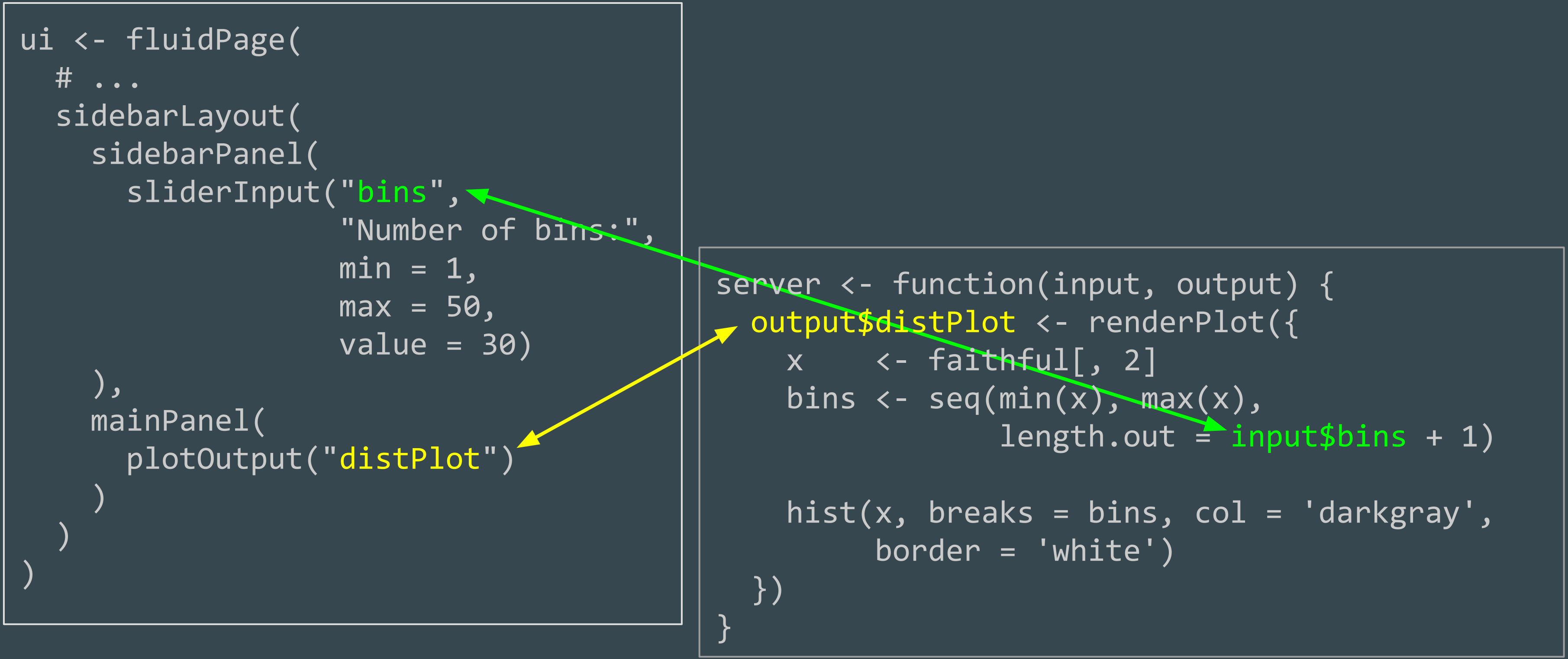
shiny.rstudio.com/articles/layout-guide.html

```
server <- function(input, output, session) {
  # server logic
  # using parts of input
  # and setting parts of output
}
```

← FUNCTION

```
# Run the app - ties together ui and server
shinyApp(ui, server)
```

Connecting inputs and outputs



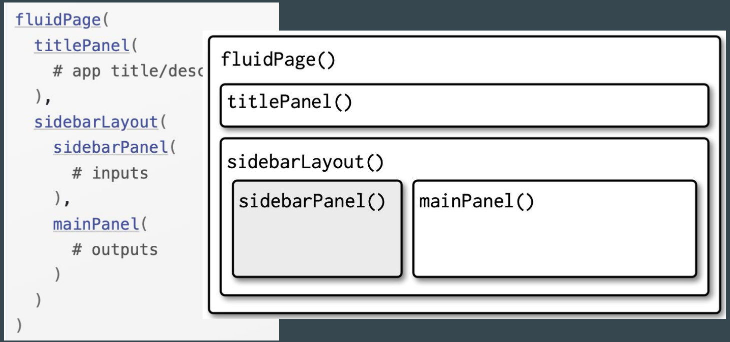
```
ui <- fluidPage(
  # ...
  sidebarLayout(
    sidebarPanel(
      sliderInput("bins",
                  "Number of bins:",
                  min = 1,
                  max = 50,
                  value = 30)
    ),
    mainPanel(
      plotOutput("distPlot")
    )
  )
)
```

```
server <- 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')
  })
}
```

Layouts: Single-page layouts

There are different layouts available:



```
fluidPage(
  titlePanel(
    # app title/desc
  ),
  sidebarLayout(
    sidebarPanel(
      # inputs
    ),
    mainPanel(
      # outputs
    )
  )
)
```

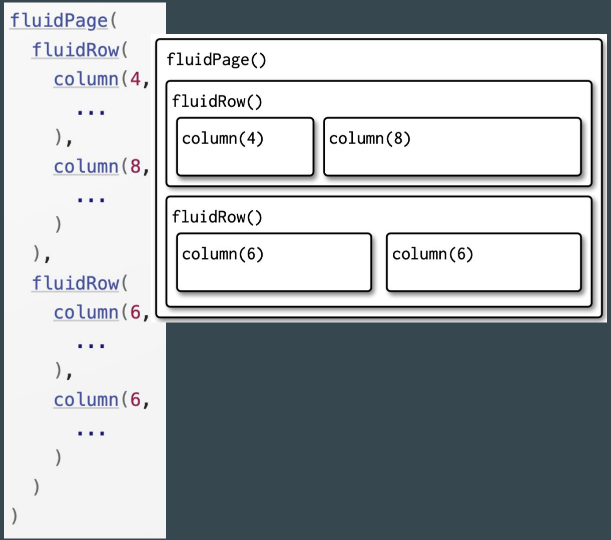
fluidPage()

titlePanel()

sidebarLayout()

sidebarPanel()

mainPanel()



```
fluidPage(
  fluidRow(
    column(4,
      ...
    ),
    column(8,
      ...
    )
  ),
  fluidRow(
    column(6,
      ...
    ),
    column(6,
      ...
    )
  )
)
```

fluidPage()

fluidRow()

column(4)

column(8)

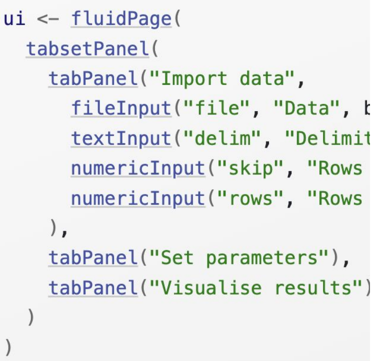
fluidRow()

column(6)

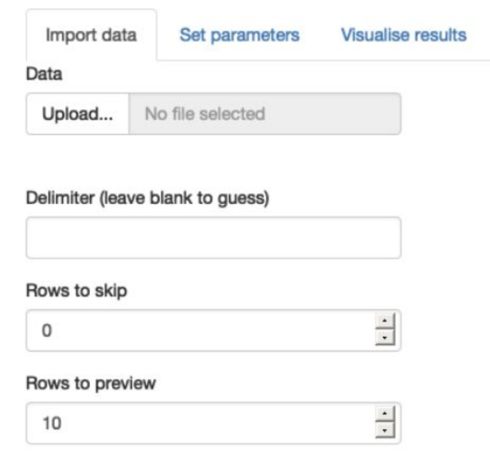
column(6)

Layouts: Multi-page layouts

tabsetPanel:



```
ui <- fluidPage(
  tabsetPanel(
    tabPanel("Import data",
      fileInput("file", "Data", b
      textInput("delim", "Delimit
      numericInput("skip", "Rows
      numericInput("rows", "Rows
    ),
    tabPanel("Set parameters"),
    tabPanel("Visualise results")
  )
)
```



Data

No file selected

Delimiter (leave blank to guess)

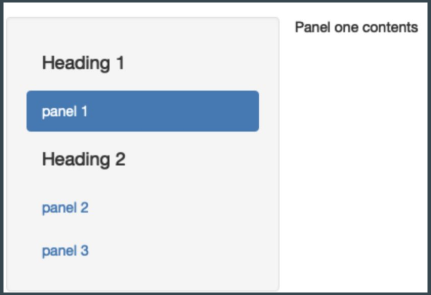
Rows to skip

0

Rows to preview

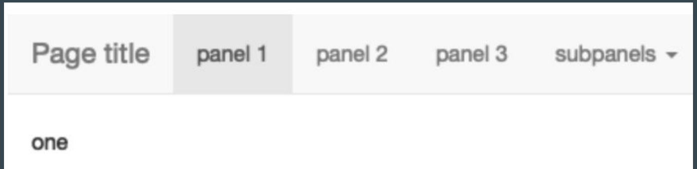
10

navlistPane



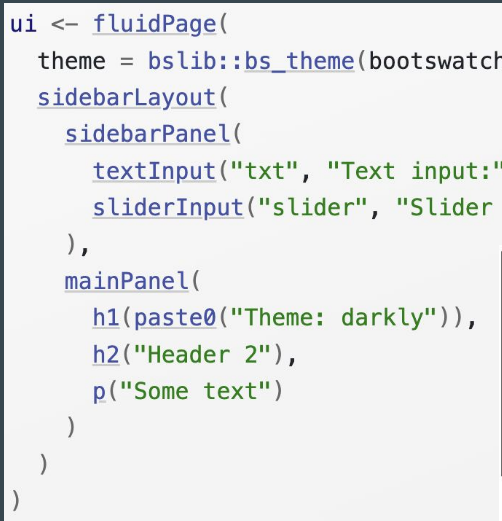
Panel one contents

navbarPage:

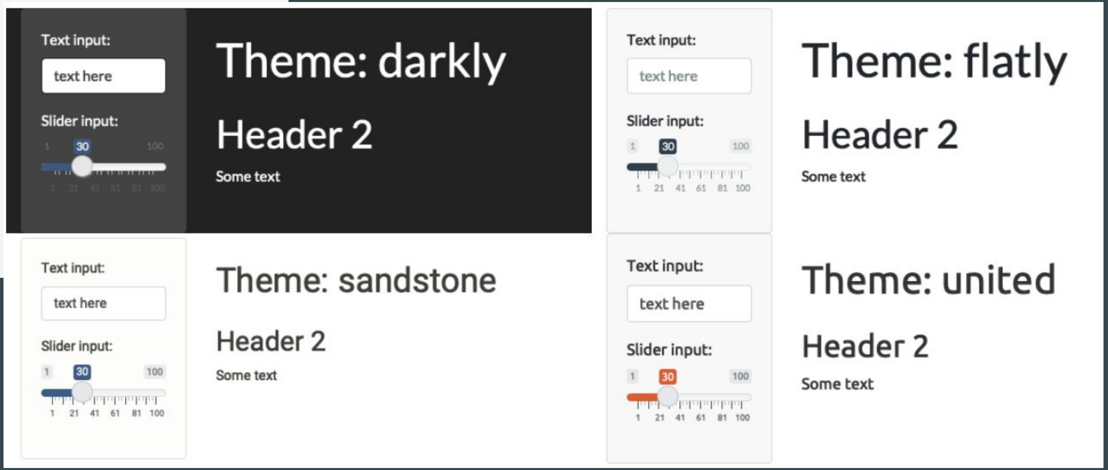


one

Themes!

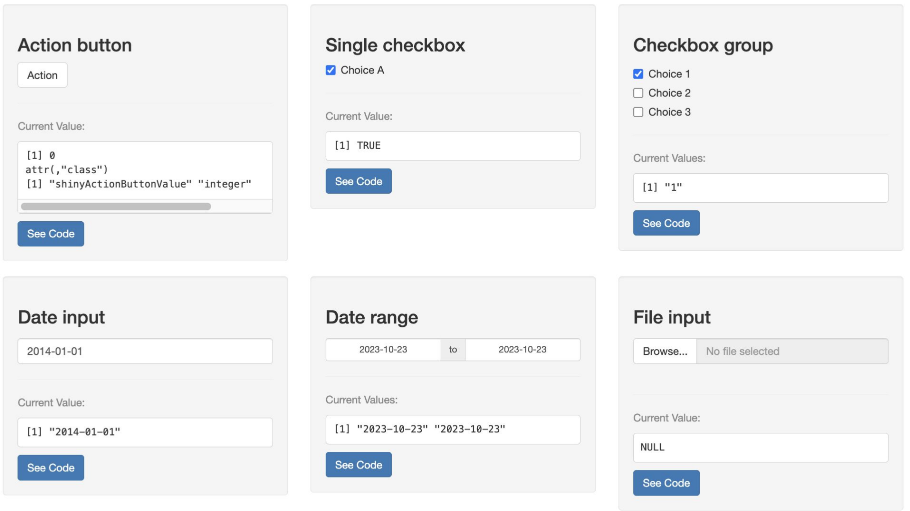


```
ui <- fluidPage(
  theme = bslib::bs_theme(bootswatch = "darkly"),
  sidebarLayout(
    sidebarPanel(
      textInput("txt", "Text input:", "text here"),
      sliderInput("slider", "Slider input:", 1, 100, 30)
    ),
    mainPanel(
      h1(paste0("Theme: darkly")),
      h2("Header 2"),
      p("Some text")
    )
  )
)
```



Input widget types

See shiny.posit.co/r/gallery/widgets/widget-gallery



Action button

Action

Current Value:

```
[1] 0
attr(,"class")
[1] "shinyActionButtonValue" "integer"
```

See Code

Single checkbox

☑ Choice A

Current Value:

```
[1] TRUE
```

See Code

Checkbox group

☑ Choice 1
☐ Choice 2
☐ Choice 3

Current Values:

```
[1] "1"
```

See Code

Date input

2014-01-01

Current Value:

```
[1] "2014-01-01"
```

See Code

Date range

2023-10-23 to 2023-10-23

Current Values:

```
[1] "2023-10-23" "2023-10-23"
```

See Code

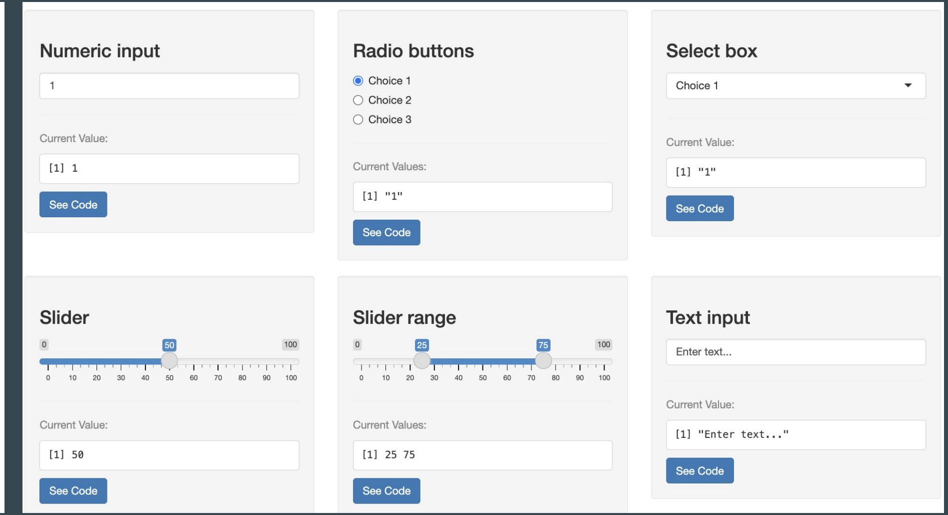
File input

Browse... No file selected

Current Value:

```
NULL
```

See Code



Numeric input

1

Current Value:

```
[1] 1
```

See Code

Radio buttons

● Choice 1
○ Choice 2
○ Choice 3

Current Values:

```
[1] "1"
```

See Code

Select box

Choice 1

Current Value:

```
[1] "1"
```

See Code

Slider

Current Value:

```
[1] 50
```

See Code

Slider range

Current Values:

```
[1] 25 75
```

See Code

Text input

Enter text...

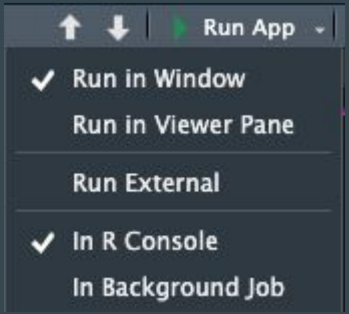
Current Value:

```
[1] "Enter text..."
```

See Code

Workflow - Steps to create an RShiny app

- Create the "empty" app
- Lay out the app and write the **`ui`**
- Add functionality bit by bit, test incrementally
- Deploy/Distribute



Options for sharing your app with others

- Share for people to run locally

  shiny.posit.co/r/articles/share/deployment-local

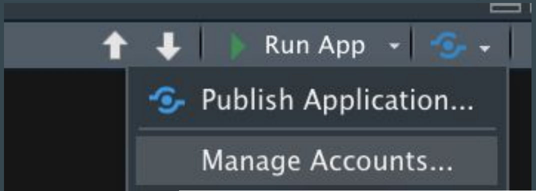
  - Post to a URL as zip file - `runUrl('https://github.com/rstudio/shiny_example/archive/master.zip')`
  - Place on GitHub
    - as repository - `shiny::runGitHub('shiny_example', 'rstudio')`
    - as gist - `shiny::runGist('3239667')`
- Share for people to access on the web

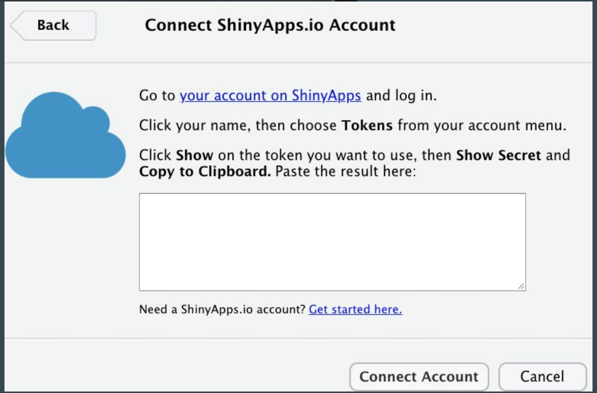
  shiny.posit.co/r/articles/share/deployment-web

  - Publish to shinyapps.io - `https://kerchner.shinyapps.io/rshiny-test/`
  - Publish to another shiny server - `https://shiny.library.gwu.edu/aciar/StatApps/Normal_prob/`

Publishing your app to shinyapps.io

- Create account
- Create a token
- Copy the token (***with secret***) into RStudio "Connect Account"
- Publish your application
- View at: *youraccount.shinyapps.io/yourapp*





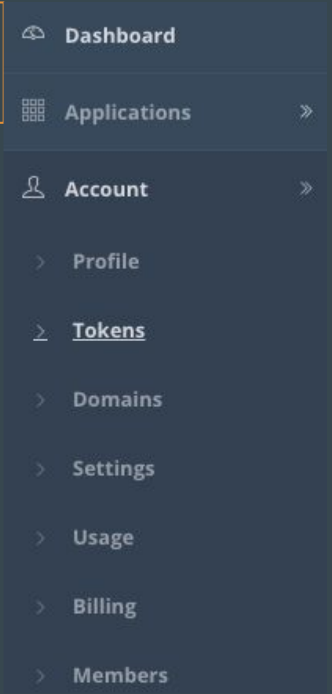
Connect ShinyApps.io Account

Go to your account on ShinyApps and log in.

Click your name, then choose **Tokens** from your account menu.

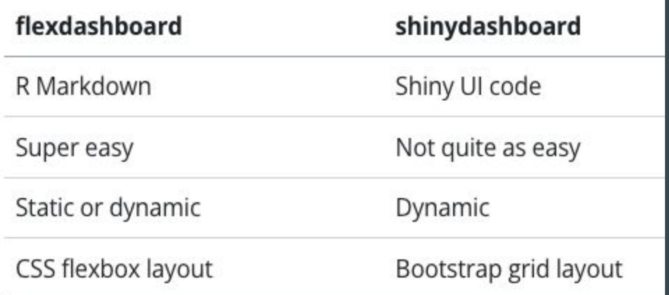
Click **Show** on the token you want to use, then **Show Secret** and **Copy to Clipboard.** Paste the result here:

Need a ShinyApps.io account? Get started here.

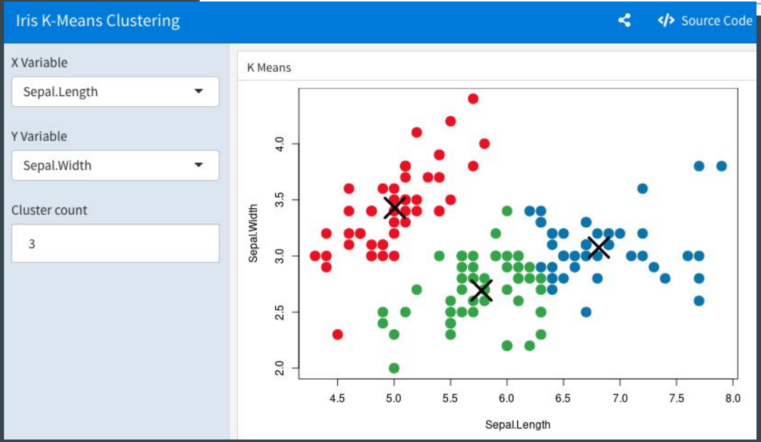


Dashboards with RShiny

- Two options (see shiny.posit.co/r/articles/build/dashboards)
  - shinydashboard
    rstudio.github.io/shinydashboard
  - flexdashboard
    pkgs.rstudio.com/flexdashboard
    This is an RMarkdown document that "knits" to HTML



flexdashboard	shinydashboard
R Markdown	Shiny UI code
Super easy	Not quite as easy
Static or dynamic	Dynamic
CSS flexbox layout	Bootstrap grid layout



Iris K-Means Clustering

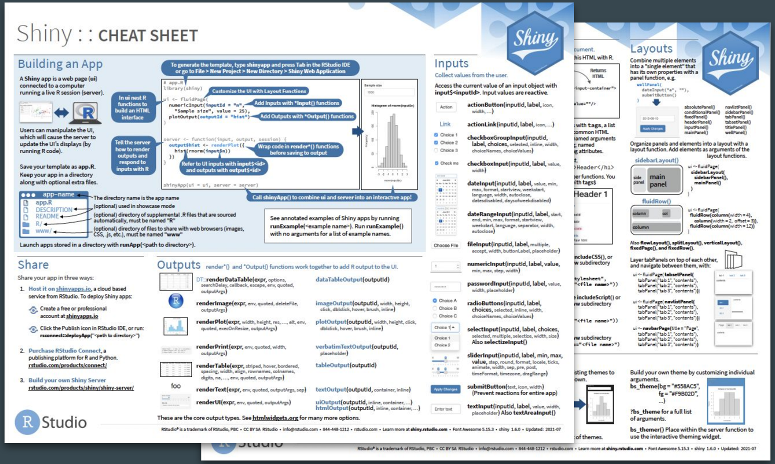
X Variable: Sepal.Length

Y Variable: Sepal.Width

Cluster count: 3

Shiny Resources

- Reference: [shiny.posit.co/reference/shiny/latest]
- Tutorial : [shiny.posit.co/tutorial]
- Online book: [mastering-shiny.org]
- Gallery : [shiny.posit.co/r/gallery]
- Cheat sheet : [shiny.posit.co/articles/cheatsheet.html]
- LinkedIn Learning
- Library Resources



Shiny : : CHEAT SHEET

Building an App

A **Shiny** app is a web page (**ui**) connected to a computer running a live R session (**server**).

Users can manipulate the UI, which will cause the server to update the UI's displays (by running R code).

Save your template as **app.R**. Keep your app in a directory along with optional extra files.

To generate the template, type shinyapp and press Tab in the RStudio IDE or go to File > New Project > New Directory > Shiny Web Application

```
# app.R
library(shiny)

ui <- fluidPage(
  numericInput(inputId = "n",
    "Sample size", value = 25),
  plotOutput(outputId = "hist")
)


server <- function(input, output, session) {
  output$hist <- renderPlot({
    hist(rnorm(input$n))
  })
}

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

In ui nest R functions to build an HTML interface

Customize the UI with Layout Functions

Add Inputs with *Input() functions

Add Outputs with *Output() functions

Tell the server how to render outputs and respond to inputs with R

Wrap code in render*() functions before saving to output

Refer to UI inputs with input$<id> and outputs with output$<id>

Call shinyApp() to combine ui and server into an interactive app!

- app-name — The directory name is the app name
- app.R
- DESCRIPTION — (optional) used in showcase mode
- README — (optional) used in showcase mode
- R/ — (optional) directory of supplemental .R files that are sourced automatically, must be named "R"
- www/ — (optional) directory of files to share with web browsers (images, CSS, .js, etc.), must be named "**www**"

Launch apps stored in a directory with **runApp**(<path to directory>).

See annotated examples of Shiny apps by running **runExample**(<example name>). Run **runExample()** with no arguments for a list of example names.

Share

Share your app in three ways:

1. Host it on shinyapps.io, a cloud based service from RStudio. To deploy Shiny apps:
   - Create a free or professional account at shinyapps.io
   - Click the Publish icon in RStudio IDE, or run: **rsconnect::deployApp**("<path to directory>")
2. Purchase RStudio Connect, a publishing platform for R and Python. rstudio.com/products/connect/
3. Build your own Shiny Server rstudio.com/products/shiny/shiny-server/

Outputs

render*() and *Output() functions work together to add R output to the UI.

DT::**renderDataTable**(expr, options, searchDelay, callback, escape, env, quoted, outputArgs)	**dataTableOutput**(outputId)
renderImage(expr, env, quoted, deleteFile, outputArgs)	**imageOutput**(outputId, width, height, click, dblclick, hover, brush, inline)
renderPlot(expr, width, height, res, …, alt, env, quoted, execOnResize, outputArgs)	**plotOutput**(outputId, width, height, click, dblclick, hover, brush, inline)
renderPrint(expr, env, quoted, width, outputArgs)	**verbatimTextOutput**(outputId, placeholder)
renderTable(expr, striped, hover, bordered, spacing, width, align, rownames, colnames, digits, na, …, env, quoted, outputArgs)	**tableOutput**(outputId)
renderText(expr, env, quoted, outputArgs, sep)	**textOutput**(outputId, container, inline)
renderUI(expr, env, quoted, outputArgs)	**uiOutput**(outputId, inline, container, …) **htmlOutput**(outputId, inline, container, …)

These are the core output types. See htmlwidgets.org for many more options.

Inputs

Collect values from the user.

Access the current value of an input object with **input$<inputId>**. Input values are **reactive**.

- **actionButton**(inputId, label, icon, width, …)
- **actionLink**(inputId, label, icon, …)
- **checkboxGroupInput**(inputId, label, choices, selected, inline, width, choiceNames, choiceValues)
- **checkboxInput**(inputId, label, value, width)
- **dateInput**(inputId, label, value, min, max, format, startview, weekstart, language, width, autoclose, datesdisabled, daysofweekdisabled)
- **dateRangeInput**(inputId, label, start, end, min, max, format, startview, weekstart, language, separator, width, autoclose)
- **fileInput**(inputId, label, multiple, accept, width, buttonLabel, placeholder)
- **numericInput**(inputId, label, value, min, max, step, width)
- **passwordInput**(inputId, label, value, width, placeholder)
- **radioButtons**(inputId, label, choices, selected, inline, width, choiceNames, choiceValues)
- **selectInput**(inputId, label, choices, selected, multiple, selectize, width, size) Also **selectizeInput()**
- **sliderInput**(inputId, label, min, max, value, step, round, format, locale, ticks, animate, width, sep, pre, post, timeFormat, timezone, dragRange)
- **submitButton**(text, icon, width) (Prevent reactions for entire app)
- **textInput**(inputId, label, value, width, placeholder) Also **textAreaInput()**

RStudio® is a trademark of RStudio, PBC • CC BY SA RStudio • info@rstudio.com • 844-448-1212 • rstudio.com • Learn more at **shiny.rstudio.com** • Font Awesome 5.15.3 • shiny 1.6.0 • Updated: 2021-07

Layouts

Combine multiple elements into a "single element" that has its own properties with a panel function, e.g.

```
wellPanel(
  dateInput("a", ""),
  submitButton()
)
```

absolutePanel()
conditionalPanel()
fixedPanel()
headerPanel()
inputPanel()
mainPanel()
navlistPanel()
sidebarPanel()
tabPanel()
tabsetPanel()
titlePanel()
wellPanel()

Organize panels and elements into a layout with a layout function. Add elements as arguments of the layout functions.

sidebarLayout()

```
ui <- fluidPage(
  sidebarLayout(
    sidebarPanel(),
    mainPanel()
  )
)
```

fluidRow()

```
ui <- fluidPage(
  fluidRow(column(width = 4),
    column(width = 2, offset = 3)),
  fluidRow(column(width = 12))
)
```

Also **flowLayout(), splitLayout(), verticalLayout(), fixedPage(), and fixedRow()**.

Layer tabPanels on top of each other, and navigate between them, with:

```
ui <- fluidPage( tabsetPanel(
  tabPanel("tab 1", "contents"),
  tabPanel("tab 2", "contents"),
  tabPanel("tab 3", "contents")))

ui <- fluidPage( navlistPanel(
  tabPanel("tab 1", "contents"),
  tabPanel("tab 2", "contents"),
  tabPanel("tab 3", "contents")))

ui <- navbarPage(title = "Page",
  tabPanel("tab 1", "contents"),
  tabPanel("tab 2", "contents"),
  tabPanel("tab 3", "contents"))
```

Build your own theme by customizing individual arguments.

```
bs_theme(bg = "#558AC5",
         fg = "#F9B02D",
         ...)
```

?bs_theme for a full list of arguments.

bs_themer() Place within the server function to use the interactive theming widget.

RStudio® is a trademark of RStudio, PBC • CC BY SA RStudio • info@rstudio.com • 844-448-1212 • rstudio.com • Learn more at **shiny.rstudio.com** • Font Awesome 5.15.3 • shiny 1.6.0 • Updated: 2021-07

Interactive Data Visualization - in R, beyond Shiny

- Libraries
  - plotly, d3heatmap, heatmaply, leaflet
- Resources
  - htmlwidgets.org - htmlwidgets for R
  - r-graph-gallery.com/interactive-charts.html
  - posit.co/resources/cheatsheets
  - ggplot2-book.org
  - rkabacoff.github.io/datavis
  - worldbank.github.io/r-econ-visual-library

Interactive Data Visualization - Not in R

- Tableau
- Power BI
- Python libraries
- ...

Statistics+R help @ GW

R-Statistics Appointments:
academiccommons.gwu.edu/data-consulting

Also...

Appointments with me:
calendly.com/kerchner

Coding consultations (Python, R, git, etc.):
calendly.com/gwul-coding

Thanks!

Dan Kerchner kerchner@gwu.edu