RStudio

MACQUARIE
University

Introduction to Shiny

Kate Dodds, Biological Sciences
Kate.dodds@mq.edu.au

Quantitative advice

- There’s a link to todays slides and code on the MQ QUANTITATIVE ADIVCE page: http://quantitative-advice.gg.mq.edu.au/
- Link:
- Ref Materials also listed there:
  - https://shiny.rstudio.com/articles/shinyapps.html
  - http://docs.rstudio.com/shinyapps.io/
  - http://shiny.rstudio.com/tutorial/
  - https://shiny.rstudio.com/articles/basics.html
  - https://shiny.rstudio.com/articles/action-buttons.html
  - https://shiny.rstudio.com/articles/layout-guide.html
  - http://rstudio.github.io/shinydshboard/ also www.rstudio.com/resources/webinars
  - Felxdashboards (easier to learn) and shinydashboard

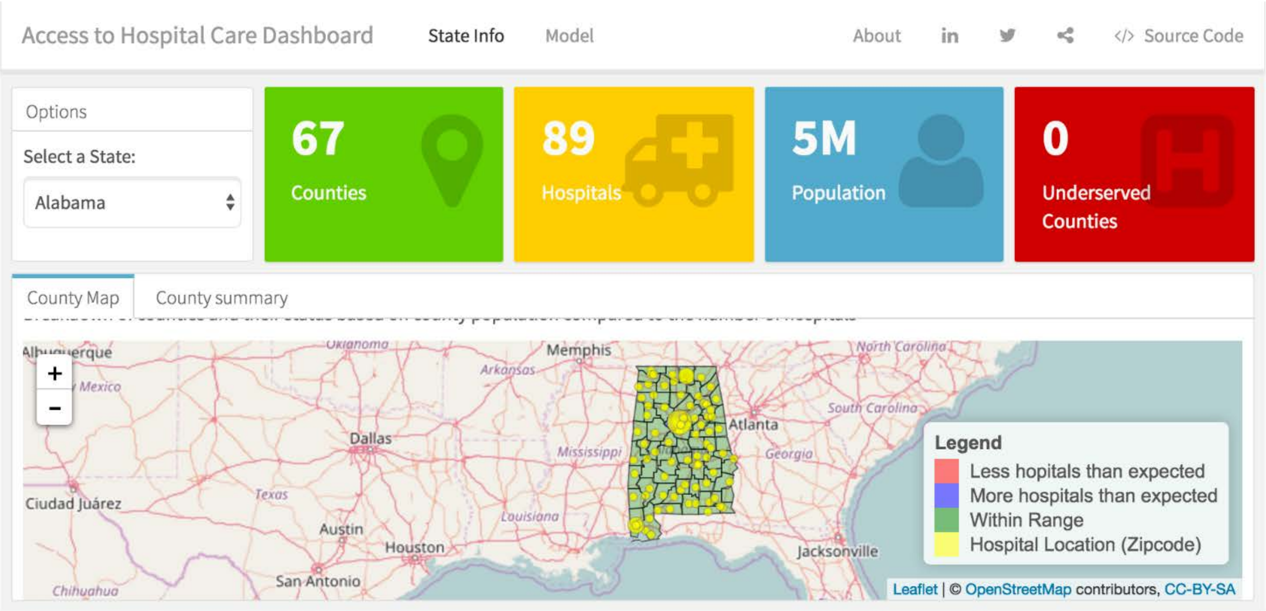
Session overview – just the tip of the iceberg

- What is Shiny?
- What can Shiny do?
- Walk through some basic Shiny apps
- Build a data explorer app
- Publish your app
- Find out where to find more info..
- Play with APPS

What is `Shiny`? http://shiny.rstudio.com/

- `Shiny` is an R package that makes it easy to build interactive web apps straight from R. Show/interact with your results.
- No web development skills are required. The shiny package functions are all built in HTML and can receive HTML as input.
- Host standalone apps on a webpage or embed them in R Markdown documents or build dashboards. You can also extend your Shiny apps with CSS themes, htmlwidgets, and JavaScript actions.
- *shinyapps.io* from Rstudio provides free hosting for open source projects. Paid options are also available – with some extra features.
- All R packages are supported, expect those that don’t work on Ubuntu linux or those that require access to the display (eg Tcl/Tk).
- Any packages installed form github, must have been installed using devtools v1.4 or later (devtools::install_github()).

What can Shiny do?


Access to Hospital Care Dashboard

Options

Select a State: Alabama

67 Counties

89 Hospitals

5M Population

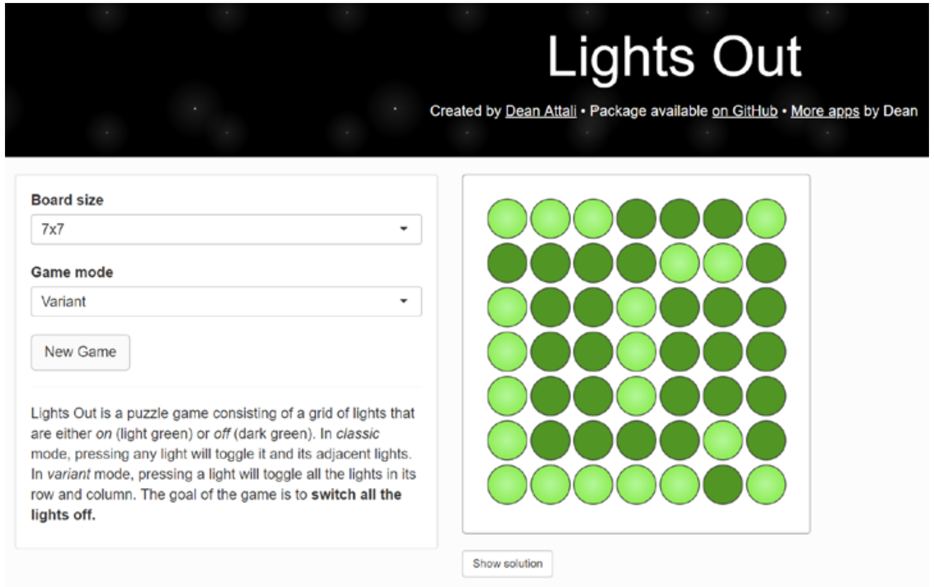
0 Underserved Counties

Legend

- 🟥 Less hopitals than expected
- 🟦 More hospitals than expected
- 🟩 Within Range
- 🟨 Hospital Location (Zipcode)

Leaflet | © OpenStreetMap contributors, CC-BY-SA

- Interactive data exploration and visualisation
- Dashboards with real-time data feeds:
  - Flexdashboards
  - Shinydashboards
- Use powerful open source Javascript visualisation libraries
- Check out the Shiny example gallery
  - https://shiny.rstudio.com/gallery/
  - https://gallery.shinyapps.io/lake_erie_fisheries_stock_assessment_app/
  - https://datasociety.com/kitamba-the-opportunity-project/
- Check out the Shiny widgets gallery
  - https://shiny.rstudio.com/gallery/widget-gallery.html
- All examples have associated code!


Lights Out

Created by Dean Attali • Package available on GitHub • More apps by Dean

Board size: 7x7

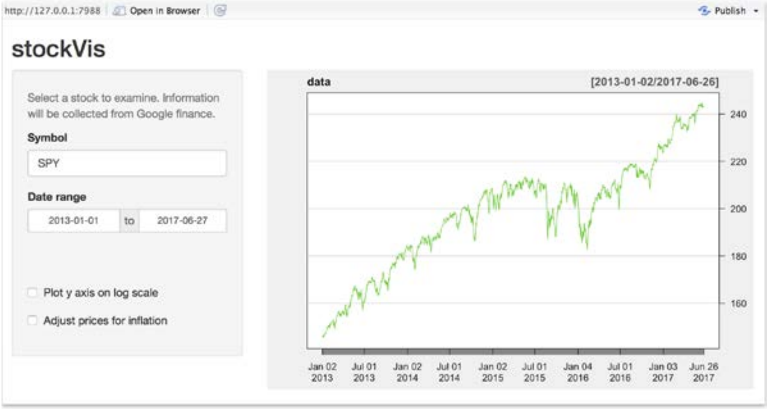
Game mode: Variant

Lights Out is a puzzle game consisting of a grid of lights that are either *on* (light green) or *off* (dark green). In *classic* mode, pressing any light will toggle it and its adjacent lights. In *variant* mode, pressing a light will toggle all the lights in its row and column. The goal of the game is to **switch all the lights off.**

Even games! https://daattali.com/shiny/lightsout/

Core components of a Shiny app

- UI – user interface (front end)
  - Input widgets collect information from the user
  - Displays outputs, such as plots or tables
  - Define how visual elements are laid out
  - Written in R syntax, but most functions won't be familiar



http://127.0.0.1:7988 Open in Browser

stockVis

Select a stock to examine. Information will be collected from Google finance.

Symbol

SPY

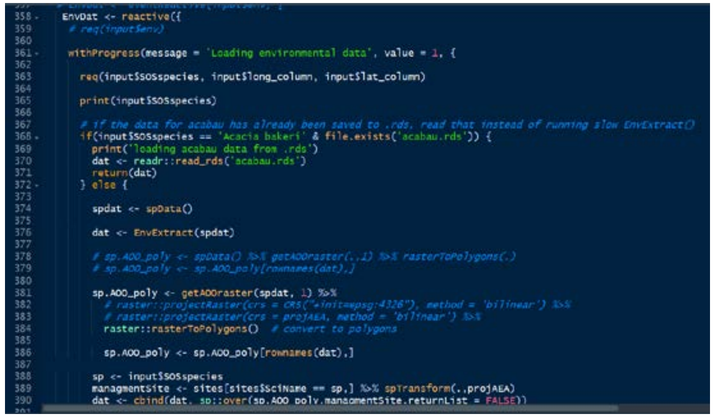
Date range

2013-01-01 to 2017-06-27

□ Plot y axis on log scale

□ Adjust prices for inflation

- Server (back end)
  - Where data is processed
  - Regular R code combined with special `shiny::` functions that implement reactivity
  - Reactive expressions are re-evaluated when their dependent values have changed



```
  EnvDat <- reactive({
   # req(input$env)

   withProgress(message = 'Loading environmental data', value = 1, {

     req(input$SOSspecies, input$long_column, input$lat_column)

     print(input$SOSspecies)

     # if the data for acabau has already been saved to .rds, read that instead of running slow EnvExtract()
     if(input$SOSspecies == 'Acacia bakeri' & file.exists('acabau.rds')) {
       print('loading acabau data from .rds')
       dat <- readr::read_rds('acabau.rds')
       return(dat)
     } else {

       spdat <- spData()

       dat <- EnvExtract(spdat)

       # sp.AOO_poly <- spData() %>% getAOOraster(.,1) %>% rasterToPolygons(.)
       # sp.AOO_poly <- sp.AOO_poly[rownames(dat),]

       sp.AOO_poly <- getAOOraster(spdat, 1) %>%
         # raster::projectRaster(crs = CRS("+init=epsg:4326"), method = 'bilinear') %>%
         # raster::projectRaster(crs = projAEA, method = 'bilinear') %>%
         raster::rasterToPolygons()  # convert to polygons

         sp.AOO_poly <- sp.AOO_poly[rownames(dat),]

       sp <- input$SOSspecies
       managmentSite <- sites[sites$SciName == sp,] %>% spTransform(.,projAEA)
       dat <- cbind(dat, sp::over(sp.AOO_poly,managmentSite,returnList = FALSE))
```

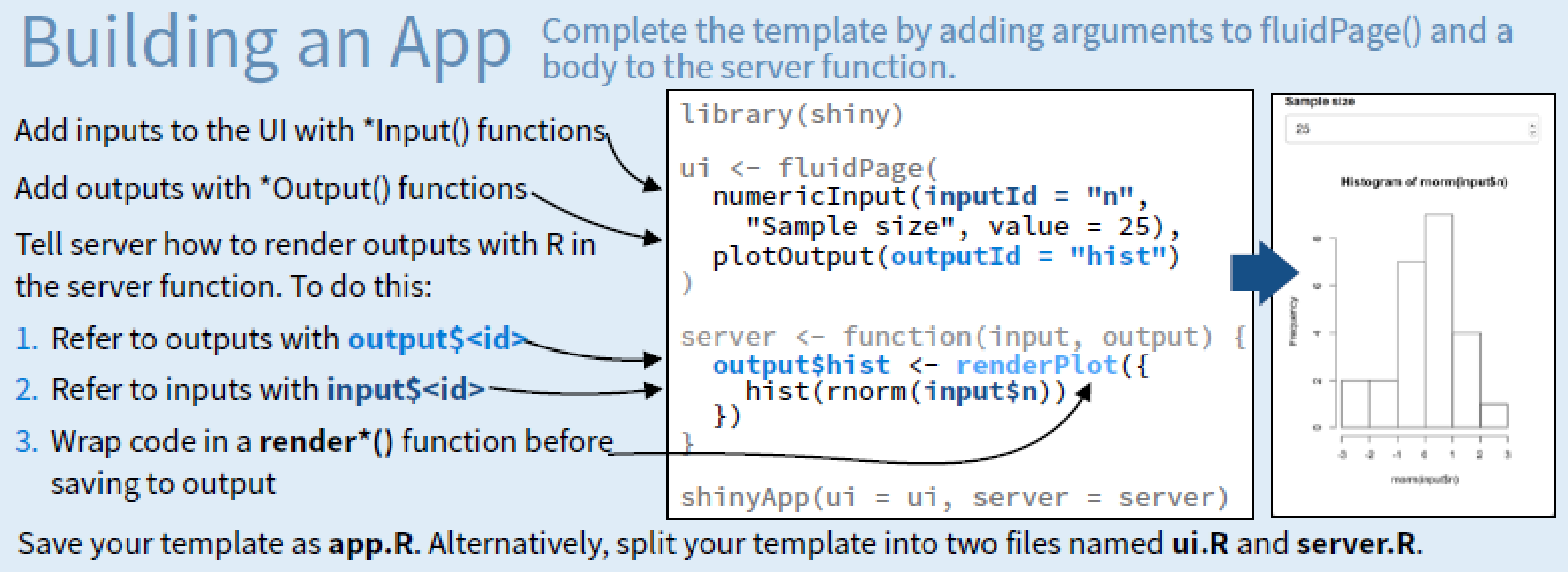
Shiny Template

- This template should be used for all the apps you develop
- It helps you remember the basic outline to stick to.. Even when things get more complicated.
- Just running these 4 lines, gets you an app!

```
library(shiny)

ui<- fluidPage("Hello World")
server<- function(input,output){}
shinyApp(ui=ui,server=server)
```

Try this!

Populating the template



Building an App

Complete the template by adding arguments to fluidPage() and a body to the server function.

Add inputs to the UI with *Input() functions

Add outputs with *Output() functions

Tell server how to render outputs with R in the server function. To do this:

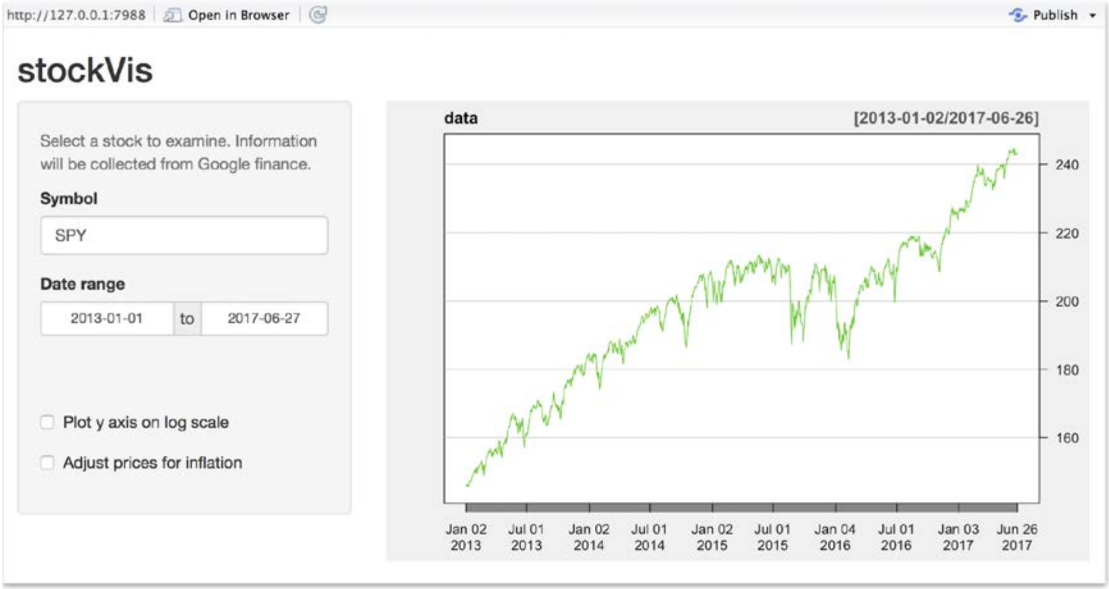
1. Refer to outputs with **output$<id>**
2. Refer to inputs with **input$<id>**
3. Wrap code in a **render*()** function before saving to output

```
library(shiny)

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

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

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

Save your template as **app.R**. Alternatively, split your template into two files named **ui.R** and **server.R**.

Reactive programming

- Reactive programming forms the basis of `Shiny` apps.
- Reactive objects change in response to changes in other objects.
- These objects can be connected in a chain of reactivity.
- A well known example is found in Microsoft Excel, where changing one cell can have consequences throughout the Workbook.



stockVis

Select a stock to examine. Information will be collected from Google finance.

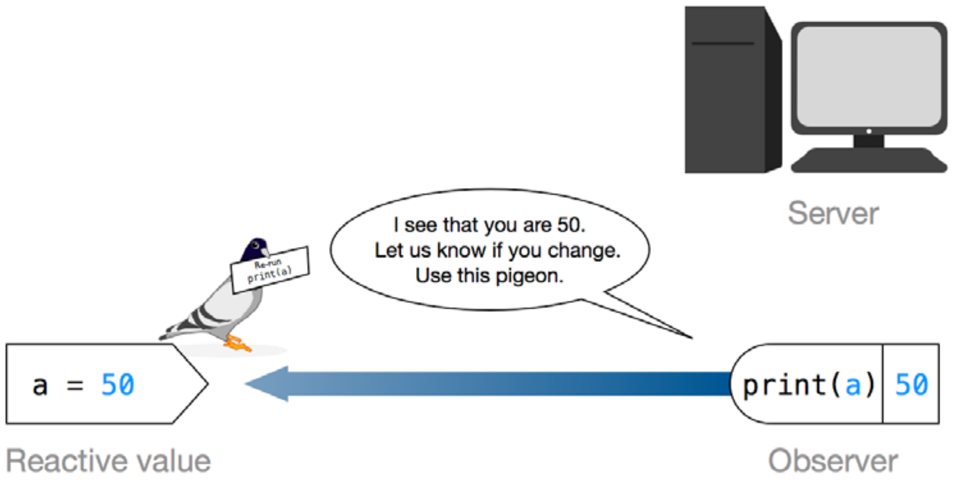
Symbol: SPY

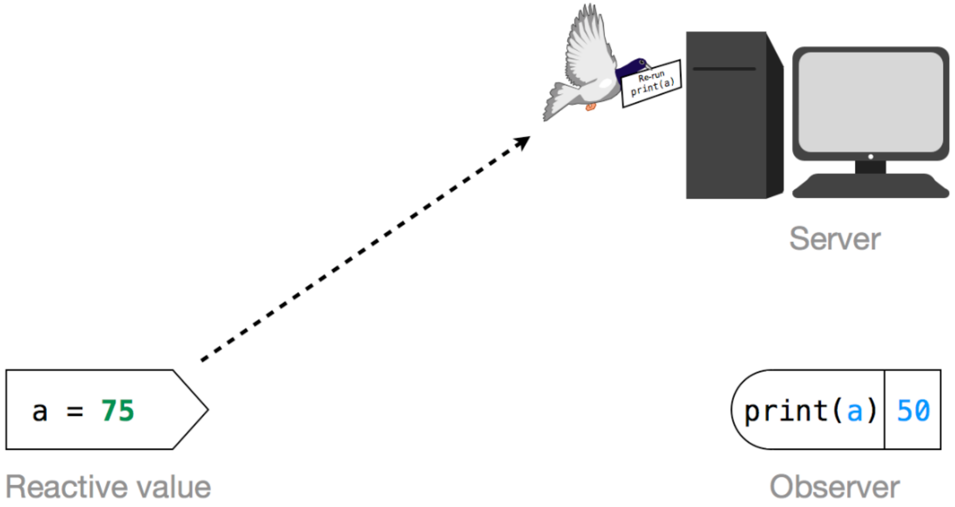
Date range: 2013-01-01 to 2017-06-27

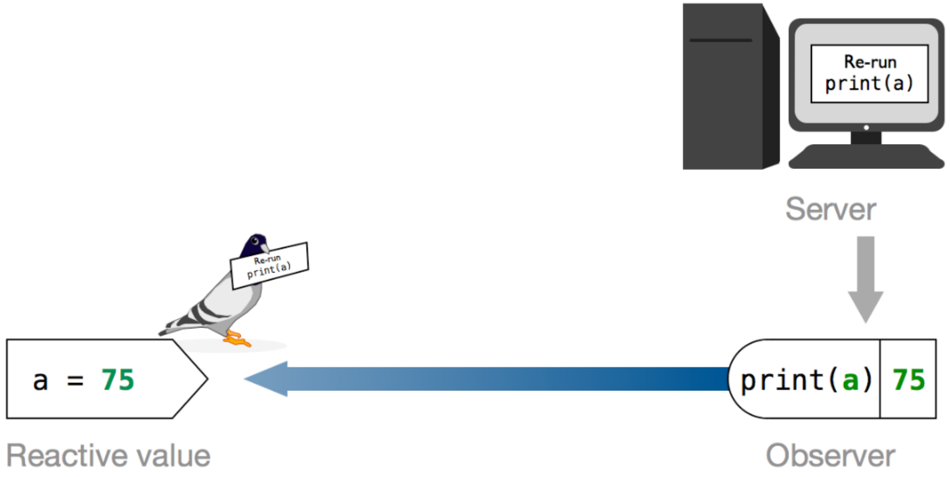
□ Plot y axis on log scale

□ Adjust prices for inflation

Understanding reactivity

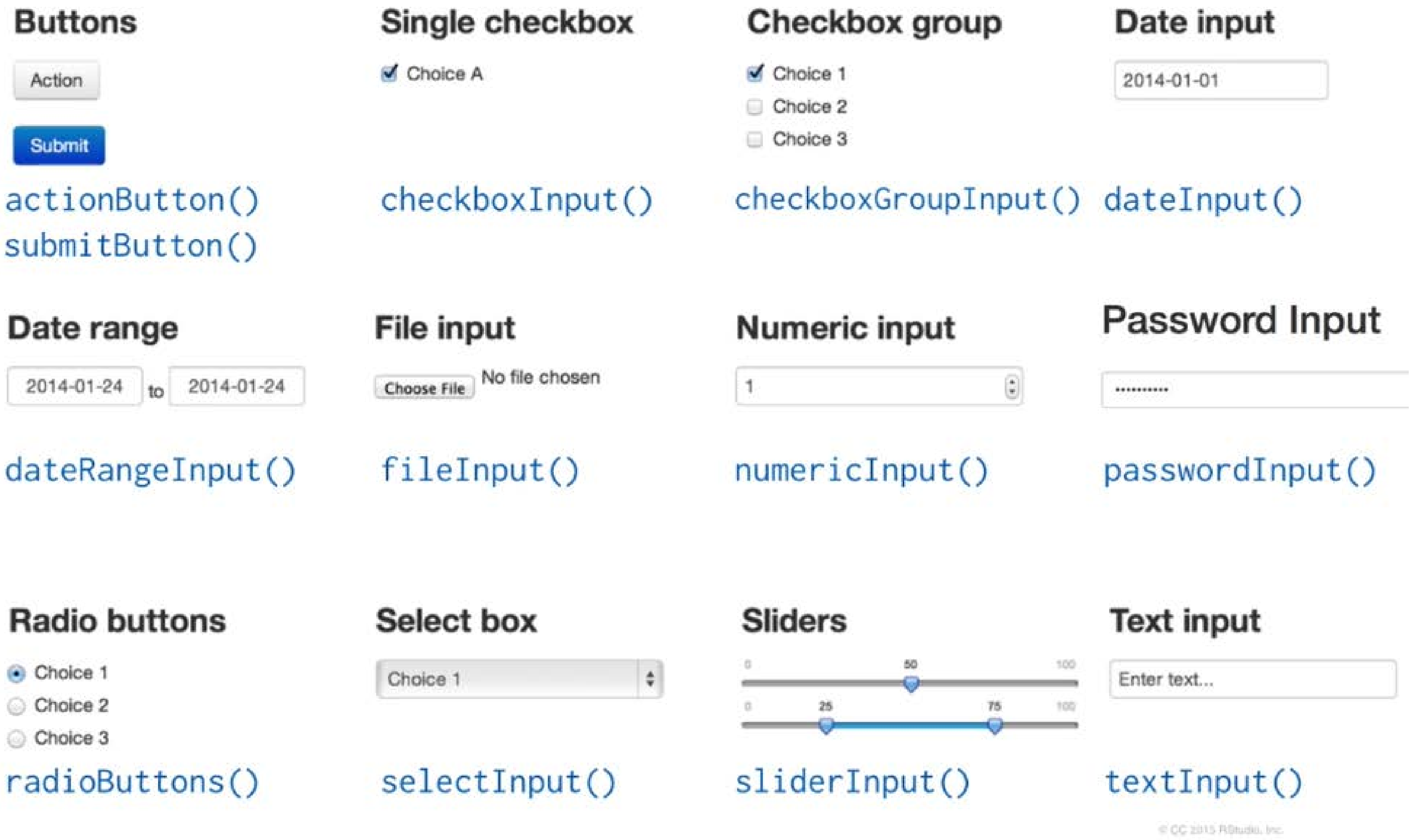






https://shiny.rstudio.com/articles/understanding-reactivity.html

Reactive functions: UI: inputs



Buttons

Action

Submit

`actionButton()`
`submitButton()`

Single checkbox

☑ Choice A

`checkboxInput()`

Checkbox group

☑ Choice 1
□ Choice 2
□ Choice 3

`checkboxGroupInput()`

Date input

2014-01-01

`dateInput()`

Date range

2014-01-24 to 2014-01-24

`dateRangeInput()`

File input

Choose File No file chosen

`fileInput()`

Numeric input

1

`numericInput()`

Password Input

••••••••••

`passwordInput()`

Radio buttons

● Choice 1
○ Choice 2
○ Choice 3

`radioButtons()`

Select box

Choice 1

`selectInput()`

Sliders

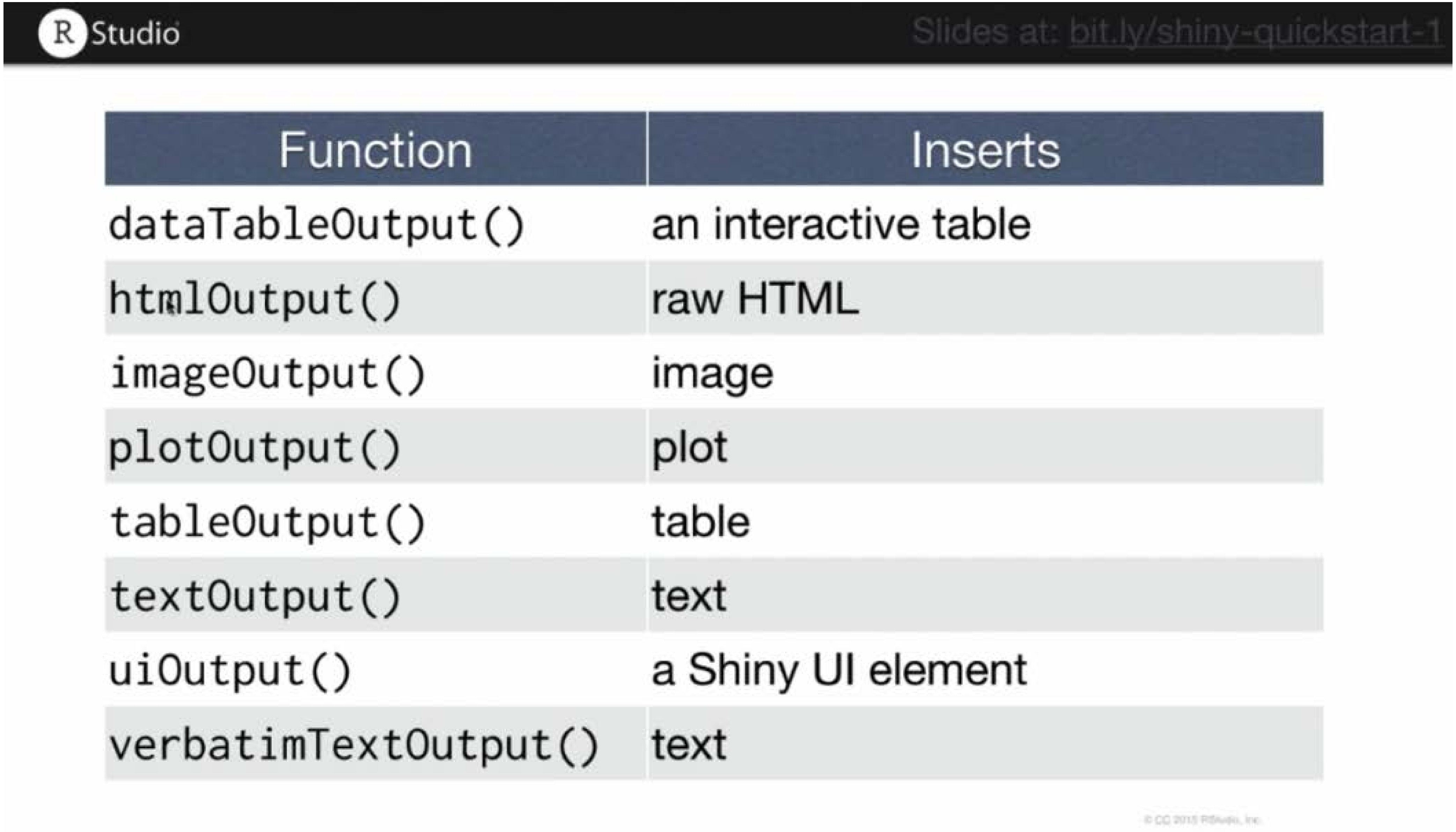
`sliderInput()`

Text input

Enter text...

`textInput()`

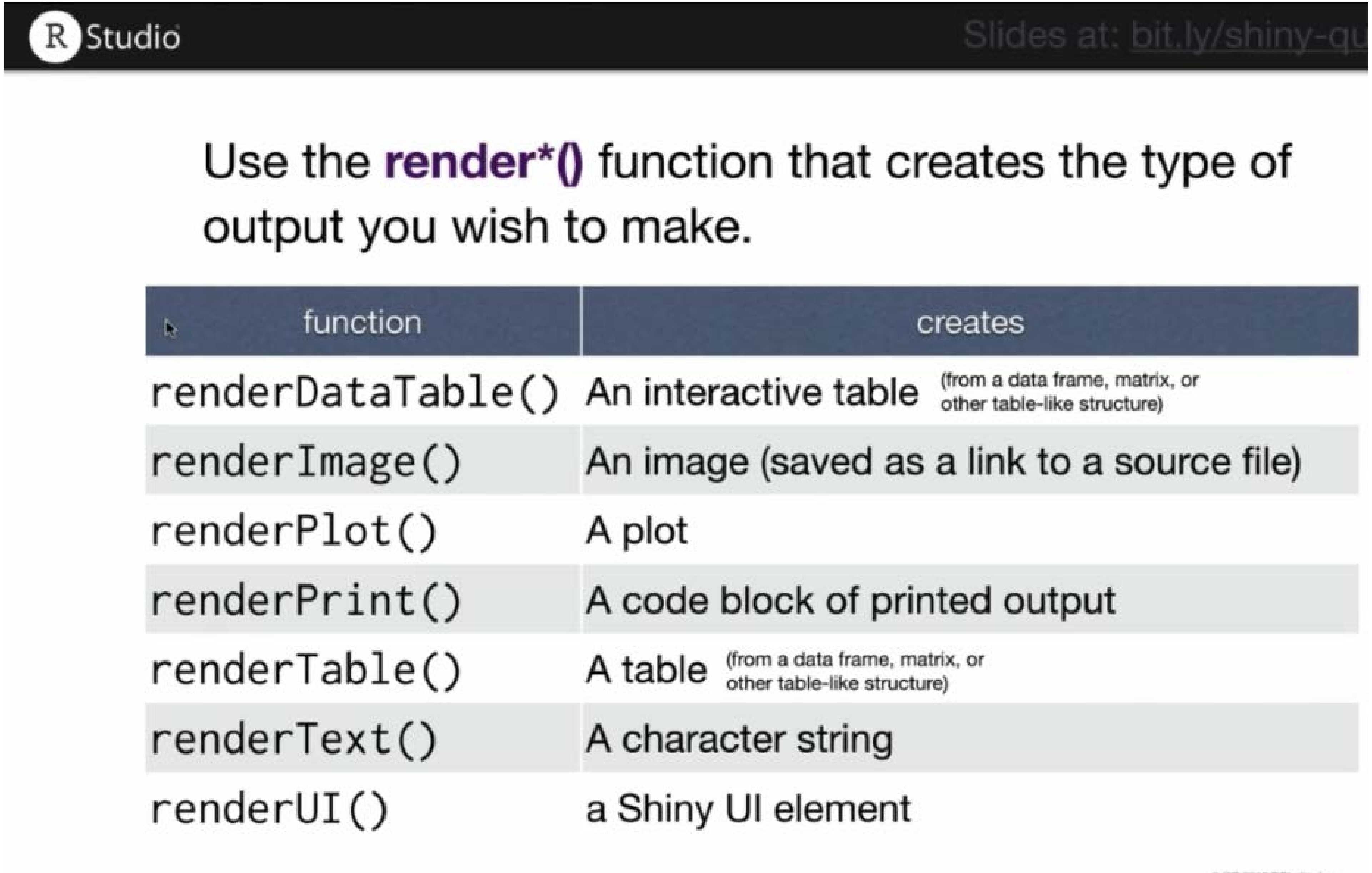
Reactive functions: UI: outputs



Slides at: bit.ly/shiny-quickstart-1

Function	Inserts
`dataTableOutput()`	an interactive table
`htmlOutput()`	raw HTML
`imageOutput()`	image
`plotOutput()`	plot
`tableOutput()`	table
`textOutput()`	text
`uiOutput()`	a Shiny UI element
`verbatimTextOutput()`	text

Reactive functions: SERVER: render



Use the **render*()** function that creates the type of output you wish to make.

function	creates
`renderDataTable()`	An interactive table (from a data frame, matrix, or other table-like structure)
`renderImage()`	An image (saved as a link to a source file)
`renderPlot()`	A plot
`renderPrint()`	A code block of printed output
`renderTable()`	A table (from a data frame, matrix, or other table-like structure)
`renderText()`	A character string
`renderUI()`	a Shiny UI element

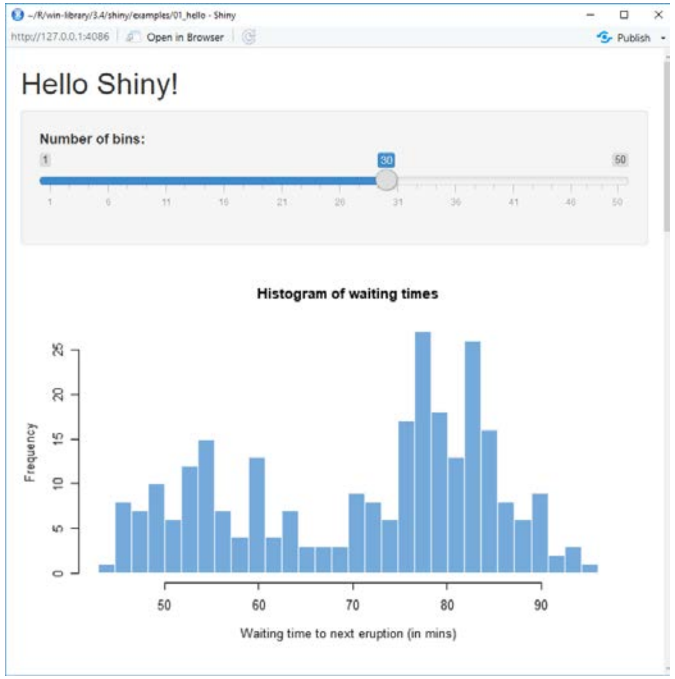
Complicating things..

- When trying to make more complicated apps it my be useful to separate the UI and SERVER components into different files
- Save these in your app's directory as ui.R and server.r – this folder name is the name of your app.
- Even better is to save a THIRD file defining variables common to both components.. This must be called global.r
- Also save any images/logos or data associated with your app in this folder. The contents of this folder will be uploaded to the hosting server when you publish your app.
- Efficient programming!

- https://shiny.rstudio.com/gallery/word-cloud.html - for an example

TEST 1 – some simple `Shiny` apps

- Load up `lesson1_under_the_hood/lesson1.R` in Rstudio
- These instructive examples come with the `Shiny` package
- Check out https://shiny.rstudio.com/articles/basics.html for their guide to the code

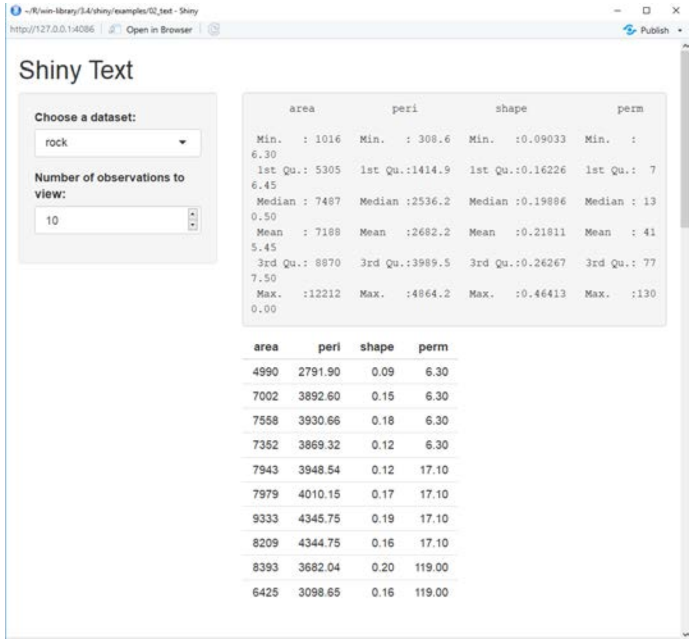


Hello Shiny!

Number of bins: 30

Histogram of waiting times

Waiting time to next eruption (in mins)



Shiny Text

Choose a dataset: rock

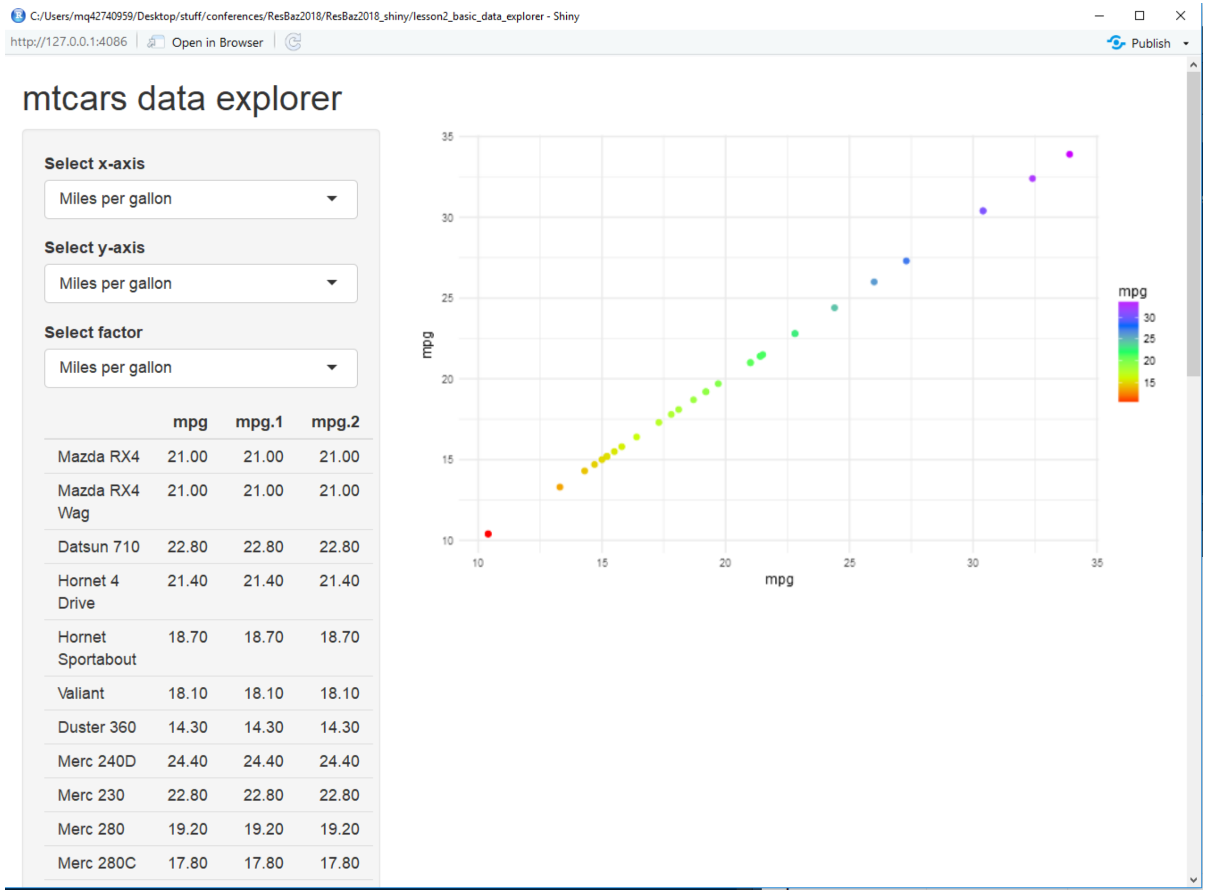
Number of observations to view: 10

```
      area            peri            shape               perm
 
 Min.   : 1016   Min.   : 308.6   Min.   :0.09033   Min.   :
6.30
 1st Qu.: 5305   1st Qu.:1414.9   1st Qu.:0.16226   1st Qu.:  7
6.45
 Median : 7487   Median :2536.2   Median :0.19886   Median : 13
0.50
 Mean   : 7188   Mean   :2682.2   Mean   :0.21811   Mean   : 41
5.45
 3rd Qu.: 8870   3rd Qu.:3989.5   3rd Qu.:0.26267   3rd Qu.: 77
7.50
 Max.   :12212   Max.   :4864.2   Max.   :0.46413   Max.   :130
0.00
```

area	peri	shape	perm
4990	2791.90	0.09	6.30
7002	3892.60	0.15	6.30
7558	3930.66	0.18	6.30
7352	3869.32	0.12	6.30
7943	3948.54	0.12	17.10
7979	4010.15	0.17	17.10
9333	4345.75	0.19	17.10
8209	4344.75	0.16	17.10
8393	3682.04	0.20	119.00
6425	3098.65	0.16	119.00

Test 2 – build a data explorer app

- Load up `lesson2_basic_data_explorer/lesson2_basic_data_explorer_TEMPLATE.R` in Rstudio
- We’ll build our first app!
- Exercise: add functionality to select a variable to colour points or change point shapes


C:/Users/mq42740959/Desktop/stuff/conferences/ResBaz2018/ResBaz2018_shiny/lesson2_basic_data_explorer - Shiny
http://127.0.0.1:4086
Open in Browser
Publish
35
30
25
20
15
10
mpg
10
15
20
25
30
35
mpg
mpg
30
25
20
15

mtcars data explorer

Select x-axis

Miles per gallon

Select y-axis

Miles per gallon

Select factor

Miles per gallon

	mpg	mpg.1	mpg.2
Mazda RX4	21.00	21.00	21.00
Mazda RX4 Wag	21.00	21.00	21.00
Datsun 710	22.80	22.80	22.80
Hornet 4 Drive	21.40	21.40	21.40
Hornet Sportabout	18.70	18.70	18.70
Valiant	18.10	18.10	18.10
Duster 360	14.30	14.30	14.30
Merc 240D	24.40	24.40	24.40
Merc 230	22.80	22.80	22.80
Merc 280	19.20	19.20	19.20
Merc 280C	17.80	17.80	17.80

Publishing your app

- What to do with this fancy new app??
- Currently its only available on your local computer.
    - There are ways to share this info across intranets too
- Broader audience:
    - Publish on the shinyapps.io site
- Steps:
    - Set up an account.. www.shinyapps.io
    - Link your IDE with the shiny account:
        - install.packages(‘rsconnect’)
        - Library(rsconnect)
        - Go to your shiny.io account, find your token, click show.. Then show secret, copy the text and run in Rstudio: rsconnect::setAccountInfo(name="<ACCOUNT>", token="<TOKEN>", secret="<SECRET>")
    - Check to see if the publishing works.
    - Set publishing options in Rstudio – tools- global options-publishing
- Ready to publish?
    - Set working directory to the directory of your app
    - Run deployApp() / select publish icon

Tips_1

- Keep install commands outside of the app
- Start very small and simple.. Then use examples to get more complicated.
- Reactivity terms and reactivity functions work in paris to give you the desired output
- Whatever is in the curly brackets of the reactive term, will be used to update that feature.
- When you have separate reactive functions looking at the same reactive dataset (ie a plot and a summary table) you should make the dataset a standalone variable, that the reactive functions can both call on: data<-reactive({}). Then simply call data() – don't forget the brackets – in your reactive function.
- Use isolate({}) to isolate the updating of certain items.
- Use a reactive action button to delay updating until all inputs are ready: EventReactive() – creates reactive expression to be paired with action button.
- Try and keep code within the server section to a bare minimum, these sections are repeated multiple times and will slow your processing speeds.
- Use HTML tags to make your apps prettier – tags$h1() = header, tags$p() = paragraph etc.
    - Can use the HTML functions to insert images etc. these must then be saved in your app's directory.

Tips_2

- You can determine how you want your app to look in the x y and even z dimensions. Layering is allowed.
    - FluidRow() – divides app into rows, always a max of 12 units wide
    - Column() – can be used with fluid row to create columns
    - wellPanel() – puts things into nice looking grey “wells”
    - Tabpanels() – make new tabs and can be layered.
    - Sidebarlayout() probably the most common.
- You can use a fixed page if you don’t want the page to auto-resize, which is what fluidpage does. You must also use fixedrow if you choose this route.
- There’s a lot of great info about dashboards!
- There are ways to check the errors of your app and to get info about how your app is responding to users - rsconnect::showLogs()