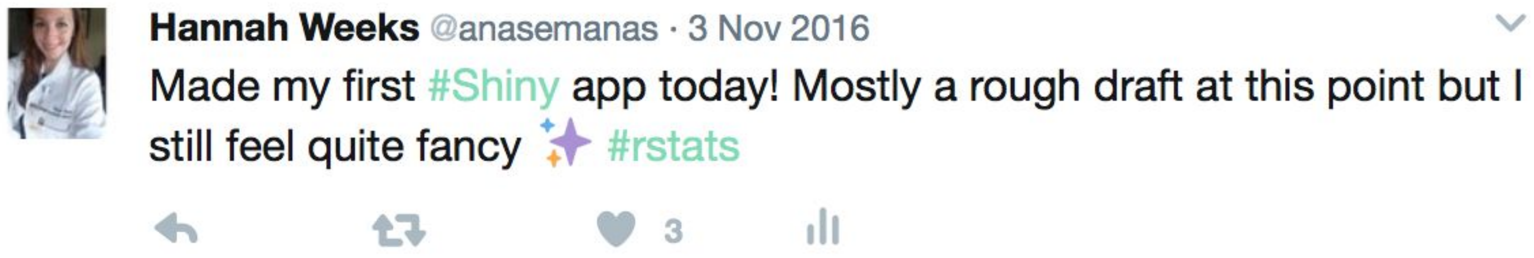
R-Ladies Introduction to Shiny

Hannah Weeks
hannah.weeks@vanderbilt.edu

https://github.com/hlweeks/shinydemo



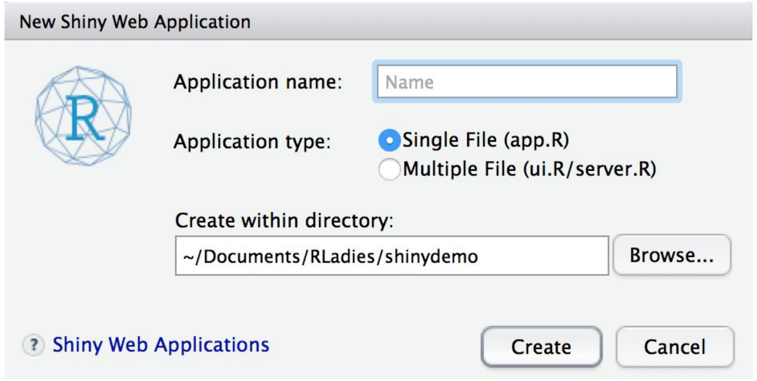
Hannah Weeks @anasemanas · 3 Nov 2016

Made my first #Shiny app today! Mostly a rough draft at this point but I still feel quite fancy ✨ #rstats

3

Create application file(s)

- RStudio: File -> New -> Shiny Web App



New Shiny Web Application

Application name: Name

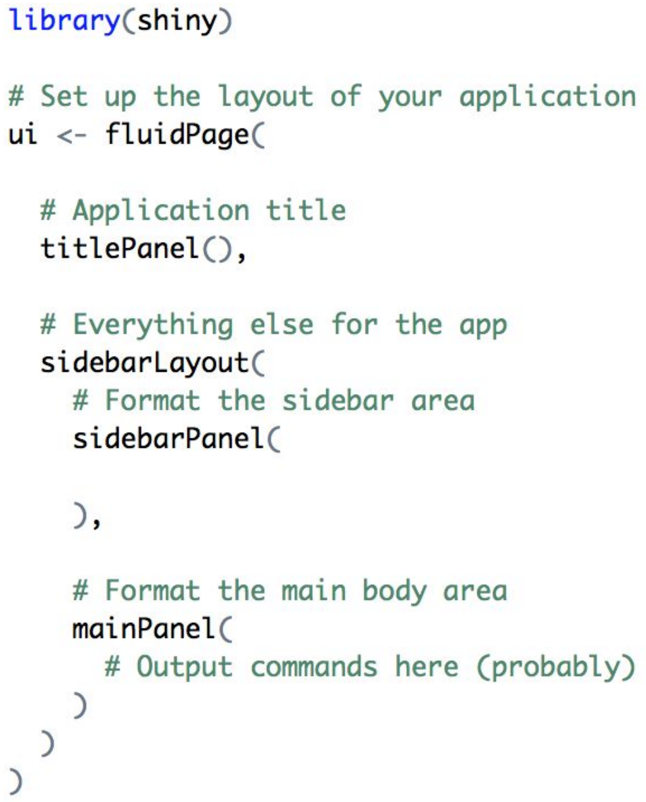
Application type: [✓ Single File (app.R)] Multiple File (ui.R/server.R)

Create within directory: ~/Documents/RLadies/shinydemo

- R: Just create R script(s) and start with `library(shiny)`

- Various ways to create and launch apps: https://shiny.rstudio.com/articles/app-formats.html

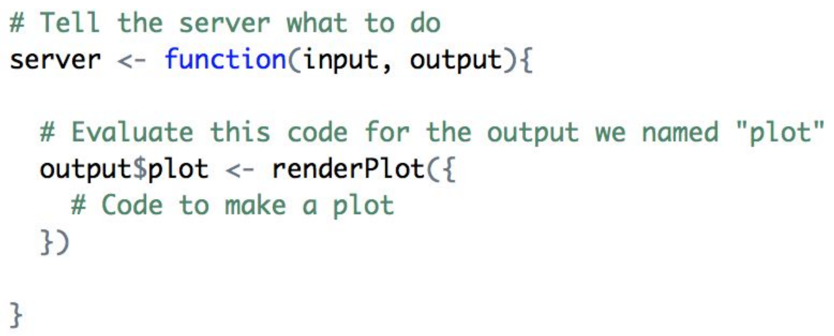
ui (user interface)



```
library(shiny)

# Set up the layout of your application
ui <- fluidPage(

  # Application title
  titlePanel(),

  # Everything else for the app
  sidebarLayout(
    # Format the sidebar area
    sidebarPanel(

    ),

    # Format the main body area
    mainPanel(
      # Output commands here (probably)
    )
  )
)
```

- Set up the application layout
- Define input and output commands
- Layout/appearance formatting goes here

server

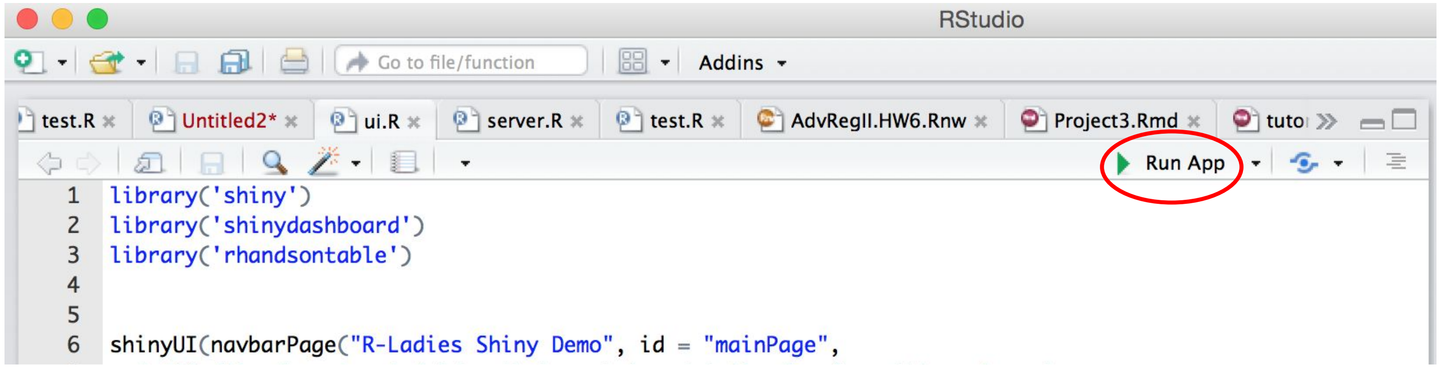


```
# Tell the server what to do
server <- function(input, output){

  # Evaluate this code for the output we named "plot"
  output$plot <- renderPlot({
    # Code to make a plot
  })

}
```

- Define outputs using inputs
- Wrap desired output with corresponding `render` function, e.g...
  - `renderPlot`
  - `renderTable`
  - `renderText`
- Usual R code goes here

Run the app

- In RStudio:



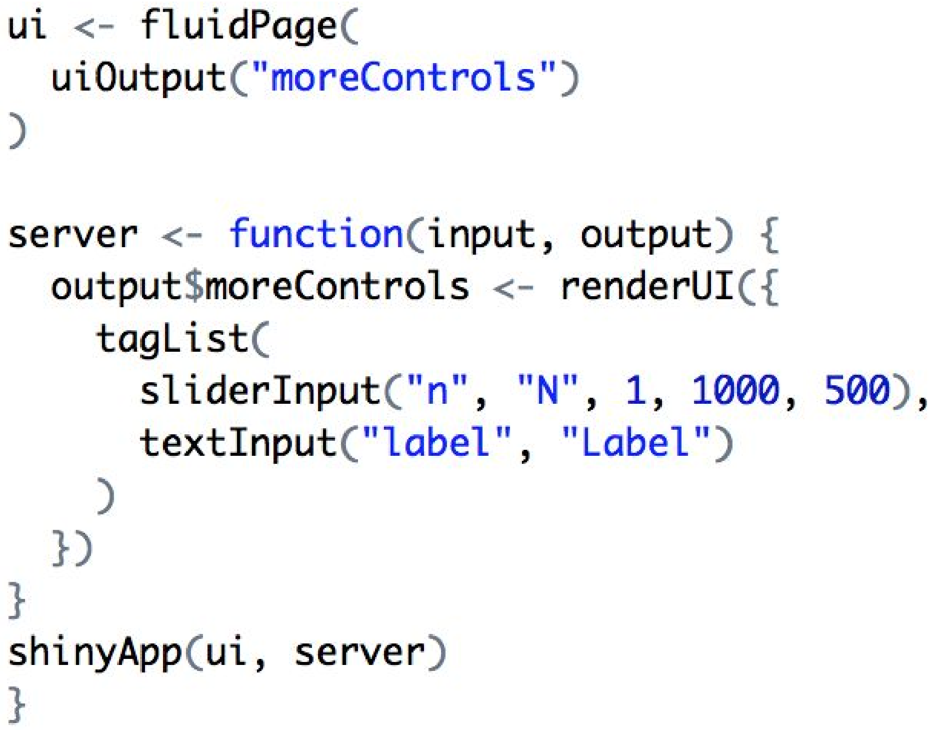
```
library('shiny')
library('shinydashboard')
library('rhandsontable')


shinyUI(navbarPage("R-Ladies Shiny Demo", id = "mainPage",
```

- In R:
    - `runApp()` when working directory is the app directory, or `runApp(“path/to/app/dir”)`
    - `shinyApp(ui = ui, server = server)`, if `ui` and `server` objects have been defined

Basic Shiny functions

- Inputs
  - General form: functionInput(input_ID, display name of input, …)
  - E.g. `sliderInput("num", "Number of Doses", min = 0, value = 5, step = 1)`
  - E.g. `checkboxInput("common", "Common Dosing Pattern")`
- Outputs
  - General form: typeOutput(output_ID, …)
  - E.g. `plotOutput("plot", hover = "plot_hover")`
  - E.g. `verbatimTextOutput("info")`

Inputs in the server function



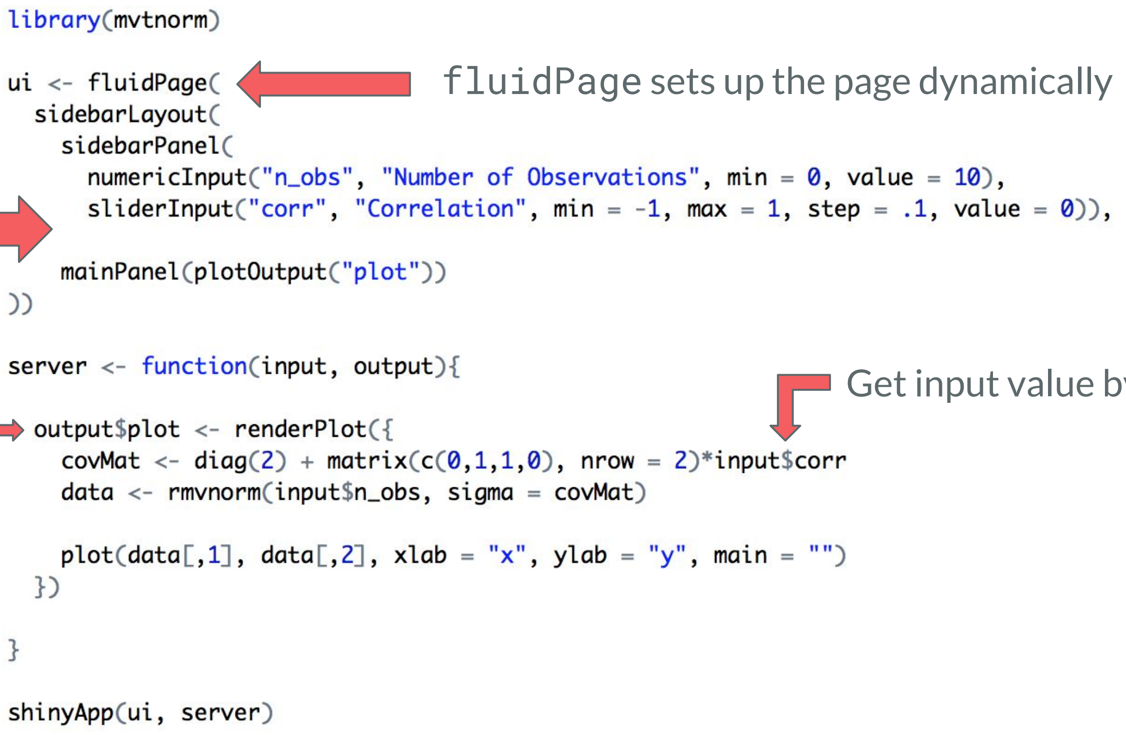
```
ui <- fluidPage(
  uiOutput("moreControls")
)

server <- function(input, output) {
  output$moreControls <- renderUI({
    tagList(
      sliderInput("n", "N", 1, 1000, 500),
      textInput("label", "Label")
    )
  })
}
shinyApp(ui, server)
}
```

- Place `uiOuput` in the ui object

- Use `renderUI` in the server function

- Example from: https://shiny.rstudio.com/reference/shiny/latest/renderUI.html

How to use inputs/outputs



```
library(mvtnorm)

ui <- fluidPage(
  sidebarLayout(
    sidebarPanel(
      numericInput("n_obs", "Number of Observations", min = 0, value = 10),
      sliderInput("corr", "Correlation", min = -1, max = 1, step = .1, value = 0)),

    mainPanel(plotOutput("plot"))
))

server <- function(input, output){

  output$plot <- renderPlot({
    covMat <- diag(2) + matrix(c(0,1,1,0), nrow = 2)*input$corr
    data <- rmvnorm(input$n_obs, sigma = covMat)

    plot(data[,1], data[,2], xlab = "x", ylab = "y", main = "")
  })

}

shinyApp(ui, server)
```

`fluidPage` sets up the page dynamically

Give IDs to inputs/output

Get input value by referencing ID

Define output by ID

Action inputs

- `actionButton` or `actionLink`

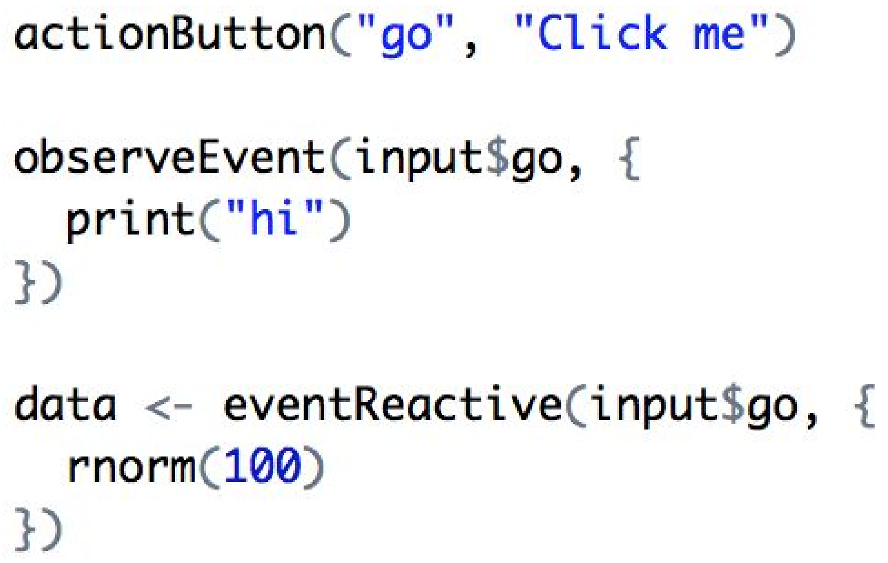
- `actionButton(“button_id”, “Click here”)`

- input$action_id = 0 when it hasn’t been clicked yet
  - Value increments by 1 every time it is pressed/clicked

```
output$plot <- renderPlot({
if(input$go == 0){
   plot(1:10, xlab = “x variable”, ylab = “y variable”, main = “”)
}
```

Event-reactive functions

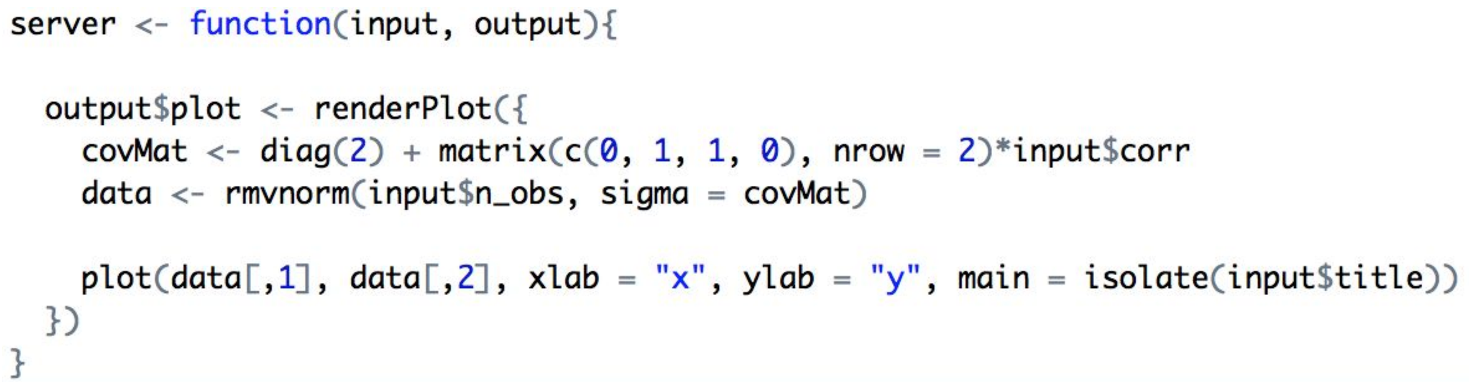
- Typically respond to action buttons or action links
- `observeEvent` - perform some action in response to an event
- `eventReactive` - calculate some value in response to an event



```r
actionButton("go", "Click me")

observeEvent(input$go, {
  print("hi")
})

data <- eventReactive(input$go, {
  rnorm(100)
})
```

Prevent reactivity

- Use the `isolate` function



```r
server <- function(input, output){

  output$plot <- renderPlot({
    covMat <- diag(2) + matrix(c(0, 1, 1, 0), nrow = 2)*input$corr
    data <- rmvnorm(input$n_obs, sigma = covMat)

    plot(data[,1], data[,2], xlab = "x", ylab = "y", main = isolate(input$title))
  })
}
```

Multi-tab apps

- `navbarPage`: use instead of fluidPage to set up app with top-side tabs
- `tabPanel`: lay out the code within each tab
- `navlistPanel`: creates a sidebar of tabs
- `tabsetPanel`: create a series of panels separated by tabs (can be embedded within a single-page app)
- `navbarMenu`: create multiple tabs in a drop-down menu

- `ui` should be structured in the order tabs are presented
- `server` does not need to be structured in a specific way

HTML/CSS in Shiny

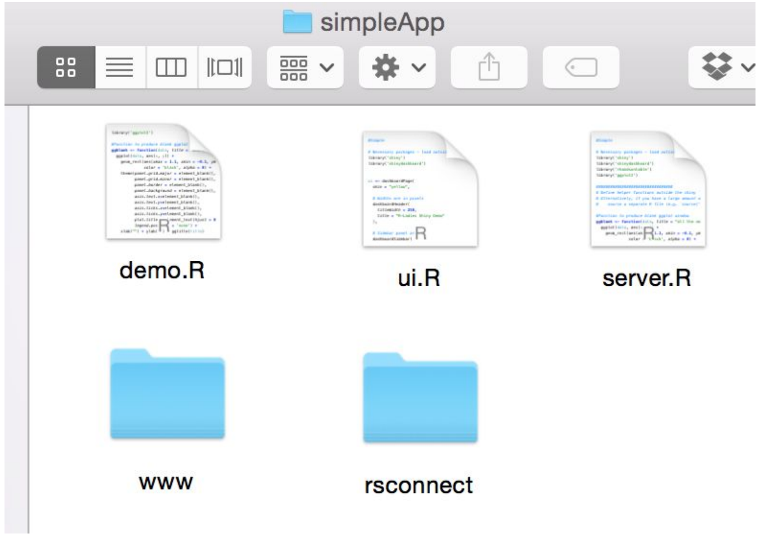
- Can be use to customize appearance

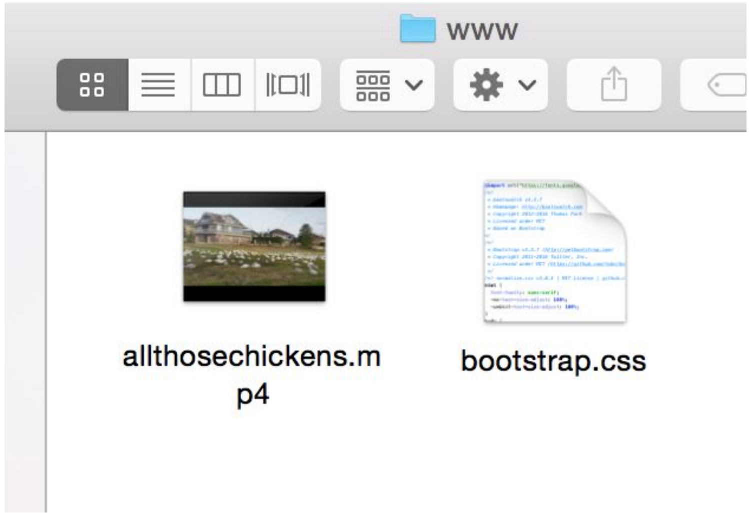
- HTML function
  - `HTML("<div style='height: 150px;'>")`
- CSS
  - `tags$style(type="text/css", ".shiny-output-error { visibility: hidden; }")`

- https://shiny.rstudio.com/articles/tag-glossary.html
- https://www.codecademy.com/learn/learn-html-css

Media

- In the directory where app files are located, add folder called ‘www’
- Place images, videos, .css files here



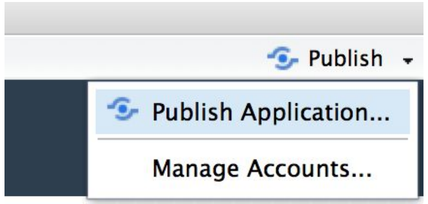


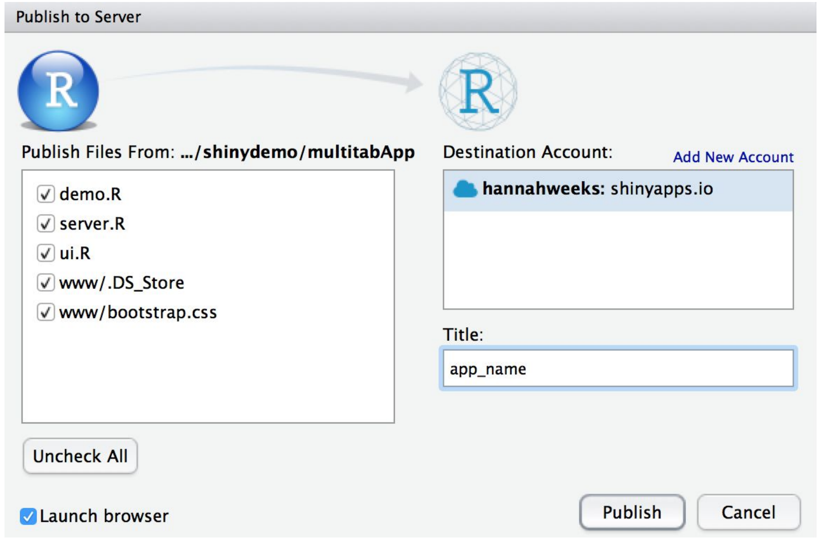
Debugging in Shiny

- Version control
- Run the app often
- Test extensive R code on actual data first (not `input$things`)
- The internet is your friend

Hosting your app online

- https://www.shinyapps.io
- 5 applications and 25 active hours for free
- Publish (deploy) apps with `rsconnect` package or through RStudio





Publish Files From: .../shinydemo/multitabApp

- ✓ demo.R
- ✓ server.R
- ✓ ui.R
- ✓ www/.DS_Store
- ✓ www/bootstrap.css

Destination Account: **hannahweeks:** shinyapps.io

Title: app_name

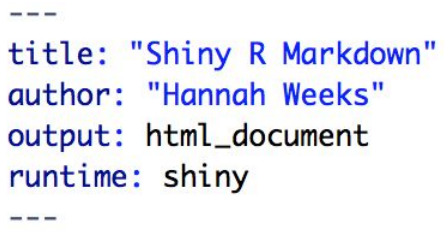
✓ Launch browser

rhandsontable package

- https://jrowen.github.io/rhandsontable/
- Interactive tables
- Important functions:
    - `renderRHandsontable`
    - `rhandsontable`: convert from an R object to an rhandsontable object
    - `hot_to_r`: convert to an R object for manipulation

Shiny with R Markdown

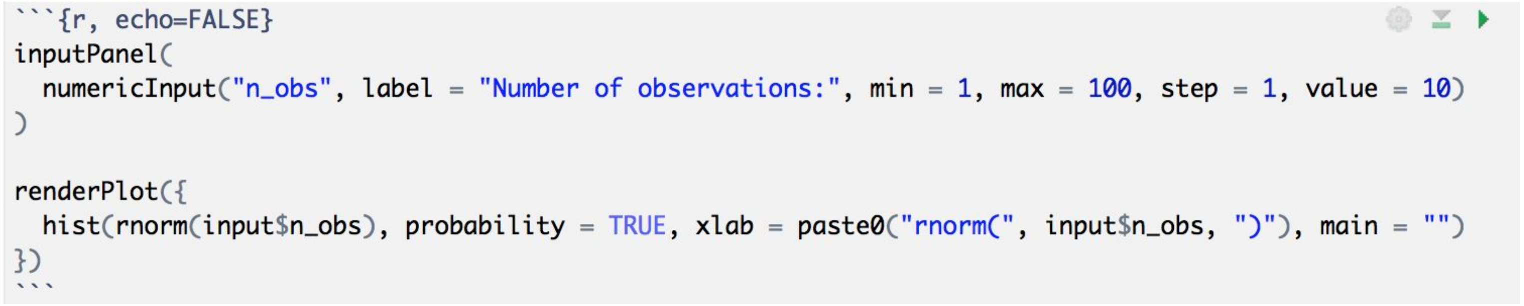
- Add `runtime: shiny` to header for R Markdown HTML document



```
---
title: "Shiny R Markdown"
author: "Hannah Weeks"
output: html_document
runtime: shiny
---
```

- Click "Run Document" (RStudio) or type `rmarkdown::run("path/to/file/filename.Rmd")`

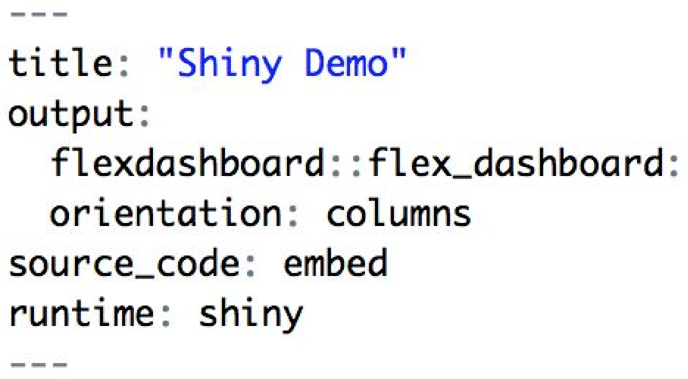
- Syntax is slightly different (no `ui`/`server` objects)



~~~
```{r, echo=FALSE}
inputPanel(
  numericInput("n_obs", label = "Number of observations:", min = 1, max = 100, step = 1, value = 10)
)

renderPlot({
  hist(rnorm(input$n_obs), probability = TRUE, xlab = paste0("rnorm(", input$n_obs, ")"), main = "")
})
```
~~~

Shiny with R Markdown

- Flexdashboard: http://rmarkdown.rstudio.com/flexdashboard/
  - Creates dashboards using R Markdown, add `runtime: shiny` to header
  - E.g. https://jjallaire.shinyapps.io/shiny-kmeans/
  - Super easy for giving source code



```
---
title: "Shiny Demo"
output:
  flexdashboard::flex_dashboard:
  orientation: columns
source_code: embed
runtime: shiny
---
```

Helpful websites

- https://shiny.rstudio.com

- https://www.shinyapps.io
- https://rstudio.github.io/shinydashboard/
- http://rmarkdown.rstudio.com/flexdashboard/

- Link to slides: https://github.com/hlweeks/shinydemo