

Using Shiny in a Modern Workflow for Statistics

Carter Allen
February 4, 2019

Outline

- **What?**
- **Why?**
- **How?**
- **Example 1:** *Mapping County Parks in Miami*
- **Example 2:** *Bayesian test for skewness*
- **Live coding (lol): Beta-Binomial app**



Materials

github.com/carter-allen/brownbag

What?

What is a Shiny app?

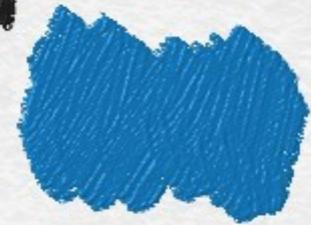
- A web application constructed using the `Shiny` package developed and maintained by RStudio
- Provides a flexible framework allowing R code to exist and run on the web
- Can run locally, on a server, or for free on the RStudio cloud
- Whatever R can do locally, it can also do on the web with Shiny!

Fun Examples

- **Papr**: The “Tinder” of academic papers
- **CDC Disease Monitor**: Display weekly case counts of various diseases from CDC data
- **Genome Browser**: Visualize genomes using Circos diagrams

Why?

Quick



Shiny

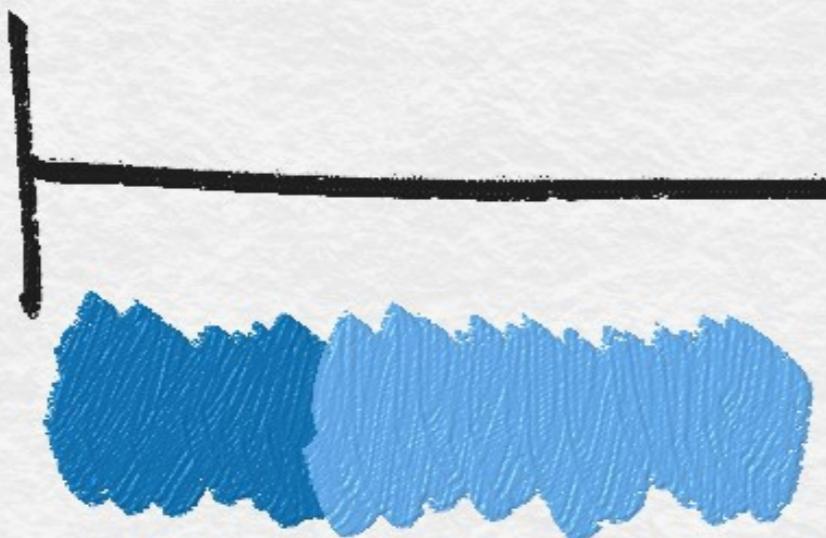
Not
worth it

Robust



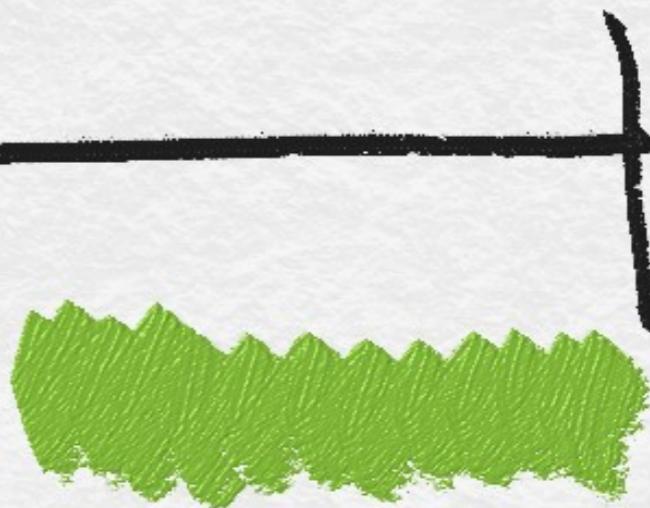
JavaScript

Quick



Shiny ↑
Shinier!

Robust

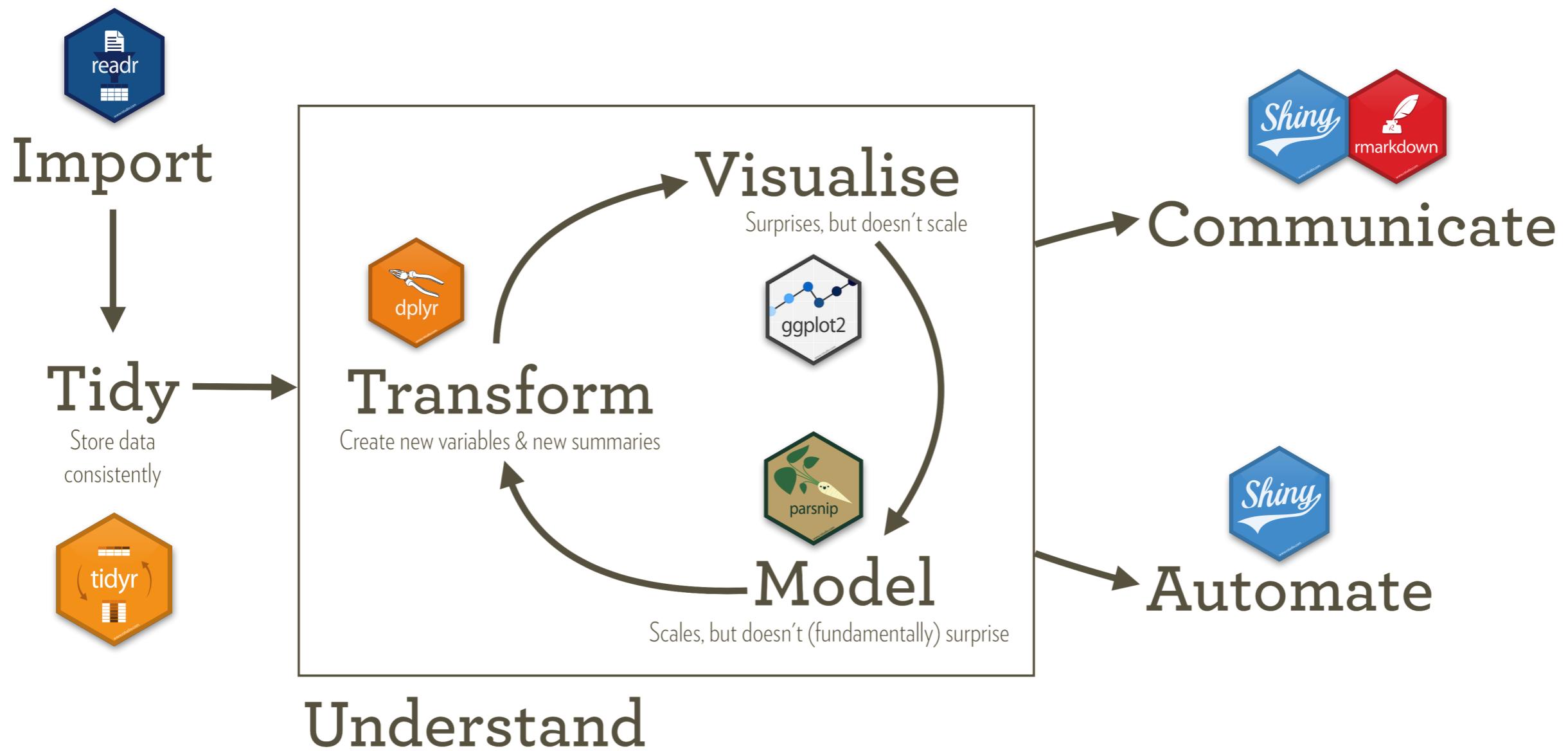


JavaScript

Why use a Shiny app?

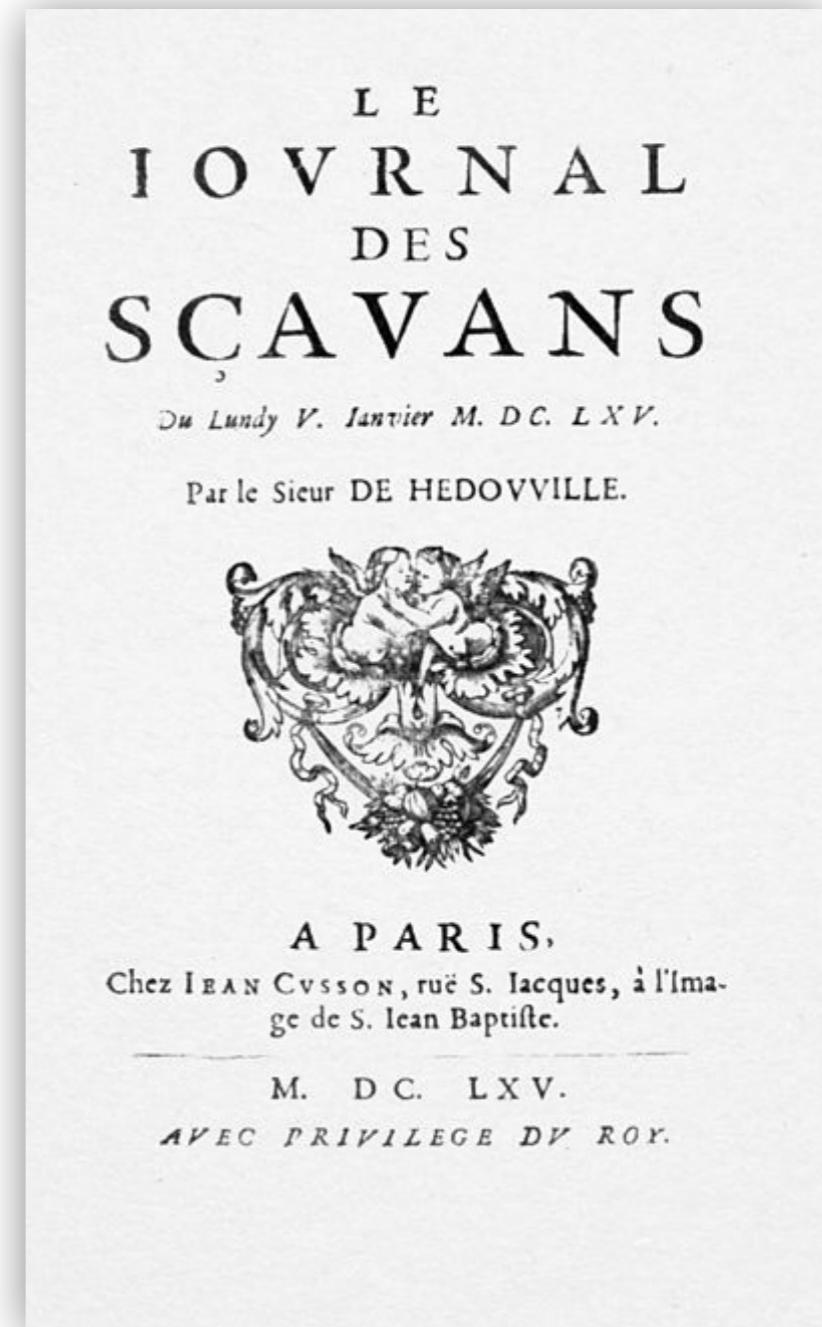
- A statistical analysis can (should) be thought of as a *system* that **must be able to iterate more than once** (reproducibility).
- Some components of the system are “high-touch” and some are more automated.
- For the entire system to work, we must dedicate thought and effort to **every** component, not just statistical methods.
- The statistician designs, constructs, and maintains the system then shares it with others.

A Statistical System



Shiny for Communication

- Academic communication is done primarily through the medium of the journal.
- Modern data analysis is often performed through the medium of highly customized software (not highly compatible with the scientific journal).
- **Our tools for communication should reflect the different media in which we work.**



First known academic journal in Europe:
Journal des Scavans (began publishing
in 1665)

Shiny for Education

- Core statistics concepts often involve lots of abstract visual imagination.
 - Conditional probability/Bayes Theorem
 - Sampling distributions/CLT
- Shiny apps can serve as interactive visual teaching aids.
 - Penn State's "Book of Apps for Teaching Statistics"

How?

Three Core Functions

```
ui <- shiny::fluidPage(...)
```



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



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

Three Core Functions

- The `ui`, `server`, and `shinyApp` functions can all be called from one single .R file, usually named `app.R`.
- Alternatively, the `ui` and `server` functions can exist in their own .R files.
- From the console, use `runApp("app.R")` to run the app. In RStudio, there is a GUI for this.
- All R packages used by the app should be loaded at the top of the `app.R` file.

User Interface

- The **user interface** allows for *transfer of data between the user and the “back-end” machinery of the application*. It is the “cell membrane” of the application.
- User communicates with the app to provide things like:
 - Input data, simulation parameters, search queries, etc...
- App communicates with the user to provide things like:
 - Output data, simulation results, search results, etc...
- **All interactions between the user and app take place through the UI**

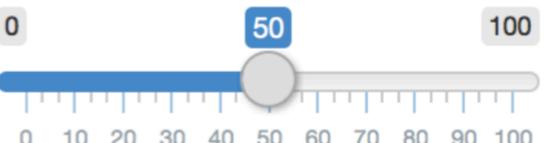
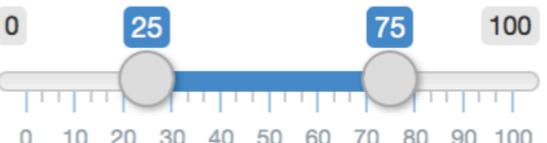
User Interface

- With `shiny`, the user interface is an R object (conventionally named `ui`) defined with several possible functions.
 - **UI Layout functions:** designate where elements live on the webpage (title page, free text, input/output).
 - **UI Input functions:** construct simple input widgets such as slider bars, text entry, file upload etc...
 - **UI Output functions:** control the application's output such as plots, text, file downloads etc...

Control widgets

http://127.0.0.1:3771 | [Open in Browser](#) | [G](#) [Publish](#) ▾

Basic widgets

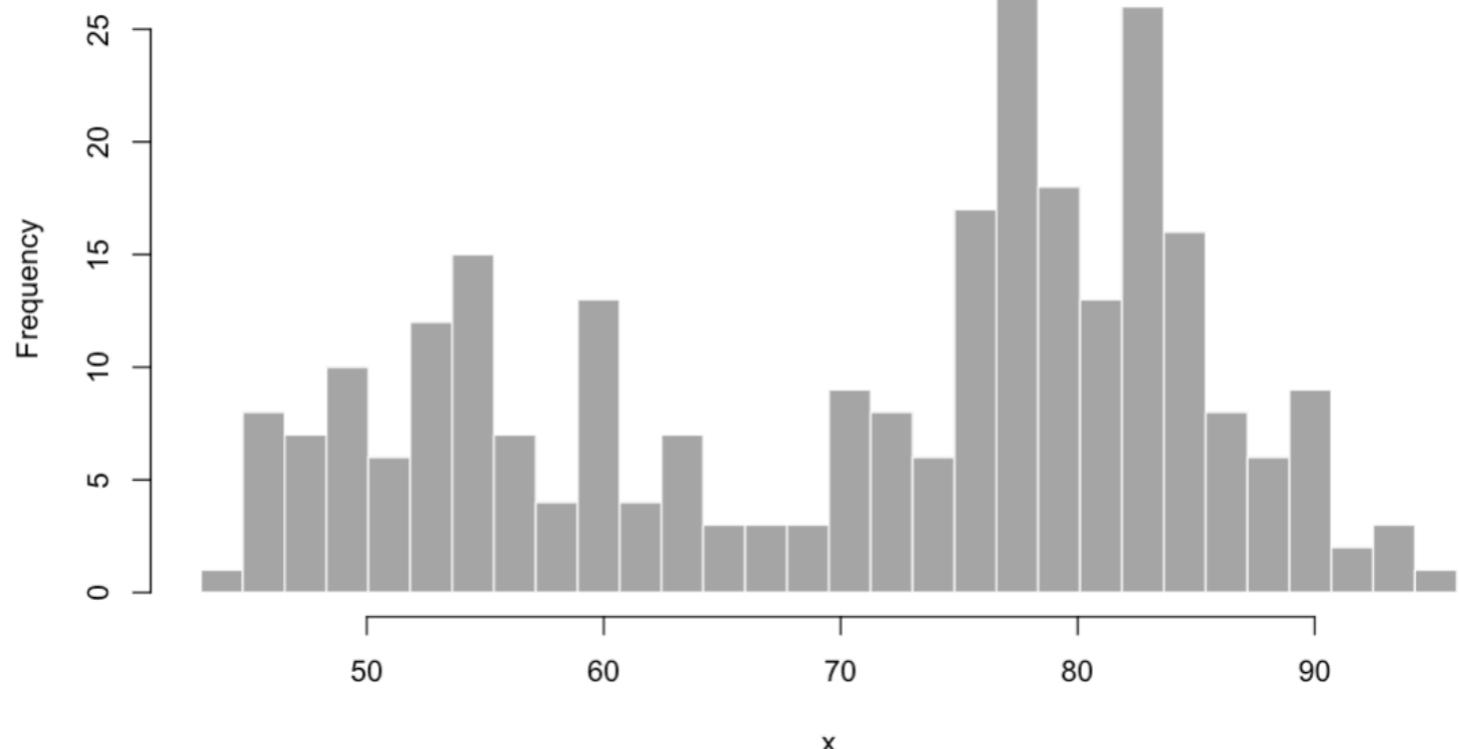
Buttons Action Submit	Single checkbox <input checked="" type="checkbox"/> Choice A	Checkbox group <input checked="" type="checkbox"/> Choice 1 <input type="checkbox"/> Choice 2 <input type="checkbox"/> Choice 3	Date input 2014-01-01
Date range 2017-06-21 to 2017-06-21	File input Browse... No file selected	Help text Note: help text isn't a true widget, but it provides an easy way to add text to accompany other widgets.	Numeric input 1
Radio buttons <input checked="" type="radio"/> Choice 1 <input type="radio"/> Choice 2 <input type="radio"/> Choice 3	Select box Choice 1 ▾	Sliders  	Text input Enter text...

Old Faithful Geyser Data

Number of bins:

1 6 11 16 21 26 30 31 36 41 46 50

Histogram of x





```
# Define UI for application that draws a histogram
ui <- fluidPage(

  # Application title
  titlePanel("Old Faithful Geyser Data"),

  # Sidebar with a slider input for number of bins
  sidebarLayout(
    sidebarPanel(
      sliderInput("bins",
                  "Number of bins:",
                  min = 1,
                  max = 50,
                  value = 30)
    ),
    mainPanel(
      plotOutput("distPlot")
    )
  )
)
```

Server

- In a shiny app, **server** is a function defined with two parameters:
 - **input**: Used to access all parameters defined in the UI.
 - Ex: `input$bins` will retrieve the current value of chosen number of histogram bins
 - **output**: Used to update the displayed objects defined in the UI layout



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

  output$distPlot <- renderPlot({
    # generate bins based on input$bins from ui.R
    x      <- faithful[, 2]
    bins   <- seq(min(x), max(x), length.out = input$bins + 1)

    # draw the histogram with the specified number of bins
    hist(x, breaks = bins, col = 'darkgray', border = 'white')
  })
}
```

Server

- Tendency is for backend to become cluttered and inefficient since not seen by the user.
- For complex apps, avoid this issue by making an R package.



Tips for good server functions

1. If code does not need to be **reactive** to user input, isolate it outside the server (ex: reading data)
2. Write an R package for any custom functions you use. Shiny will only need to load the package once (rather than re-allocating memory for the function after each refresh).
3. Use delayed evaluation when possible to avoid unnecessary computation (ex: ggplots)

General Advice

- Shiny apps work very well with R Projects and R Packages. These tools take effort to learn but will ease the organizational burden.
- If not using R Projects & R Packages, put apps in their own directory on your computer and define any custom functions in a separate file (called `helper.R` or something).
 - Source this file at the top of the app with `source("helper.R")` to be able to use the custom functions.

Advanced Topics

Reactive Programming

- Reactive expressions are those that fetch data from the user interface and cache their values.
- Much of the computational burden in a Shiny app comes with transferring data between the UI and the served side.
- By using the `reactive({})` function, you can **control when the app fetches data from the UI**.
- **Shiny will even know to skip fetching parameters which are already up to date.**

Dependencies

- Reactive expressions are like links in a chain connecting `input` values to `output` objects.
- Using a reactive programming will eliminate issues of dependency (ex. A plot is updated but the data it relies on is not).
- Shiny will update an output object if:
 1. An `input` value in the object's `render*` function changes
 2. A reactive expression in the object's `render*` function becomes obsolete

Testing apps

- Once your Shiny app is running well locally, the next step is to test it before going live.
- `shinytest` is an R package to help automate the testing process.
- See RStudio's [vignette](#) for `shinytest`

Deploying Apps

- RStudio provides a range of hosting options for Shiny apps ranging from free and easy to use solutions to more powerful solutions that require a subscription fee and technical skill.
- <https://www.shinyapps.io/> is the best solution for most users

Helpful Links

A compilation of links and materials used here
can be found at [github.com/carter-allen/
brownbag](https://github.com/carter-allen/brownbag)

Examples