AN EXPLORATION OF THE CAPABILITIES IN *shiny* FOR DATA VISUALIZATION

WHAT IS SHINY?

- An R package that allows the creation of interactive web apps
- Consists of User Interface (UI) functions that include HTML, CSS and JavaScript
- Uses reactive programming (v. functional) to specify a network of dependencies

OUTLINE

In this presentation, we will learn about:

(1) The basic components to a Shiny script (UI v. Server)

(2) Creating a reactive graph

(3) Creating and launching a magnitude estimation study

```
library(shiny)

ui <- fluidPage(
  # user interface
)

server <- function(input, output, session) {
  # behind the scenes
}

shinyApp(ui, server)
```

USER INTERFACE (UI)

- Allows HTML inputs and outputs (e.g., sliderInput and plotOutput)
- All input functions have 'inputId' and 'label' as their first and second arguments, respectively
- Likewise, output functions have 'inputId' as their first argument for objects specified in the server (e.g., output$plot)

SERVER

- server() is called for every new session
- Takes in three arguments: input, output and session

INPUT

- A read-only object that stores information from UI inputId
- Need to use a reactive function to access its content

SERVER

OUTPUT

- Always used in conjunction with a render function
- Used to send output into the UI

```
ui <- fluidPage(plotOutput("plot"))

server <- function(input, output, session) {
output$plot <- renderPlot(...)}
```

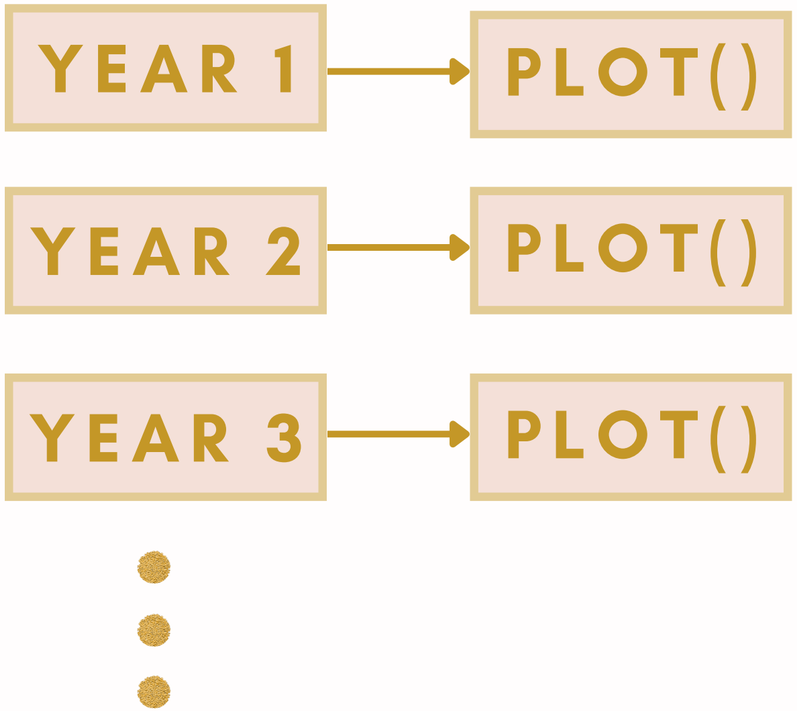
SERVER

REACTIVITY

- Also known as declarative programming (vs. imperative), this is the key element that makes Shiny unique, and, well... shiny!
- Imperative language is assertive and read top-down whereas declarative is passive-aggressive (and lacks order)
- Another way to think about it: you give R scripts commands, whereas you give Shiny recipes

SERVER

REACTIVITY

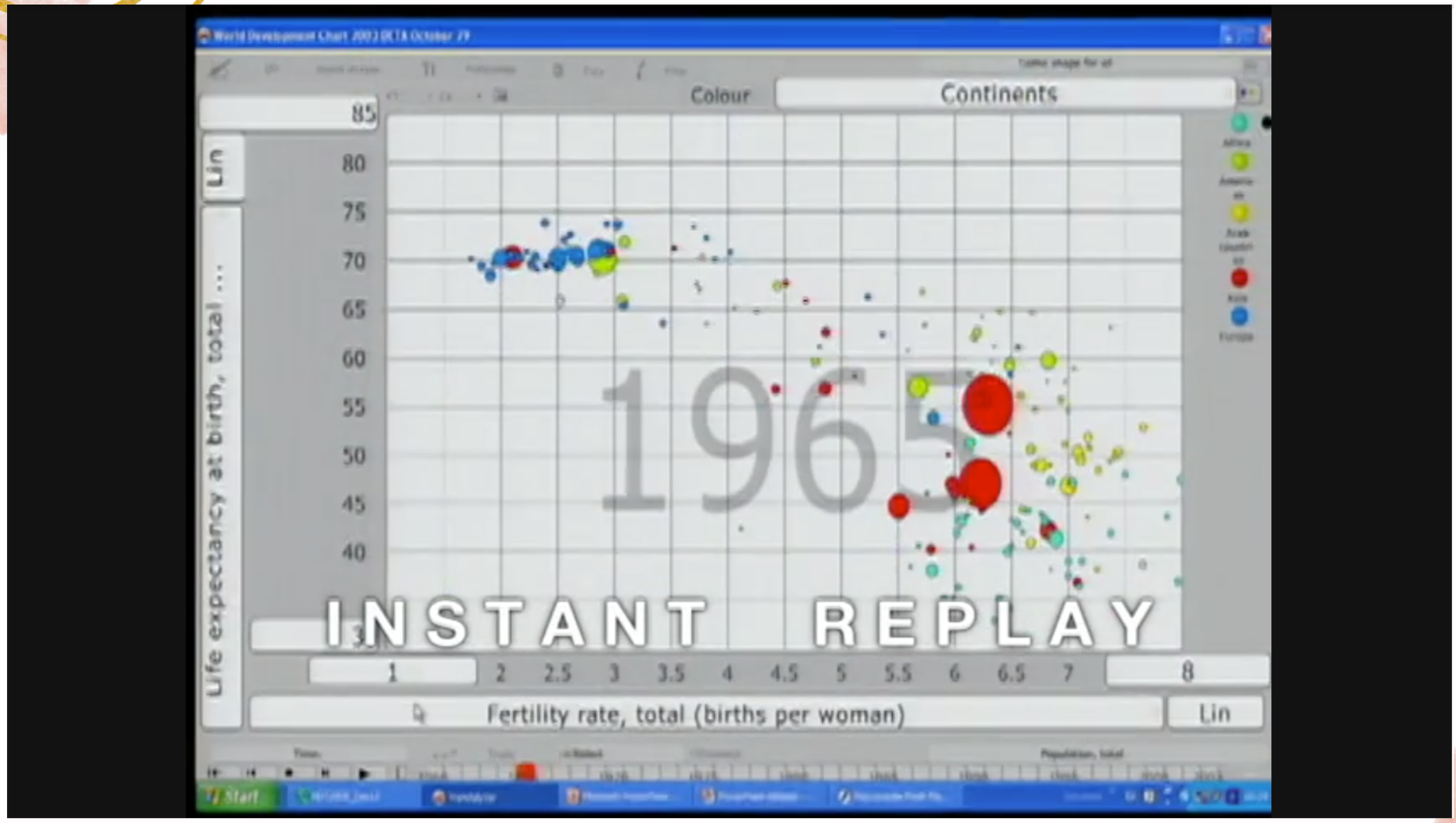


V.

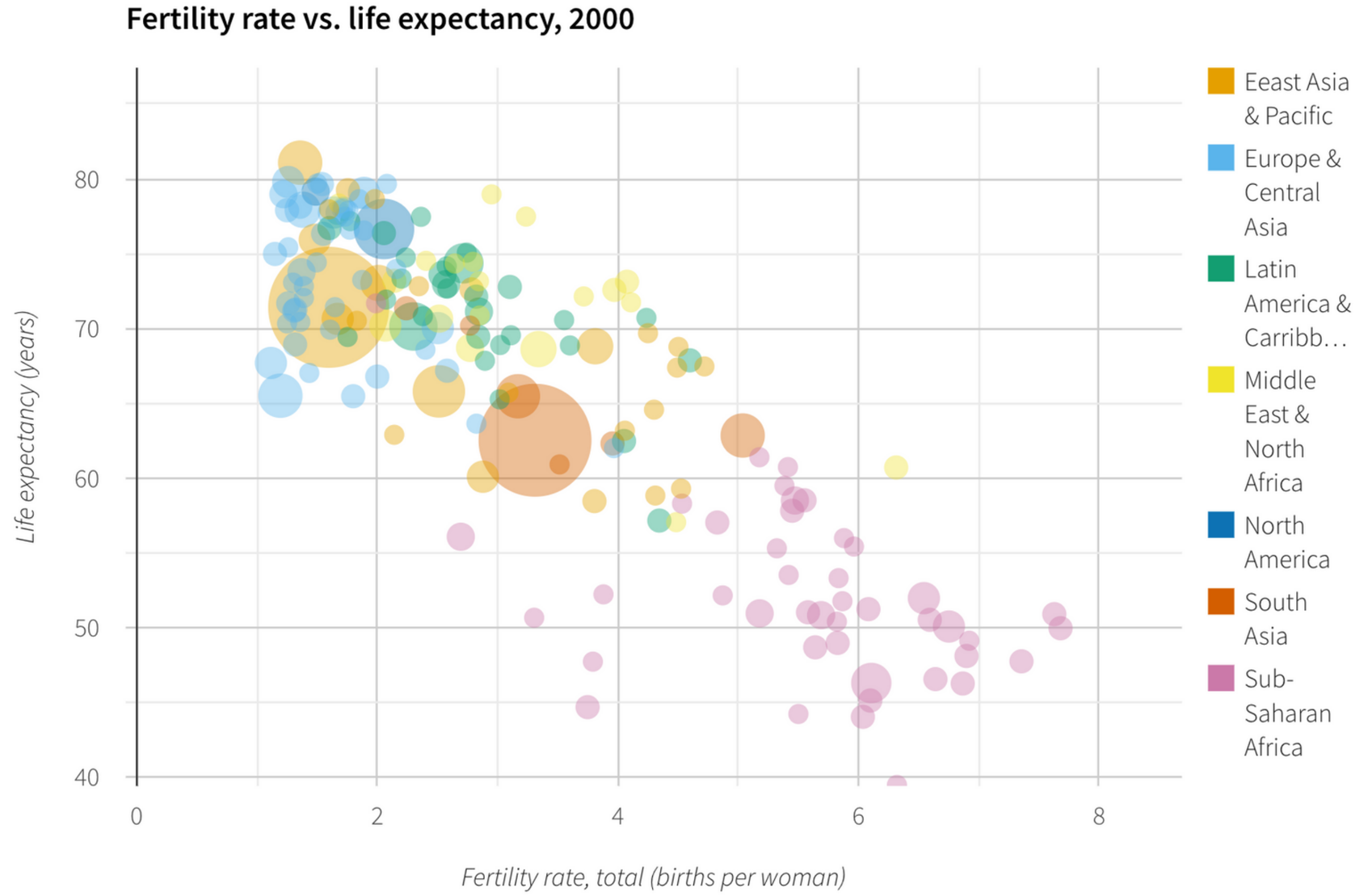
THE REACTIVE PLOT

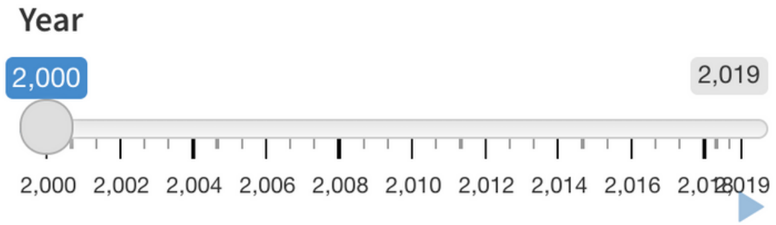
- You can load other R packages in your Shiny script: e.g., ggplot2, googleCharts,
- plotOutput() is a reactive functions that not only outputs your graph but can respond to four pointer events: click, dblclick, hover and brush

REMEMBER THIS TED TALK?

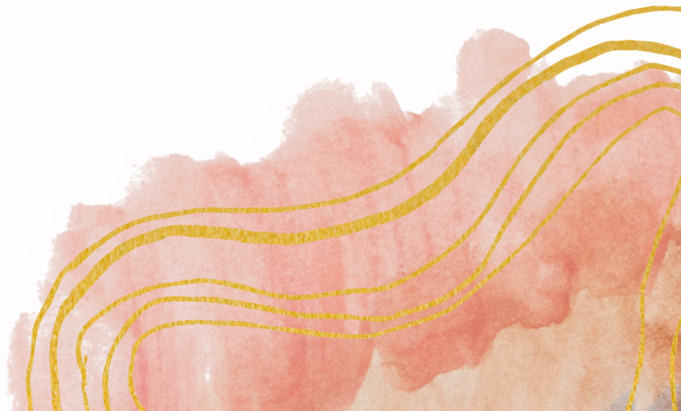


LET'S RE-CREATE THIS PLOT IN SHINY





Data (2000-2019) and Shiny script cant be found in my GH repository

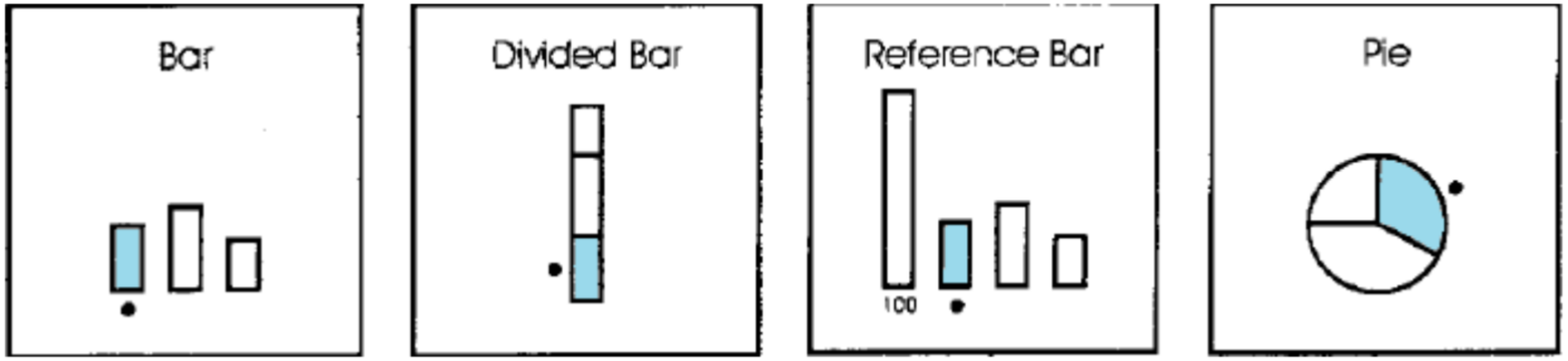


MAGNITUDE ESTIMATION

- Using what we've learned about reactive plots, we can create stimuli that assess perceived magnitude
- That is, we can create a magnitude estimation study

MAGNITUDE ESTIMATION

- Recall experiments by Hollands & Spence (1992, 1998)



MAGNITUDE ESTIMATION

Shiny brings forth advantages:

- Random (in regards to trial order and data generation for plots)
- More parameters to manipulate generates new research questions (e.g., can allow the participant to manipulate the graph)

Let's create a small magnitude estimation study in Shiny to showcase these advantages

MAGNITUDE ESTIMATION

KEY FUNCTIONS

- reactive() allows random trials; i.e., no two users are shown the same graphs in the same order
    - NOTE: dataframes or variables within reactive() are treated like functions e.g., dataframe_name()
- reactiveValues() generates each new perception task

MAGNITUDE ESTIMATION

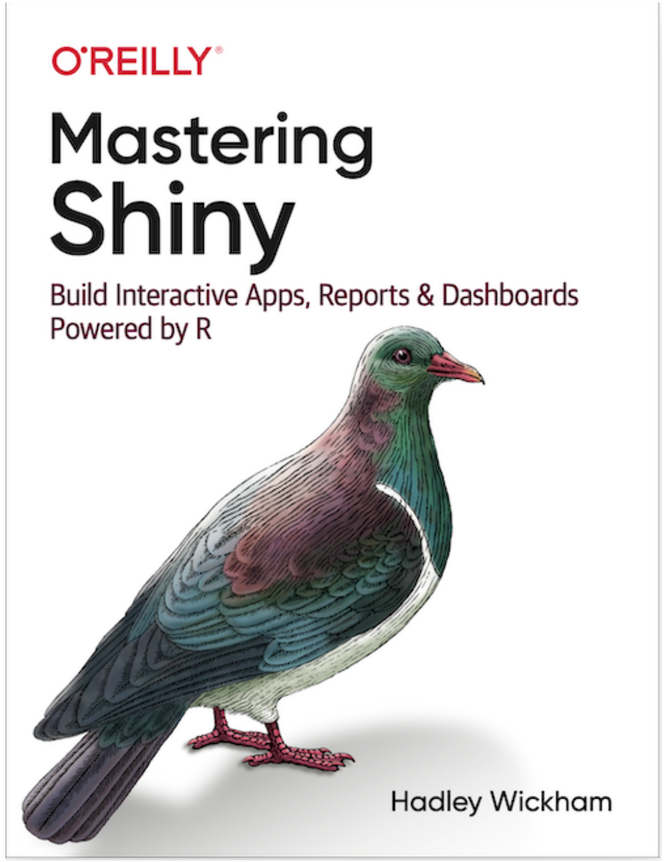
HOSTING

- Shinyapps.io makes your study/app public for free (lim. five apps)
- Alternatives; AWS, SQL

SAVING

- You will need to save participant data remotely; one option is the R package rdrop2 (saves it to your dropbox)

EXTRA



- FREE!
- Includes exercises and goes more in depth into theory
- Unfortunately does not touch on how to save data (you can look at this resource instead)


thank
you