# Building dashboards in R/Shiny

Kimberly Zhang

June 10, 2024

## Presentation Overview

## Why R/Shiny?

- Shiny gives R users the power to build a dashboard without prior knowledge of HTML, CSS, and JavaScript, but retain the ability to use them if needed

## Shiny Basics

- Start with some basic examples from the Shiny gallery like the telephones by region dashboard
- ui.R is your “road map” for every feature in the dashboard
    - Widgets[1,2]
    - Progress bars
- server.R
    - server.R connects user inputs from the widgets set up in the UI to calculations in the server through the inputId argument
    - Caching

## Getting Data

- Read Excel files from your working directory using fread(), read.csv(), read.xlsx()
- Load RData files using load()
- Use `RODBC::odbcDriverConnect()` to get data from SQL

## Visuals

- Charting libraries built on underlying principle of "layering" visualization elements, giving developers tremendous flexibility
  - `ggplot`
  - `plotly` (interactive, also available in Python)

## Tables

- Make pretty tables with the `reactable` package (e.g., 2019 Women's World Cup Predictions)
- For very large tables, use `DT::datatable()` with the option server set to TRUE so that the browser receives only the displayed data.

## Data manipulation

- Aggregation (e.g., operations like summing by desired grouping) using `dplyr` or `data.table`
- melt() and dcast() functions from `reshape2` to transform data between “long” and “short” format
- merge() for joining tables

## reactive() vs observe()

- `reactive()` expressions aren't executed until they're explicitly called by something else; they also return a value
- `observe()` is similar to `reactive()`, but continuously "listens" for changes in its dependencies (e.g., user inputs)
- Key advantage of `observe()`: Make the server more organized/efficient with the priority argument

## Caching

- Use bindCache() to improve performance via caching
- Important to carefully select cache keys, which will determine when cache needs to be refreshed. For example:
  - Sys.Date() (today's date) to refresh cache file once per day
  - Last modified date and time for a file
  - Input values

## Click and hover events

## Key Advantage of reactive()

What's the difference?

- getData <- reactive({
  Pull data based on input$a
  Filter, sort data based on input$b
  Run calculations data based on input$c
  })

- pullData <- reactive({Pull data based on input$a })
- filterSortData <- reactive({ Filter, sort pullData() based on input$b })
- calcData <- reactive({ Run calculations on filterSortData() based on input$c })

## Debugging

- Place the `browser()` function inside the server wherever you want to pause the server and investigate further
- Use `renderPrint()` and `verbatimTextOutput()` to print values and display them directly in the UI
- Use reactive log to understand order in which reactives are being called
- For more details: https://shiny.posit.co/r/articles/improve/debugging/