An Intro to Shiny

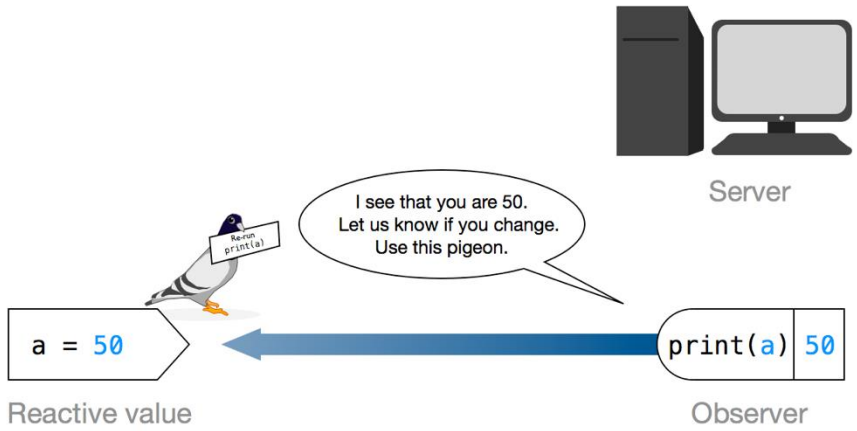
J. David Aponte

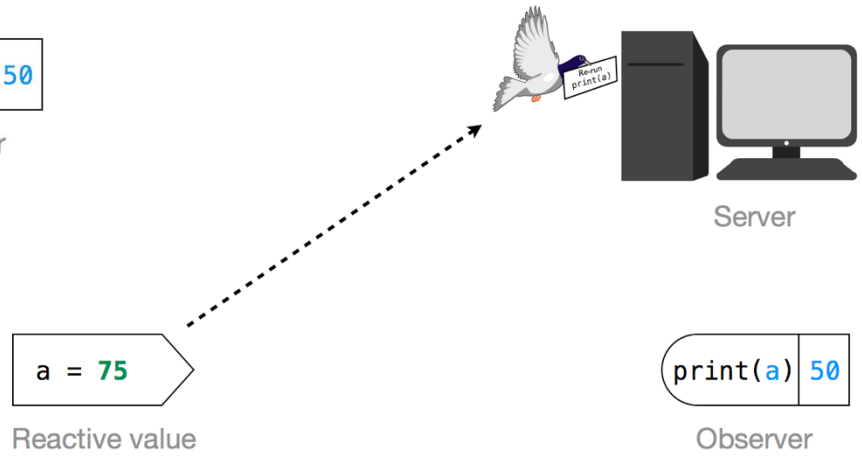
What is Shiny?

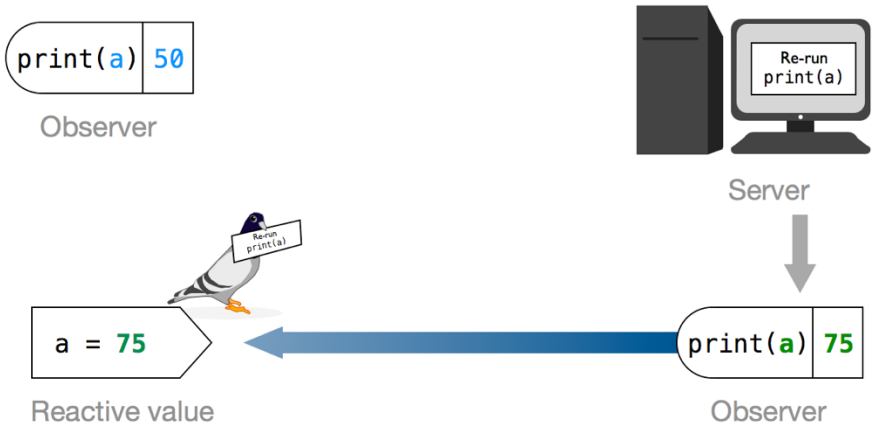
- Shiny is an R package that makes it easy to build interactive web apps straight from R
- Interactivity is valuable when the underlying data is too complex to tell the whole story about
- Case in point: https://www.washingtonpost.com/health/interactive/2021/unvaccinated-case-rate-delta-surge/

How does Shiny work?

Reactivity – messages via virtual carrier pigeon







4 ways Shiny is different from R

1. Make your inputs dynamic

Use food_choice to select a row from a data frame

```
measure_df <- ca_food_name %>%
    filter(FoodDescription == food_choice) %>%
    select(FoodID) %>%
    left_join(ca_conversion_factor) %>%
    left_join(ca_measure_name) %>%
    select(numeric, units, description,
ConversionFactorValue, MeasureID, FoodID)
```

in an R script:

```
food_choice <- "Coffee, brewed, prepared with tap water"
```

in a Shiny script:

```
food_choice <- input$ingredient
```

4 ways Shiny is different from R

2. You have to write some R code to make a user interface (UI)

```
# Let's define our user interface (UI)####
ui ← fluidPage(
  # Application title
  titlePanel("Health Canada Nutrient Calculator"),

  # Sidebar with a selector for food item and for the amount of food
  sidebarLayout(
    sidebarPanel(
      selectInput(inputId = "ingredient", label = "Which item?", choices = ca_food_name$FoodDescription, multiple = F),
      sliderInput(inputId = "amount", label = "How much?", min = 2, max = 500, step = 20, value = 250)
    ),

    # Show interactive tables and plots of the mineral and macronutrient content of the selected food item
    mainPanel(
      dataTableOutput("nutrientTable"),
      plotlyOutput("nutrientPlot")
    )
  )
)
```

4 ways Shiny is different from R

3. You have to define what kinds of outputs you will produce in a server function – table, plot, etc

```
# Let's define our server logic####
server ← function(input, output){
  # The calculations for each output we've defined for the mainPanel() should go here

  # An important difference between Shiny and normal R. Inputs can only be passed to an actively listening context.
  # Uncomment this line and try to run the app: food_choice ← input$ingredient

  output$nutrientTable ← renderDataTable({ })

  output$nutrientPlot ← renderPlotly({ })

} # end server logic
```

4 ways Shiny is different from R

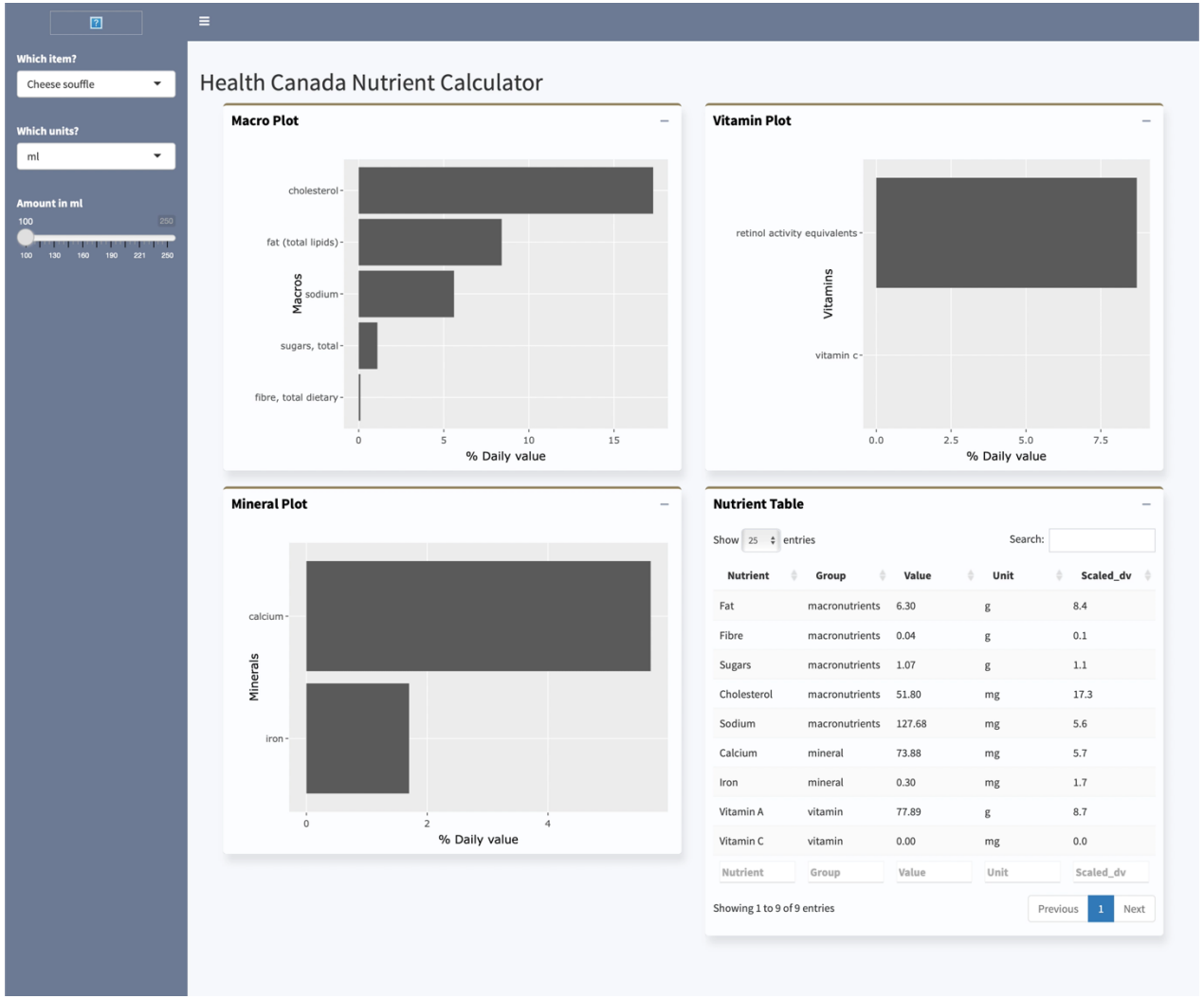
4. R actively listens to your inputs

This blocks you from running non-app code while the app is running

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

Listening on http://127.0.0.1:4250
Warning: Error in path.expand: invalid 'path' argument
  [No stack trace available]
```

Our sample application today



How this workshop is organized

1. **Don’t start in Shiny!** Converting an existing analysis to a barebones app
2. **Thinking in the Shiny paradigm** – the basics of reactivity
3. **Make it look cool** – packages for styling Shiny apps
4. **Share your work** – deploying your work to shinyapps.io

Build it yourself with me: https://github.com/J0vid/CalgaryR_Shiny

Suggestions for making your own stuff

1. Think about what you want the app to look like and write a normal R script version first
2. Make sure your code works with a few hardcoded test examples
3. After you have a script that does what you want, plug in your inputs to make it interactive
4. If you're new to R/Shiny, don't try to add lots of inputs at the same time because it can be really hard to diagnose problems
5. If you didn't heed my advice and are trying to solve errors from within Shiny, print(), browser(), and breakpoints are your friends