Making any R code interactive – engaging stakeholders using shiny apps

LEC PGR Geospatial Data Science Working Group

Michael Tso (mtso *at* ceh.ac.uk)

October 2019

CEH Environmental Data Science Group

- Environmental Infomatics Data Centre
- Speicialist data resources, portals and information products
- Software development
- National capability
- DataLab initiative and the data science framework
- Close collaboration with LU (e.g. LEC, DSI, DSNE) and other partners
- My role

Centre for
Ecology & Hydrology
NATURAL ENVIRONMENT RESEARCH COUNCIL

What is `shiny`?

- R package from RStudio

> "A simple, in-browser, markdown-driven slideshow tool targeted at people who know their way around HTML and CSS"

- Web application framework for R
- R code $\longrightarrow$ interactive web page
- No HTML/CSS/JavaScript knowledge required
- Great for sharing R analysis with someone scared of R (and others)

Why sharing your work interactively using web apps/dashboards?

- Let the user explore the datset themselves (Guided tour)
- Let the user try out your methods (e.g. using their own dataset)
- Illustrate impact and engage stakeholders
- Python equivalent? Dash by plotly

Today's goal:

- Showcase a few R shiny apps (a flavour of what they can do)
- Basic R shiny app concepts
- Key resources and related R packages
- A guide to make your first shiny app
- Q&A

Movie explorer

Filter

Minimum number of reviews on Rotten Tomatoes

10 80 300

Year released

1,940 1,970 2,014

Minimum number of Oscar wins (all categories)

0 4

Dollars at Box Office (millions)

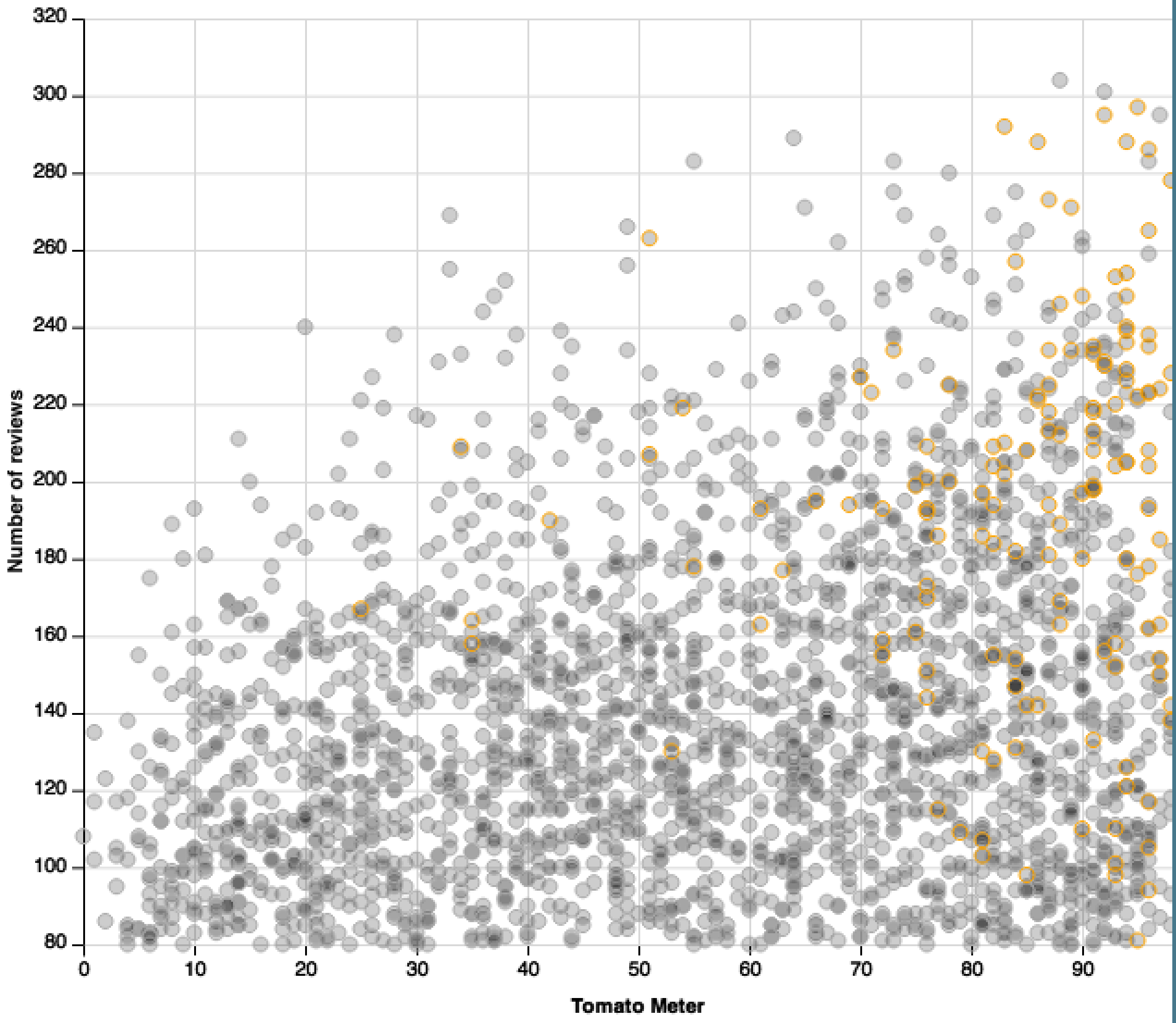
0 800

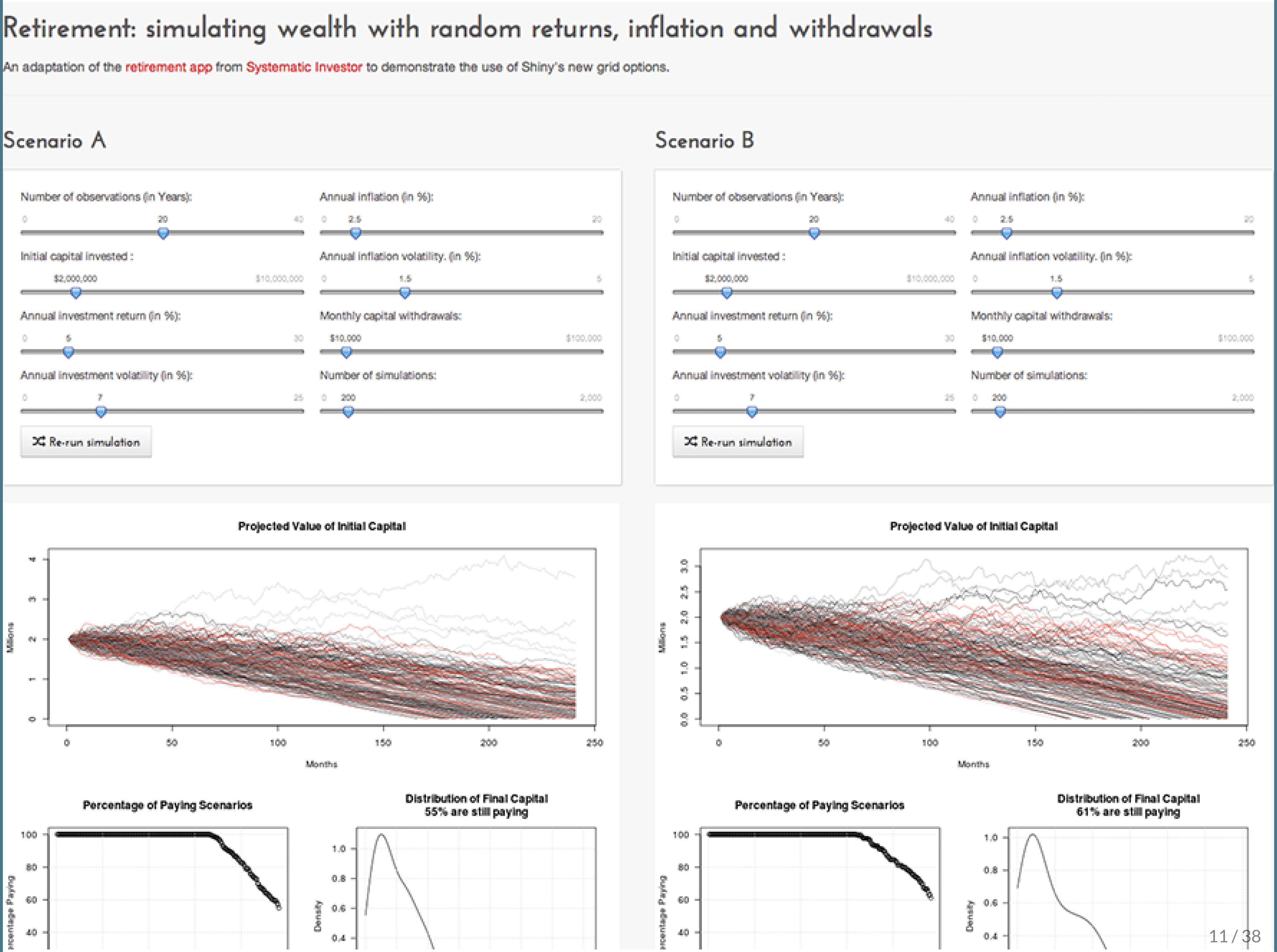
Genre (a movie can have multiple genres)

All

X-axis variable

Tomato Meter





Retirement: simulating wealth with random returns, inflation and withdrawals

An adaptation of the retirement app from Systematic Investor to demonstrate the use of Shiny's new grid options.

Scenario A

Number of observations (in Years): 20 (0–40)

Initial capital invested : $2,000,000 ($10,000,000)

Annual investment return (in %): 5 (0–30)

Annual investment volatility (in %): 7 (0–25)

Annual inflation (in %): 2.5 (0–20)

Annual inflation volatility. (in %): 1.5 (0–5)

Monthly capital withdrawals: $10,000 ($100,000)

Number of simulations: 200 (0–2,000)

Re-run simulation

Scenario B

Number of observations (in Years): 20 (0–40)

Initial capital invested : $2,000,000 ($10,000,000)

Annual investment return (in %): 5 (0–30)

Annual investment volatility (in %): 7 (0–25)

Annual inflation (in %): 2.5 (0–20)

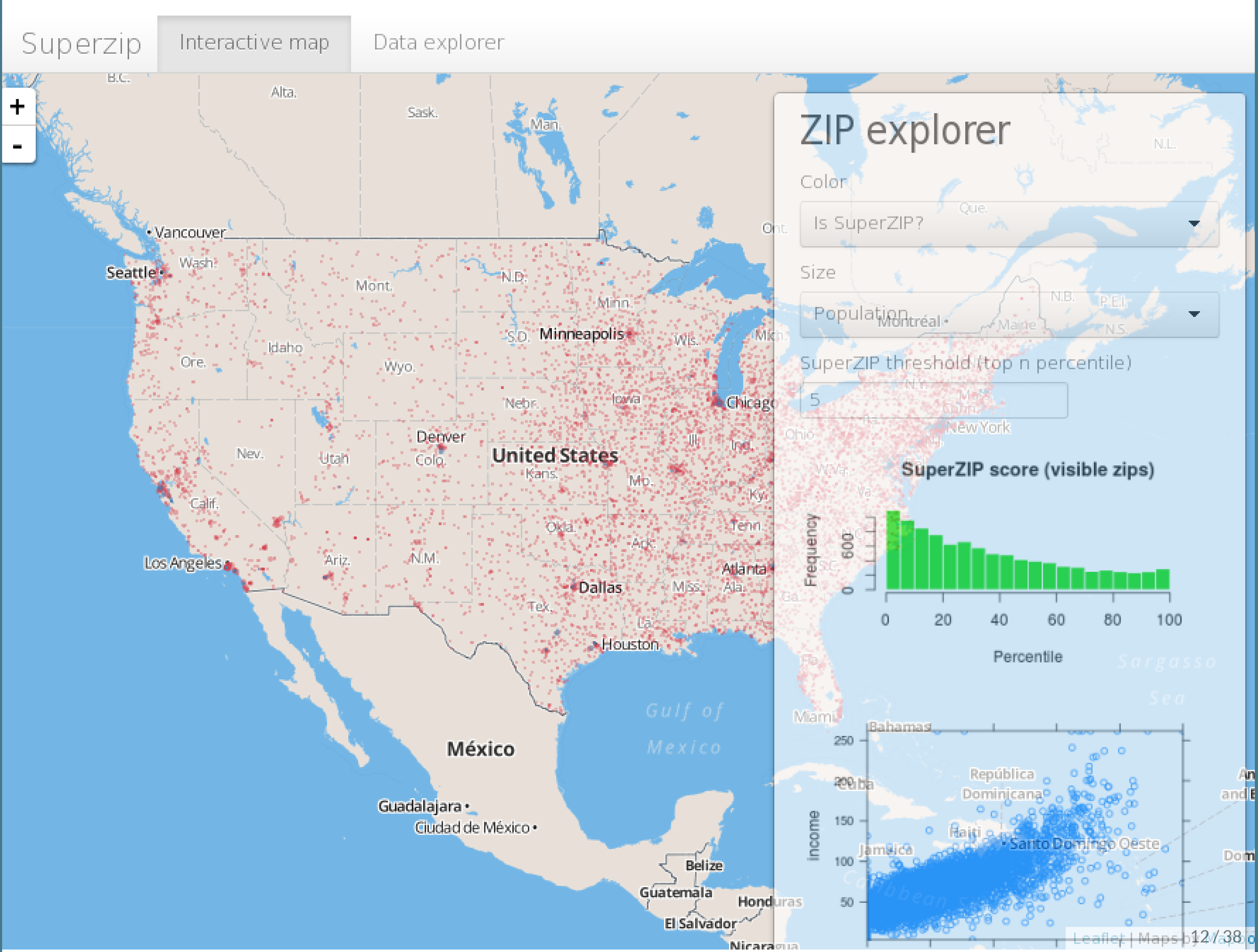
Annual inflation volatility. (in %): 1.5 (0–5)

Monthly capital withdrawals: $10,000 ($100,000)

Number of simulations: 200 (0–2,000)

Re-run simulation

Projected Value of Initial Capital
Millions
Months
Percentage of Paying Scenarios
Percentage Paying
Distribution of Final Capital
55% are still paying
Density
Projected Value of Initial Capital
Millions
Months
Percentage of Paying Scenarios
Percentage Paying
Distribution of Final Capital
61% are still paying
Density



ZIP explorer

Color: Is SuperZIP?

Size: Population

SuperZIP threshold (top n percentile): 5

Changepoint analysis

for the UK National River Flow Archive (NRFA)

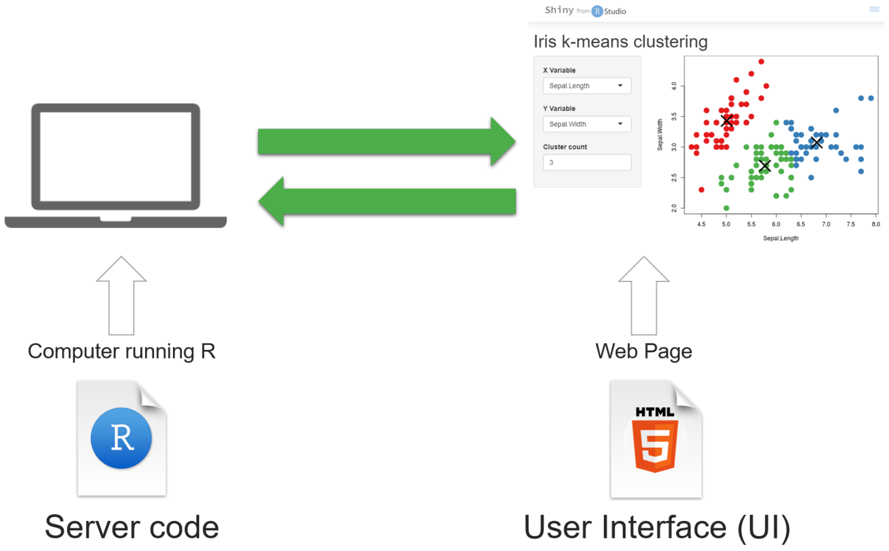
Nature's postcodes:

Which neigborhood is doing best in this species recording citizen science project?

Crucial `shiny` concepts

(1) UI + server, (2) reactivity

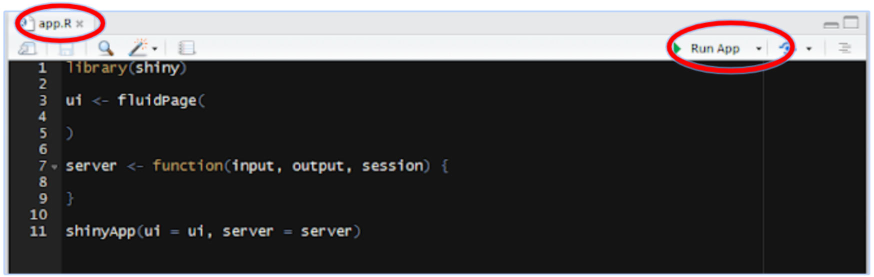
What's in a `shiny` app?



- plus optional web elements (e.g. static images, css files)

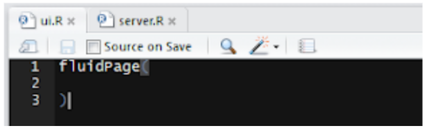
Two ways to run a `shiny` app

- single file option

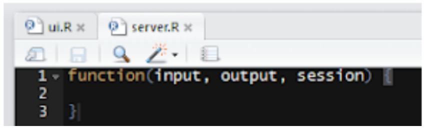


```
library(shiny)

ui <- fluidPage(

)

server <- function(input, output, session) {

}

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

- two file option - save UI as "ui.R" and server as "server.R" in the same directory



```
fluidPage(

)
```



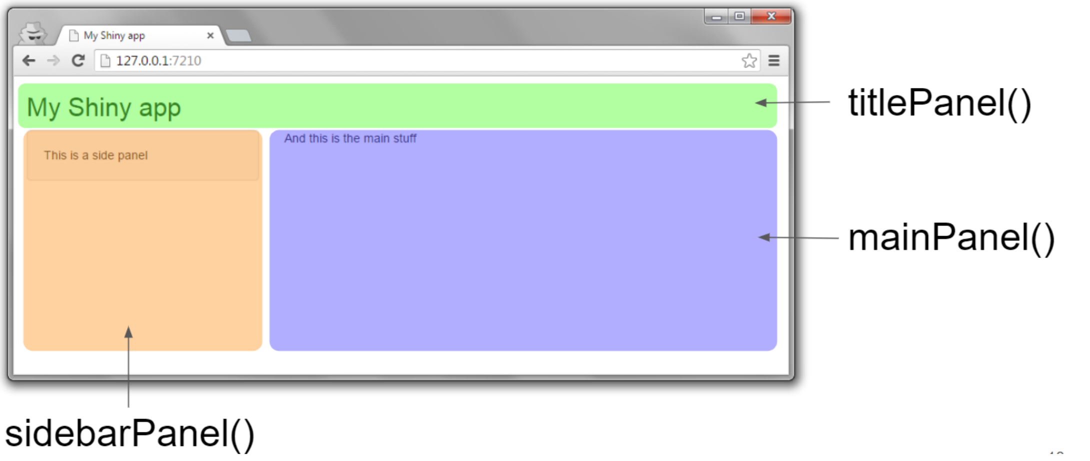
```
function(input, output, session) {

}
```

Making your `shiny` app available

- Run it locally
- Serve it on http://www.shinyapps.io/
- Use other servers: Amazon Cloud, your institution etc.

UI components

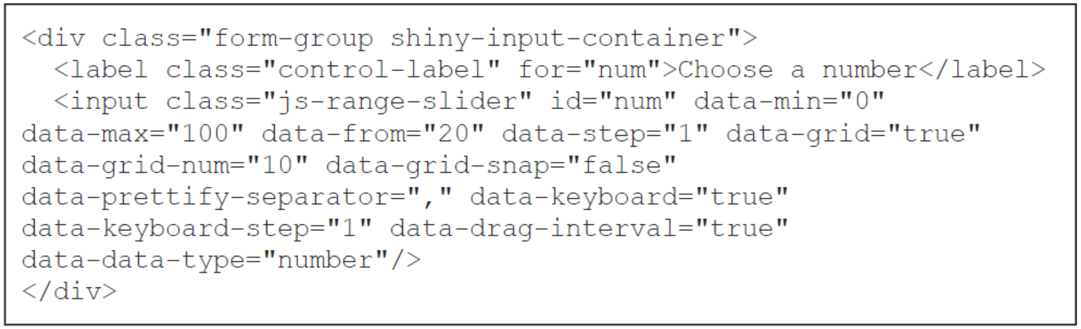
- inputs (e.g. buttons, sliders, checkboxes, file upload box) ...`input$*`
- outputs (e.g. plots, tables)
- layouts (how do you want everything to be arranged) and HTML
- check out the package `shinythemes`



UI components

- everything is turned into HTML + JavaScript for you (it's just a webpage)

```r
sliderInput("num","Choose a number", min = 0, max=100, value = 20)
```



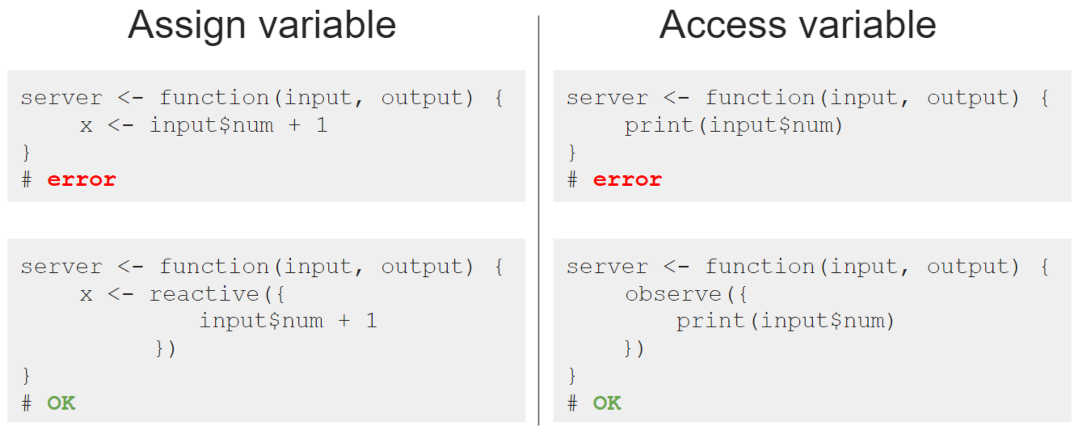
```html
<div class="form-group shiny-input-container">
  <label class="control-label" for="num">Choose a number</label>
  <input class="js-range-slider" id="num" data-min="0"
data-max="100" data-from="20" data-step="1" data-grid="true"
data-grid-num="10" data-grid-snap="false"
data-prettify-separator="," data-keyboard="true"
data-keyboard-step="1" data-drag-interval="true"
data-data-type="number"/>
</div>
```

Server components

- your analysis (does it react to user input?)
- If so, use `reactive` (assign), `observe` (access), `isoluate`
- output components...`output$*`, use `render*( {} )`

Reactivity

- Shiny uses reactive programming and supports reactive variables
- Unlike regular R, if x changes, anything that relies of x is re-evaluated.



Assign variable

```
server <- function(input, output) {
     x <- input$num + 1
}
# error
```

```
server <- function(input, output) {
     x <- reactive({
           input$num + 1
         })
}
# OK
```

Access variable

```
server <- function(input, output) {
     print(input$num)
}
# error
```

```
server <- function(input, output) {
     observe({
          print(input$num)
     })
}
# OK
```

Run your first `shiny` code

"Learning is doing."

Run your first shiny code

```r
shinyAppDir(
  system.file("examples/06_tabsets", package="shiny"),
  options = list(width = "100%", height = 700)
)
```

View the code:

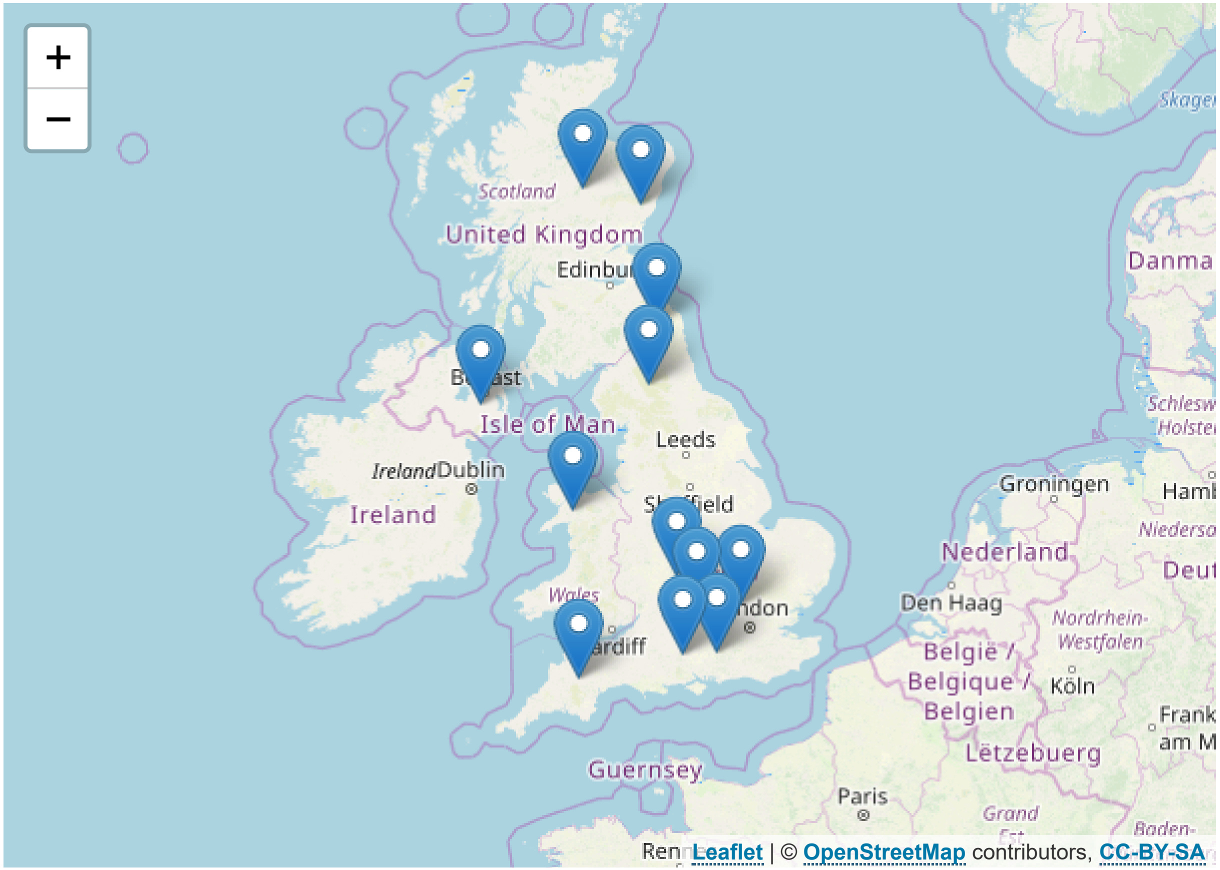
```r
# file.show(system.file("examples/06_tabsets/app.R", package="shi
# https://dr-harper.github.io/rmarkdown-cookbook/html-scroll.html

library(shiny)

# Define UI for random distribution app ----
ui <- fluidPage(

  # App title ----
  titlePanel("Tabsets"),

  # Sidebar layout with input and output definitions ----
  sidebarLayout(

    # Sidebar panel for inputs ----
    sidebarPanel(
```

Useful R packages to go with R
`shiny`

"Make your app interactive and look pretty!"

Adding interactive maps using **leaflet**

```
leaflet(data=ECN_site_info) %>% addTiles() %>%addMarkers(~long,~lat
```



Presenting tables

If you want to generate a table, make sure it is in the HTML format (instead of Markdown or other formats), e.g.,

```
knitr::kable(head(MEMSS::Theoph), format = 'html')
```

Subject	Wt	Dose	Time	conc
A	79.6	4.02	0.00	0.74
A	79.6	4.02	0.25	2.84
A	79.6	4.02	0.57	6.57
A	79.6	4.02	1.12	10.50
A	79.6	4.02	2.02	9.66
A	79.6	4.02	3.82	8.58

Using the package DT

```r
DT::datatable(
  head(MEMSS::Theoph, 120),
  fillContainer = FALSE, options = list(pageLength = 7)
)
```

Show 7 ▾ entries Search:

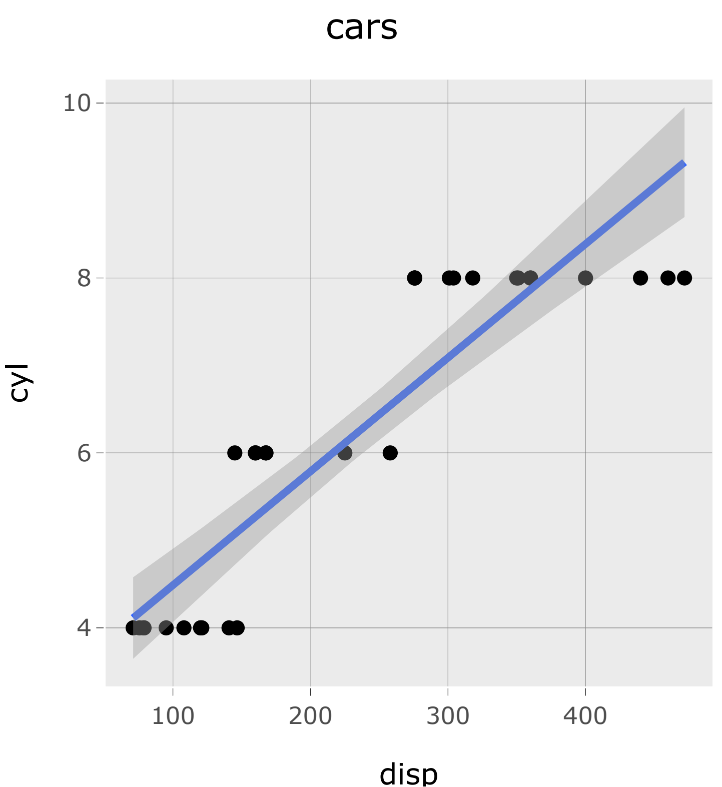
	Subject	Wt	Dose	Time	conc
1	A	79.6	4.02	0	0.74
2	A	79.6	4.02	0.25	2.84
3	A	79.6	4.02	0.57	6.57
4	A	79.6	4.02	1.12	10.5
5	A	79.6	4.02	2.02	9.66
6	A	79.6	4.02	3.82	8.58
7	A	79.6	4.02	5.1	8.36

Showing 1 to 7 of 120 entries

`plotly` for nice interactive plots

- Works for most `ggplot2` functions
- You may try `ggvis` and `googlevis` packages too

```
p = ggplot(data = mtcars, aes(x = disp, y = cyl)) +
      geom_point() + geom_smooth(method=lm) + ggtitle('cars')
ggplotly(p)
```



How to design your `shiny` app?

Things to think about:

- what is your audience?
- What data/ data product would you like them to interact with?
- what would you like them to change (i.e. inputs)?
- what would you like to show them (i.e. outputs)?
- how to fit your analysis behind the scene (i.e. in server.R)
- how would you like the different elements appear (e.g. layout)?

Useful resources

Links to ebooks/blogs/doc/repo pages to learn more.

Lists of resources

- STAT545 ebook shiny tutorial
- RStudio shiny cheat sheet
- slides by Dean Attali
- R x Plotly ebook
- blogs by Zev Ross
- RStudio GitHub: 100+ examples
- Stack Overflow

Thanks!

Slides created via the R package xaringan.

This presentation is made using the R package **xaringan**

- **Free** and Open Source (Slides are composed in R Markdown + css)--> HTML document
- Uses **reveal.js** library
- Portable and easily share (Web-based slide) and easily print to PDF from Chrome
- Repodicibe research result
- Limited capability to embed shiny apps at the moment