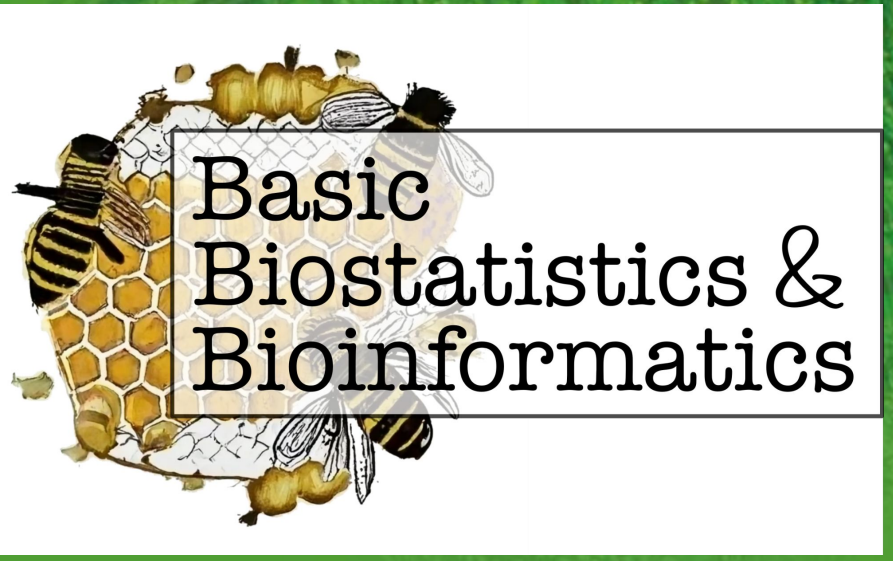


3 B's Session 12

R Shiny

Lizel Potgieter
lizel.potgieter@slu.se
Department of Plant Breeding, SLU Alnarp

Who we are

SLU bioinformatics Infrastructure

Weekly online drop-in (Wednesdays at 13.00)

slubi@slu.se,

Alnarp: Lizel Potgieter (Dept. of Plant Breeding)

Statistics at SLU

SLU statistics center

Free consultations for all SLU staff

statistics@slu.se

Alnarp: Jan-Eric Englund and Adam Flöhr (Dept. of Biosystems and Technology)

Introduction

Shiny can be used to build web apps and interactive documents

Useful if you need a live environment

Gallery

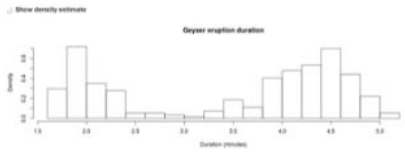
Shiny

Home Get Started Shiny for R Shiny for Python Blog

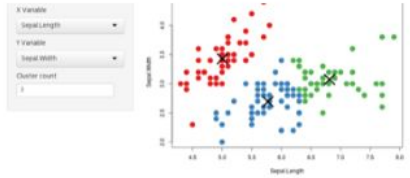
Get Started Articles Gallery Reference Help Deploy Contribute

Start Simple

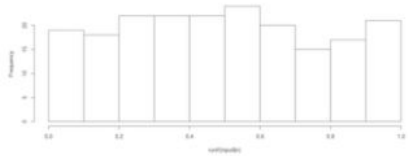
If you're new to Shiny, these simple but complete applications are designed for you to learn from.



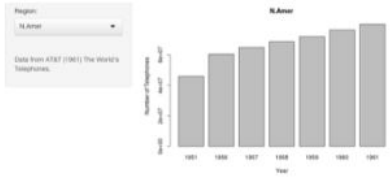
Faithful



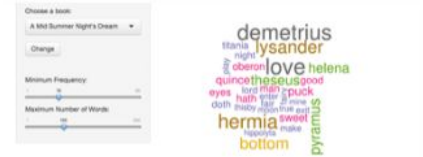
Kmeans example



Single-file shiny app



Telephones by region



Word cloud

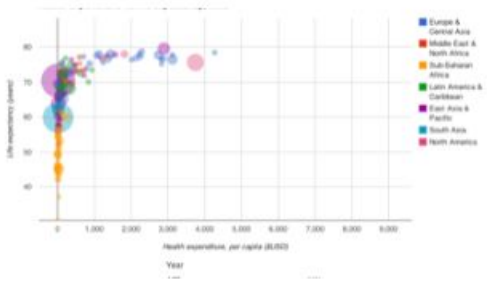
See: https://shiny.posit.co/r/gallery/#feature-demos

Gallery

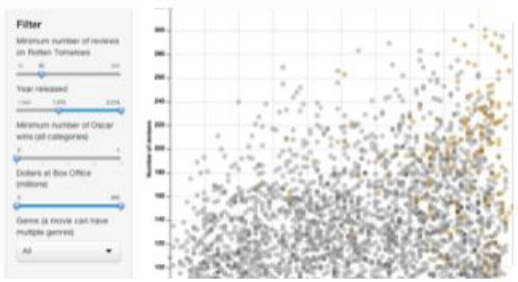
Interactive visualizations

Shiny is designed for fully interactive visualization, using JavaScript libraries like d3, Leaflet, and Google Charts.

Bus dashboard



Google Charts



Movie explorer

SuperZip example

See: https://shiny.posit.co/r/gallery/#feature-demos

App Structure

In its most basic form: UI and a Server

UI block: this is what your user can change

Server: uses reactive programming so that when something in the UI/input changes, all related outputs are automatically updated

```
library(shiny)

ui <- fluidPage(
  # front end interface
)

server <- function(input, output, session) {
  # back end logic
}

shinyApp(ui, server)
```

```
library(shiny)
ui <- fluidPage(
  "Hello, world!"
)
server <- function(input, output, session) {
}
shinyApp(ui, server)
```

See: https://mastering-shiny.org/basic-app.html

UI Block

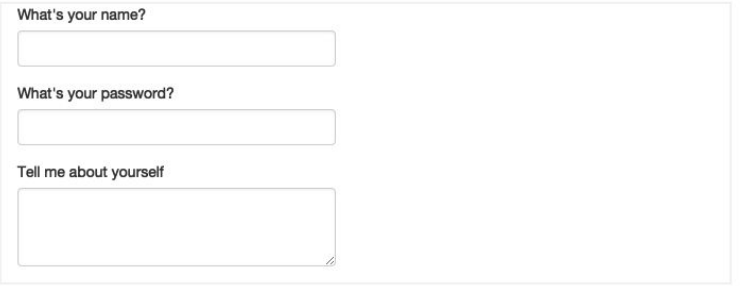
There are may input formats you can use (textInput, dateInput, dateRangeInput etc)

You can use sliders, lists, and multiple choice lists

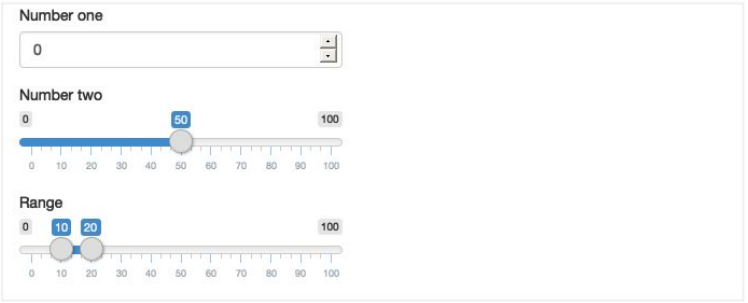
This in and of itself is not particularly useful beyond clicking a few buttons. Combining this with the server makes it really cool!

See: https://mastering-shiny.org/basic-ui.html

```
ui <- fluidPage(
  textInput("name", "What's your name?"),
  passwordInput("password", "What's your password?"),
  textAreaInput("story", "Tell me about yourself", rows = 3)
)
```

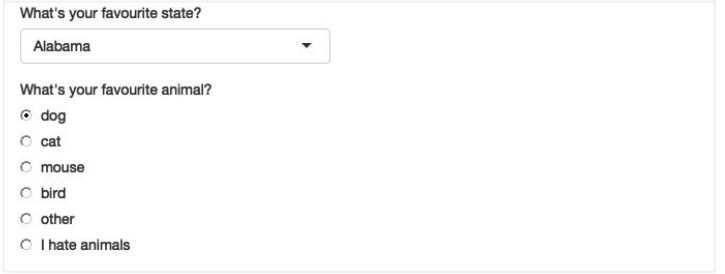



```
ui <- fluidPage(
  numericInput("num", "Number one", value = 0, min = 0, max = 100),
  sliderInput("num2", "Number two", value = 50, min = 0, max = 100),
  sliderInput("rng", "Range", value = c(10, 20), min = 0, max = 100)
)
```




```
animals <- c("dog", "cat", "mouse", "bird", "other", "I hate animals")

ui <- fluidPage(
  selectInput("state", "What's your favourite state?", state.name),
  radioButtons("animal", "What's your favourite animal?", animals)
)
```



Server Block

Relies on reactive programming where inputs are directly connected to output

```
ui <- fluidPage(
  textInput("name", "What's your name?"),
  textOutput("greeting")
)

server <- function(input, output, session) {
  output$greeting <- renderText({
    paste0("Hello ", input$name, "!")
  })
}
```

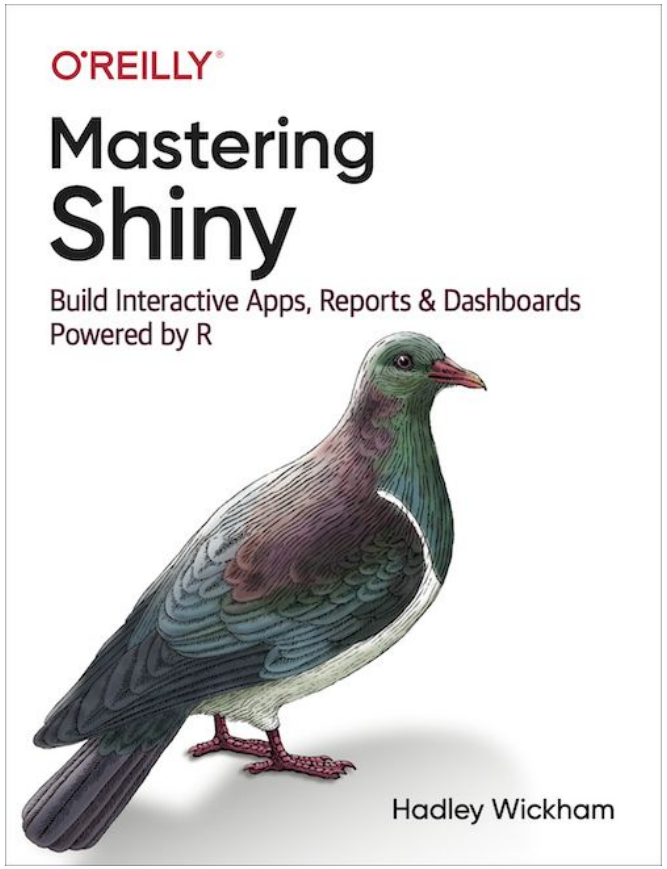
See: https://mastering-shiny.org/basic-reactivity.html

Additional Reading

https://mastering-shiny.org/index.html

https://ourcodingclub.github.io/tutorials/shiny/

https://jcoliver.github.io/learn-r/016-intro-shiny.html




SLU
SCIENCE AND EDUCATION FOR SUSTAINABLE LIFE