How to create a Shiny web app in R

Elisabeth Dahlqwist

Department of Medical Epidemiology and Biostatistics, Karolinska Institutet
elisabeth.dahlqwist@ki.se

March 15, 2018

Outline

- What is Shiny and how to get started?
- Simple example
- My experience with Shiny
- Using Shiny as an user interface for an `R`- package.
- How to share your Shiny

What is Shiny?

"A Shiny app is a web page connected to a computer running a live R session" - Shiny cheat sheet

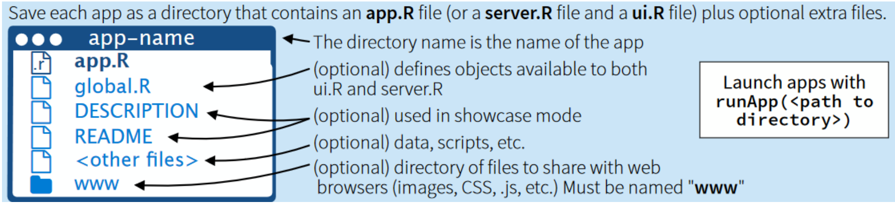
And all you need is:

- `R-studio`
- `install.packages("shiny")` and `library(shiny)`
- Basic R knowledge (for using templates) and good R knowledge for doing more tailored apps

Templates
`https://shiny.rstudio.com/gallery/`

Shiny file structure

- `ui.R` define the user interface
- `server.R` compute the function, graph etc. depending on the input from the user
- `global.R` contain all additional code for the app
- All files are saved in a folder with the name of the app


Save each app as a directory that contains an **app.R** file (or a **server.R** file and a **ui.R** file) plus optional extra files.

app–name
app.R
global.R
DESCRIPTION
README
<other files>
www
The directory name is the name of the app
(optional) defines objects available to both ui.R and server.R
(optional) used in showcase mode
(optional) data, scripts, etc.
(optional) directory of files to share with web browsers (images, CSS, .js, etc.) Must be named "www"
Launch apps with runApp(<path to directory>)

How to learn Shiny (except for this tutorial)

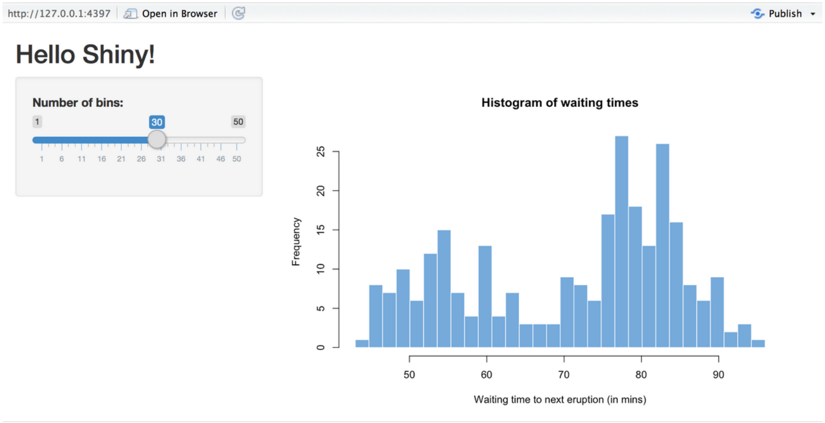
Basic tutorials
https://shiny.rstudio.com/tutorial/

https://www.youtube.com/watch?v=sJl0EE_RE4o&list=PLH6mU1kedUy-aGYi-w1XqSiGtViFK9NpI

https://github.com/aagarw30/R-Shinyapp-Tutorial

Shiny-cheatsheet
http://shiny.rstudio.com/images/shiny-cheatsheet.pdf

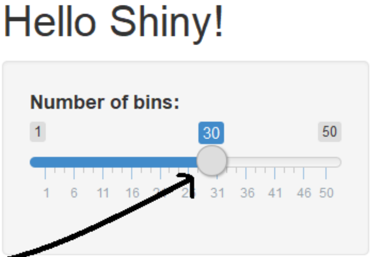
Simple example



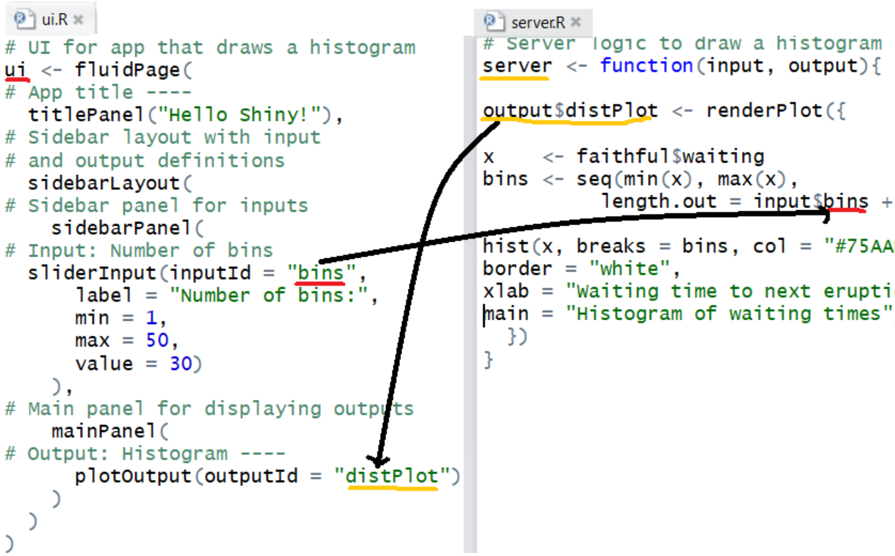
https://shiny.rstudio.com/articles/basics.html

ui.R

```
# UI for app that draws a histogram
ui <- fluidPage(
# App title ----
  titlePanel("Hello Shiny!"),
# Sidebar layout with input
# and output definitions
  sidebarLayout(
# Sidebar panel for inputs
    sidebarPanel(
# Input: Number of bins
  sliderInput(inputId = "bins",
      label = "Number of bins:",
      min = 1,
      max = 50,
      value = 30)
    ),
# Main panel for displaying outputs
    mainPanel(
# Output: Histogram ----
      plotOutput(outputId = "distPlot")
    )
  )
```



ui.R and server.R



```
# UI for app that draws a histogram
ui <- fluidPage(
# App title ----
  titlePanel("Hello Shiny!"),
# Sidebar layout with input
# and output definitions
  sidebarLayout(
# Sidebar panel for inputs
    sidebarPanel(
# Input: Number of bins
  sliderInput(inputId = "bins",
      label = "Number of bins:",
      min = 1,
      max = 50,
      value = 30)
    ),
# Main panel for displaying outputs
    mainPanel(
# Output: Histogram ----
      plotOutput(outputId = "distPlot")
    )
  )
)
```

```
# Server logic to draw a histogram
server <- function(input, output){

output$distPlot <- renderPlot({

x     <- faithful$waiting
bins <- seq(min(x), max(x),
          length.out = input$bins +

hist(x, breaks = bins, col = "#75AA
border = "white",
xlab = "Waiting time to next erupti
main = "Histogram of waiting times"
  })
}
```

My Shiny application

Imagine you want to visualize this:

$$AF(p, b, k, h) = \frac{\Phi\{\Phi^{-1}(p), -b; h\} - \Phi\{\Phi^{-1}(p) - kh, -b; h\}}{p}$$

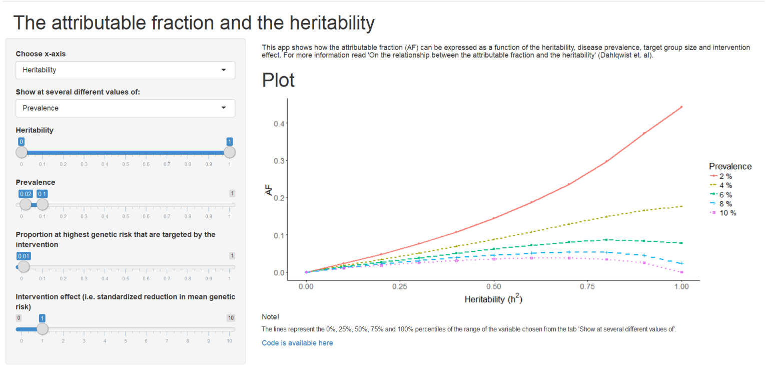
where

- $\Phi(\cdot)$ is the standard (i.e. mean 0, variance 1) normal distribution function

Problem: A function that depends on 4 parameter is difficult to visualize.

Solution: Interactive graph

My Shiny app



The attributable fraction and the heritability

Choose x-axis

Heritability

Show at several different values of:

Prevalence

Heritability 0 – 1

Prevalence 0.02 – 0.1

Proportion at highest genetic risk that are targeted by the intervention 0.01

Intervention effect (i.e. standardized reduction in mean genetic risk) 1

This app shows how the attributable fraction (AF) can be expressed as a function of the heritability, disease prevalence, target group size and intervention effect. For more information read 'On the relationship between the attributable fraction and the heritability' (Dahlqwist et. al).

Plot

Note!

The lines represent the 0%, 25%, 50%, 75% and 100% percentiles of the range of the variable chosen from the tab 'Show at several different values of'.

Code is available here

https://afheritability.shinyapps.io/afheritability/

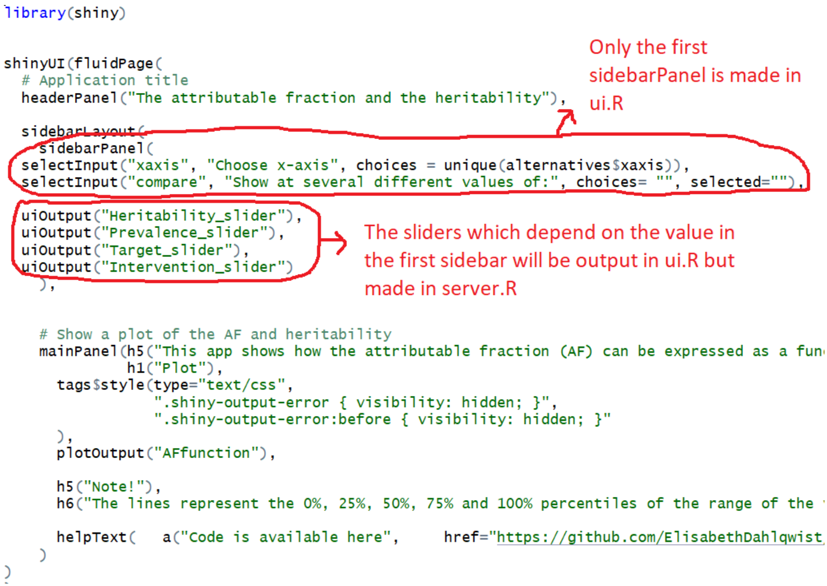
Creating interactive panels

`conditionalPanel()` looks intuitive since the value choosen in the first condition give you different conditions depending on the value.

My problem: A unique id is needed within every level of the `conditionalPanel()` to make it interactive $\rightarrow$ was not possible for me

My solution: Create the a function that create sliders depending on the first input arguments.

My solution - ui.R

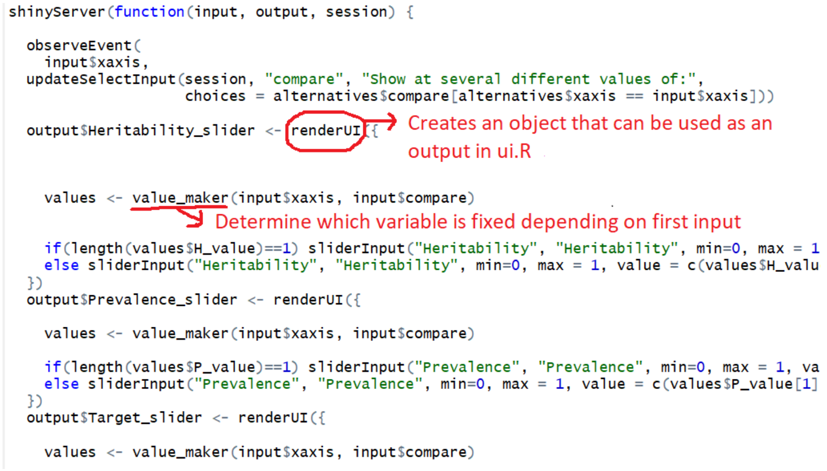


```
library(shiny)


shinyUI(fluidPage(
  # Application title
  headerPanel("The attributable fraction and the heritability"),

  sidebarLayout(
   sidebarPanel(
  selectInput("xaxis", "Choose x-axis", choices = unique(alternatives$xaxis)),
  selectInput("compare", "Show at several different values of:", choices= "", selected=""),

  uiOutput("Heritability_slider"),
  uiOutput("Prevalence_slider"),
  uiOutput("Target_slider"),
  uiOutput("Intervention_slider")
    ),


    # Show a plot of the AF and heritability
    mainPanel(h5("This app shows how the attributable fraction (AF) can be expressed as a fun
                 h1("Plot"),
      tags$style(type="text/css",
                 ".shiny-output-error { visibility: hidden; }",
                 ".shiny-output-error:before { visibility: hidden; }"
      ),
      plotOutput("AFfunction"),

      h5("Note!"),
      h6("The lines represent the 0%, 25%, 50%, 75% and 100% percentiles of the range of the

      helpText(   a("Code is available here",     href="https://github.com/ElisabethDahlqwist
    )
)
```

Only the first sidebarPanel is made in ui.R

The sliders which depend on the value in the first sidebar will be output in ui.R but made in server.R

My solution - server.R

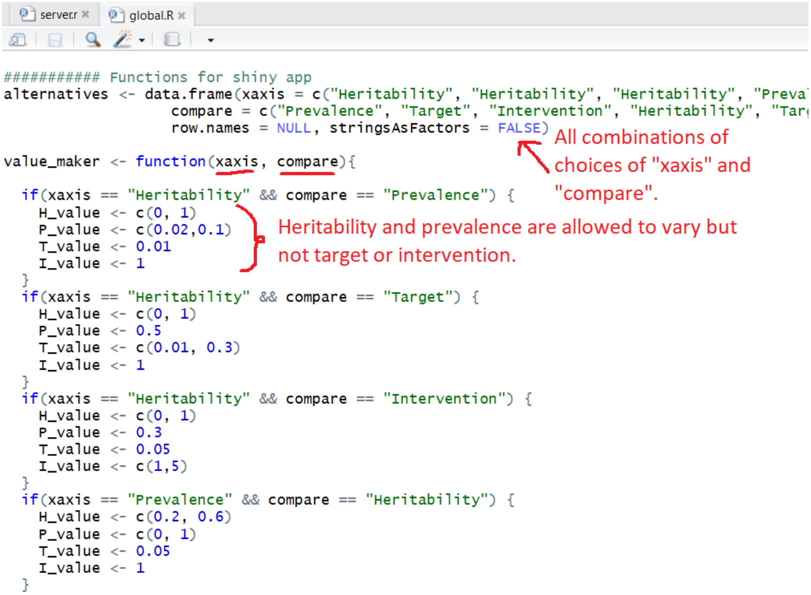


```
shinyServer(function(input, output, session) {

  observeEvent(
    input$xaxis,
  updateSelectInput(session, "compare", "Show at several different values of:",
                    choices = alternatives$compare[alternatives$xaxis == input$xaxis]))

  output$Heritability_slider <- renderUI({


    values <- value_maker(input$xaxis, input$compare)

    if(length(values$H_value)==1) sliderInput("Heritability", "Heritability", min=0, max = 1
    else sliderInput("Heritability", "Heritability", min=0, max = 1, value = c(values$H_valu
  })
  output$Prevalence_slider <- renderUI({

    values <- value_maker(input$xaxis, input$compare)

    if(length(values$P_value)==1) sliderInput("Prevalence", "Prevalence", min=0, max = 1, va
    else sliderInput("Prevalence", "Prevalence", min=0, max = 1, value = c(values$P_value[1]
  })
  output$Target_slider <- renderUI({

    values <- value_maker(input$xaxis, input$compare)
```

Creates an object that can be used as an output in ui.R

Determine which variable is fixed depending on first input

My solution - global.R

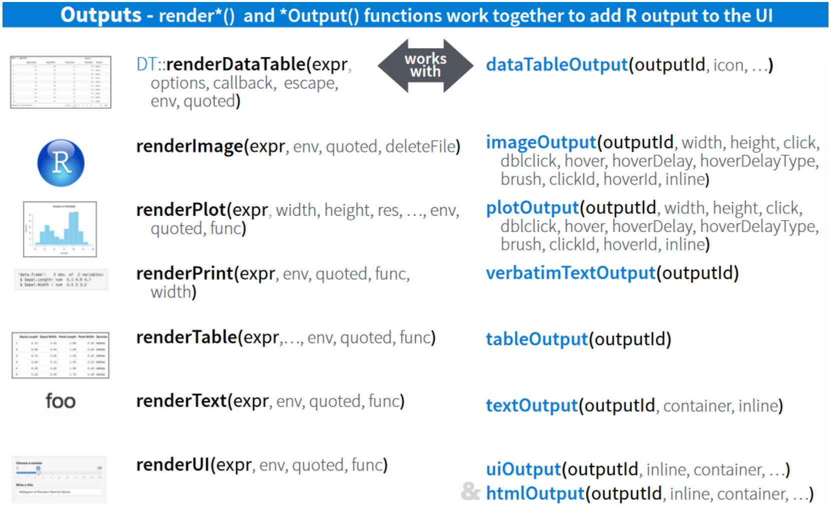


```
########### Functions for shiny app
alternatives <- data.frame(xaxis = c("Heritability", "Heritability", "Heritability", "Preva
                           compare = c("Prevalence", "Target", "Intervention", "Heritability", "Tar
                           row.names = NULL, stringsAsFactors = FALSE)

value_maker <- function(xaxis, compare){

  if(xaxis == "Heritability" && compare == "Prevalence") {
    H_value <- c(0, 1)
    P_value <- c(0.02,0.1)
    T_value <- 0.01
    I_value <- 1
  }
  if(xaxis == "Heritability" && compare == "Target") {
    H_value <- c(0, 1)
    P_value <- 0.5
    T_value <- c(0.01, 0.3)
    I_value <- 1
  }
  if(xaxis == "Heritability" && compare == "Intervention") {
    H_value <- c(0, 1)
    P_value <- 0.3
    T_value <- 0.05
    I_value <- c(1,5)
  }
  if(xaxis == "Prevalence" && compare == "Heritability") {
    H_value <- c(0.2, 0.6)
    P_value <- c(0, 1)
    T_value <- 0.05
    I_value <- 1
  }
```

All combinations of choices of "xaxis" and "compare".

Heritability and prevalence are allowed to vary but not target or intervention.

The render function



Outputs - render*() and *Output() functions work together to add R output to the UI

	render*()	works with	*Output()
	DT::**renderDataTable(expr**, options, callback, escape, env, quoted**)**		**dataTableOutput(outputId**, icon, ...**)**
	renderImage(expr, env, quoted, deleteFile**)**		**imageOutput(outputId**, width, height, click, dblclick, hover, hoverDelay, hoverDelayType, brush, clickId, hoverId, inline**)**
	renderPlot(expr, width, height, res, ..., env, quoted, func**)**		**plotOutput(outputId**, width, height, click, dblclick, hover, hoverDelay, hoverDelayType, brush, clickId, hoverId, inline**)**
	renderPrint(expr, env, quoted, func, width**)**		**verbatimTextOutput(outputId)**
	renderTable(expr,..., env, quoted, func**)**		**tableOutput(outputId)**
foo	**renderText(expr**, env, quoted, func**)**		**textOutput(outputId**, container, inline**)**
	renderUI(expr, env, quoted, func**)**		**uiOutput(outputId**, inline, container, ...**)** & **htmlOutput(outputId**, inline, container, ...**)**

Shiny as an interface for an R-package

`EpiEstim`: a package to estimate disease transmissibility during an infectious disease outbreak

Info: `https://github.com/jstockwin/EpiEstimApp/wiki`

```
install.packages("devtools")
library(devtools)
devtools::install_github("jstockwin/EpiEstimApp",
 ref = "recon-update", force = TRUE)
EpiEstimApp::runEpiEstimApp()
```

Share your Shiny!

There are several alternatives in how to share your Shiny app: `https://shiny.rstudio.com/articles/deployment-web.html`

- Code available on GitHub - require that the user have `R` installed
- `Shinyapps.io` - you share your code with `Shinyapps.io`. Different alternatives for scaling.
- Shiny Server open-source software - requires a Linux server that you will need to set up and maintain
- Shiny Server Pro - extension of Shiny Server with higher security, control, and support that workgroups and enterprises need

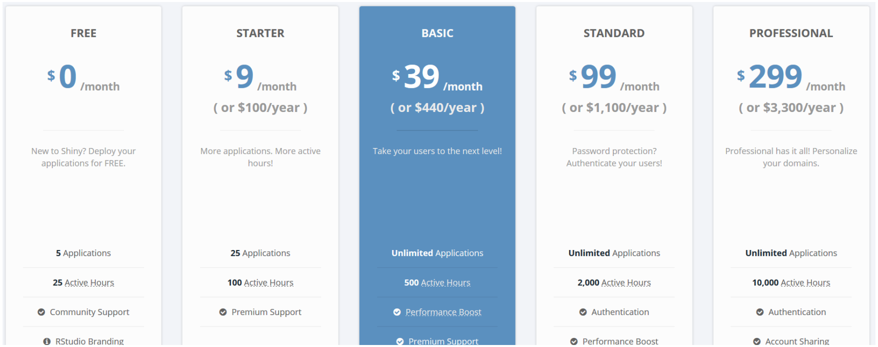
Publish the Shiny app at `Shinyapps.io`

Where to create an account: `http://www.shinyapps.io/`
How to do it:
`https://shiny.rstudio.com/articles/shinyapps.html`

Steps to make it happen!!

- You need:
  `install.packages('rsconnect')` and
  `library(rsconnect)`
- Create a `Shinyapps.io` account
- Configure `rsconnect`:
  `rsconnect::setAccountInfo(name="<ACCOUNT>", token="<TOKEN>", secret="<SECRET>")`
- PUBLISH: `rsconnect::deployApp('Directory of your Shiny app folder')`

Prices at `shinyapps.io`



FREE	STARTER	BASIC	STANDARD	PROFESSIONAL
$ 0 /month	$ 9 /month	$ 39 /month	$ 99 /month	$ 299 /month
	(or $100/year)	(or $440/year)	(or $1,100/year)	(or $3,300/year)
New to Shiny? Deploy your applications for FREE.	More applications. More active hours!	Take your users to the next level!	Password protection? Authenticate your users!	Professional has it all! Personalize your domains.
5 Applications	**25** Applications	**Unlimited** Applications	**Unlimited** Applications	**Unlimited** Applications
25 Active Hours	**100** Active Hours	**500** Active Hours	**2,000** Active Hours	**10,000** Active Hours
✓ Community Support	✓ Premium Support	✓ Performance Boost	✓ Authentication	✓ Authentication
ℹ RStudio Branding		✓ Premium Support	✓ Performance Boost	✓ Account Sharing

My experience of Shiny in summary

- Easy to get started
- With more advanced operations it can be a bit tricky
- BUT, information, tutorials etc. are really really good!
- Surprizingly easy to make the Shiny into a web app - but probably more complicated for some more advanced Shiny

My grade to Shiny as a very beginner:

R R R R out of 5

Thank you!