BIO8068 Data visualisation and management in ecology

Further shiny development

Roy Sanderson

## 1. Introduction

When designing and writing shiny applications it is easy to become confused by the R code, as the code in the user-interface (ui) and server functions is difficult to debug. The simplest approach is to compartmentalise different tasks, testing them in standard R scripts, and either calling these at the start of the main app.R code using the source function to setup various initial parameters, creating functions than can be re-used later, or integrating them into the server code once you know the original R code is robust. Good coding practices will help you do this effectively and well. Finally, you want others to be able to use your shiny app, so you have to know how to publish it on the web. The main aims of this practical are to:

* learn how to plan a shiny app, how to structure and test it
* good coding practice, shiny reactive programming
* publishing your app on the web

## 2. Planning your shiny app

Decide on who will be your users: how much technical skills will they have, and what will they want to gain through using your shiny app? For your assignment, assume that your end-users might be informed members of the public, wanting to gain a more in-depth understanding of the environment and ecology in Cumbria. Sketch out on paper what you would like your shiny app to be able to show, what data (maps, graphs, tables) do you want displayed. How might you want your user to interact with your shiny app? Do you want the user to be presented with a single page, that they can scroll down (much simpler to build), or separate tabs on a dashboard (harder to code).

## 3. Prototyping your shiny app

Spend some time thinking carefully about the user interface (UI) part of your application. Ideally you want to design this and check that it *roughly* behaves as expected before you invest a lot of time on the server code. A good place to start is the Shiny Gallery where you can look at example interfaces, read their code, and even see which lines of code are responding to different user-interactions dynamically. You can find Shiny Gallery at <https://shiny.rstudio.com/gallery/>

* Begin by looking at the the section marked Start Simple where you can learn about simple methods, particularly the “iris dataset” <https://shiny.rstudio.com/gallery/kmeans-example.html> scatterplot and “Telephones by region” <https://shiny.rstudio.com/gallery/telephones-by-region.html>. **Note** Originally shiny apps had to be written with the ui and server in separate R files, but this is now optional and they can be both stored in a single app.R file.
* The section entitled Widgets gives simple examples of how to use check boxes, drop-down lists, sliders, graphs, tables of data etc. Explore some of these, particularly those that you think might be useful to your app. **Exercise**: Copy the code of one or two widget examples and test them: this is the only real way of learning what they do.
* Decide whether you want a long-style of website or a dashboard. An example long-style website is that of the USGS Biological Monitoring Station at Lake Erie <https://gallery.shinyapps.io/lake_erie_fisheries_stock_assessment_app/>. This is a single long page, with all the different graphical interfaces visible as you scroll down. Dashboards are more difficult to code, requiring the use of the shinydashboard add-on package <https://rstudio.github.io/shinydashboard/index.html>. As this is more complex, have a look at some of the examples before deciding whether to go down this route.

## 4. Testing your prototype

It is difficult to test a prototype user-interface without having all the server components written already, which can be frustrating if you want to focus on the user-interface design initially. The shinipsum package comes with random plots, images, tables of data etc. that you can use in an interface, without needing to worry about the detail of the backend. The package is not available (yet) on CRAN but can be installed from github:

# Note: installing shinipsum will also install lots of extra packages  
remotes::install\_github("Thinkr-open/shinipsum")

You can then experiment by creating a very simple shiny app with some random components:

library(shiny)  
library(shinipsum)  
  
library(shiny)  
library(shinipsum)  
library(DT)  
ui <- fluidPage(  
 h2("A Random DT"),  
 DTOutput("data\_table"),  
 h2("A Random Image"),  
 plotOutput("image", height = "300px"),  
 h2("A Random Plot"),  
 plotOutput("plot"),  
 h2("A Random Print"),  
 verbatimTextOutput("print"),  
 h2("A Random Table"),  
 tableOutput("table"),  
 h2("A Random Text"),  
 tableOutput("text")  
)  
  
server <- function(input, output, session) {  
 output$data\_table <- DT::renderDT({  
 random\_DT(10, 5)  
 })  
 output$image <- renderImage({  
 random\_image()  
 })  
 output$plot <- renderPlot({  
 random\_ggplot()  
 })  
 output$print <- renderPrint({  
 random\_print("model")  
 })  
 output$table <- renderTable({  
 random\_table(10, 5)  
 })  
 output$text <- renderText({  
 random\_text(nwords = 50)  
 })  
}  
shinyApp(ui, server)