Shiny

An introduction

What is Shiny

- Shiny is a package from R that can be used to build interactive web pages without knowing how to code in HTML (what a bonus)

- Install packages:
  - shiny
  - shinydashboard

Basics

- Every Shiny app is composed of two parts:
  - UI – ui.R
  - Server – server.R
- UI: interactive web document that the user gets to see:
  - HTML that you write using Shiny's functions
  - layout of the app
- Server: responsible for the logic of the app
  - works under the hood and tells the web page what to show when the user interacts with the page.

Lets dive right in

- All Shiny apps follow the sample template

```
library(shiny)
library(shinydashboard)

ui <- dashboardPage(
  dashboardHeader(),
  dashboardSidebar(),
  dashboardBody()
)

server <- function(input, output) { }

shinyApp(ui, server)
```

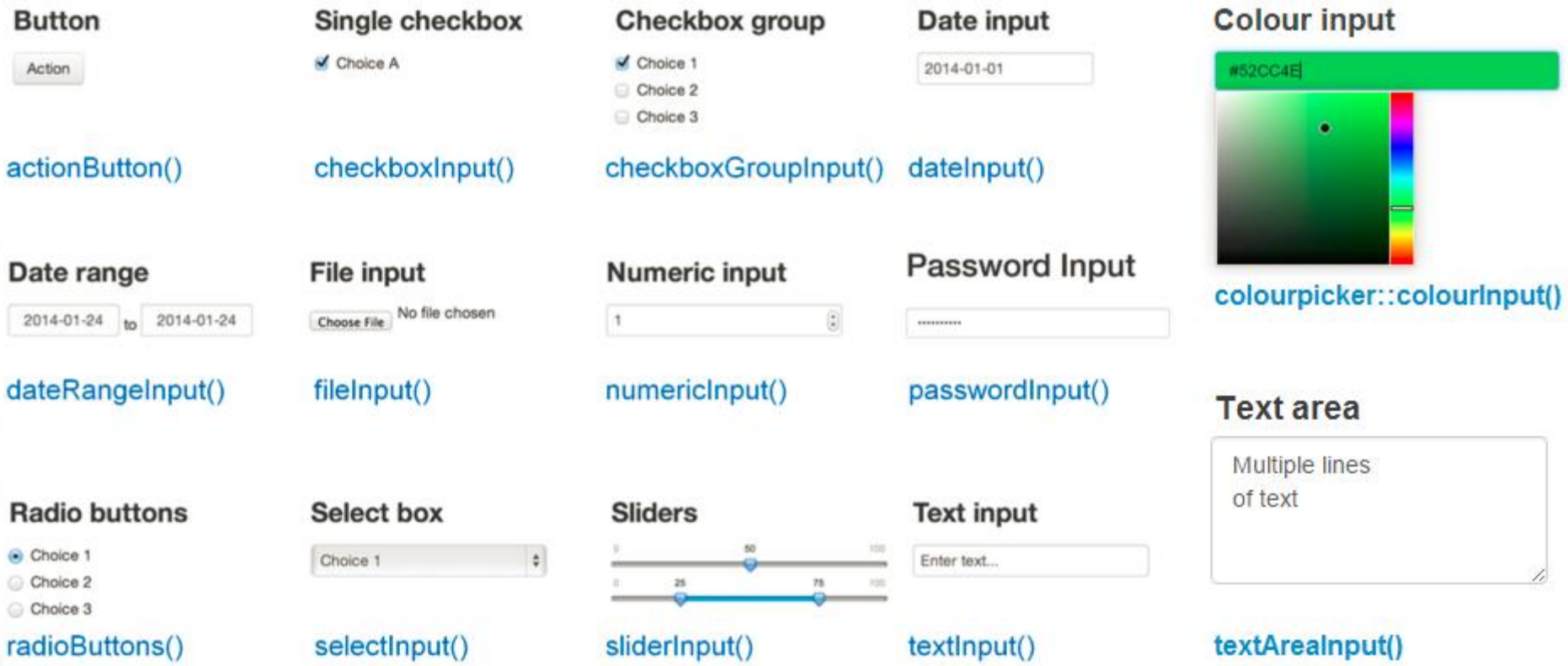
- Save as an app.R file so that RStudio recognises the program as a Shiny app
- Another way to define a Shiny app is to separate the UI and server code into two files: ui.R and server.R
- This is the preferable way when the app is complex and involves more code

Building the UI

1. Add a title

2. Add inputs to the UI
   - Inputs give users a way to interact with a Shiny app
   - Many different input functions

Building the UI



Lets add some input functions

- Shiny has many functions that are wrappers around HTML tags that format text h1(), h2() etc.
- We'll add some titles radio buttons and sliders to input some of the risk factors associated with chronic diseases
- Rows are created by the fluidRow() function and include columns defined by the column() function. Column widths are based on the Bootstrap 12-wide grid system, so should add up to 12 within a fluidRow() container.

Implement server logic to create outputs

- Now we have to write the server function, which will be responsible for listening to changes to the inputs and creating outputs to show in the app

- Notice the two arguments: *input* and *output*
  - input is a list you will read values from (user inputs at a given time)
  - output is a list you will write values to - save output objects (such as tables and plots) to display in your app.

Implement server logic to create outputs – functions to take note of

- reactive()
  - A reactive expression is an R expression that uses widget input and returns a value. The reactive expression will update this value whenever the original widget changes.
- req()
  - Ensure that values are available
- reactiveValues()
  - This function returns an object for storing reactive values
- render* (where * is the output type)
  - Renders a reactive plot that is suitable for assigning to an output slot.

How to build output

There are three rules to build an output in Shiny.

1. Access input values using the input list
2. Build the object with a render* function, where * is the type of output
3. Save the output object into the output list

Building the server function for the Disease Prediction App

- We have a saved model – created in h2o
- We want to:
  - Pass a vector of user input to the model
  - Run the model and obtain the prediction
  - Visualise the results

How to reference input values

- Remember we defined a radio button for gender in the UI

```
radioButtons("GENDER",
             label = h3("Gender"),
             choiceNames = c("Female", "Male"),
             choiceValues = c("F", "M"),
             selected = "M",
             inline = T),
```

- We can refer to this in the server function like this:

```
GENDER = input$GENDER
```

Rendering output

```
output$hypertensionPlot <- flexdashboard::renderGauge({
  val <- req(case_pred_ci_hypertension())

  flexdashboard::gauge(round(val,2),
                       min = 0, max = 1,
                       label = "Risk category",
                       flexdashboard::gaugeSectors(
                         success = c(0, 0.35), warning = c(0.351, 0.75), danger = c(0.751, 1)
                       )
  )
})
```