Website building in R markdown and Shiny

Edith Invernizzi

3rd July 2017

(Obviously) required packages:
- rmarkdown
- shiny

Build your index.Rmd document

rmarkdown.rstudio.com
Your webpage. Equivalent to HTML - but with the possibility of embedding functioning R code.

```
title: "R-group - test"
author: "Edith Invernizzi"
date: "3rd July 2017"
output: html_document
runtime: shiny
```

Hide R code in the full document:

```
knitr::opts_chunk$set(echo = FALSE)
```

Unless you wish otherwise -in which case you can override document-wide settings in single chunks:
“‘{r chunk, echo=T}

Personalised layout - as in any good html: CSS

Two things needed:
- CSS subfolder in the same folder as your Rmd document
- link to CSS document in Rmd body

```
<link rel=stylesheet type="text/css" href="CSS/webstyles.css">
```

or, in the YAML header,

```
title: "index.Rmd"
author: "Edith Invernizzi"
date: "3rd July 2017"
output:
  html_document:
    css: CSS/webstyles.css
runtime: shiny
```

Internal CSS can be used for anything in the document.

```
<div class="titleSection">
  ##Search database
<p> This is a catalogue of research data. </p>
</div>
```

Including the YAML header.

```
title: "<div style= 'background-color: LightSeaGreen; margin-left: 20px;'> <h1> index.Rmd </h1> </div>"
author: "Edith Invernizzi"
date: "3rd July 2017"
output:
  html_document:
    css: CSS/webstyles.css
runtime: shiny
```

Build your search filter in Shiny

shiny.rstudio.com

Shiny apps have two components: a server-side script -computing the calculations and the R functionalities that you want the user to perform- and a user interface script -creating the graphic interface that the user will interact with, unaware of the underlying R.

Two options:
- inline app
- external app

Inline:

```
{r searchFilter, message=F, warning=F}
 require(shiny)
 shinyApp(
   ui=fluidRow(textInput("name",label = "Car model", value="")),
   server = function(input,output) {

   },
   options = list(width="100%", height= "800px")
  )
```

External:

App folder as a subdirectory of the index.Rmd folder. The `shinyAppDir()` function refers to the subdirectory.

```
{r searchFilter, message=F, warning=F}
 require(shiny)
 shinyAppDir(
  appDir = "search",
  options=list(width="100%", height= "800px")
         )
```

NB: this option can work either with a single-script (yourApp.R) app or a two-script (ui.R and server.R) app. To call a single-script app, use the shinyAppFile function:

```
{r searchFilter, message=F, warning=F}
 require(shiny)
 shinyAppFile(
  appFile = "myApp",
  options=list(width="100%", height= "800px")
         )
```

Below, I will use a two-script app. More information on how to build a single-script app here: shiny.rstudio.com/articles/single-file.html

ui.R

Shiny widgets: shiny.rstudio.com/gallery/widget-gallery.html

```
require(shiny)

nameList=as.character(rownames(mtcars))

shinyUI <- fluidPage(

  selectInput("name",label = "Car",choices = c("Select model",nameList)),
  sliderInput("hp","Horsepower range",min =50 ,max=350,value=c(50,150)),
  actionButton("submit",label = "Submit"),

  dataTableOutput("res")
)
```

Improved UI layout:

```
require(shiny)

nameList=as.character(rownames(mtcars))

shinyUI <- fluidPage(

  sidebarPanel(
  fluidRow(selectInput("name",label = "Model",choices = c("",nameList))),
  fluidRow(sliderInput("hp","Horsepower range",min =50 ,max=350,value=c(50,150))),
  fluidRow(actionButton("submit",label = "Submit"))
  ),

  mainPanel(
    dataTableOutput("res")
    )
)
```

Possibility of fiddling with js to personalise the format, e.g. to modify the icon of `submitButton`. See: fontawesome.io/examples.

```
submitButton("submit",icon = icon("star", lib="glyphicon"))
```

submit

Make content downloadable:

```
require(shiny)

nameList=as.character(rownames(mtcars))

shinyUI <- fluidPage(

  sidebarPanel(
  fluidRow(selectInput("name",label = "Model",choices = c("",nameList))),
  fluidRow(sliderInput("hp","Horsepower range",min =50 ,max=350,value=c(50,150))),
```

```
  fluidRow(actionButton("submit",label = "Submit"))
  ),

  mainPanel(
    dataTableOutput("res"),

    conditionalPanel(
      condition="output.res",fluidRow(downloadLink("mtcars","Download table"),fluidRow(downloadLink("pr
    )
    )
)
```

server.R

```
require(shiny)

shinyServer <- function(input, outputs) {

  res=eventReactive(input$submit,{

    name=as.character(input$name)

    res=mtcars
    if (name!="") {
      res=mtcars[name,]
    }
    res=res[res$hp<maxHp & res$hp>minHp,]
    #error messages can be returned
    validate(need(nrow(res)>0, message="No matches found"))

  return(res)

  })

  output$res=renderDataTable({
    res=res()
    },options=list(hover = T, bordered = T, align="c", colnames = T, rownames = T, na="NA"))

}
```

Make content downloadable:

```
require(shiny)

shinyServer <- function(input, output, data=mtcars) {

  res=eventReactive(input$submit,{

    name=as.character(input$name)
    minHp=input$hp[1]
    maxHp=input$hp[2]

    res=mtcars
```

```
    if (name!="") {
      res=mtcars[name,]
    }
    res=res[res$hp<maxHp & res$hp>minHp,]

    validate(need(nrow(res)>0, "No matches found"))

  return(res)

  })

  output$res=renderDataTable({
    res=res()
    },options=list(hover = T, bordered = T, align="c", colnames = T, rownames = T, na="NA"))

  output$mtcars=downloadHandler(filename="data_table.csv",
                                      content = function(file) {
                                       write.csv(res,file,row.names = F)
                                      }, contentType = "text/csv")

  output$priceList=downloadHandler(filename = "cars_prices.csv",
                                   content=function(file){
                                     file.copy("price_list.csv")
                                     },contentType = "text/csv")

}
```

Javascript in Shiny app

js scripts can be added to the app, to add extra functionalities. Example: read the Enter key as the Action Button. The script is part of the HTML page - hence, it goes in ui.R, and it is part of the document's head. The js script can be internal or external.

Internal

```
shinyUI <- fluidPage(

  enterButton.js="your js script",

  sidebarPanel(
    tags$head(tags$script(HTML(enterButton.js))),
    fluidRow(selectInput("name",label = "Model",choices = c("",nameList))),
    fluidRow(sliderInput("hp","Horsepower range",min =50 ,max=350,value=c(50,150))),
    fluidRow(actionButton("submit",label = "Submit"))
  ),

  mainPanel(
    dataTableOutput("res"),

    conditionalPanel(
      condition="output.res",fluidRow(downloadLink("mtcars","Download table"),fluidRow(downloadLink("pri
    )
```

```
    )
)
```

External

Shiny by default looks for the script in the www subdirectory, which does not need to be indexed in the path.

```
shinyUI <- fluidPage(

  sidebarPanel(
    tags$head(tags$script(src="enterButton.js")),
    fluidRow(selectInput("name",label = "Model",choices = c("",nameList))),
    fluidRow(sliderInput("hp","Horsepower range",min =50 ,max=350,value=c(50,150))),
    fluidRow(actionButton("submit",label = "Submit"))
  ),

  mainPanel(
    dataTableOutput("res"),

    conditionalPanel(
      condition="output.res",fluidRow(downloadLink("resTable","Download table"),fluidRow(downloadLink("
    )
    )
)
```

Deploy the website on server

Needs both R and Shiny server (www.rstudio.com/products/shiny/download-server) installed on the host server for the website to be built from rmarkdown.

The index.Rmd file stays in .Rmd format (no pre-knitting).

On Shiny server:
website folder:
–> index.Rmd
–> App folder (if the app is external): -server.R -ui.R -www folder (HTMLs, js, downloadable files)

Downloadable files on Shiny server

Call `addResourcePath("myDirectory","path/to/myDirectory")` sometime during app startup (i.e. in global.R or at the top of ui.R, server.R or app.R) to expose the directory containing the files publicly.