Cours de Shiny
Création d'applications interactives avec R

François

28 avril 2015

1 Détails techniques

L'application doit être rangée dans un dossier. Dans ce dossier il faut avoir :

- ui.R le fichier de définition de l'interface. Il début avec du code qui ressemble à :

```
shinyUI(fluidPage(
  titlePanel("My Shiny App"),
  sidebarLayout(
    sidebarPanel(
      h2("Installation"),
```

- server.R le fichier qui execute le code

On lance ensuite l'application en utilisant :

```
library("shiny")
runApp("FolderName")
```

2 Widgets

Pour créer des widgets, qui permettent à l'utilisateur de rentrer des informations, choisir des dates, etc... il faut utiliser au sein d'un appel à une fonction de type `*Panel` des appels à des fonctions de type `*Input` . Par exemple :

```
shinyUI(fluidPage(
        titlePanel("census Vis"),
        sidebarPanel(p("Create demographic maps with information from the 2010 US Census"),
                     selectInput("variable", label = h3("Choose a variable to display"),
                                 choices = list( "Percent White" = 1,"Percent Black" =2,
                                                 "Percent Hispanic" = 3, "Percent Asian"=4),
                     br(),
                     sliderInput("range",label="range of interest", min=0,max=100,value=c(0,
                     ),
```

```
fluidRow(

        column(3,
               h3("Buttons"),
               actionButton("action", label = "Action"),
               br(),
               br(),
               submitButton("Submit")),

        column(3,
               h3("Single checkbox"),
               checkboxInput("checkbox", label = "Choice A", value = TRUE)),

        column(3,
               checkboxGroupInput("checkGroup",
                                  label = h3("Checkbox group"),
                                  choices = list("Choice 1" = 1,
                                                 "Choice 2" = 2, "Choice 3" = 3),
                                  selected = 1)),

        column(3,
               dateInput("date",
                         label = h3("Date input"),
                         value = "2014-01-01"))
),

fluidRow(

        column(3,
               dateRangeInput("dates", label = h3("Date range"))),

        column(3,
               fileInput("file", label = h3("File input"))),

        column(3,
               h3("Help text"),
               helpText("Note: help text isn't a true widget,",
                        "but it provides an easy way to add text to",
                        "accompany other widgets.")),

        column(3,
               numericInput("num",
                            label = h3("Numeric input"),
                            value = 1))
),
```

```
        fluidRow(

                column(10,
                       radioButtons("radio", label = h3("Radio buttons"),
                                    choices = list("Choice 1" = 1, "Choice 2" = 2,
                                                   "Choice 3" = 3),selected = 1)),

                column(3,
                       selectInput("select", label = h3("Select box"),
                                   choices = list("Choice 1" = 1, "Choice 2" = 2,
                                                  "Choice 3" = 3), selected = 1)),

                column(3,
                       sliderInput("slider1", label = h3("Sliders"),
                                   min = 0, max = 100, value = 50),
                       sliderInput("slider2", "",
                                   min = 0, max = 100, value = c(25, 75))
                ),

                column(3,
                       textInput("text", label = h3("Text input"),
                                 value = "Enter text..."))
        )

))
```

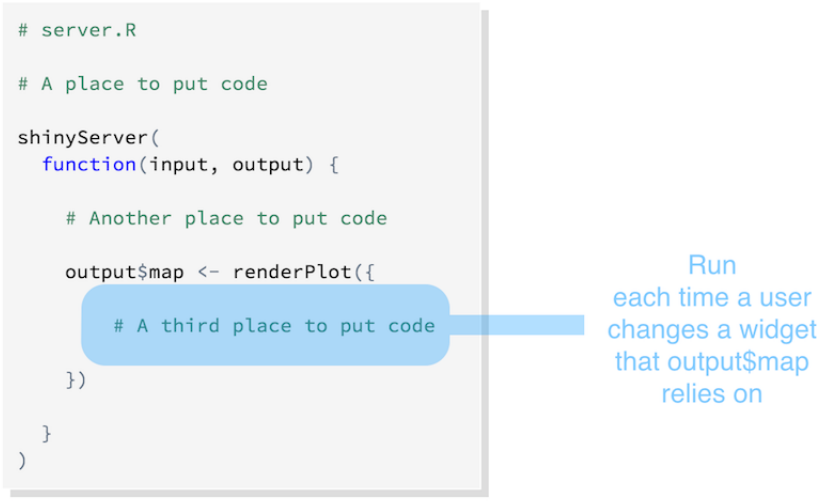
3 Objects intéractifs

Pour créer des object interactifs :

- Utiliser les fonctions `*Output` dans `ui.R` pour placer des object réactifs dans l'application :
  ```
  mainPanel(
        textOutput("text1"),
        textOutput("text2")
  )
  ```
- Utiliser une fonction `render*` dans `server.r` pour dire comment construire l'object :
  ```
  output$text2 <- renderText({
        paste("You have chosen a range that goes from",
          input$range[1], "to", input$range[2])
  })
  ```
- Entourer les expressions par des accolades `{}` dans chaque fonction `render*`
- Assigner chaque appel de fonction `render*` à un élément de la liste `output`. Chaque élément de cette liste doit correspondre à un object réactif crée

dans `ui.R`
- Rendre le tout réactif en utilisant un élément de la liste `input` dans l'expression entre accolades

Il y a trois endroits où mettre du code, toujours dans le fichier `server.R` :



1. Le premier endroit n'est exécuté qu'au lancement de l'application, une seule fois.
2. Le deuxième est exécuté à chaque arrivée d'un nouvel utilisateur (?)
3. Le troisième est exécuté à chaque fois que l'utilisateur modifie un widget, soit, très, très, souvent.

L'outil `switch` est cool :

```
data <- switch(input$var,
        "Percent White" = counties$white,
        "Percent Black" = counties$black,
        "Percent Hispanic" = counties$hispanic,
        "Percent Asian" = counties$asian)
```

Permet de faire un «mappage» entre différents éléments. On peut également appeler une fonction avec une liste d'argument en utilisant `do.call` ce qui permet de créer un code plus compact.

```
library(maps)
library(mapproj)
counties <- readRDS("data/counties.rds")
source("helpers.R")
```

```
shinyServer(
  function(input, output) {
    output$map <- renderPlot({
      args <- switch(input$var,
        "Percent White" = list(counties$white, "darkgreen", "% White"),
        "Percent Black" = list(counties$black, "black", "% Black"),
        "Percent Hispanic" = list(counties$hispanic, "darkorange", "% Hispanic"),
        "Percent Asian" = list(counties$asian, "darkviolet", "% Asian"))

      args$min <- input$range[1]
      args$max <- input$range[2]

      do.call(percent_map, args)
    })
  }
)
```