Shiny

Rindra LUTZ / William ROBACHE

16/12/2020

Introduction

Envie de partager votre projet R ? Bonne nouvelle : l'équipe RStudio a développé Shiny, un package qui vous permet de créer des applications dynamiques pour le web… partagez vos projets avec le monde ! Le package est facile d'accès dans sa version de base et vous pouvez créer des applications Web sans faire appel à du HTML ou du CSS.

Shiny, comment ça marche ?

Tout commence par l'installation (veuillez noter que le package est pré-installé dans RStudio) : après le chargement de la bibliothèque, vous diviserez l'application Shiny en deux éléments, une partie "ui" (utilisée pour l'interface utilisateur, c'est à dire tout ce qui sera affiché à l'utilisateur) et une partie "serveur" (qui contiendra le contenu de toutes les commandes). Le côté serveur de R contiendra les éexcutions. Bon, c'est bien beau tout ça, mais concrètement, comment ça se passe ? Tout commence par l'installation (à noter que le package est préinstallé dans RStudio) :

```
install.packages("shiny")
library(shiny)
```

Commençons par la partie "ui". Cela fait partie du code qui contiendra les éléments graphiques de l'interface de l'application Shiny, tels que les éléments de mise en page (titre, nombre et taille des éléments, boutons et curseurs, etc.). Mais surtout, cette partie contiendra les composants de l'application, c'est à dire les objets qui communiquent avec la partie serveur pour afficher ces éléments (tables, dataviz, code…) dans une interface graphique.

Important

Afin de tester le code dans son intégralité, il est nécessaire de le lancer par petit bout, et non pas de Knit le fichier .rmd

UI

On ne vous le répétera jamais assez : rien de mieux qu'un exemple pour bien comprendre ! Alors, pour découvrir Shiny, nous allons travailler ensemble sur la création d'une application (très simple) qui vous permet de voir les X premières lignes d'un des jeux de données proposés, X étant un chiffre entré par l'utilisateur. Commençons par UI :

Rien de mieux qu'un exemple pour bien comprendre! Ainsi, pour découvrir Shiny, nous allons créer ensemble une application (très simple) qui vous permet de voir les X premières lignes d'un ensemble de données fournies, où X est le nombre saisi par l'utilisateur. Commençons par l'interface utilisateur :

```
 # Définition de l'interface utilisateur de notre application
  ui <- shinyUI(fluidPage(

    # Le titre de votre application
    titlePanel("Aperçu d'un dataset"),

    #Indiquer le 'layout' de votre application : autrement dit le squelette
visuel de l'application
    sidebarLayout(
      #Composants de la région gauche de l'application
      sidebarPanel(
        #Ici, nous insérons un champ pour entrer un chiffre, ainsi qu'un menu
déroulant
        textInput(inputId = "lignes",
                  label = "Combien de lignes voulez-vous voir ? ",
                  value = 10),
        selectInput(inputId = "labs",
                    label = "Dataset",
                    choice = c("cars","rock","beaver1", "sleep")
        )
      ),

      #Ici, nous indiquons l'élement qui sera présent dans la fenêtre
principale : l'element "dataset", qui est un graph
      mainPanel(
        tableOutput("dataset")
      )
    )))
```

Server

Regardons maintenant la partie serveur. Ici, vous entrerez tous les codes R, qui s'exécuteront avant tout affichage dans votre application. En d'autres termes, c'est lui qui crée les éléments qui seront affichés dans l'interface. C'est là que les composantes sont appelées.

```
server <- shinyServer(function(input, output) {
  #On retrouve ici l'élement "dataset", qui communique avec ui par 'output'.
  output$dataset <- renderTable({
    #Nous imprimons les éléments d'après les données en entrée : ces derniers
sont appellés avec `input` puis le nom de la composante de ui (ici 'labs' et
'lignes')
    if(input$labs == "cars"){
      print(head(cars, input$lignes))
    } else if(input$labs == "rock"){
```

```
      print(head(rock, input$lignes))
    } else if(input$labs == "sleep"){
      print(head(sleep, input$lignes))
    } else {
      print(head(beaver1, input$lignes))
    }
  })
})
```

Enfin, pour voir votre application en local, c'est très simple :

```
shinyApp(ui = ui, server = server)
```

Cette ligne de commande permet de lancer l'interface Shiny en spécifiant quelle UI et quel server nous voulons utiliser. Il est en effet possible qu'il y ait plusieurs serveurs pour une seule interface générale et que l'on appelle différentes fois cette ligne de code pour afficher le serveur désiré.

Une fenêtre s'ouvre, et vous pouvez désormais admirer votre application !

Shiny, un peu plus loin

Comme vous pouvez l'imaginer, on peut aller un peu plus loin que cette première application, notamment en termes de design (mais pour cela, vous devrez peut-être faire des efforts en HTML et/ou CSS). Vous pouvez également charger des thèmes prédéfinis, tels que la création de tableaux de bord. Bien entendu, en plus de la visualisation locale, l'application peut également être déployée sur le Web : via https://www.shinyapps.io/ ou en installant votre propre serveur - la solution choisie dans le contexte professionnel mais c'est une autre histoire.

Complément utiles

Exactement comme pour HTML et d'autres langages Web, Shiny fonctionne via des balises qui vont permettrent d'imbriquer des éléments les uns dans les autres, tels que dashboardPage(), dashboardHeader(), dashboardSidebar(). Ces balises servent à créer respectivement une page, un bandeau d'en-tête et une barre latérale.

Il faut également faire attention aux parenthèses qui seront à l'origine des imbrications.

Voici d'autres balises : sidebarMenu(),dashboardBody(), tabItems()

Encore une fois ces balises vont servir à créer des objets biens spécifiques dans l'interface.
Sidebar menu va permettre de configurer la barre latérale.
Le Dashboard est lui comme une page blanche sur l'interface. Au lieu d'écrire dessus, on va afficher des graphes dessus.
Le tabItems va lui nous servir a créer plusieurs items différents dans chaque page blanche de notre interface. Chaque Item renseignera l'onglet de la sidebar auquel il veut sera lié.

Il suffit ensuite de renseigner les graphes, les inputs s'il y en a et les dimensions dans chaque Item pour qu'ils puissent s'afficher correctement au moment du Run.