Cours sur Shiny et Shinydashboard

Nolan

November 8, 2024

Introduction à Shiny

Shiny est un package R permettant de créer des applications web interactives et des tableaux de bord dynamiques. Ces applications permettent de visualiser des données en temps réel et d'interagir avec les éléments de l'interface. Contrairement aux rapports statiques, les tableaux de bord Shiny offrent une expérience utilisateur enrichie en proposant des filtres, des graphiques et des indicateurs mis à jour en direct.

1 Structure d'une Application Shiny

Une application Shiny est structurée autour de deux composantes principales :

- **UI (User Interface)** : L'interface utilisateur, définie avec `fluidPage`, contient les éléments visibles par l'utilisateur, comme les graphiques, les menus déroulants et les boutons.
- **Server** : La logique de l'application, définie dans une fonction `server`, récupère les valeurs d'entrée (inputs) et produit les sorties (outputs).

L'interface et le serveur sont assemblés avec la fonction `shinyApp(ui, server)` pour lancer l'application.

1.1 Définir l'UI dans Shiny

La structure de base d'une application Shiny commence par la définition de l'UI et du serveur. Dans l'UI, chaque composant est ajouté via des fonctions spécifiques, comme `titlePanel` pour afficher un titre, `textOutput` pour du texte, et `plotOutput` pour des graphiques.

```
library(shiny)
ui <- fluidPage(
    titlePanel("Titre de l'application"),
    sidebarLayout(
        sidebarPanel(
```

```
            selectInput("variable", "Choisissez une variable :", choices = c("mpg", "hp")),
            sliderInput("bins", "Nombre de bins :", min = 1, max = 50, value = 30)
        ),
        mainPanel(
            plotOutput("histPlot")
        )
    )
)
```

1.2 Définir le Serveur dans Shiny

Dans la fonction `server`, les sorties sont générées en fonction des entrées de l'utilisateur. Les objets de sortie sont créés via des fonctions de rendu telles que `renderPlot` et `renderText`, placées dans le serveur.

```
server <- function(input, output) {
    output$histPlot <- renderPlot({
        hist(mtcars[[input$variable]], breaks = input$bins, col = "darkgray", border = "whit
    })
}
```

2 Les Entrées (Inputs) et Sorties (Outputs) dans Shiny

Shiny propose une variété d'éléments d'entrée pour recueillir des informations auprès des utilisateurs et des éléments de sortie pour afficher des résultats en fonction de ces informations.

2.1 Types d'Entrées (Inputs)

Les types d'entrées permettent aux utilisateurs de personnaliser les données qu'ils visualisent.

- `selectInput` : Crée un menu déroulant pour choisir une option parmi plusieurs.
- `sliderInput` : Crée un curseur pour sélectionner une valeur numérique dans une plage.
- `textInput` : Crée un champ pour saisir du texte.
- `checkboxInput` et `checkboxGroupInput` : Fournit des cases à cocher pour les sélections multiples.

2.2 Types de Sorties (Outputs)

Les sorties permettent d'afficher des résultats à partir des entrées de l'utilisateur.

- `textOutput` : Affiche des chaînes de texte dynamiquement mises à jour.
- `plotOutput` : Affiche des graphiques, générés dans le serveur avec `renderPlot`.
- `tableOutput` : Affiche des tables de données.

2.3 Fonctions de Rendu pour les Sorties

Chaque type de sortie nécessite une fonction de rendu spécifique dans le serveur :

- `renderText` : Utilisé pour le texte.
- `renderPlot` : Utilisé pour les graphiques.
- `renderTable` : Utilisé pour les tables.

3 Shinydashboard : Créer des Tableaux de Bord Interactifs

`shinydashboard` est un package R complémentaire de Shiny, conçu pour créer des tableaux de bord organisés en plusieurs sections. Il introduit des composants comme le `dashboardHeader`, `dashboardSidebar` et `dashboardBody` pour organiser l'interface utilisateur.

3.1 Structure de base d'un Shinydashboard

Un tableau de bord Shiny est structuré autour de trois éléments principaux :

- `dashboardHeader` : L'en-tête, où l'on place le titre et les menus déroulants pour les notifications.
- `dashboardSidebar` : La barre latérale pour naviguer entre différentes sections, créée avec `sidebarMenu` et `menuItem`.
- `dashboardBody` : Le corps, qui contient les éléments interactifs principaux, organisés avec `fluidRow` et `box`.

3.2 Exemple de Code Shinydashboard

```
library(shiny)
library(shinydashboard)

ui <- dashboardPage(
    dashboardHeader(title = "Tableau de bord"),
```

```
    dashboardSidebar(
        sidebarMenu(
            menuItem("Accueil", tabName = "home", icon = icon("home")),
            menuItem("Graphiques", tabName = "charts", icon = icon("chart-bar")),
            menuItem("Statistiques", tabName = "stats", icon = icon("file"))
        )
    ),
    dashboardBody(
        fluidRow(
            box(title = "Graphique", status = "primary", plotOutput("plot1"), width = 6),
            box(title = "Statistiques", status = "warning", verbatimTextOutput("stats"), wid
        )
    )
)
```

4 Les Composants Interactifs dans Shinydashboard

`shinydashboard` permet également l'ajout de composants interactifs pour enrichir l'expérience utilisateur, comme les `valueBox` et `infoBox`.

4.1 Les `valueBox` et `infoBox`

Ces composants affichent des informations synthétiques en nombre ou en texte pour attirer l'attention de l'utilisateur.

- `valueBox` : Affiche une valeur clé comme une statistique importante.
- `infoBox` : Affiche des informations supplémentaires, avec des options de couleur et d'icône.

4.2 Exemple d'Utilisation de `valueBox` et `infoBox`

```
body <- dashboardBody(
    fluidRow(
        valueBox(value = 3, subtitle = "Total des cartons rouges", icon = icon("flag"), colo
        infoBox(value = 158, title = "Total des buts marqués", icon = icon("futbol"), color
    )
)
```

5 Fonctions de Rendu dans le Serveur pour les Composants Dynamiques

Les composants dynamiques comme `valueBoxOutput` et `infoBoxOutput` permettent des mises à jour en temps réel. Ils utilisent les fonctions `renderValueBox`

et `renderInfoBox`.

```
output$valuebox1 <- renderValueBox({
    valueBox(value = 3, subtitle = "Cartons rouges", icon = icon("flag"), color = "red")
})
```

6 Création d'un Tableau de Bord Complet

Pour créer un tableau de bord complet, il est nécessaire de combiner les éléments d'UI, les sorties de type `valueBox` et `plotOutput`, et les rendus de serveur.

```
ui <- dashboardPage(
    dashboardHeader(title = "Tableau de bord complet"),
    dashboardSidebar(
        sidebarMenu(
            menuItem("Accueil", tabName = "home", icon = icon("home")),
            menuItem("Données", tabName = "data", icon = icon("table")),
            menuItem("Graphiques", tabName = "charts", icon = icon("chart-bar"))
        )
    ),
    dashboardBody(
        fluidRow(
            valueBoxOutput("valuebox1"),
            plotOutput("mainPlot")
        )
    )
)

server <- function(input, output) {
    output$valuebox1 <- renderValueBox({
        valueBox(value = 3, subtitle = "Cartons rouges", icon = icon("flag"), color = "red")
    })
    output$mainPlot <- renderPlot({
        hist(mtcars$mpg, col = "blue")
    })
}

shinyApp(ui, server)
```

7 Conclusion

Shiny et Shinydashboard sont dzdz des outils puissants pour créer des tableaux de bord dynamiques et interactifs en R. Ils permettent de visualiser des données de manière engageante et d'intégrer une large gamme de composants pour

répondre aux besoins spécifiques de chaque utilisateur. Grâce à leur modularité, ces applications sont adaptées aux projets de data science, d'analyse de données, et de suivi en temps réel.