# Création de tableaux de bord avec R Shiny

Tamsir NDONG, Jean-Luc BATABATI, Ange Rayan RAHERINASOLO

ENSAE Pierre NDIAYE - ANSD Sénégal

2025-04-17

# Plan

- Introduction
- Chargement des packages
- Généralités sur R Shiny
- Structure d'une application
- Interface & Widgets
- Outputs et render
- Réactivité
- Apparence et Thèmes
- Shinydashboard
- Publication
- Conclusion

# Introduction

- À l'ère du numérique, la visualisation interactive des données est cruciale pour la prise de décision.
- R Shiny permet de créer des applications web interactives sans HTML/CSS/JS.
- Objectif : présenter les fondamentaux de Shiny, ses avantages et ses bonnes pratiques.

# Chargement des packages

- tidyverse : manipulation et visualisation de données
- shiny : création d'applications interactives
- bslib : personnalisation de l'apparence (Bootstrap)
- shinydashboard : création de tableaux de bord
- datasets : jeux de données intégrés
- DT : affichage de tableaux interactifs
- rsconnect : publication d'applications sur shinyapps.io

```
packages <- c("tidyverse", "shiny", "bslib", "shinydashboard",
for (package in packages) {
  if (!requireNamespace(package, quietly = TRUE)) {
    install.packages(package)
  }
  library(package, character.only = TRUE)
}
```

# Généralités sur R Shiny

- Une application Shiny contient :
  - Une interface utilisateur (UI)
  - Une logique de traitement (serveur)
- Structure recommandée :
  - ui.R, server.R ou un fichier unique app.R

# Structure d'une application

- UI : organise les éléments interactifs (inputs, outputs)
- Serveur : exécute les traitements et rend les outputs
- Exemple d'app minimale avec `fluidPage`, `sidebarLayout`, `renderPlot`

```
ui <- fluidPage(...)
server <- function(input, output) {...}
shinyApp(ui = ui, server = server)
```

# Interface utilisateur (UI)

- Layouts :
  - `fluidPage`, `fixedPage`, `sidebarLayout`, `tabsetPanel`
- Widgets :
  - `textInput`, `sliderInput`, `selectInput`, etc.

```
ui <- fluidPage(
  titlePanel("Application Shiny"),
  sidebarLayout(
    sidebarPanel(...),
    mainPanel(...)
  )
)
```

# Outputs et render

- Types d'outputs : `textOutput`, `plotOutput`, `DTOutput`, `verbatimTextOutput`
- À associer avec `renderText`, `renderPlot`, `renderDT`, etc.
- L'outputId doit être unique.

```
output$hist <- renderPlot({
  hist(iris[[input$var]])
})
```

# Réactivité

- Utiliser `reactive()` pour créer des objets dynamiques
- `observeEvent()` pour des actions déclenchées par événement
- `isolate()` pour ignorer temporairement la réactivité

```
df <- reactive({ iris })
output$iris_data <- renderDT({ df() })
```

# Apparence et Thèmes

- `shinythemes` : thèmes prédéfinis via Bootswatch
- Fichier CSS personnalisé possible via `theme = "style.css"`
- Autres outils : `bslib`, `shinydashboard`

```
ui <- fluidPage(theme = shinytheme("superhero"), ...)
```

# Shinydashboard

- Structure :
  - `dashboardPage(header, sidebar, body)`
- Éléments :
  - `menuItem`, `tabItem`, `box`, `valueBox`, `infoBox`

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

# Exemple complet dashboard

```
ui <- dashboardPage(
  dashboardHeader(title = "Dashboard"),
  dashboardSidebar(
    sidebarMenu(menuItem("Histogramme", tabName = "hist"))
  ),
  dashboardBody(
    tabItems(
      tabItem(tabName = "hist",
        fluidRow(
          box(plotOutput("histogramme")),
          valueBoxOutput("moyenne")
        )
      )
    )
  )
)
```

# Publication

- Service recommandé : shinyapps.io
- Étapes :
  1. Créer un compte
  2. Installer `rsconnect`
  3. Lier le compte avec `setAccountInfo`
  4. Publier avec `deployApp`

```
rsconnect::setAccountInfo(name='nom', token='token', secret='s
rsconnect::deployApp("chemin/app")
```

# Conclusion

- Shiny est un outil puissant pour créer des visualisations interactives
- Permet de diffuser facilement des résultats analytiques
- La personnalisation avec shinydashboard, shinythemes, bslib le rend très flexible

# Merci de votre attention !