Kampala dashboard

2024-08-14

Summary

This script builds a fully interactive dashboard for visualizing Antibiotic Resistance data, with multiple features: - **Species Filtering**: Dynamic data filtering based on species selection. - **Interactive Visualizations**: Bar charts, pie charts, and maps. - **Data Exploration**: Display and download of filtered data.

It's structured using `shiny` and `shinydashboard` to ensure a clean, organized interface, with the use of libraries like `plotly` and `leaflet` to provide interactive and visually appealing data representations.

1. Install Required Packages

```r
install.packages("shiny")
install.packages("shinydashboard")
install.packages("leaflet")
install.packages("plotly")
install.packages("tidyverse")
install.packages("DT")
install.packages("RColorBrewer")
```

This section install the required R packages:

2. Loading Required Libraries

```r
library(shiny)
library(shinydashboard)
library(leaflet)
library(plotly)
library(tidyverse)
library(DT)
library(RColorBrewer)
```

This section loads the required R libraries: - `shiny`: For building interactive web applications. - `shinydashboard`: For creating a dashboard layout. - `leaflet`: For interactive maps. - `plotly`: For interactive plots. - `tidyverse`: For data manipulation (particularly `dplyr` and `ggplot2`). - `DT`: For rendering data tables. - `RColorBrewer`: For generating color palettes.

3. Loading Data

```
data <- read_csv("Kampala_data.csv")
```

This line loads the AMR data from a CSV file called `Kampala_data.csv`. This dataset is used for all the visualizations in the dashboard.

4. User Interface (UI)

The UI defines how the dashboard will look and how the user will interact with it. It uses `shinydashboard` components such as `dashboardPage`, `dashboardHeader`, `dashboardSidebar`, and `dashboardBody`.

```
dashboardHeader(title = "AMR Dashboard")
dashboardSidebar(
  sidebarMenu(
    menuItem("Dashboard", tabName = "dashboard", icon = icon("dashboard")),
    selectInput("selected_species", "Select Species:", choices = unique(data$Species)),
    menuItem("Download Results", tabName = "isolate_table", icon = icon("table"))
  )
)
```

Header and Sidebar

- `dashboardHeader`: Sets the title of the dashboard.
- `dashboardSidebar`: Adds a sidebar with two menu items:
  - **Dashboard**: The main tab for visualizing AMR data.
  - **Download Results**: Provides access to a table and a download button.
- `selectInput`: Adds a dropdown menu to select a species. The unique species names are pulled from the data.

Dashboard Body The body of the dashboard is divided into different sections, each with various visualizations and data displays.

```
dashboardBody(
  tabItems(
    tabItem(tabName = "dashboard",
      fluidRow(
        box(title = "An Interactive Dashboard...", status = "info", solidHeader = TRUE, width = 14, ...
      ),
      fluidRow(
        box(title = "Source Distribution", plotlyOutput("source_plot")),
        box(title = "Geographical Distribution by Region", leafletOutput("map"))
      ),
      fluidRow(
        box(title = "Samples by Year", plotlyOutput("year_plot")),
        box(title = "Gender Distribution", plotlyOutput("gender_plot"))
      ),
      fluidRow(
        box(title = "Amikacin Resistance", plotlyOutput("amikacin_plot")),
        box(title = "Ampicillin Resistance", plotlyOutput("ampicillin_plot")),
        box(title = "Erythromycin Resistance", plotlyOutput("erythromycin_plot")),
        box(title = "Doripenem Resistance", plotlyOutput("doripenem_plot"))
      )
    ),
    ...
  )
)
```

- **`fluidRow` and `box`**: Used to structure the layout into rows and boxes, containing different visualizations:
  - **Source Distribution**: Bar chart showing the number of samples per source.
  - **Geographical Distribution**: An interactive map showing sample distribution by region.
  - **Samples by Year**: Bar chart showing the number of samples per year.
  - **Gender Distribution**: Pie chart displaying gender distribution.
  - **Resistance plots**: Bar charts for different antibiotics (Amikacin, Ampicillin, Erythromycin, and Doripenem), with colors representing resistance/susceptibility.

5. Server Logic

The `server` function defines the logic behind the dashboard, generating dynamic content based on user inputs.

```
filtered_data <- reactive({
  req(input$selected_species)
  data %>% filter(Species == input$selected_species)
})
```

Reactive Data Filtering

- **Reactive Expressions**: The `filtered_data` function is a reactive expression that filters the dataset based on the selected species from the dropdown menu. This ensures that the plots and tables are updated dynamically whenever the species changes.

```
output$source_plot <- renderPlotly({
  source_count <- filtered_data() %>% count(Source)
  plot_ly(source_count, x = ~Source, y = ~n, type = 'bar')
})
```

Source Distribution Plot

- This renders a `plotly` bar chart showing the count of samples per source, based on the filtered species.

```
output$map <- renderLeaflet({
  region_data <- filtered_data() %>%
    group_by(Region) %>%
    summarise(x = mean(x, na.rm = TRUE), y = mean(y, na.rm = TRUE), count = n())

  color_palette <- colorFactor(palette = brewer.pal(n = length(unique(region_data$Region)), name = "Set3

  leaflet(region_data) %>%
    addTiles() %>%
    addCircles(lng = ~x, lat = ~y, radius = 700, popup = ~paste(Region, "<br>", "Count:", count), color
})
```

Geographical Distribution by Region

- This generates an interactive `leaflet` map, with circles showing sample counts per region. The circles are color-coded using the `RColorBrewer` palette.

Other Plots (Year, Gender, Resistance)

- **`year_plot`**: A bar chart that counts the number of samples by year.
- **`gender_plot`**: A pie chart showing gender distribution.
- **Resistance Plots**: Bar charts for different antibiotics (Amikacin, Ampicillin, Erythromycin, Doripenem).

6. Data Table and Download

```
output$data_table <- DT::renderDataTable({
  DT::datatable(data)
})

output$download_data <- downloadHandler(
  filename = function() {
    paste("filtered_data_", Sys.Date(), ".csv", sep = "")
  },
  content = function(file) {
    write.csv(filtered_data(), file, row.names = FALSE)
  }
)
```

- **Data Table**: Renders a searchable, sortable data table using `DT`.
- **Download Handler**: Allows users to download the filtered dataset as a CSV file, customized with the current date.
