R Shinny APP

HITHESRAJA NAVARETHINAM

2025-04-12

# Allow large file uploads (up to 100MB)

options(shiny.maxRequestSize = 100 \* 1024^2)  
  
library(shiny)  
library(dplyr)

##   
## Attaching package: 'dplyr'

## The following objects are masked from 'package:stats':  
##   
## filter, lag

## The following objects are masked from 'package:base':  
##   
## intersect, setdiff, setequal, union

library(ggplot2)  
library(lubridate)

## Warning: package 'lubridate' was built under R version 4.4.3

##   
## Attaching package: 'lubridate'

## The following objects are masked from 'package:base':  
##   
## date, intersect, setdiff, union

library(gridExtra)

## Warning: package 'gridExtra' was built under R version 4.4.3

##   
## Attaching package: 'gridExtra'

## The following object is masked from 'package:dplyr':  
##   
## combine

ui <- fluidPage(  
 titlePanel("IMPACT OF WEATHER ON TRAFFIC ACCIDENTS"),  
 sidebarLayout(  
 sidebarPanel(  
 fileInput("accident\_file", "Upload Accident Data CSV", accept = ".csv"),  
 fileInput("weather\_file", "Upload Weather Data CSV", accept = ".csv"), # Still allows weather file upload  
 actionButton("run\_btn", "Run Analysis", icon = icon("play")),  
 selectInput("season", "Select Season:", choices = NULL),  
 selectInput("time\_of\_day", "Select Time of Day:", choices = NULL),  
 width = 3  
 ),  
 mainPanel(  
 tabsetPanel(  
 tabPanel("Summary", verbatimTextOutput("summary\_out")),  
 tabPanel("Time Visualizations", plotOutput("plot1"), plotOutput("plot2")),  
 tabPanel("Seasonal Accidents", plotOutput("accident\_season\_plot")),  
 tabPanel("Time of Day Accidents", plotOutput("accident\_tod\_plot"))  
 )  
 )  
 )  
)  
  
server <- function(input, output, session) {  
 observeEvent(input$run\_btn, {  
 req(input$accident\_file)  
   
 # Load accident data  
 accidents\_data <- read.csv(input$accident\_file$datapath, stringsAsFactors = FALSE)  
   
 # Load weather data (but won't use it in the analysis)  
 if (!is.null(input$weather\_file)) {  
 weather\_data <- read.csv(input$weather\_file$datapath, stringsAsFactors = FALSE)  
 }  
   
 # Date/time processing function  
 process\_datetime <- function(data, start\_col, end\_col) {  
 data$start\_time <- as.POSIXct(data[[start\_col]], format="%m/%d/%Y %H:%M", tz="UTC")  
 data$end\_time <- as.POSIXct(data[[end\_col]], format="%m/%d/%Y %H:%M", tz="UTC")  
 data$date <- as.Date(data$start\_time)  
 data$hour <- hour(data$start\_time)  
 data$day\_of\_week <- wday(data$start\_time, label=TRUE)  
 data$month\_num <- month(data$start\_time)  
 data$month <- month(data$start\_time, label=TRUE)  
 data$year <- year(data$start\_time)  
 data$season <- case\_when(  
 data$month\_num %in% c(12, 1, 2) ~ "Winter",  
 data$month\_num %in% c(3, 4, 5) ~ "Spring",  
 data$month\_num %in% c(6, 7, 8) ~ "Summer",  
 data$month\_num %in% c(9, 10, 11) ~ "Fall",  
 TRUE ~ NA\_character\_  
 )  
 data$time\_of\_day <- case\_when(  
 data$hour >= 5 & data$hour < 12 ~ "Morning",  
 data$hour >= 12 & data$hour < 17 ~ "Afternoon",  
 data$hour >= 17 & data$hour < 21 ~ "Evening",  
 TRUE ~ "Night"  
 )  
 return(data)  
 }  
   
 # Process the data  
 accidents\_data <- process\_datetime(accidents\_data, "StartTime.UTC.", "EndTime.UTC.")  
   
 # Update input choices dynamically based on the dataset  
 updateSelectInput(session, "season", choices = unique(accidents\_data$season), selected = "Winter")  
 updateSelectInput(session, "time\_of\_day", choices = unique(accidents\_data$time\_of\_day), selected = "Morning")  
   
 # Summary output  
 output$summary\_out <- renderPrint({  
 paste0("✅ Accident Records: ", nrow(accidents\_data),  
 "\n📅 Date Range: ", min(accidents\_data$start\_time, na.rm = TRUE),  
 " to ", max(accidents\_data$start\_time, na.rm = TRUE))  
 })  
   
 # Visualization: Hour of Day & Day of Week  
 output$plot1 <- renderPlot({  
 p1 <- ggplot(accidents\_data, aes(x = hour)) +  
 geom\_histogram(binwidth = 1, fill = "steelblue", color = "black") +  
 labs(title = "Accidents by Hour of Day", x = "Hour", y = "Count") +  
 theme\_minimal()  
 p2 <- ggplot(accidents\_data, aes(x = day\_of\_week)) +  
 geom\_bar(fill = "darkred", color = "black") +  
 labs(title = "Accidents by Day of Week", x = "Day", y = "Count") +  
 theme\_minimal()  
 grid.arrange(p1, p2, ncol = 2)  
 })  
   
 # Time-of-day visual  
 output$plot2 <- renderPlot({  
 tod <- accidents\_data %>%   
 group\_by(time\_of\_day) %>%   
 summarize(count = n(), .groups = "drop")  
 ggplot(tod, aes(x = time\_of\_day, y = count, fill = time\_of\_day)) +  
 geom\_bar(stat = "identity", color = "black") +  
 labs(title = "Accidents by Time of Day", x = "Time of Day", y = "Count") +  
 theme\_minimal() + theme(legend.position = "none")  
 })  
   
 # Seasonal Accidents Plot  
 output$accident\_season\_plot <- renderPlot({  
 seasonal\_data <- accidents\_data %>%  
 filter(season == input$season) %>%  
 count(day\_of\_week)  
 ggplot(seasonal\_data, aes(x = day\_of\_week, y = n)) +  
 geom\_bar(stat = "identity", fill = "steelblue") +  
 labs(title = paste("Accidents by Day of Week in", input$season), x = "Day", y = "Count")  
 })  
   
 # Time of Day Accidents Plot  
 output$accident\_tod\_plot <- renderPlot({  
 tod\_data <- accidents\_data %>%  
 filter(time\_of\_day == input$time\_of\_day) %>%  
 count(hour)  
 ggplot(tod\_data, aes(x = hour, y = n)) +  
 geom\_bar(stat = "identity", fill = "darkred") +  
 labs(title = paste("Accidents by Hour during", input$time\_of\_day), x = "Hour", y = "Count")  
 })  
 })  
}  
  
shinyApp(ui = ui, server = server)