Class Activity 20

Your name here

May 10 2024

Group Activity 1

Explore COVID-19 vaccination rates across the United States with the following R script, which scrapes data, processes it, and creates an interactive, state-level leaflet map for clear visualization.

```
library(sf)
library(tigris)
library(rvest)
library(janitor)
library(dplyr)
library(leaflet)
library(readr)
library(stringr)

# Set options to get data in sf format
options(tigris_class = "sf")
options(tigris_use_cache = TRUE)

# Load U.S. state shapes
states_sf <- states()

states_sf <- states_sf %>%
  mutate(state = tolower(NAME))

# Reading and preprocessing the COVID data
covid_final <- read_html("https://usafacts.org/visualizations/covid-vaccine-tracker-states/state/minnes
  html_elements(css = "table") %>%
  html_table() %>%
  .[[1]] %>%
  janitor::clean_names() %>%
  mutate(across(c(2, 3, 4), parse_number)) %>%
  mutate(state = tolower(state))

diff <- c(setdiff(states_sf$state, covid_final$state), "alaska", "hawaii")
states_sf <- states_sf %>%
  filter(!state %in% diff)

# Merge COVID data with sf data
states_sf <- left_join(states_sf, covid_final, by = ("state" = "state"))
states_sf <- st_transform(states_sf, crs = "+proj=longlat +datum=WGS84")

# Create the leaflet map
leaflet(states_sf) %>%
```

```
  addProviderTiles(providers$Stamen.TonerLite) %>%
  addPolygons(fillColor = ~colorNumeric("viridis", percent_fully_vaccinated)(percent_fully_vaccinated),
              color = "#000000", weight = 1, opacity = 1,
              fillOpacity = 0.7, smoothFactor = 0.5,
              highlightOptions = highlightOptions(weight = 3, color = "#666666", bringToFront = TRUE),
              label = ~paste(str_to_title(state), ":", percent_fully_vaccinated, "% fully vaccinated"))
  addLegend(pal = colorNumeric("viridis", range(states_sf$percent_fully_vaccinated, na.rm = TRUE)),
            values = ~percent_fully_vaccinated, opacity = 1,
            title = "Vaccination Rate (%)", position = "bottomright")
```

Now, let's create an interactive Leaflet map leveraging the tools in the Shiny ecosystem. This Shiny app allows users to dynamically choose variables related to COVID-19 vaccination rates across U.S. states, as well as the color scheme for visualizing these data. Users can interact with the dashboard to select their preferred data variable and color palette via user-friendly dropdown menus. The map updates in real-time based on user selections, providing a versatile tool for detailed and customized data visualization.

```
library(shiny)
library(tigris)
library(shinydashboard)
library(shinyWidgets)
library(leaflet)
library(sf)
library(dplyr)
library(rvest)
library(janitor)
library(RColorBrewer)
library(readr)
library(stringr)
library(ggplot2)
library(shinybusy)


options(tigris_class = "sf")
options(tigris_use_cache = TRUE)

# Load U.S. state shapes
states_sf <- states() %>%
  mutate(state = tolower(NAME)) %>%
  left_join(read_html("https://usafacts.org/visualizations/covid-vaccine-tracker-states/state/minnesota
            html_elements(css = "table") %>%
            html_table() %>%
            .[[1]] %>%
            janitor::clean_names() %>%
            mutate(across(2:4, parse_number), state = tolower(state)), by = "state") %>%
  filter(!state %in% c("alaska", "hawaii", "united states virgin islands", "commonwealth of the northern
  st_transform(crs = "+proj=longlat +datum=WGS84")

head(states_sf)
```

```
server <- function(input, output, session) {

  values <- reactiveValues()

  # Load and prepare the initial data
```

```
values$data <- states_sf
values$variable <- "percent_fully_vaccinated"
values$colorScheme <- "Spectral"

# Function to update data based on current inputs
updateData <- function() {
  values$data$selected_var <- values$data[[values$variable]]
  values$data
}

# Render the map initially and upon changes
output$map <- renderLeaflet({
  # This will fetch the data and apply transformations based on the selected variable
  data <- updateData()

  # Create a color palette using the currently selected color scheme
  pal <- colorNumeric(brewer.pal(8, values$colorScheme), data$selected_var)

  # Build and render the map
  leaflet(data) %>%
    addProviderTiles(providers$Stamen.TonerLite) %>%
    addPolygons(
      fillColor = ~pal(selected_var),
      color = "#000000", weight = 1, opacity = 1,
      fillOpacity = 0.7, smoothFactor = 0.5,
      highlightOptions = highlightOptions(weight = 3, color = "#666666", bringToFront = TRUE),
      popup = ~paste(str_to_title(NAME), ":", round(selected_var, 2), "%")
    ) %>%
    addLegend(
      pal = pal,
      values = ~selected_var,
      opacity = 1,
      title = values$variable,
      position = "bottomright"
    )
})

# Observe the Update Map button
observeEvent(input$updateMap, {
  # Update the variables and color scheme based on the input from the UI when the button is pressed
  values$variable <- input$variable
  values$colorScheme <- input$colorScheme

  # Force re-render of the map
  output$map <- renderLeaflet({
    # Re-fetch the updated data
    data <- updateData()

    # Recreate the color palette with possibly a new color scheme
    pal <- colorNumeric(brewer.pal(8, values$colorScheme), data$selected_var)

    # Rebuild and render the updated map
    leaflet(data) %>%
```

```
        addProviderTiles(providers$Stamen.TonerLite) %>%
        addPolygons(
          fillColor = ~pal(selected_var),
          color = "#000000", weight = 1, opacity = 1,
          fillOpacity = 0.7, smoothFactor = 0.5,
          highlightOptions = highlightOptions(weight = 3, color = "#666666", bringToFront = TRUE),
          popup = ~paste(str_to_title(NAME), ":", round(selected_var, 2), "%")
        ) %>%
        addLegend(
          pal = pal,
          values = ~selected_var,
          opacity = 1,
          title = values$variable,
          position = "bottomright"
        )
    })
  })
}


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