Class Activity 20

Your name here

February 19 2024

Group Activity 1

Explore the magic of `leaflet` firsthand! Simply paste the code provided in the slides into the field below and run it to witness the output in action.

```
# paste the code here
```

Group Activity 2

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

Q1: How do we obtain and prepare the COVID-19 vaccination data?

To analyze COVID-19 vaccination rates, we start by scraping the necessary data from a webpage. This involves selecting the specific HTML elements that contain the data, parsing it into a readable format, and cleaning the column names for easier manipulation. We also filter the data to include only certain states for focused analysis.

```
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_at(2:4, parse_number) %>%
  mutate(state = str_to_lower(state)) %>%
  filter(state %in% c("ohio", "indiana", "michigan", "illinois", "missouri", "wisconsin", "minnesota",
                      "iowa", "kansas", "nebraska", "south dakota", "north dakota"))
```

Q2: How do we represent geographical data for mapping?

Next, we create a spatial representation of the Midwest states. This involves mapping state boundaries and converting them into a spatial polygons format, which `Leaflet` can use to plot the data geographically.

```
USA_Midwest <- maps::map("state",
                         regions = c("ohio", "indiana", "michigan", "illinois", "missouri", "wisconsin"
                                     "iowa", "kansas", "nebraska", "south dakota", "north dakota"),
```

```
                          plot = FALSE, fill = TRUE) %>%
  map2SpatialPolygons(IDs = str_remove(.$names, "(?=:).+"))
```

Q3: How do we integrate the vaccination data with the geographical map?

Merging the COVID-19 vaccination data with the geographical map allows us to visualize vaccination rates across different regions. This step combines the spatial polygons with the vaccination data based on state names.

```
map <- SpatialPolygonsDataFrame(USA_Midwest, covid_final, match.ID = FALSE)
```

Q4: How do we categorize vaccination rates for visualization?

To visually differentiate regions based on their vaccination rates, we create bins of vaccination percentages and assign a color palette. This helps in categorizing states into different levels of vaccination coverage.

```
bins <- seq(min(map$percent_fully_vaccinated), max(map$percent_fully_vaccinated), length.out = 6)
pal <- colorBin("viridis", domain = map$percent_fully_vaccinated, bins = bins)
```

Q5: How do we add informative labels to the map?

Labels enhance the map's interactivity by providing detailed information on hover or click. Here, we format the labels to display the state name and its observed vaccination rate.

```
labels <- str_glue("<strong> {map$county} </strong> <br/> Observed: {map$cases}") %>%
  lapply(htmltools::HTML)
```

Q6: How do we plot the finalized map with Leaflet?

```
leaflet(map) %>%
  addTiles() %>%
  setView(lng = -93.1616, lat = 44.4583, zoom = 4) %>%
  addPolygons(
    color = "grey",
    weight = 1,
    fillColor = ~pal(percent_fully_vaccinated),
    fillOpacity = 0.7,
    highlightOptions = highlightOptions(weight = 5),
    label = labels
  ) %>%
  addLegend(
    pal = pal,
    values = ~percent_fully_vaccinated,
    opacity = 0.5,
    title = "Percent Vaccinated",
    position = "bottomright"
  )
Error in sum(sapply(label, function(x) {: invalid 'type' (list) of argument
```

Q7: Think about how would you include your favorite state beside these midwestern states to the leaflet plot. How would you go about including all the states? Please exclude `Alaska` and `Hawaii` from the analysis.

```
# your r-code
```

Q8. Deploying a Shiny App to `shinyapps.io`

Open an account on shinyapps.io and follow the steps to deploy one of the apps to shinyapps.io.

Step 1: Create Your Shiny App Create a file named app.R in a new folder. This folder will contain all the necessary files for your app.

Step 2: Organize Your App Folder Ensure your app.R file is saved in a dedicated folder for your app. This folder can also include any additional data files, data folder for data, or other resources your app needs.

Step 3: Deploy Your App to `shinyapps.io` Before deploying, you'll need to set up an account on `shinyapps.io`. Use the rsconnect package to authenticate your R session with your `shinyapps.io` account. You will need the API key from your `shinyapps.io` account dashboard.

```
library(rsconnect)
rsconnect::setAccountInfo(name='<account-name>',
                          token='<api-token>',
                          secret='<api-secret>')
```

Replace `<account-name>`, `<api-token>`, and `<api-secret>` with your actual account name and API key details. Navigate to your app folder in RStudio, or set the working directory to your app folder in R:

```
setwd("/path/to/your/app")
```

Deploy your app using the `deployApp` function:

```
rsconnect::deployApp()
```

Follow the prompts in the R console. Once the deployment process completes, you will receive a URL to access your app hosted on `shinyapps.io`.