app.R

samsaltman

2022-04-03

```
#### Load packages ----
library(shiny)
library(shinythemes)
library(tidyverse)
```

```
## -- Attaching packages --------------------------------------- tidyverse 1.3.1 --
```

```
## v ggplot2 3.3.5     v purrr   0.3.4
## v tibble  3.1.4     v dplyr   1.0.7
## v tidyr   1.1.3     v stringr 1.4.0
## v readr   2.0.1     v forcats 0.5.1
```

```
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```
#### Load data ----
nutrient_data <- read_csv("Data/NTL-LTER_Lake_Nutrients_PeterPaul_Processed.csv")
```

```
## Rows: 2770 Columns: 13
```

```
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr  (2): lakeid, lakename
## dbl  (9): year4, daynum, depth_id, depth, tn_ug, tp_ug, nh34, no23, po4
## lgl  (1): comments
## date (1): sampledate
```

```
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
nutrient_data$sampledate <- as.Date(nutrient_data$sampledate, format = "%Y-%m-%d")
nutrient_data <- nutrient_data %>%
  filter(depth_id > 0) %>%
  select(lakename, sampledate:po4)

#### Define UI ----
ui <- fluidPage(theme = shinytheme("superhero"),
  titlePanel("Nutrients in Peter Lake and Paul Lake"),
  sidebarLayout(
    sidebarPanel(

      # Select nutrient to plot
      selectInput(inputId = "y",
                  label = "Nutrient",
```

```
                  choices = c("tn_ug", "tp_ug", "nh34", "no23", "po4"),
                  selected = "nh34"),

      # Select depth
      checkboxGroupInput(inputId = "fill",
                         label = "Depth ID",
                         choices = unique(nutrient_data$depth_id),
                         selected = c(1, 7)),

      # Select lake
      checkboxGroupInput(inputId = "shape",
                         label = "Lake",
                         choices = c("Peter Lake", "Paul Lake"),
                         selected = "Paul Lake"),

      # Select date range to be plotted
      sliderInput(inputId = "x",
                  label = "Date",
                  min = as.Date("1991-05-01"),
                  max = as.Date("2016-12-31"),
                  value = c(as.Date("2011-01-01"), as.Date("2016-12-31")))),

    # Output
    mainPanel(
      plotOutput("scatterplot", brush = brushOpts(id = "scatterplot_brush")),
      tableOutput("mytable")
    )))

#### Define server  ----
server <- function(input, output) {

    # Define reactive formatting for filtering within columns
     filtered_nutrient_data <- reactive({
       nutrient_data %>%
         filter(sampledate >= input$x[1] & sampledate <= input$x[2]) %>%
         filter(depth_id %in% input$fill) %>%
         filter(lakename %in% input$shape)
     })

    # Create a ggplot object for the type of plot you have defined in the UI
       output$scatterplot <- renderPlot({
        ggplot(filtered_nutrient_data(),
               aes_string(x = "sampledate", y = input$y,
                          fill = "depth_id", shape = "lakename")) +
          geom_point(alpha = 0.5, size = 3) +
          theme_classic(base_size = 17) +
          scale_shape_manual(values = c(21, 24)) +
          labs(x = "Date", y = expression(Concentration ~ (mu*g / L)), shape = "Lake", fill = "Depth ID
          #scale_fill_distiller(palette = "YlOrBr", guide = "colorbar", direction = 1)
          scale_fill_viridis_c(option = "viridis", begin = 0, end = 0.8, direction = -1)
      })

    # Create a table that generates data for each point selected on the graph
```

```
      output$mytable <- renderTable({
        brush_out <- brushedPoints(filtered_nutrient_data(), input$scatterplot_brush)
      })

  }


#### Create the Shiny app object ----
shinyApp(ui = ui, server = server)
```

```
## PhantomJS not found. You can install it with webshot::install_phantomjs(). If it is installed, pleas
```

```
#### Questions for coding challenge ----
#1. Play with changing the options on the sidebar.
    # Choose a shinytheme that you like. The default here is "yeti"
    # How do you change the default settings?
    # How does each type of widget differ in its code and how it references the dataframe?
#2. How is the mainPanel component of the UI structured?
    # How does the output appear based on this code?
#3. Explore the reactive formatting within the server.
    # Which variables need to have reactive formatting?
    # How does this relate to selecting rows vs. columns from the original data frame?
#4. Analyze the similarities and differences between ggplot code for a rendered vs. static plot.
    # Why are the aesthetics for x, y, fill, and shape formatted the way they are? Continuous vs discre
    # Note: the data frame has a "()" after it. This is necessary for reactive formatting.
    # Adjust the aesthetics, playing with different shapes, colors, fills, sizes, transparencies, etc.
#5. Analyze the code used for the renderTable function.
    # Notice where each bit of code comes from in the UI and server.
    # Note: renderTable doesn't work well with dates. "sampledate" appears as # of days since 1970.
```