# Mini Project 2

## PSTAT100: Data Science Concepts and Analysis

Instructor: Ali Abuzaid

2024-11-15

**STUDENT NAME**

- Harry He (7133085)
- Yue Zhu (A1F8C52)
- Tina Zhou (48761653)
- Xihua Yao (5927777)

Instructions

- This mini project aims to familiarize you with real-life data sourced from various resources.
- The mini project includes narrative questions. While these questions are primarily based on lecture material and prerequisites, they may also require independent thinking and investigation.
- Collaborate in groups of **3-4** students from the **same discussion session**; individual submissions **will not be accepted**.
- Ensure that all `R` code, mathematical formulas, and workings are presented clearly and appropriately.
- Please submit a .zip file of the folder you create, containing your `app.R` file along with the report.

**Information on Grading** To grade your project submissions, the grader will download your project files and run the app you created locally.
**Important:** If your app fails to open due to issues like incorrect file format or improper zipping, your group will receive a score of ZERO for this project. It's essential to submit a functioning app for fair grading. Thank you for your understanding!**

Due Date

**Due Date:** Friday, November 15, 2024, 11:59 PM

# 1 Overview

**Project Title**: Interactive Exploration of the ‘CO2’ Dataset

## 1.1 Objectives:

**Data Exploration**: Students will gain hands-on experience in data exploration and visualization using a classic dataset available in `R`.

**Shiny App Development**: Students will learn the basics of building a Shiny app, including UI design and server logic.

**User Interaction**: Students will implement interactive features allowing users to filter and visualize the dataset.

**Presentation of Results**: Students will summarize their findings and demonstrate the app to their peers. Dataset:

## 1.2 Dataset

Choose any dataset available in R (e.g., `mtcars`, `iris`, `diamonds`, etc.). You can load datasets directly using built-in `R` functions.

# 2 Tasks:

## 2.1 Data Loading and Preparation:

1- Load your chosen dataset using the following R code:

```r
data(CO2)
head(CO2)
```

```
  Plant   Type  Treatment conc uptake
1   Qn1 Quebec nonchilled   95   16.0
2   Qn1 Quebec nonchilled  175   30.4
3   Qn1 Quebec nonchilled  250   34.8
4   Qn1 Quebec nonchilled  350   37.2
5   Qn1 Quebec nonchilled  500   35.3
6   Qn1 Quebec nonchilled  675   39.2
```

2- Explore the dataset to understand its structure and perform any necessary data cleaning. Use functions like summary(mtcars) and str(mtcars) for insights.

```r
summary(CO2)
```

```
     Plant             Type         Treatment           conc          uptake
 Qn1    : 7   Quebec     :42   nonchilled:42   Min.   :  95   Min.   : 7.70
 Qn2    : 7   Mississippi:42   chilled   :42   1st Qu.: 175   1st Qu.:17.90
 Qn3    : 7                                    Median : 350   Median :28.30
 Qc1    : 7                                    Mean   : 435   Mean   :27.21
 Qc3    : 7                                    3rd Qu.: 675   3rd Qu.:37.12
 Qc2    : 7                                    Max.   :1000   Max.   :45.50
 (Other):42
```

```r
str(CO2)
```

```
Classes 'nfnGroupedData', 'nfGroupedData', 'groupedData' and 'data.frame':  84 obs. of  5 va
 $ Plant    : Ord.factor w/ 12 levels "Qn1"<"Qn2"<"Qn3"<..: 1 1 1 1 1 1 1 2 2 2 ...
 $ Type     : Factor w/ 2 levels "Quebec","Mississippi": 1 1 1 1 1 1 1 1 1 1 ...
 $ Treatment: Factor w/ 2 levels "nonchilled","chilled": 1 1 1 1 1 1 1 1 1 1 ...
 $ conc     : num  95 175 250 350 500 675 1000 95 175 250 ...
 $ uptake   : num  16 30.4 34.8 37.2 35.3 39.2 39.7 13.6 27.3 37.1 ...
 - attr(*, "formula")=Class 'formula'  language uptake ~ conc | Plant
  .. ..- attr(*, ".Environment")=<environment: R_EmptyEnv>
 - attr(*, "outer")=Class 'formula'  language ~Treatment * Type
  .. ..- attr(*, ".Environment")=<environment: R_EmptyEnv>
 - attr(*, "labels")=List of 2
  ..$ x: chr "Ambient carbon dioxide concentration"
  ..$ y: chr "CO2 uptake rate"
 - attr(*, "units")=List of 2
  ..$ x: chr "(uL/L)"
  ..$ y: chr "(umol/m^2 s)"
```

```r
names(CO2)
```

```
[1] "Plant"     "Type"      "Treatment" "conc"      "uptake"
```

```r
dim(CO2)
```

```
[1] 84  5
```

```r
sum(is.na(CO2))
```

```
[1] 0
```

```r
range(CO2$conc)
```

```
[1]   95 1000
```

## 2.2 Shiny App Development

**User Interface (UI):**

1- Create a sidebar layout that allows users to select:

- A variable to analyze.
- Type of plot (e.g., scatter plot, histogram, box plot).
- Display the selected plot and relevant summary statistics.

```r
library(shiny)
data('CO2')

ui <- fluidPage(
    titlePanel("CO2 Dataset Interactive Explorer"),

    sidebarLayout(
        sidebarPanel(
            selectInput("variable", "Analyze CO2 Data By:",
                        choices = c("CO2 Uptake Rate" = "uptake", "CO2 Concentration Level

            selectInput("group", "Group Data By:",
                        choices = c("Treatment", "Type")),

            selectInput("plotType", "Type of Plot to Display:",
                        choices = c("Scatter Plot", "Histogram", "Box Plot")),

            sliderInput("concRange", "Concentration Range Filter:",
                        min = min(CO2$conc), max = max(CO2$conc),
            value = c(min(CO2$conc), max(CO2$conc))),

      sliderInput("uptakeRange", "Uptake Range Filter:",
            min = min(CO2$uptake), max = max(CO2$uptake),
            value = c(min(CO2$uptake), max(CO2$uptake)))
    ),
    mainPanel(
      plotOutput("plot"),
      verbatimTextOutput("summary")
    )
  )
)
```

**Server Logic:** Write server code to respond to user inputs and generate the selected plot dynamically.

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

 filteredData <- reactive({
   CO2[CO2$conc >= input$concRange[1] &
     CO2$conc <= input$concRange[2] &
     CO2$uptake >= input$uptakeRange[1] &
     CO2$uptake <= input$uptakeRange[2], ]
 })


 output$plot <- renderPlot({
  data <- filteredData()

  if (input$plotType == "Scatter Plot") {
       colors <- c("skyblue", "orange", "green", "orchid4")

    layout(matrix(c(1, 2), nrow = 1), widths = c(4, 1))

    par(mar = c(5, 4, 4, 2) + 0.1)
    plot(data$conc, data$uptake,
      col = colors[as.factor(data[[input$group]])],
      pch = 16,
      xlab = "Concentration", ylab = "Uptake",
      main = paste("Scatter Plot of Uptake vs Concentration by", input$group))


```

```r
       tryCatch({model <- lm(uptake ~ conc, data = data)
                 abline(model, col = "firebrick", lwd = 2)},
                error = function(err) {
                showNotification("Best-fit line cannot be found!", type = "error")})


       par(mar = c(5, 0, 4, 0))
       plot.new()
       legend("center",
              legend = levels(as.factor(data[[input$group]])),
              col = colors[1:length(levels(as.factor(data[[input$group]])))],
              pch = 16,
              title = input$group,
              bty = "n")

    } else if (input$plotType == "Histogram") {
      hist(data[[input$variable]],
           xlab = input$variable, main = paste("Histogram of", input$variable),
           col = "lightcyan3")
    } else if (input$plotType == "Box Plot") {
      boxplot(data[[input$variable]] ~ data[[input$group]],
              xlab = "Group", ylab = input$variable,
              main = paste("Box Plot of", input$variable,"by", input$group),
              col = "bisque4",
            border = "black")
    }
  })

  output$summary <- renderPrint({
    if (input$variable == "uptake")
    {
      print("Uptake Rate")
    }
    else
    {
      print("Concentration Level")
    }

    data <- filteredData()
    summary(data[[input$variable]])
  })
```

```
}


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

### 2.3 Interactivity

1- Implement features such as sliders to adjust numerical variables and checkboxes for selection.

2- Consider adding tooltips or hover functionality for enhanced user experience.

In the updated Shiny app, the labels in the UI have been changed to make it clearer for users to pick options, like choosing between different CO2 measurements. A regression line was added to the scatter plot to show the relationship between concentration and uptake, and it appears when there are enough data points. Colors were also added to the scatter plot, histogram, and box plot to make it easier to read the data. Finally, tooltips and plot titles were improved to give users more context and make the app easier to understand. If the regression line does not exist, a “Best-fit line cannot be found!” reminder will show up.

## 3 Documentation and Presentation

The Documentation of the dataset “CO2” is also inside the zip file.

## 4 References

1- Shiny Documentation: Welcome to Shiny

2- Hadley Wickham (2021) Mastering Shiny, O’Reilly Media. (https://mastering-shiny.org/)