

A simple shiny app to explore Palmer Penguin data using ChatGPT to prototype.

Ronald (Ryy) Glenn Thomas

2025-01-14

Table of contents

1 Introduction	1
1.2 R code to launch.	2

1 Introduction

Shiny apps are a great way to explore data interactively. In this example, we will use ChatGPT to prototype a simple Shiny app to explore the Palmer Penguin data set from the `palmerpenguins` package.

1.1

chatGPT Prompts:

“I want to use the Palmer Penguin dataset to create a Shiny app for data exploration.”

“Update shiny app. Add a dropdown menu to select categorical variables, sex, species or island. Also add a dropdown menu to select continuous variables Use selected categorical variable as



Figure 1: chatGPT

a grouping variable in side-by-side boxplots of selected continuous variables.”

Specifics that could be added (but weren’t. ChatGPT inferred them):

Features:

- The legend is positioned at the bottom for better clarity.
- The x aesthetic is mapped to the grouping variable (groupvar), and the y aesthetic is mapped to the selected continuous variable (xvar).
- Includes filtering by species, island, and body mass.

1.2 R code to launch.

From inside R.

```
shiny::runApp("app.R", launch.browser = TRUE)
```

From shell.

```
R -e "shiny::runApp('app.R', launch.browser=T)"
```

```
library(shiny)          # Load the Shiny package for building web apps
library(palmerpenguins) # Load the Palmer Penguins dataset
library(ggplot2)        # Load ggplot2 for creating plots

# Load the dataset and remove rows with missing values for simplicity
data <- na.omit(penguins)

# User Interface (UI) Definition
ui <- fluidPage(
  # App title displayed at the top
  titlePanel("Palmer Penguins Explorer"),

  # Layout with a sidebar for controls and main panel for outputs
```

```

sidebarLayout(
  # Sidebar containing input controls
  sidebarPanel(
    # Dropdown for selecting the continuous variable to plot
    selectInput(
      "xvar",
      "Continuous Variable for Boxplot:",
      choices = names(data)[3:6] # Select columns 3 to 6 as options
    ),

    # Dropdown for selecting the grouping categorical variable
    selectInput(
      "groupvar",
      "Group by (Categorical Variable):",
      choices = c("species", "sex", "island"), # Options for grouping
      selected = "species" # Default selection
    ),

    # Slider for filtering body mass range
    sliderInput(
      "body_mass",
      "Body Mass Range (g):",
      min = min(data$body_mass_g), # Minimum body mass in dataset
      max = max(data$body_mass_g), # Maximum body mass in dataset
      value = range(data$body_mass_g) # Default slider range
    ),

    # Checkbox group for filtering species
    checkboxGroupInput(
      "species",
      "Species:",
      choices = unique(data$species), # Unique species in dataset
      selected = unique(data$species) # All species selected by default
    ),

    # Checkbox group for filtering islands
    checkboxGroupInput(
      "islands",
      "Islands:",
      choices = unique(data$island), # Unique islands in dataset

```

```

    selected = unique(data$island) # All islands selected by default
  )
),

# Main panel for displaying outputs
mainPanel(
  # Boxplot output
  plotOutput("boxPlot"),

  # Text output for displaying summary statistics
  verbatimTextOutput("summary")
)
)
)

# Server Logic
server <- function(input, output) {
  # Reactive expression to filter data based on user inputs
  filteredData <- reactive({
    filtered <- data
    # Filter by selected species
    filtered <- filtered[filtered$species %in% input$species, ]
    # Filter by selected islands
    filtered <- filtered[filtered$island %in% input$islands, ]
    # Filter by body mass range from the slider
    filtered <- filtered[
      filtered$body_mass_g >= input$body_mass[1] &
      filtered$body_mass_g <= input$body_mass[2],
    ]
    filtered # Return the filtered dataset
  })

  # Render the boxplot based on filtered data and user inputs
  output$boxPlot <- renderPlot({
    ggplot(
      filteredData(), # Use the filtered data
      aes_string(x = input$groupvar, y = input$xvar, fill = input$groupvar)
    ) +
    geom_boxplot(alpha = 0.7) + # Add boxplots with some transparency
    theme_minimal() +         # Use a minimal theme for the plot
  })
}

```

```

    labs(x = input$groupvar, y = input$xvar, fill = input$groupvar) +
    theme(legend.position = "bottom") # Place the legend at the bottom
  })

  # Render summary statistics for the filtered data
  output$summary <- renderPrint({
    summary(filteredData()) # Print a summary of the filtered dataset
  })
}

# Run the Shiny app
shinyApp(ui, server)

```