Session 2 on Interacting with data for research and teaching.

In this one-hour session, we'll delve into the practical applications of leveraging your data to enhance research and teaching purposes using three chatgpt tools: TheOpenAir, CodeLingo and RTutor. By the end of the session, you'll have beginner knowledge and experience to effectively utilize these tools in your own data analysis projects and educational endeavors.

1. TheOpenAIR Package

TheOpenAir package provides seamless integration of ChatGPT technology into R applications, allowing for interactive exploration and analysis of data. With TheOpenAir, you can harness the power of ChatGPT to generate code, insights, and visualizations tailored to your data analysis needs.

A. Functions Overview

Functions provided by TheOpenAIR package:

- chat(): Interacts with the OpenAI API to send requests and retrieve responses.
- count_tokens(): Counts the number of ChatGPT tokens required for a given string.
- write_code(): Generates code in a specified language, useful for automating tasks. *We will not use this function in the session because it is used for chatgpt conversations in the R console, and may take time to load. It is replaced with the* **code_generation_prompt** for generating code related to a specific task or analysis.
- extract_r_code(): Extracts R code from a ChatGPT response containing code snippets.
- refactor(): Performs R-specific code refactoring for improved efficiency.
- **Others:**
    - get_chatlog_id(): Retrieves the ID of the current ChatGPT session for tracking purposes.
    - For detailed usage examples, refer to the documentation via help(package = "TheOpenAIR").

B. Load libraries and connect to OpenAI API

```
# Step 1: Install and load necessary packages
# install.packages(c("TheOpenAIR", "httr", "jsonlite"))

library(httr)
library(jsonlite)
library(TheOpenAIR)

# Step 2: API Key Set Up
openai_api_key("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"
```

C. Example: Analyzing Data with ChatGPT

In this example, we utilize the essential functions provided by TheOpenAIR package to interact with ChatGPT and fetch insights, count tokens, extract R code, refactor the code, and generated a scatter plot of fuel consumption vs. horsepower. This approach demonstrates the versatility and usefulness of TheOpenAIR package in real-world data analysis tasks.

Step 1: Interact with ChatGPT

```
# Define the question to ask ChatGPT
question <- "Based on the summary of the mtcars dataset, could you provide insights or analysis rega
rding the relationship between fuel consumption and horsepower? Additionally, are there any notable
patterns or outliers in the data that warrant further investigation?"

# Call the chat function to get a response
chat_response <- chat(question, output = "message")

# Print ChatGPT response
cat("ChatGPT Response:\n", chat_response, "\n\n")
```

```
ChatGPT Response:
 Certainly! In the mtcars dataset, we can explore the relationship between fuel consumption (measure
d as Miles Per Gallon - mpg) and horsepower. Generally, there is an inversely proportional relationshi
p between fuel consumption and horsepower in a vehicle - higher horsepower usually corresponds to
lower fuel efficiency.

To analyze this relationship, we can plot a scatter plot to visualize how mpg changes with horsepower
. By examining the data points in the plot and fitting a line or curve to the points, we can identify any tr
ends or patterns in the data.

Additionally, we can investigate outliers in the dataset by looking at data points that deviate significant
ly from the overall trend. Outliers could indicate errors in data collection, anomalies in the vehicles, or
specific characteristics of certain models that warrant further investigation.

Would you like to see a visual representation of the relationship between fuel consumption and horse
power in the mtcars dataset?
```

Step 2: Count Tokens

```
# Count tokens for the prompt
token_count <- count_tokens(chat_response)
print(token_count)
```

```
[1] 168
```

Step 3: Instruct ChatGPT to Generate Scatterplot Code

```
# Prompt ChatGPT to generate R code
code_generation_prompt <- "Based on the summary of the mtcars dataset, please generate R code to analyze the relationship between fuel consumption and horsepower. For example, you could write code to create a scatter plot or calculate correlation coefficients."

# Chat with ChatGPT
chat_response <- chat(code_generation_prompt, output = "message")

# Print ChatGPT Response
cat("ChatGPT Response:\n", chat_response, "\n\n")
```

ChatGPT Response:
Certainly! Here is an example of R code that you can use to analyze the relationship between fuel consumption (mpg) and horsepower in the mtcars dataset:

```R
# Load the mtcars dataset
data(mtcars)

# Create a scatter plot of mpg vs. horsepower
plot(mtcars$hp, mtcars$mpg, xlab = "Horsepower", ylab = "Miles Per Gallon", main = "Scatter Plot of MPG vs. Horsepower")

# Calculate the correlation coefficient between mpg and horsepower
correlation <- cor(mtcars$hp, mtcars$mpg)
cat("Correlation coefficient between MPG and Horsepower:", correlation, "\n")
```

You can run this code in an R environment to generate a scatter plot showing the relationship between fuel consumption (mpg) and horsepower in the mtcars dataset, as well as calculate the correlation coefficient between the two variables. This will help you understand the strength and direction of the relationship between fuel consumption and horsepower in the dataset.

Step 4: Extract code

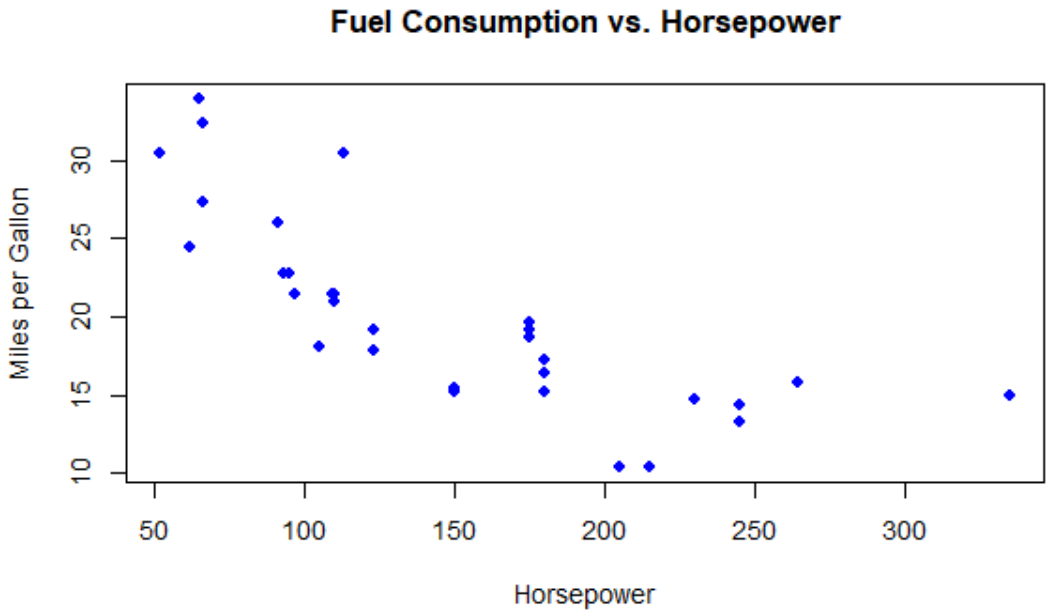
```
extract_r_code(chat_response)
```

[1] "Certainly! Here is an example of R code that you can use to analyze the relationship between fuel consumption (mpg) and horsepower in the mtcars dataset:"
[2] "# Load the mtcars dataset
"
[3] "data(mtcars)
"
[4] "# Create a scatter plot of mpg vs. horsepower"
[5] "plot(mtcars$hp, mtcars$mpg, xlab = \"Horsepower\", ylab = \"Miles Per Gallon\", main = \"Scatter Plot of MPG vs. Horsepower\")"

[6] "# Calculate the correlation coefficient between mpg and horsepower
"
[7] "correlation <- cor(mtcars$hp, mtcars$mpg)
"
[8] "cat(\"Correlation coefficient between MPG and Horsepower:\", correlation, \"\\n\")
"
[9] "You can run this code in an R environment to generate a scatter plot showing the relationship bet
ween fuel consumption (mpg) and horsepower in the mtcars dataset, as well as calculate the correlati
on coefficient between the two variables. This will help you understand the strength and direction of th
e relationship between fuel consumption and horsepower in the dataset."
Certainly! Here is an example of R code that you can use to analyze the relationship between fuel con
sumption (mpg) and horsepower in the mtcars dataset:
# Load the mtcars dataset
data(mtcars)
# Create a scatter plot of mpg vs. horsepower
plot(mtcars$hp, mtcars$mpg, xlab = "Horsepower", ylab = "Miles Per Gallon", main = "Scatter Plot of
MPG vs. Horsepower")
# Calculate the correlation coefficient between mpg and horsepower
correlation <- cor(mtcars$hp, mtcars$mpg)
cat("Correlation coefficient between MPG and Horsepower:", correlation, "\n")
You can run this code in an R environment to generate a scatter plot showing the relationship betwee
n fuel consumption (mpg) and horsepower in the mtcars dataset, as well as calculate the correlation c
oefficient between the two variables. This will help you understand the strength and direction of the rel
ationship between fuel consumption and horsepower in the dataset.

Step 5: Refactor code (if needed)

```
# Define a simple R code snippet
code_snippet <-

# Load the mtcars dataset
data(mtcars)

# Create a scatter plot of Fuel Consumption vs. Horsepower
plot(mtcars$hp, mtcars$mpg, main = "Fuel Consumption vs. Horsepower",
     xlab = "Horsepower", ylab = "Miles per Gallon", col = "blue", pch = 19)

# Calculate the correlation coefficient between Fuel Consumption and Horsepower
correlation <- cor(mtcars$mpg, mtcars$hp)
cat("Correlation between Fuel Consumption and Horsepower:", correlation, "\n")

# Use the refactor() function
refactored_code <- refactor(code_snippet)

# Print the refactored code
cat("Refactored Code:\n")
cat(refactored_code)
```



D. Can TheOpenAir Create an R Shiny App in minutes with ChatGPT?

```
# Prompt ChatGPT to generate R code
code_generation_prompt <- "Create an R Shiny App with ChatGPT that allows users to select and plot bivariate combinations of numeric variables from the diamonds dataset. Add code to connect to a publicly available REST API for ChatGPT. I have the endpoint and any required authentication details, and I can modify the code to use httr to communicate with the API. "

# Chat with ChatGPT
chat_response2 <- chat(code_generation_prompt, output = "message")

# Print ChatGPT Response
cat("ChatGPT Response:\n", chat_response2, "\n\n")
```

ChatGPT Response:
To create an R Shiny App with ChatGPT integration and allow users to select and plot bivariate combinations of numeric variables from the diamonds dataset, you can follow the steps below:

1. First, ensure you have the shiny and ggplot2 packages installed. If not, you can install them using `install.packages("shiny")` and `install.packages("ggplot2")`.

2. Here is the code template for the Shiny App with ChatGPT integration and plotting bivariate combinations:

```R
library(shiny)
library(ggplot2)
library(httr)

# UI
ui <- fluidPage(
  titlePanel("Interactive Data Visualization"),
```

```
  sidebarLayout(
    sidebarPanel(
      selectInput("x_var", "X-axis:", choices = names(diamonds)[sapply(diamonds, is.numeric)]),

      selectInput("y_var", "Y-axis:", choices = names(diamonds)[sapply(diamonds, is.numeric)])
    ),

    mainPanel(
      plotOutput("plot")
    )
  )
)

# Server logic
server <- function(input, output, session) {
  output$plot <- renderPlot({
    ggplot(diamonds, aes_string(x = input$x_var, y = input$y_var)) +
      geom_point()
  })

  # Code to connect to ChatGPT API using httr
  chatGPT_response <- httr::GET("API_ENDPOINT_HERE",
                                add_headers(Authorization = "Bearer YOUR_AUTH_TOKEN_HERE"))

  # Parse the response from ChatGPT
  chatGPT_text <- httr::content(chatGPT_response, "text")

  # Display ChatGPT response in the Shiny app
  output$gpt_output <- renderText({
    chatGPT_text
  })
}

# Run the application
shinyApp(ui = ui, server = server)
```

3. Replace `"API_ENDPOINT_HERE"` with the actual REST API endpoint provided to you, and `"YOUR_AUTH_TOKEN_HERE"` with the authentication details required to access the API.

4. You can customize and enhance the Shiny app further, e.g., by adding more interactive elements, error handling for API requests, or displaying additional information from ChatGPT.

5. Execute the app in your R environment, and users will be able to select and plot bivariate combinations of numeric variables from the diamonds dataset while also interacting with ChatGPT through the REST API.

Feel free to modify the code according to your specific requirements and data visualization preferences.

Step 1. Install required packages (if not already installed):

```
# install.packages(c("shiny", "ggplot2", "httr"))
```

Step 2. Load the libraries.

```
library(shiny)
library(ggplot2)
library(httr)
library(jsonlite)
```

Step 3. UI for the Shiny App

```
ui <- fluidPage(
  titlePanel("Diamonds Dataset Analysis"),
  sidebarLayout(
    sidebarPanel(
      selectInput("x", "X-axis variable", choices = colnames(diamonds),
                  selected = "carat"),
      selectInput("y", "Y-axis variable", choices = colnames(diamonds),
                  selected = "price")
    ),
    mainPanel(
      plotOutput("plot")
    )
  )
)
```

Step 4. Function to connect to OpenAI GPT-3 API

```
chatgpt_api_call <- function(input_text) {
  open_ai_key <- "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"
  request_body <- list(
    model = "gpt-3.5-turbo-0125",
    messages = list(
      list(role = "user", content = input_text)
    ),
    max_tokens = 500,
    temperature = 0.7
  )

  response <- tryCatch({
    httr::POST(
      url = "https://api.openai.com/v1/chat/completions",
      add_headers(
        "Content_Type" = "application/json",
```

```
      "Authorization" = paste("Bearer", open_ai_key)
    ),
    body = request_body,
    encode = "json"
  )
 }, error = function(e) {
  print(paste("Error:", e$message))
  NULL
 })

 if (!is.null(response)) {
  content(response)
 } else {
  NULL
 }
}
```

Step 4. Function to connect to OpenAI GPT-3 API

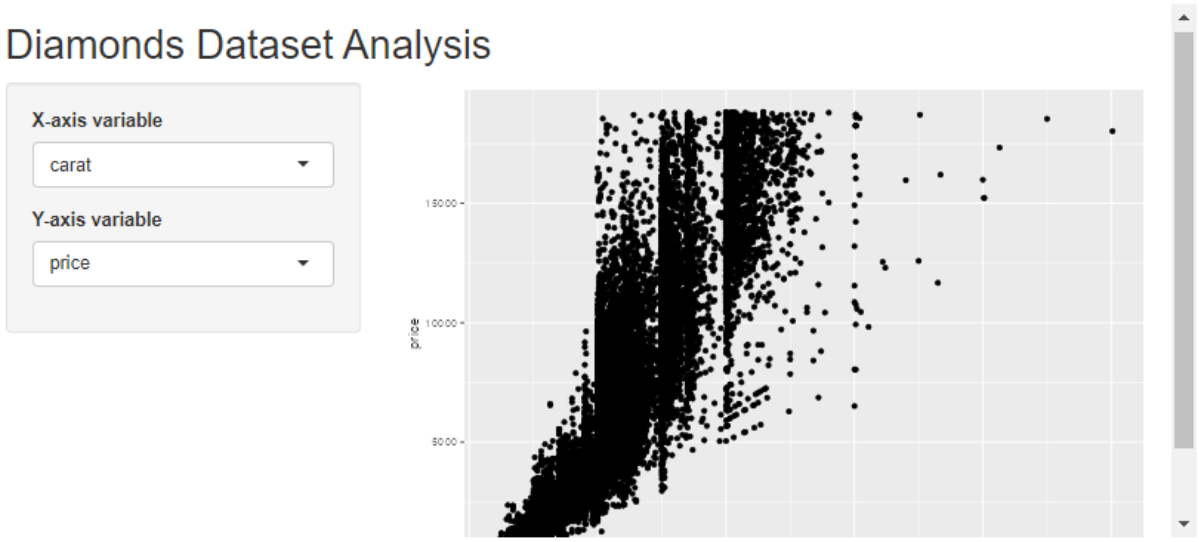
```
chatgpt_api_call <- function(input_text) {
 open_ai_key <- "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"
 request_body <- list(
  model = "gpt-3.5-turbo-0125",
  messages = list(
   list(role = "user", content = input_text)
  ),
  max_tokens = 500,
  temperature = 0.7
 )
 response <- tryCatch({
  httr::POST(
   url = "https://api.openai.com/v1/chat/completions",
   add_headers(
    "Content_Type" = "application/json",
    "Authorization" = paste("Bearer", open_ai_key)
   ),
   body = request_body,
   encode = "json"
  )
 }, error = function(e) {
  print(paste("Error:", e$message))
  NULL
 })

 if (!is.null(response)) {
  content(response)
 } else {
  NULL
 }
}
```

Step 5. Server for the Shiny App

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

  output$plot <- renderPlot({
    ggplot(diamonds, aes_string(x = input$x, y = input$y)) +
      geom_point()
  })

  # Interact with ChatGPT API
  observe({
    # Define the question for ChatGPT
    question <- "Based on the summary of the mtcars dataset, could you provide insights or analysis re
garding the relationship between fuel consumption and horsepower? Additionally, are there any notab
le patterns or outliers in the data that warrant further investigation?"

    # Call ChatGPT API
    chat_response <- chatgpt_api_call(question)

    # Print ChatGPT response
    if (!is.null(chat_response)) {
      print(chat_response)
    }
  })
}
```

Step 6. Run the application.

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



2. **CodeLingo**

CodeLingo was developed by Analytica Data Science Solutions to serve as a tool for facilitating code translation across different programming languages.

Here's a summary of the key points:

- **Functionality**: CodeLingo allows users to translate code between various programming languages, including Java, Python, JavaScript, C, C++, PHP, and R.
- **Integration with OpenAI API**: Users are required to input their OpenAI API key to use CodeLingo. This integration enables the application to leverage ChatGPT technology for code translation.

Use the previous code generated using TheOpenAir package to convert it to Python.

3. RTutor.ai

RTutor is an innovative R package designed for creating interactive tutorials and teaching materials using real-world datasets using ChatGPT and R. The application's open-source code is available on GitHub. It was developed by Dr. Steven Ge, a bioinformatics professor. While primarily designed for educational and non-commercial use, RTutor offers a valuable tool for streamlining data analysis workflows.

RTutor streamlines the coding process, making it accessible to users with some R experience. RTutor is a:

- GPT -4 powered data chatbot
- Good for exploratory and pleliminary analysis
- Interactive and reproduciable
- Free, locally as an R package

NB: We will use the online version because the offline version requires large memory and may hang during the session.

RTutor enables:

- Educators to develop engaging and interactive learning experiences for students by embedding R code, explanations, and exercises directly into the tutorials.
- Users can benefit from RTutor's functionality without requiring a ChatGPT API key for basic usage.
- Users to upload datasets, ask questions, and receive R or Python code along with visualizations.