Paper MH32

Integration of AI with R Studio: Advantages and Disadvantages

Mahesh Minnakanti, Vita Global Science, Waltham, US
Karthik Perala, Presbyterian Health Care, Albuquerque, US

ABSTRACT

Learning and writing code for any new programming language is a crucial yet daunting task characterized by its complexity, demanding significant time and effort from programmers. But with the recent AI evolution, it has become easy to understand and acquire code, at the same time it can be challenging to shuffle between AI and coding platforms. Having AI integrated with the coding platforms can make it more convenient and efficient. R can be more flexible for AI integration as it is an open-source tool. In this paper we will show ways to integrate AI with R studio, its usage for quick learning and efficiency, along with its advantages and disadvantages.

INTRODUCTION

There are always some challenges in learning new programming languages, but the all-new AI platforms simplified understanding and writing code. SAS and R play critical roles in the clinical industry, particularly in the field of clinical research. Both are widely used for statistical analysis, data management, and reporting in clinical trials. R is open source, which makes it a cost-effective alternative to SAS. R offers a vast library of packages for advanced statistical modeling and visualization. Also Known for creating high-quality, customizable visualizations using packages like ggplot2. Although not as widely accepted as SAS for regulatory submissions, its usage is growing as agencies, including the FDA, start accepting R-based analyses. Integration of AI with RStudio offers numerous advantages, such as improved efficiency, scalability, and the ability to handle complex datasets. However, it also comes with certain challenges, including computational resource demands, algorithm interpretability, and the need for technical expertise. In this paper, we will explore one of the few options for integrating AI with RStudio using available R packages, along with a few usage examples.

WAYS TO INTEGRATE AI INTO R:

There are several ways to integrate AI with R such as using Pre-built AI Packages in R for AI and machine learning, Connecting R with Python for AI Models, Using Cloud-Based AI Services, Developing Custom AI Models in R, Automating AI Workflows in R, Integrating AI with R for Data Processing & Visualization.

In this paper we will focus on integrating OpenAI ChatGPT in R studio by leveraging APIs for Advanced AI Models along with R packages like "**httr**" for API requests and "**chattr**" shiny package to interact with AI within RStudio.

PREREQUISITES FOR INTEGRATING AI INTO RSTUDIO:

To integrate AI with RStudio we need API access and authentication along with few R packages. **Application Programming Interface** referred to as API allows different software systems to communicate with each other. AI APIs (e.g., OpenAI's GPT, Google's Vertex AI) let you send data (like text or images) to a model and receive predictions or responses. API Keys are unique identifiers that authenticate and authorize your access to an API. Most AI services require authentication to ensure secure and controlled access. API keys help track usage and prevent unauthorized access. In this paper we will use Open AI's GPT to integrate with RStudio. API Key for Open AI can be found or created on your OpenAI website (https://platform.openai.com/settings/organization/api-keys).

R package "**httr**" is needed as it helps make HTTP requests (GET, POST, etc.), which is how we interact with APIs. R package like "**chattr"** can be helpful if integrating AI chat-based applications.

INTEGRATING AI INTO RSTUDIO USING CHATTR PACKAGE:

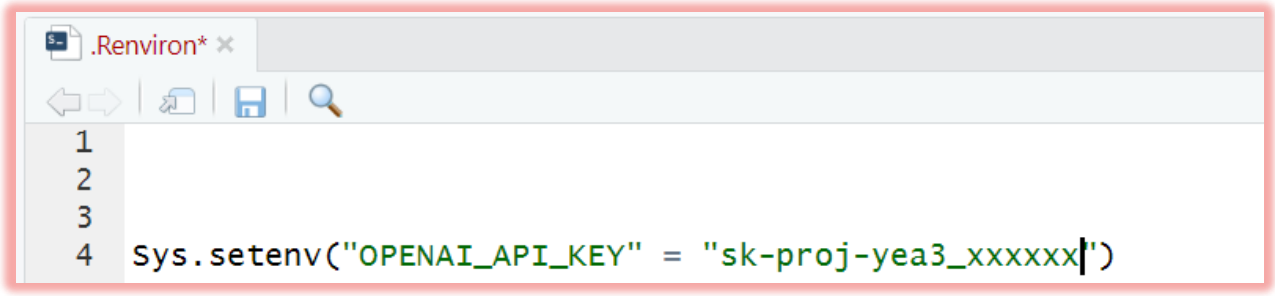
Integrating "chattr" into your RStudio workflow allows you to interact with AI using chattr's shiny App or by chattr prompts in R Console.

Step by step integration process:

- Install "chattr" package and call the library.

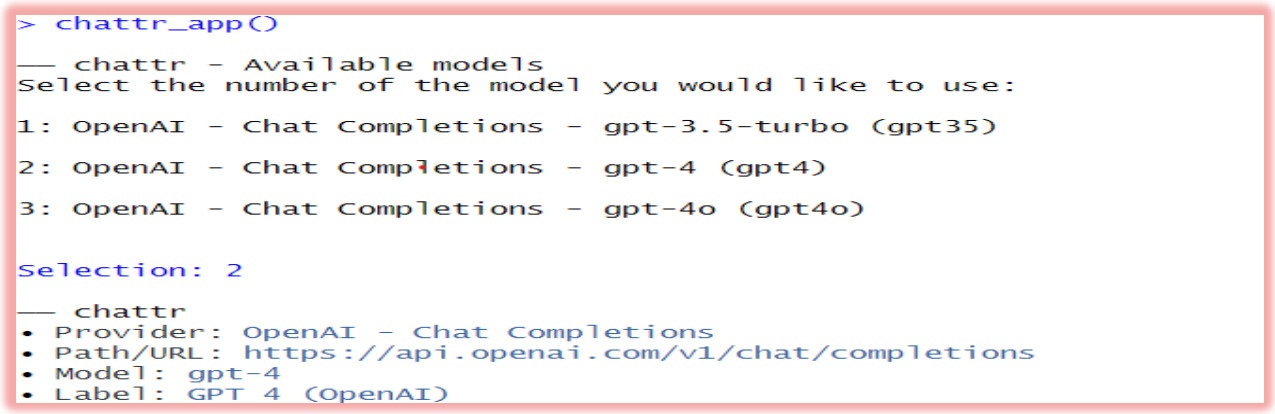
```
> install.packages("chattr")
> library(chattr)
```

- Authentication: To use the chattr package, you must authenticate with OpenAI by providing your API key. That can be stored under the OPENAI_API_KEY environment variable using Sys.setenv().

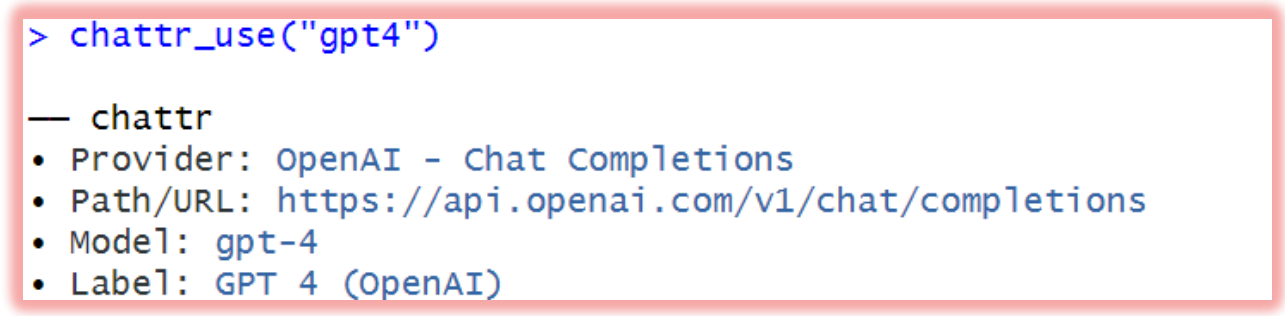


WAYS TO INTERTACT WITH AI USING CHATTR PACKAGE:
We can interact with AI using chattr application within R studio in multiple ways, one is through chattr_app shiny application and the other is using chattr prompts within RStudios console window.

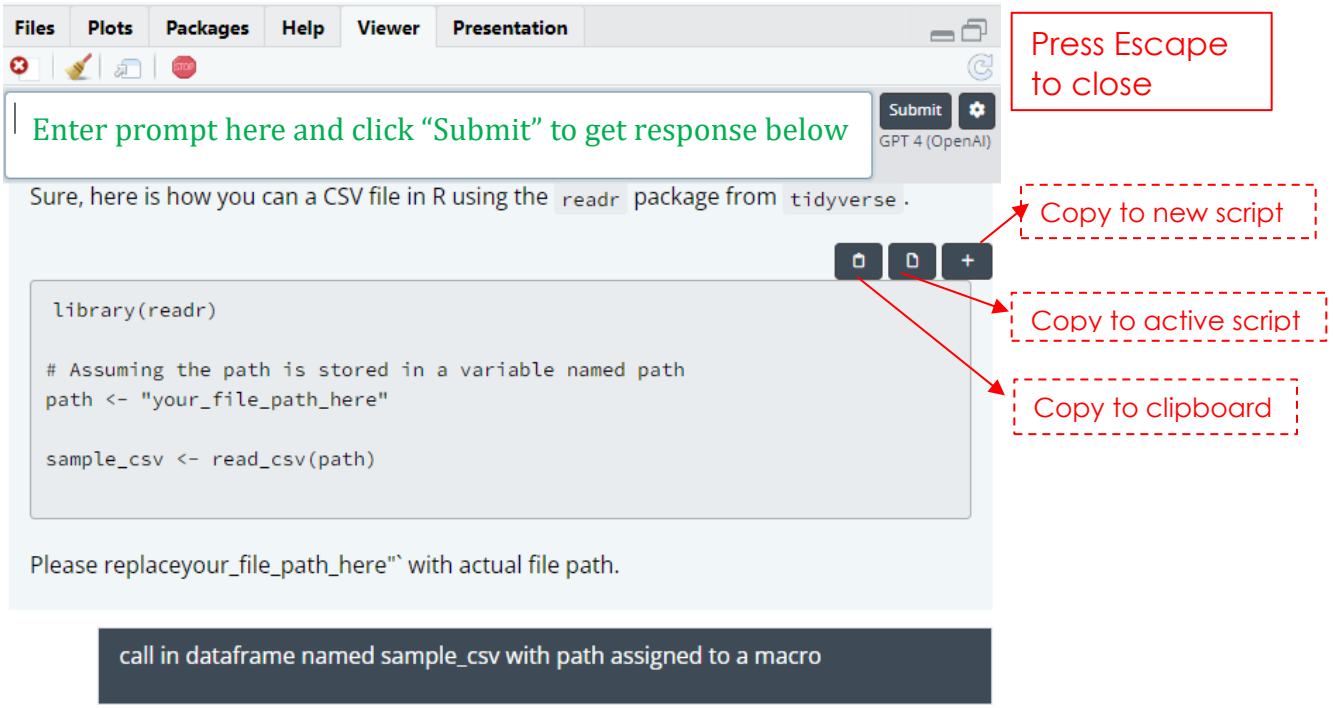
Interacting with chattr_app() application:
When ran chattr_app() for the first time, you'll be prompted to select your preferred model like shown in the below screenshot. Whichever model is selected will be used by default in subsequent sessions unless you change it manually. In the below example we have selected option 2 to use GPT version 4.



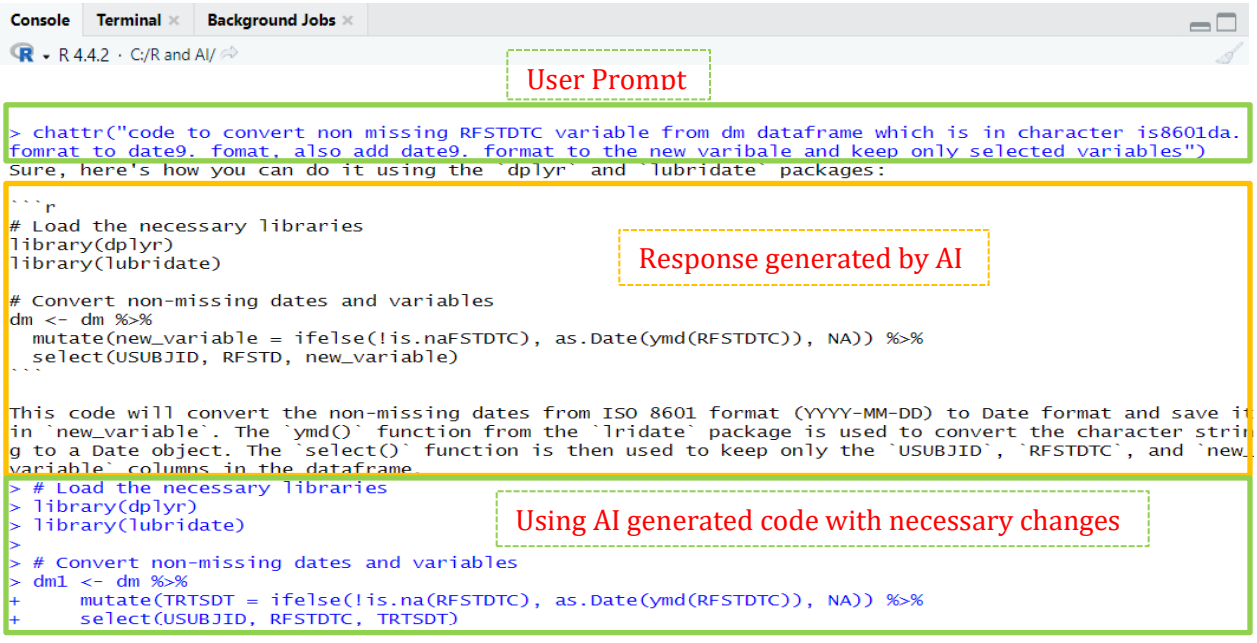
We can also manually assign the version as provided in the below screenshot by using chattr_use command.



Once the chattr_app() code is ran a prompt window is displayed in the bottom right viewer window as below. The viewer window present user-friendly options to copy AI generated code to the current or to the new script as needed. There are few other options under setting if we want to save the chat. The RStudio console code will be inactive while we are working on the shiny app and to activate the R console you need to hit escape.



Interacting with AI using chattr prompts within RStudios console window:
If we intend to stick within the R console to interact with AI and shift between chattr shiny application we can use chattr prompts by using command **chattr("prompt here")**, where the AI responses are generated in the console as shown in the below figure.



EXAMPLE OF DATA VISUALIZATION BY LEVERAGING INTEGRATED AI:

Along with data analysis and statistical modeling, R is widely used for data visualization due to the availability of powerful and flexible libraries like ggplot2, Shiny and Plotly. Though R Studio provides a user-friendly interface which makes programming easy, but it will be a heavy lift for the beginners who just start their journey in developing data visualization programming in R. By integrating AI into R Studio, it can significantly reduce the amount of time to learn the visualization packages and programming approaches to get the desired output.

BUILDING STATISTICAL PROCESS CONTROL (SPC) CHART:

SPC chart is like a series plot but has more statistical parameters like highlighted control limit area with a mean reference line while signifying outliers as necessary. This chart is plotted with the averages of Closed Coverage Gaps in a month for past 24 months. The sample data is also created using R and the mean values are manually randomized. For this type of data with Individual observations over a period, we typically use Individual – Moving Range (I-MR) chart to monitor the process over a period. To create I-MR chart in R we have listed a step-by-step approach on how we leveraged integrated AI to help build this chart.

Below is the R code used to create sample data for SPC chart:

```
start_date <- floor_date(today() - months(24), "month")
end_date <- floor_date(today() - months(1), "month")
Month <- seq(start_date, end_date, by = "month")

Coverage_Gap <- sample(3000:5000, length(Month), replace = TRUE)

df <- data.frame(Month, Coverage_Gap)
```

Initially we wanted to see what AI suggests based on the data we have and the packages that we can use to build the SPC chart.

User Prompt

```
> chattr('I have dataset with individual Mean values and dates, what is the best suggested SPC chart for this data')
```

Response generated by AI

````
For individual Mean values and dates, the best SPC chart would be the Individual () and Moving Range (MR) chart. This chart is suitable for tracking changes over time when your sample size is one. It's also useful when you can't rationally subgroup your data.

Here is how you can create an I-MR chart using the `qicharts2` package in R:

```R
# Install and load the packages
install.packages("qicharts2")
library(qicharts2)

# Assuming your data is in a data frame called df, with 'dates and 'mean_values' columns
# Convert dates to Date class
df$dates <- as.Date(df$dates)

# Create the I-MR chart
qic(mean_values, dates, data = df, chart = 'i', title = 'I-MR chart')
```

This will create an I-MR chart, which is a type of SPC chart suitable for tracking changes to the mean of a process over time. The `qic` function from the `qicharts2` package makes easy to create this and other types of SPC charts.
````

After installing the required packages, we prompted AI on how to verify if the “Month” variable is in date format or not.

```
> install.packages("qicharts2")
> library(qicharts2)
```

Using AI generated code with necessary changes

```
> chattr('how to check if the variable is in date format or not')
```

User Prompt

```
In R, you can use the `class()` function to check the data type of a variable. Here is an example:

R
# Assuming your data frame is df and variable is dates
class(df$dates)

This will return the class of the variable. If the variable is in date format, it will return "Date".
```

Response generated by AI

```
> class(df$Month)
[1] "POSIXct" "POSIXt"

> df$Month <- as.Date(df$Month)
> class(df$Month)
[1] "Date"
```

Using AI generated code with necessary changes

After confirming the format of the “Month” variable, the next step is to use the “qicharts2” package and build the SPC chart.

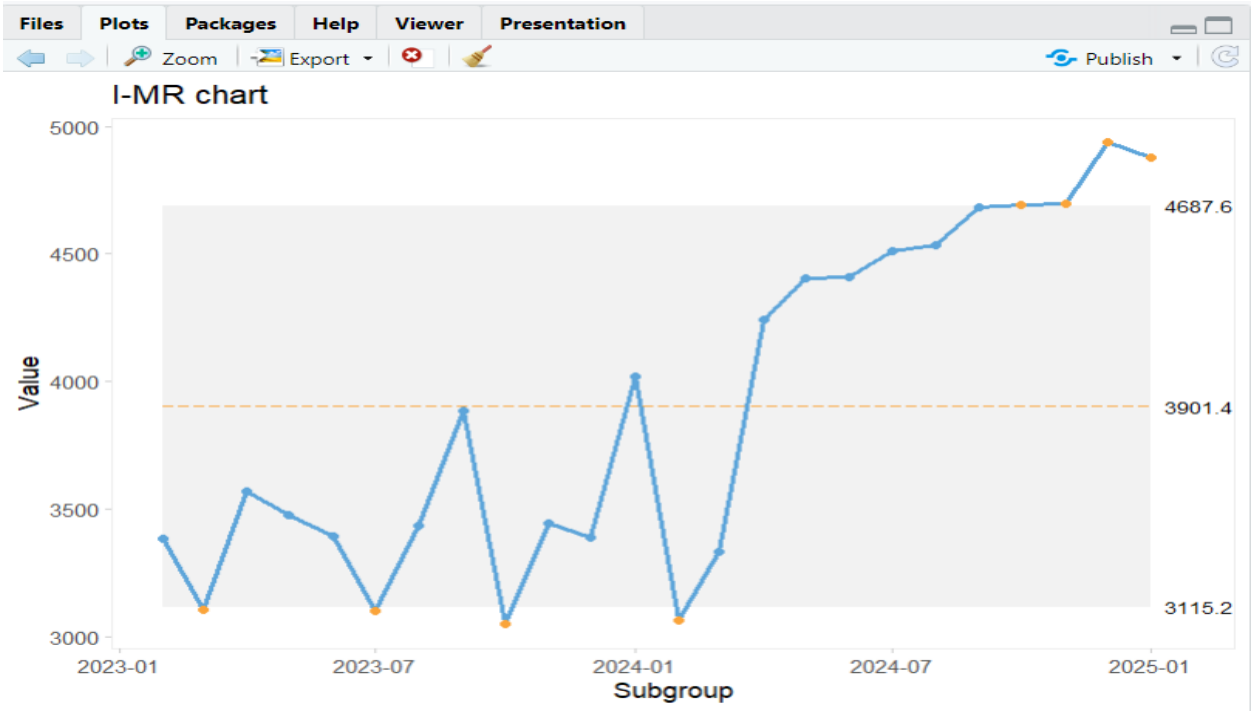
Code suggestion from Integrated AI:

```
# Load the necessary packages
library(qicharts2)

# Assuming your data is in a data frame called df, with 'dates' and 'mean_values' columns
df$dates <- as.Date(df$dates)

# Create the SPC chart
qic(mean_values, data = df, x = dates, chart = 'i', title = 'SPC chart')
```

Response generated by AI

Code to generate SPC chart based on AI suggestion in R console:

```
> qic(df$Coverage_Days, data=df, x = df$Month, chart = 'I', title = 'I-MR chart')
```



We want to change the axis labels and title of the chart that best conveys the chart information and on top of it we also wanted to adjust the title of the chart, add the data labels to the datapoints and adjust the size of the data labels to make sure that the SPC chart does not look overcrowded. AI suggested to use package “qicharts2” in combination with “ggplot2” as “qicharts2” does not have the functionality to add and control the size of data labels and adjust the label location in the chart.

- Adding Axis labels to the Chart:

```
> chattr('how to add axis labels to the plot in qic')
# Load the necessary packages
library(qicharts2)

# Assuming your data is in a data frame called df, with 'dates' and 'mean_values' columns
df$dates <- as.Date(df$dates)

# Create the SPC chart
qic(mean_values, data = df, x = dates, chart = 'i', title = 'SPC chart',
    ylab = "Your Y-Axis Label", xlab = "Your X-Axis Label")
> qic(df$Coverage_Days, data = df, x = df$Month, chart = 'i', title = 'I-MR chart for Closed Coverage Gap
s', ylab = "Closed_Coverage_Gaps", xlab = "Month")
```

Response generated by AI

- Moving the chart title to Center

```
> chattr('how to use ggplot2 along with qicharts2 to move title to the center of the plot')
# Load the necessary packages
library(qicharts2)
library(ggplot2)

# Assuming your data is in a data frame called df, with 'dates' and 'mean_values' columns
df$dates <- as.Date(df$dates)

# Create the SPC chart using qic and convert it to a ggplot object
spc_plot <- q(mean_values, data = df, x = dates, chart = 'i')$plot

# Add title and center it
spc_plot + ggtitle("SPC Chart") + theme(plot.title = element_text(hjust = 0.5))
```

User Prompt

Response generated by AI

```
> install.packages('ggplot2')
> library(ggplot2)

> plot <- qic(df$Coverage_Days, data = df, x = df$Month, chart = 'i', title = 'I-MR chart for Closed Cove
rage Gaps', ylab = "Closed_Coverage_Gaps", xlab = "Month") + theme(plot.title = element_text(hjust =0.5))
>
>
>
> print(plot)
```

- Adjusting size of data labels

Response generated by AI

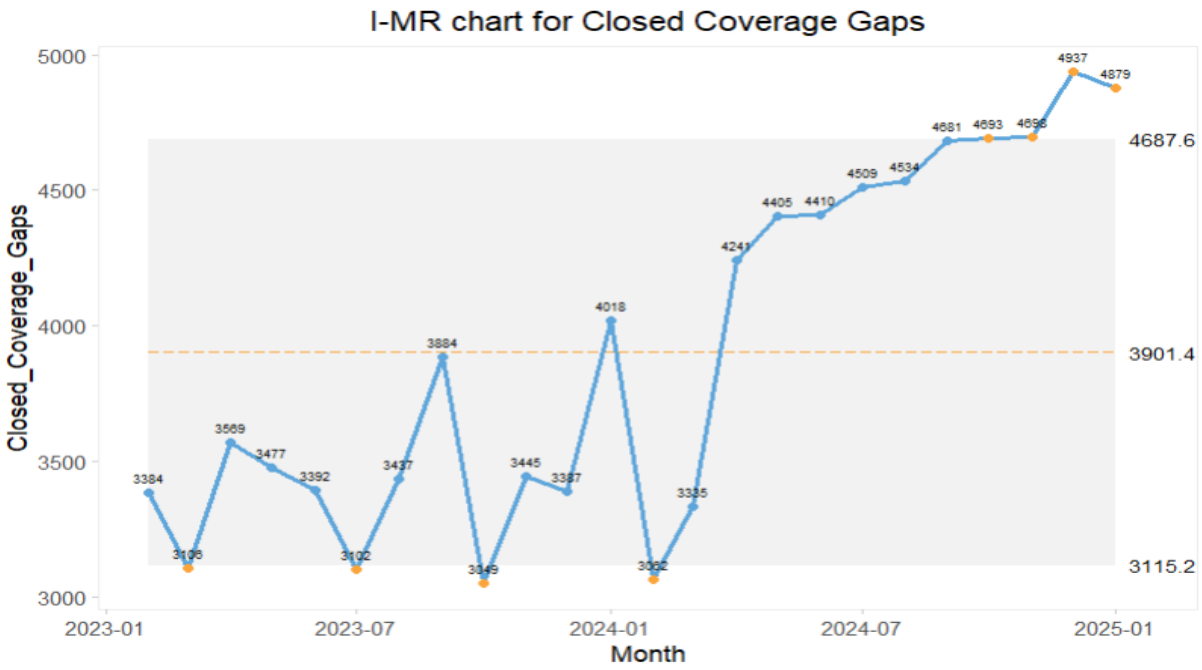
```
# Create plot
ggplot(df, aes(x = dates, y = mean_values)) +
  geom_line() +
  geom_point() +
  geom_text(aes(label = mean_values), vjust = -1, size = 3)


This will create a line plot with points for each data point and labels for each point. The `vjust = -1`
argument in `geom_text` adjusts the vertical position of the labels to avoid overlapping with the points.
The `size = 3` argument reduces the size of the labels.
```

```
> plot + geom_text(aes(label = df$Coverage_Days), vjust = -1, size = 2)
```

Below is the overall combined code from the above screenshots to generate the SPC chart:

```
> install.packages('qicharts2')
> install.packages('ggplot2')
> library(qicharts2)
> library(ggplot2)
> Plot <- qic(df$Coverage_Days, data=df, x = df$Month, chart = 'I', title = 'I-MR
chart      for Closed Coverage Gaps', ylab = "Closed_Coverage_Gaps", xlab = "Month") +
theme(plot.title = element_text(hjust = 0.5))
> plot + geom_text(aes(label = df$Coverage_Days), vjust = -1, size = 2 )
```

Final Output:



ADVANTAGES AND DISADVANTAGES:

ADVANTAGES:

- Improves Productivity: Based on the requirement, AI can provide code suggestions, create code with the specified parameters, simplify adding comments to the code and suggest code introduction based on whole block of code, this will improve the efficiency and reduce the amount of time in code development.
- Efficient Code: AI can provide suggestions in writing efficient code by suggesting good practices and styles
- Quick Learning: using AI we can learn new techniques based on the AI suggested code. AI can provide information on the packages and functions programmers are unaware of which can be applied to the solutions developers are looking for in a project.
- Code debugging: AI can easily identify if there are any inconsistencies in the code and suggest the necessary changes that needs to be applied to the code. AI can help the code validator to easily understand the whole block of code, which makes the reviewer job easy.
- Simplifying Complex Tasks: We can take AI's help in laying down step by step process in talking complex tasks like building statistical models, data visualization.

DISADVANTAGES:

However, while AI offers tremendous benefits, its improper implementation can lead to significant challenges if not used in a standardized and regulated manner.

- Data Privacy and Security: We must be cognizant while we are interacting with AI, as the prompts that we ask AI can be used to re-engineer their AI models. We cannot blindly trust the information provided by AI as sometimes it will provide the information from unknow sources and by usage of such information might lead to copyright and plagiarism issues, so it's better to verify the references when we use the information provided by AI.

- Associated Costs: Though in the preliminary stages most of the AI platforms provide their AI models for free but once the AI models are sound, they will charge us based on the usage. Also, to run and maintain the AI integration the device we use need to have high computational power if not the R Studio will slowdown.

Cost Associated with AI Integration:

Unlike some AI platforms like Google cloud AI, IBM Watson, hugging face and some others provide their AI models for free with limited number of API calls. We have used Open AI's Chat-GPT4 model for our paper, and they charge 3 cents for 1000 input (the Information we ask AI) tokens and 6 cents for 1000 output (Information that AI provides to us). Each individual word (Hello, World), sub words (Able, Un, or) and individual characters (a, I, 1, ?) are considered as one token. Overall cost is calculated in combination of number of tokens we provide and number of tokens we receive as a response.

CONCLUSION

The integration of AI into RStudio offers significant advantages, making it a powerful tool for data scientists, researchers, and analysts. The combination of R's statistical capabilities with AI techniques allows for more advanced analytics and decision-making. The continuous advancements in AI and the growing support for AI integration in R Studio make it an increasingly viable and beneficial approach for data-driven problem-solving. With the right tools and expertise, users can maximize the potential of AI within the RStudio environment. However, this integration also presents challenges. The need for high computational resources, the complexity of AI model interpretation, and the requirement for technical expertise can create barriers for some users.

REFERENCES

https://openai.com/chatgpt/pricing/
https://help.openai.com/en/articles/4936850-where-do-i-find-my-openai-api-key
https://platform.openai.com/api-keys
https://blogs.rstudio.com/ai/posts/2024-04-04-chat-with-llms-using-chattr/?utm_source=chatgpt.com
https://intro2r.library.duke.edu/ai.html?utm_source=chatgpt.com

CONTACT INFORMATION

Your comments and questions are valued and encouraged. Contact the author at:

Author Name: Mahesh Minnakanti
Company: Vita Global Sciences
Address: 281 Winter St, Waltham, MA
Email: Minnakantimahesh@gmail.com