



Vita Global Sciences

FROM RESEARCH TO REALITY

Integration of AI with R Studio: Advantages and Disadvantages

PHUSE US Connect 2025

**By :— Mahesh Minnakanti
Karthik Perala**



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.

Ways to Integrate AI into R

Integrating artificial intelligence (AI) into R can significantly enhance data analysis, modeling, and application development capabilities. Below are few approaches to achieve this integration:

- Utilizing AI-Focused R Packages such as **caret**, **RandomForest**, etc.
- **Leveraging APIs for Advanced AI Models.**
- **Enhancing RStudio with AI-Powered Tools.**
- Implementing AI in Data Analysis Workflows.

Prerequisites for Integrating AI into RStudio Using

➤ API Access and Authentication

- API Key: API key can be generated from each individual's OpenAI account.
- Environment Setup: Store API keys securely using **Sys.setenv()** or **.Renv** file.

➤ Required R Packages

- **httr** (for API requests)
- **chatr** (if integrating AI chat-based applications)

➤ Optional R packages to make the integration user friendly

- Utlis
- Crayon

Integrating AI into RStudio Using chattr package:

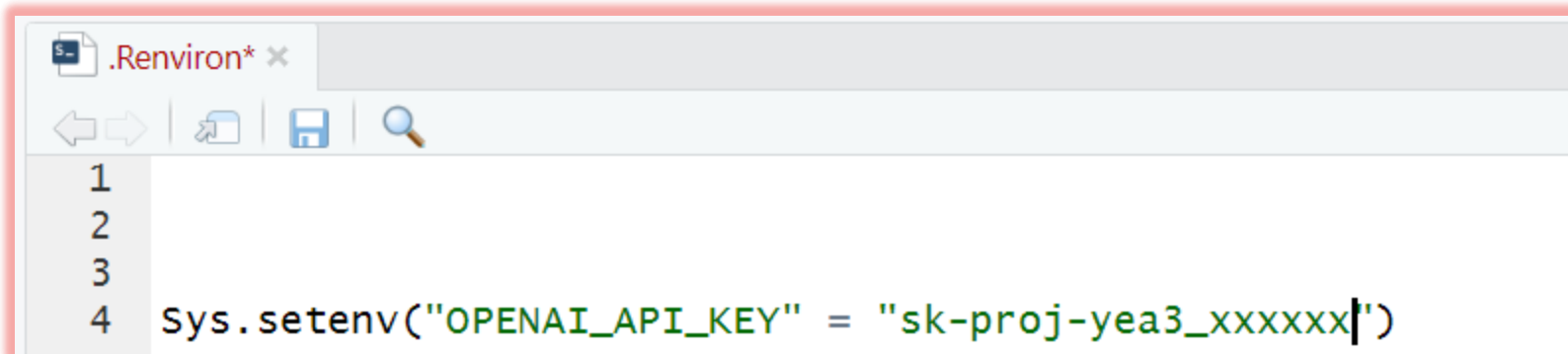
- Integrating "**chattr**" into your RStudio workflow allows you to interact with AI using chattr's shiny App or by **chattr** prompts in R Console.

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()**.

A screenshot of the RStudio editor window. The title bar shows ".Renviron* x". The toolbar includes icons for back, forward, copy, paste, save, and search. The editor area shows a script with four lines: line 1 is empty, line 2 is empty, line 3 is empty, and line 4 contains the code `Sys.setenv("OPENAI_API_KEY" = "sk-proj-yea3_XXXXXX")`. The code is highlighted in green.

```
1  
2  
3  
4 Sys.setenv("OPENAI_API_KEY" = "sk-proj-yea3_XXXXXX")
```

Integrating AI into RStudio Using chattr package:

- When ran `chattr_app()` for the first time, you'll be prompted to select your preferred model. This model will be used by default in subsequent sessions unless you change it manually.

```
> chattr_app()
— chattr - Available models
Select the number of the model you would like to use:

1: OpenAI - Chat Completions - gpt-3.5-turbo (gpt35)
2: OpenAI - Chat Completions - gpt-4 (gpt4)
3: OpenAI - Chat Completions - gpt-4o (gpt4o)

Selection: 2

— chattr
• Provider: OpenAI - Chat Completions
• Path/URL: https://api.openai.com/v1/chat/completions
• Model: gpt-4
• Label: GPT 4 (OpenAI)
```

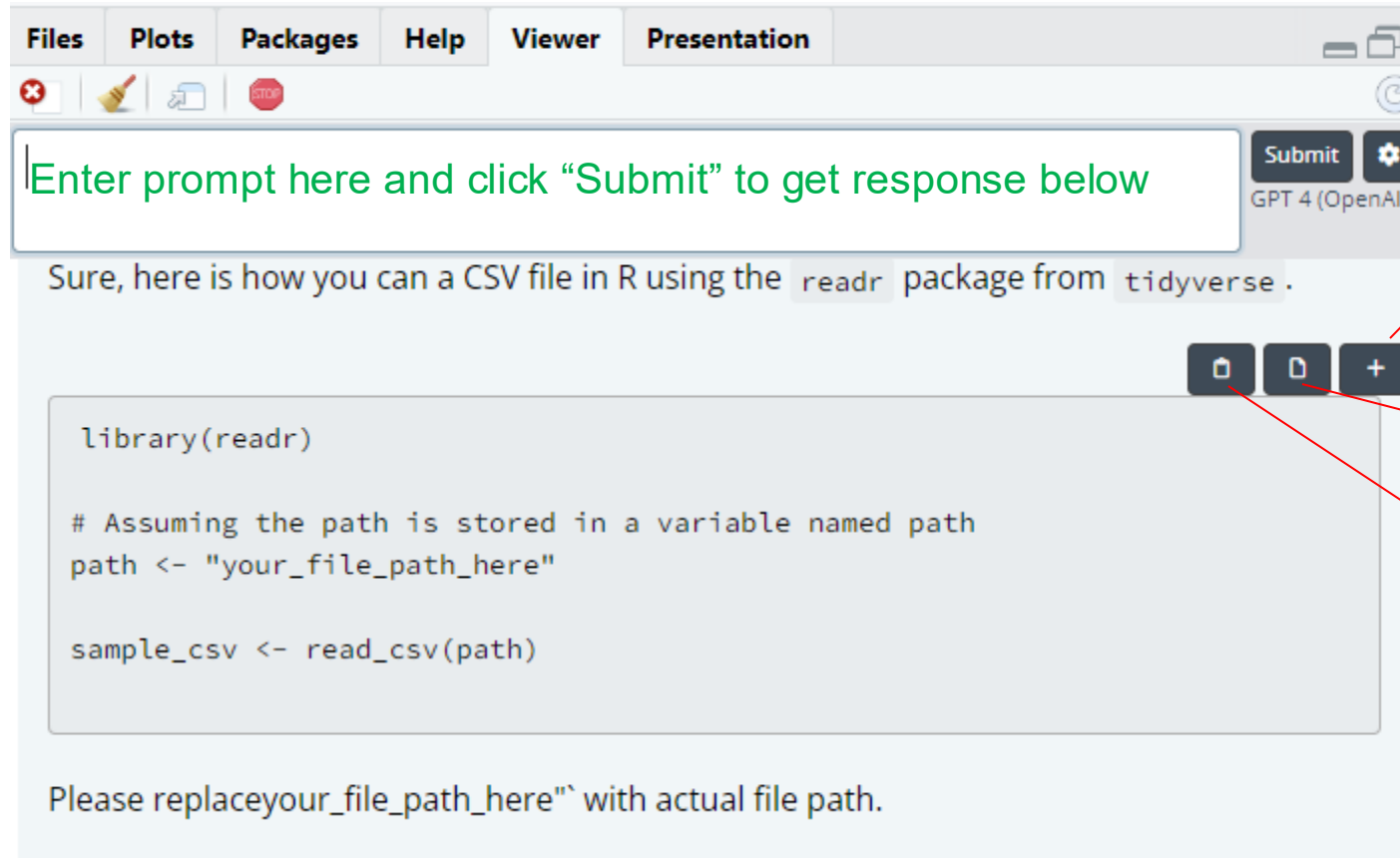
- Manually assigning preferred model.

```
> chattr_use("gpt4")

— chattr
• Provider: OpenAI - Chat Completions
• Path/URL: https://api.openai.com/v1/chat/completions
• Model: gpt-4
• Label: GPT 4 (OpenAI)
```

Ways to Interact with AI after Integration

- Interacting with AI using chattr's shiny app



Press Escape to close

Copy to new script

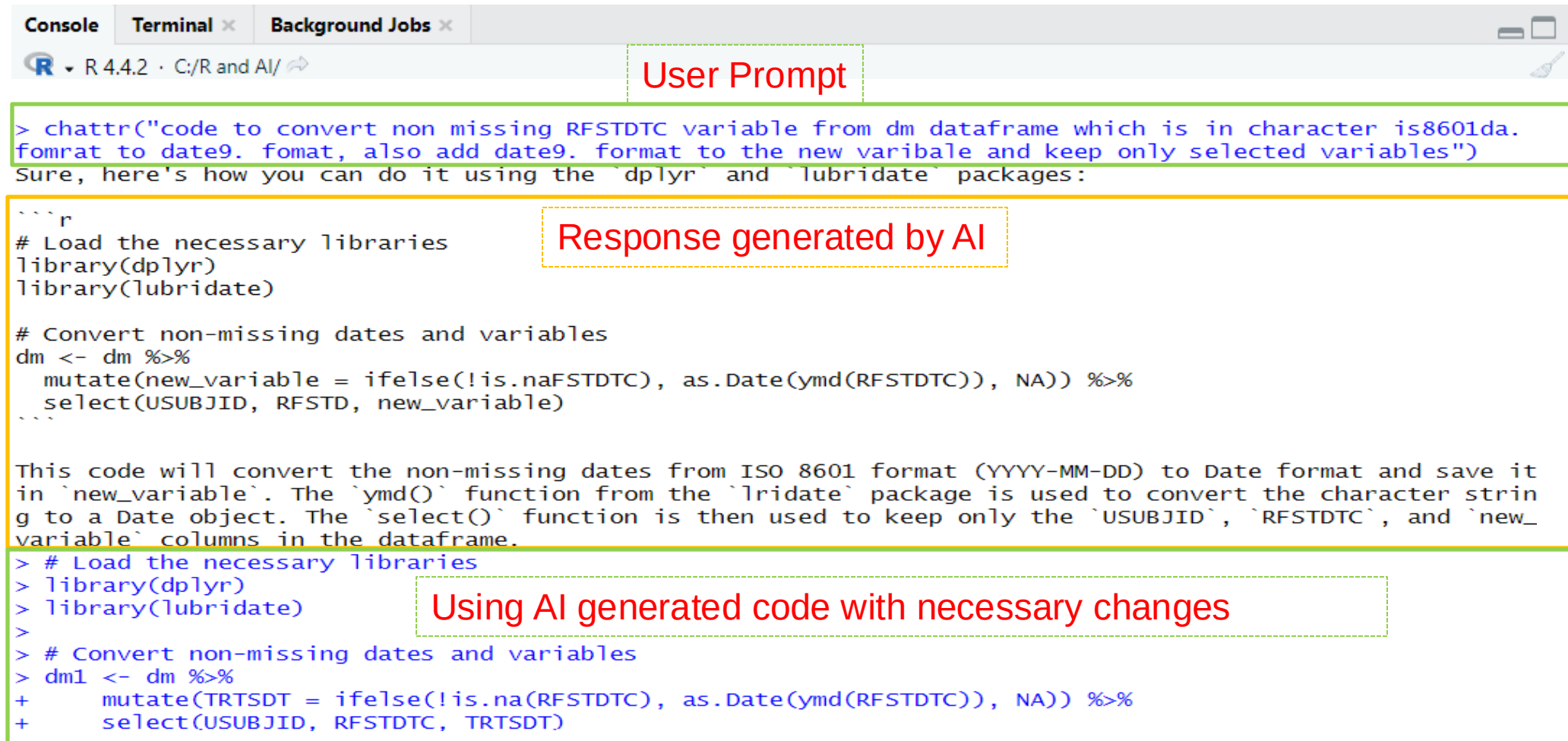
Copy to active script

Copy to clipboard

call in dataframe named sample_csv with path assigned to a macro

Ways to Interact with AI after Integration

- Interacting with AI using chattr prompts within RStudios console window.



The screenshot shows the RStudio interface with the 'Console' tab selected. The top bar indicates 'R 4.4.2 · C:/R and AI/'. A red box labeled 'User Prompt' highlights the input text in the console. Below it, a green box labeled 'Response generated by AI' contains the AI's output, which includes R code and an explanation. A third green box labeled 'Using AI generated code with necessary changes' shows the user's modified R code.

```
> chattr("code to convert non missing RFSTDTC variable from dm dataframe which is in character is8601da. fomrat to date9. fomrat, also add date9. format to the new varibale and keep only selected variables")
Sure, here's how you can do it using the `dplyr` and `lubridate` packages:
```

```
```r
Load the necessary libraries
library(dplyr)
library(lubridate)

Convert non-missing dates and variables
dm <- dm %>%
 mutate(new_variable = ifelse(!is.na(RFSTDTC), as.Date(ymd(RFSTDTC)), NA)) %>%
 select(USUBJID, RFSTD, new_variable)
```
```

This code will convert the non-missing dates from ISO 8601 format (YYYY-MM-DD) to Date format and save it in `new\_variable`. The `ymd()` function from the `lubridate` package is used to convert the character string to a Date object. The `select()` function is then used to keep only the `USUBJID`, `RFSTDTC`, and `new\_variable` columns in the dataframe.

```
> # Load the necessary libraries
> library(dplyr)
> library(lubridate)
>
> # Convert non-missing dates and variables
> dm1 <- dm %>%
+   mutate(TRTSDT = ifelse(!is.na(RFSTDTC), as.Date(ymd(RFSTDTC)), NA)) %>%
+   select(USUBJID, RFSTDTC, TRTSDT)
```

Accelerated Learning Basic Examples:

The screenshot displays the RStudio interface with the following components:

- Console:** Shows the installation of the 'crayon' package and the configuration of the 'chatr' package for OpenAI chat completions. The output includes the installation path, the download of the binary package, and the successful connection to the OpenAI API.
- Environment:** Displays the global environment with the 'response' object, which is a list of 10 values. The 'my_api_key' and 'OPENAI_API_KEY' are both set to the same value: "sk-proj-yea3_sq0Era0ECewHkFQOYo75dnKL65vQ-vwv3161vNQI7GpZF...".
- Files:** Shows the 'chatr' package installed in the user library.
- Plots:** Empty.
- Packages:** Shows the 'chatr' package installed in the user library.
- Help:** Empty.
- Viewer:** Empty.
- Presentation:** Empty.

```
R is a collaborative project with many contributors.
Type 'contributors()' for more information and
'citation()' on how to cite R or R packages in publications.

Type 'demo()' for some demos, 'help()' for on-line help, or
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.

Installing package into 'C:/Users/chunc/AppData/Local/R/win-library/4.4'
(as 'lib' is unspecified)
--- Please select a CRAN mirror for use in this session ---
trying URL 'https://cloud.r-project.org/bin/windows/contrib/4.4/crayon_1.5.3.zip'
Content type 'application/zip' length 166861 bytes (162 KB)
downloaded 162 KB

package 'crayon' successfully unpacked and MD5 sums checked

The downloaded binary packages are in
C:\Users\chunc\AppData\Local\Temp\RtmpQhDAEM\downloaded_packages

Do you want to connect to OpenAI?
1: No
2: Yes

[workspace loaded from ~/.RData]

> 2
Connected to OpenAI successfully!
> chatr("Assign work folder")

— chatr - Available models
Select the number of the model you would like to use:

1: OpenAI - Chat Completions - gpt-3.5-turbo (gpt35)
2: OpenAI - Chat Completions - gpt-4 (gpt4)
3: OpenAI - Chat Completions - gpt-4o (gpt4o)

Selection: 2

— chatr
• Provider: OpenAI - Chat Completions
• Path/URL: https://api.openai.com/v1/chat/completions
• Model: gpt-4
• Label: GPT 4 (OpenAI)
Sure, here is how you can set your working directory in R:

```R
working directory
setwd("/path/to/your/directory")
```

> # working directory
> setwd("C:/Users/chunc/Downloads/SDTM-MSG_v2.0_Sample_Submission_Package/m5/datasets/cdiscpilot01/tabulations/sdtm")
>
```

| Name | Description | Version |
|------------|---|---------|
| askpass | Password Entry Utilities for R, Git, and SSH | 1.2.0 |
| assertthat | Easy Pre and Post Assertions | 0.2.1 |
| backports | Reimplementations of Functions Introduced Since R-3.0.0 | 1.5.0 |
| base64enc | Tools for base64 encoding | 0.1-3 |
| bench | High Precision Timing of R Expressions | 1.1.4 |
| bit | Classes and Methods for Fast Memory-Efficient Boolean Selections | 4.0.5 |
| bit64 | A S3 Class for Vectors of 64bit Integers | 4.0.5 |
| blob | A Simple S3 Class for Representing Vectors of Binary Data ('BLOBs') | 1.2.4 |
| brew | Templating Framework for Report Generation | 1.0-10 |
| brio | Basic R Input Output | 1.1.5 |
| broom | Convert Statistical Objects into Tidy Tibbles | 1.0.7 |
| bsicons | Easily Work with 'Bootstrap' Icons | 0.1.2 |
| bslib | Custom 'Bootstrap' 'Sass' Themes for 'shiny' and 'rmarkdown' | 0.8.0 |
| cachem | Cache R Objects with Automatic Pruning | 1.1.0 |
| callr | Call R from R | 3.7.6 |
| caret | Classification and Regression Training | 7.0-1 |
| cellranger | Translate Spreadsheet Cell Ranges to Rows and Columns | 1.1.0 |
| chatr | Interact with Large Language Models in 'RStudio' | 0.2.1 |
| checkmate | Fast and Versatile Argument Checks | 2.3.2 |
| cli | Helpers for Developing Command Line Interfaces | 3.6.3 |
| clipr | Read and Write from the System Clipboard | 0.8.0 |
| clisymbols | Unicode Symbols at the R Prompt | 1.2.0 |

Accelerated Learning for Data Visualization

Building SPC Chart in R:

User Prompt

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

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:

Response generated by AI

```
```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.

# Building SPC Chart in R:

```
> 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)
...
```

Response generated by AI

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

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

Using AI generated code with necessary changes

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

# Building SPC Chart in R:

Using AI generated code with necessary changes

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

```
Error in Summary.Date(19631, na.rm = FALSE) :
 sum not defined for "Date" objects
```

```
> chattr('Getting error: sum not defined for "Date" objects')
```

User Prompt

It seems like you're trying to perform a sum operation on a Date object, which is not allowed in R. If you want to calculate the mean of a Date object, you can convert it to a numeric value first. Here's how you can do it:

Response generated by AI

```
> chattr('how to add x axis variable in qic')
```

User Prompt

```
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)
```

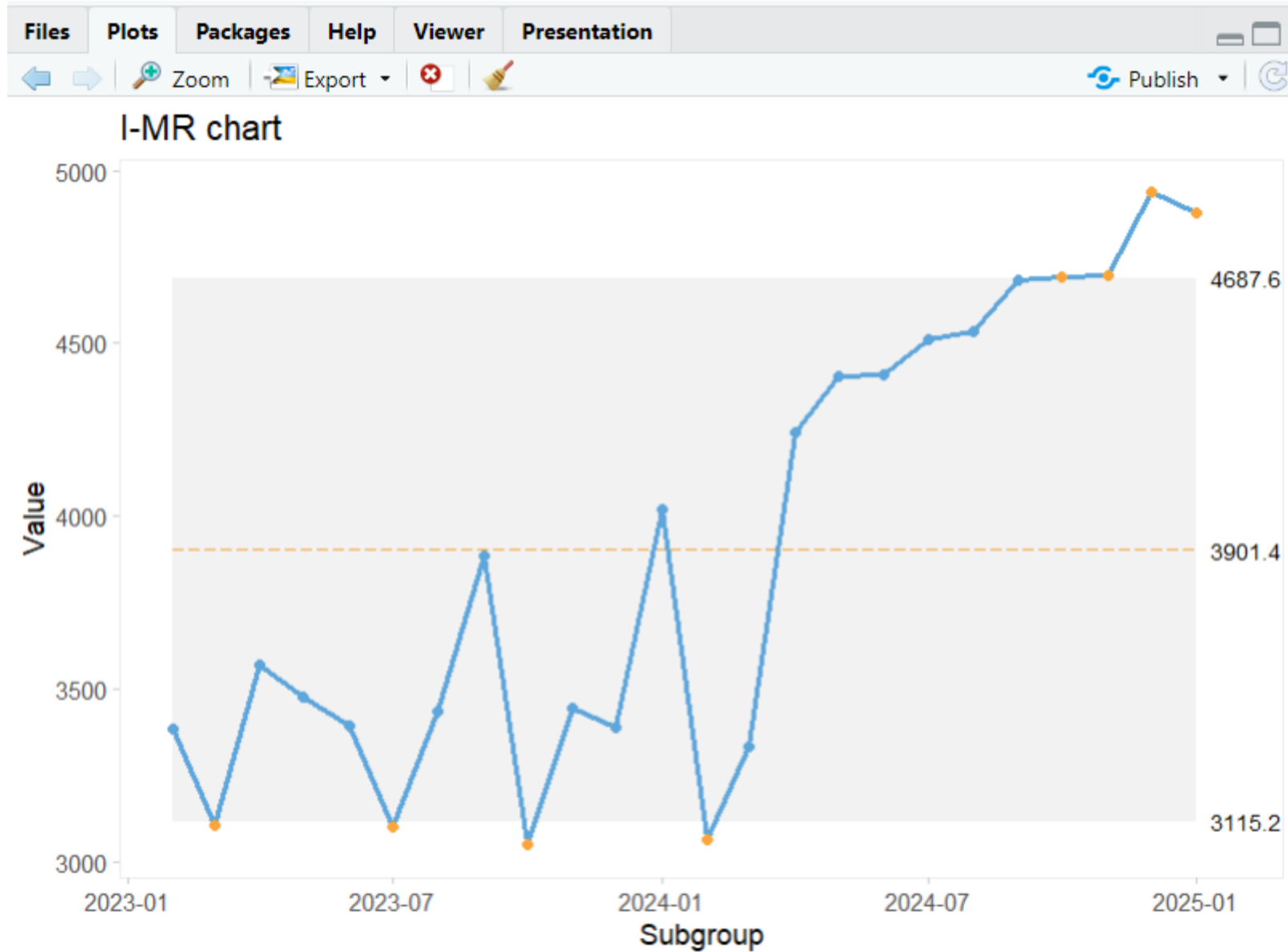
Response generated by AI

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

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

Using AI generated code with necessary changes

# Individual Moving Range Chart:



# Adding Axis Labels to Chart

User Prompt

```
> 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)
```

Response generated by AI

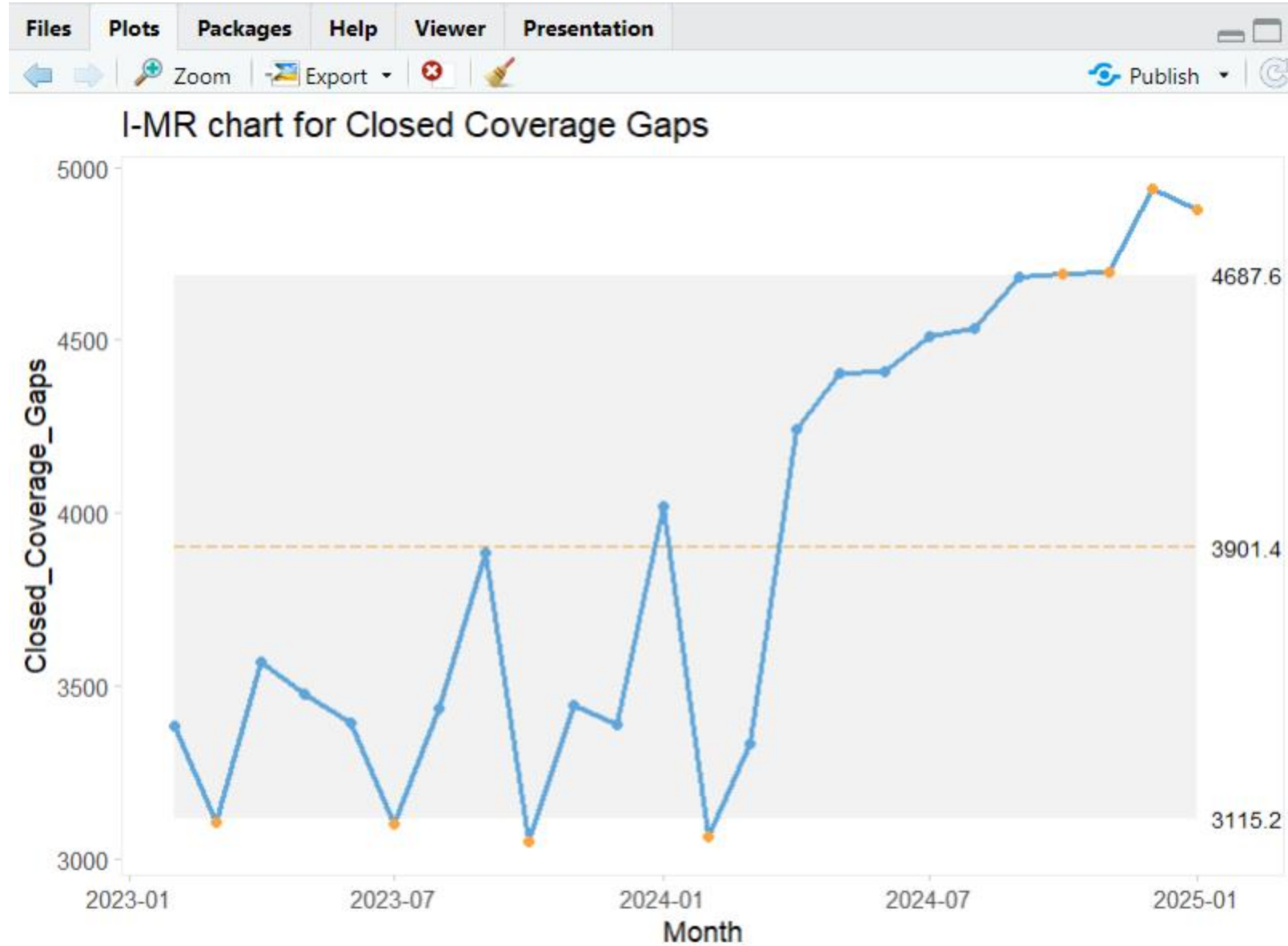
```
Create the SPC chart
```

```
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 Gaps',
 ylab = "Closed_Coverage_Gaps", xlab = "Month")
```

Using AI generated code with necessary changes

# Chart with New Axis Labels



# Moving Title to Center of the Chart

```
> chattr('how to move the title to center of the plot in qic')
```

User Prompt

```
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)
```

Response generated by AI

```
Create the SPC chart
plot <- qic(mean_values, data = df, x = dates, chart = 'i', title = 'SPC chart',
 ylab = "Your Y-Axis Label", xlab = "Your X-Axis Label")
```

```
Move the title to the center
+ theme(plot.title = element_text(hjust = 0.5))
```

```
> 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))
Error in theme(plot.title = element_text(hjust = 0.5)) :
 could not find function "theme"
```

Using AI generated code with necessary changes

```
> chattr('Error: could not find function "theme"')
```

User Prompt

```
I apologize for confusion. The `theme` is part of the `ggplot2` package, which is not when using `qicharts2`. Unfortunately, `qicharts2` does not provide a built-in way to center title.
```

# Moving Title to Center of the Chart

```
> chattr('how to use ggplot2 along with qicharts2 to move title to the center of the plot')
```

User Prompt

```
Load the necessary packages
```

```
library(qicharts2)
```

```
library(ggplot2)
```

Response generated by AI

```
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))
```

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

```
> 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))
```

```
>
```

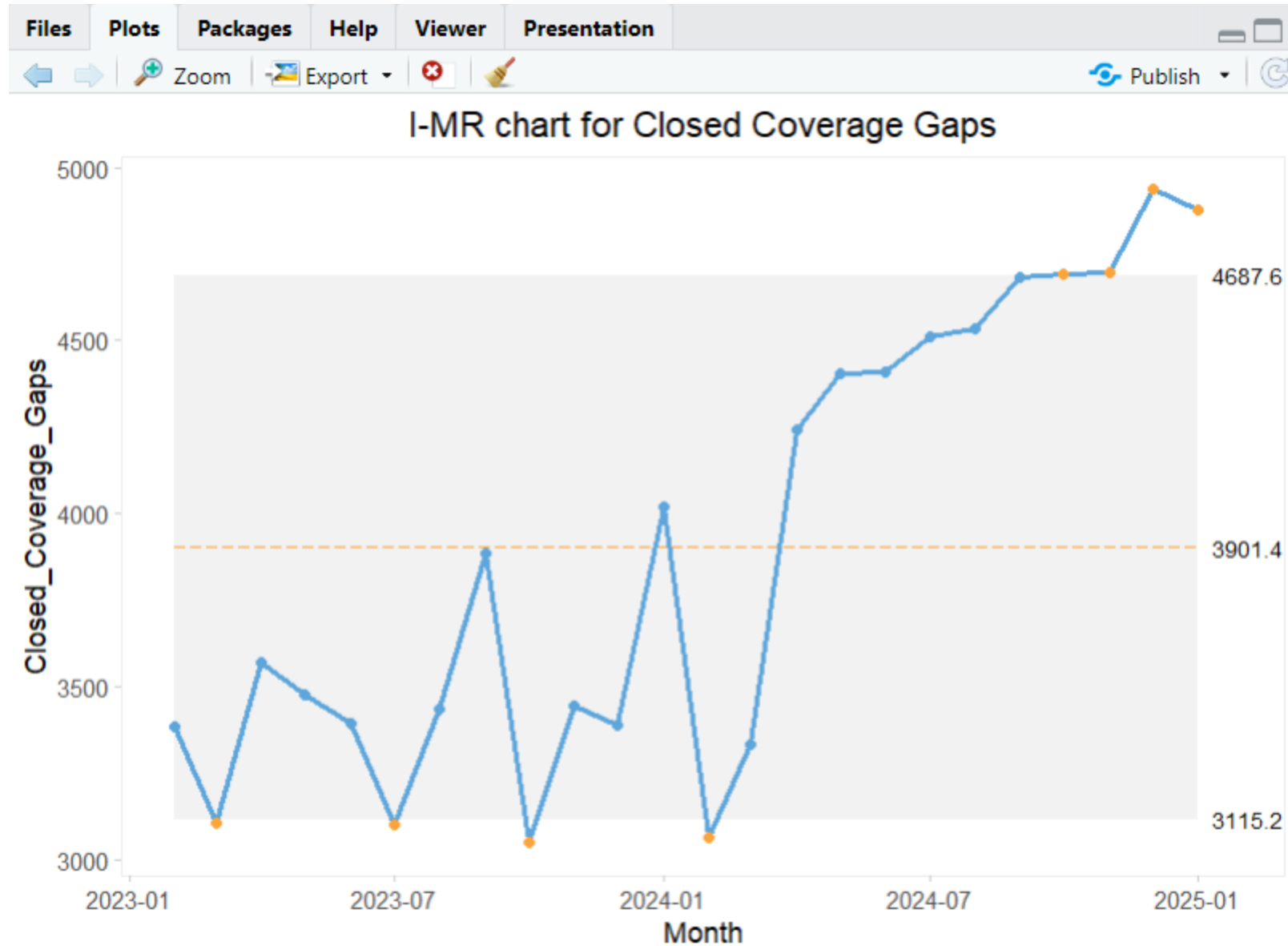
```
>
```

```
>
```

```
> print(plot)
```

Using AI generated code with necessary changes

# Moving Title to Center of the Chart:



# Adding Data Labels to the Chart:

User Prompt

```
> chattr('how to add data labels to the plot in qic')
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 data labels
spc_plot + geom_textes(label = mean_values, vjust = -1)
```

Response generated by AI

```
> plot + geom_textes((label = df$Coverage_Days), vjust = -1)
Error in geom_textes((label = df$Coverage_Days), vjust = -1) :
 could not find function "geom_textes"
```

Using AI generated code with necessary changes

# Adding Data Labels to the Chart:

```
> chattr('error with the code: geom_textes(label = df$Coverage_Days), vjust = -1)')
```

User Prompt

I apologize for confusion. It seems like there's a typo in the function name. It should be `geom\_text`, not `geom\_textes`. Here's the corrected code:

```
```R
# Load the necessary packages
(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 plot
gg(df, aes(x = dates, y = mean_values)) +
  geom_line() +
  geom_point() geom_text(aes(label = mean_values), vjust = -1)
```
```

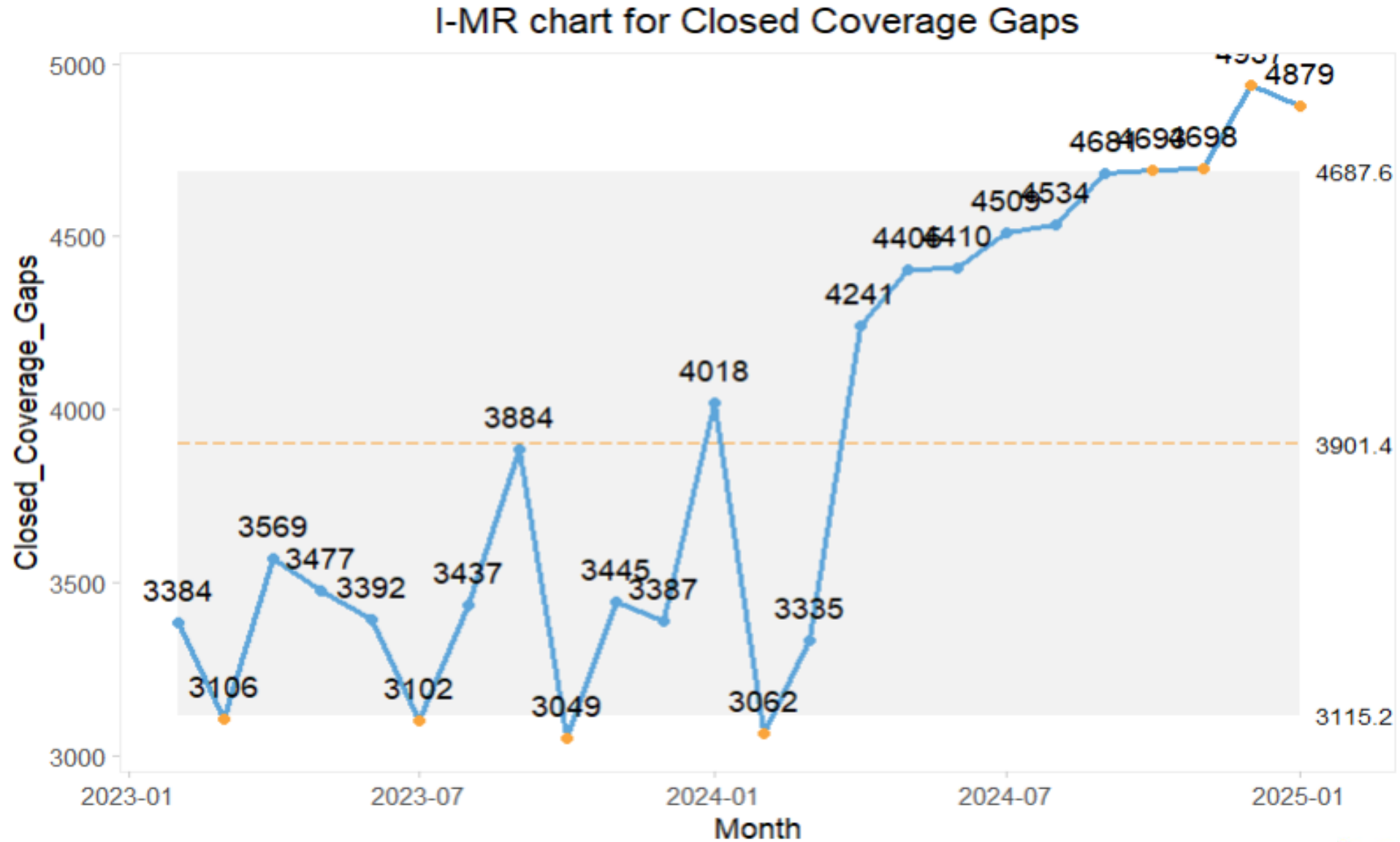
Response generated by AI

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.

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

Using AI generated code with necessary changes

# Adding Data Labels to the Chart:



# Reducing Data Labels Size:

```
> chattr('how to reduce the size of data labels with qic')
```

User Prompt

```
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
```

Response generated by AI

```
Add data labels with reduced size
spc_plot + geom_text(aes(label = mean_values), vjust = -1 size = 3)
```

```
> plot + geom_text(aes(label = df$Coverage_Days), vjust = -1 size = 3)
```

```
Error: unexpected symbol in "plot + geom_text(aes(label = df$Coverage_Days), vjust = -1 size"
```

Using AI generated code with necessary changes

# Reducing Data Labels Size:

User Prompt

```
> chattr('error in code: geom_text(aes(label = df$Coverage_Days), vjust = -1 size = 3)')
```

I apologize for confusion. It seems like there's a missing comma in the function arguments. Here's the corrected code:

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

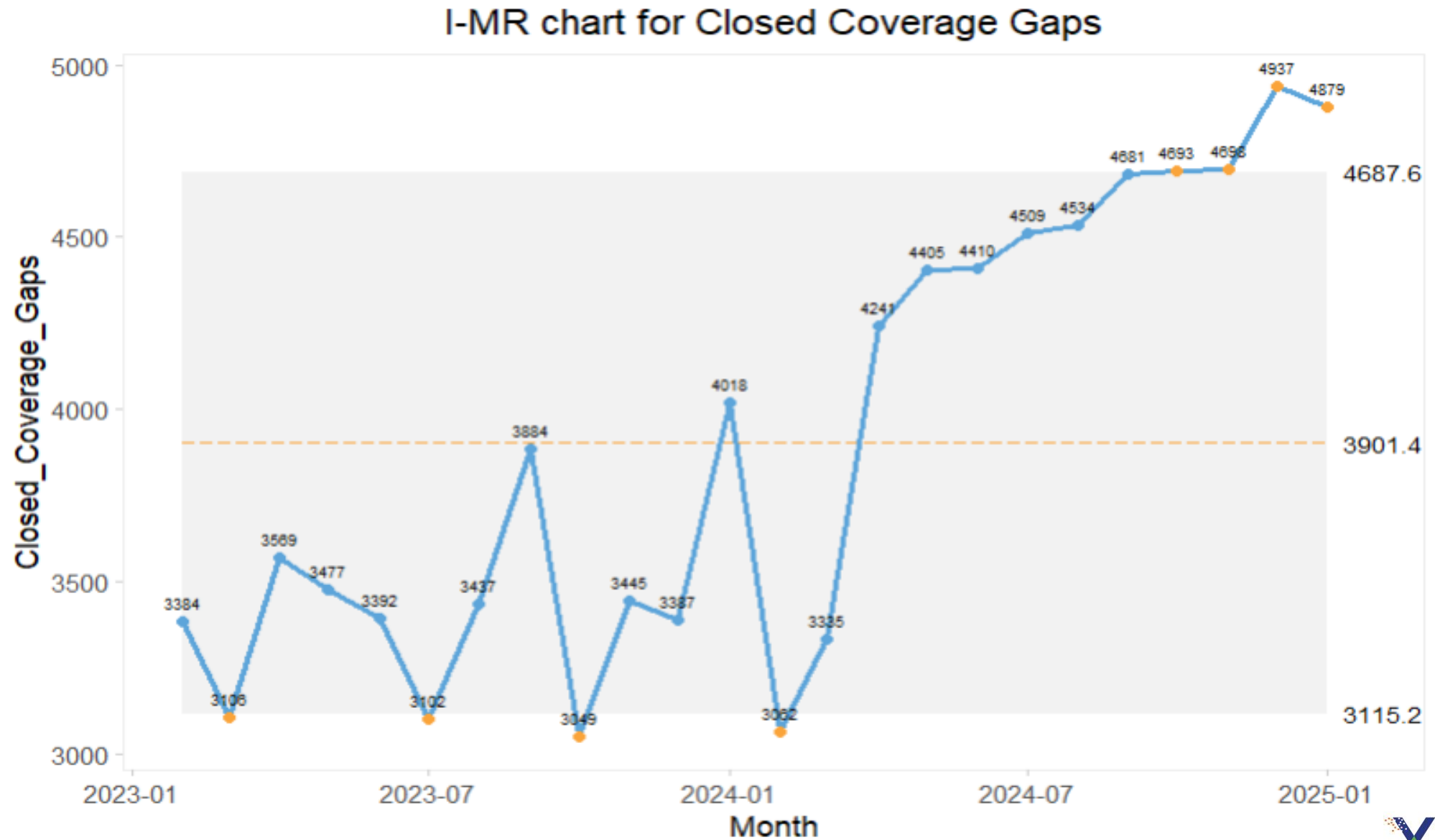
Response generated by AI

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)
```

Using AI generated code with necessary changes

Reducing Data Labels Size



Advantages of AI Integration:

- Accelerated Learning
- Code Generation and Completion
- Error Detection and Debugging
- Exposure to Best Practices

Disadvantages of AI Integration

- Overdependency on AI might result in skill degradation.
- Sharing of proprietary code with external servers
- AI generated inefficient code will lead to security vulnerabilities.
- There are costs associated with AI Integration.
- Utilizing AI extensions and models need high computing power.
- Copyright and plagiarism issues.

Security Concerns

- Overreliance on AI can lead to security and data sharing issues.
- AI extensions itself are vulnerable to security risks. Using high reputed Ai extensions are recommended.
- Prompts used while interacting with AI might be used for Re-engineering of AI models.
- Refrain to use sensitive or Personal Identifiable information in the prompts and code while interacting with AI.
- Make sure to keep your communication with AI platforms hidden when you create your API's.

Cost associated with AI Integration

- Some of AI platforms like Google cloud AI, Hugging face, IBM Watson provide free usage with limited number of API calls.
- Extensive use of AI extensions has its associated costs.
- Most of the AI platforms with focus on language models charge based on Tokens.
- Example: Open Ai's Chat GPT-4 charges 3 cents for Input 1000 tokens and 6 cents for 1000 output tokens.
- We used Chat GPT-4 for our paper and it costed us around \$15.

Questions ?

THANK YOU