PharmaSUG 2024
Paper HT-152

GenAI to Enhance Your Statistical Programming

Phil Bowsher, Posit; Boston, USA
Cole Arendt, Posit, Boston, USA

Abstract

This paper will cover examples of using generative artificial intelligence (GenAI) with an emphasis on statistical programming tasks. The information below offers insight and information on using AI in the clinical domain. The following examples are focused on GenAI tools available in an IDE that can help users be a more productive coder in creating new content.

Many organizations are looking to GenAI to help in accelerating the learning of open source languages often while transitioning from other languages. Andy Nicholls (GSK) discusses this topic in his Phuse talk "*Heading to base camp on our way up the open-source mountain…*":
https://phuse.s3.eu-central-1.amazonaws.com/Archive/2023/Connect/EU/Birmingham/PRE_TT09.pdf

Brittany Long (IQVIA) discussed at Phuse how AI can support programming tasks and outlined the following benefits:

- Encourage you to think of new options
- Review your code
- Chain of thought prompting

https://phuse.s3.eu-central-1.amazonaws.com/Archive/2023/Connect/EU/Birmingham/POS_PP14.pdf

The slides for this paper are here:
https://colorado.posit.co/rsc/ai-open-source/

Much of the information below is adapted from the paper "AI Exploration and Innovation for the Clinical Data Scientist" and slides. The following information is focused on information for program leaders and managers, while this paper is for end-users:
https://phuse.s3.eu-central-1.amazonaws.com/Archive/2024/Connect/US/Bethesda/PAP_ET08.pdf
https://phuse.s3.eu-central-1.amazonaws.com/Archive/2024/Connect/US/Bethesda/PRE_ET08.pdf

Introduction

Many statistical programmers are migrating from commercial software to open source. Below will provide examples for using various GenAI tools with an emphasis on IDE programming aids. Each option below has benefits and challenges. Programmers will also have personal preferences, so it is recommended to experiment to find the right mix.

GenAI to Enhance Your Code-First Data Science

It is our belief that software development happens in an IDE, and the focus of this paper will be on tools that incorporate GenAI in an IDE like RStudio. This concept is often referred to as **Code-First Data Science**: Building your analyses with code as the workflows can be reproduced, adapted, and scaled to solve similar problems in the future.

While there are popular public GenAI LLMs that have public interfaces like **chat.openai.com**, this paper will focus on tools incorporated in an IDE.These tools bring the power of public LLMs within the coding environment.

Many of the examples below will highlight Github Copilot. “Copilot” is becoming a popular term in GenAI. For example, a popular tool used by programmers is Obsidian, a writing app that adapts to the way you think. There is a tool called “Copilot” for Obsidian that is not connected to Github Copilot, but rather a ChatGPT interface inside Obsidian. Other popular tools like JetBrains IDE do have GitHub Copilot support.

The are 5 popular ways of using GenAI for programming in an IDE:

I. GitHub Copilot
   A. RStudio
   B. GitHub Copilot in VS Code
II. ChatGPT or other public LLM via chattr or other R package
III. cursor.sh or other AI IDE
IV. Local Large Language Models via an interface or IDE

Sample Tasks to Try in GenAI:

Below we will cover various tools for using GenAI. Here are examples to try using in each tool:

- Try “Write R code that generates an Adverse Summary Table with GT R package”
- Try “Write R code that uses the adae pharmaverse adam dataset with Age greater than 18. Then use the gt R package to generate an Adverse Summary Table with GT R package. Also create a ggplot2 graphic on AESEV count. The first table should show the total number of patients with at least one adverse event. The ggplot2 graphic should show a bar chart with total counts by AGE. Add thorough documentation to your code.”

Andreas Handel has great advice on using GenAI for programming. It is recommended to read the sections on **Good prompting and Iterating** here:
https://andreashandel.github.io/MADAcourse/content/module-ai/ai-write-code-r/ai-write-code-r.html

Andreas Handel has some great examples of AI tools for fixing code with examples here:
https://andreashandel.github.io/MADAcourse/content/module-ai/ai-analysis-r/ai-analysis-r.html

The examples point to code examples that users can test here:
https://github.com/andreashandel/MADAcourse/tree/main/code

We encourage users to test out the examples in the various methods outlined below in an IDE.

I. GitHub Copilot

GitHub Copilot Information

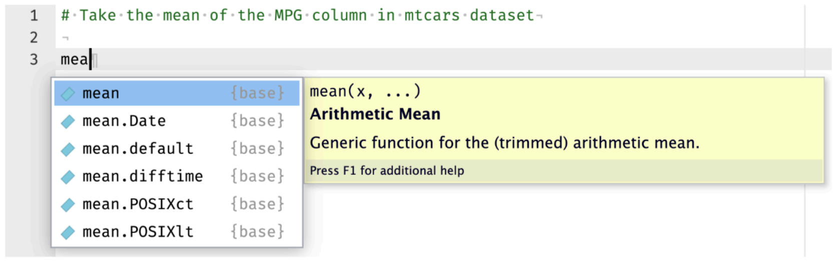
GitHub Copilot is an AI pair programmer tool that is available in many IDEs. The tool provides autocomplete-style suggestions while programming. GitHub Copilot is not free and requires a subscription and an active GitHub Copilot account.

A. Copilot in RStudio

Please see the presentation below from Posit Conf 2023 that introduced Github Copilot in RStudio:
https://colorado.posit.co/rsc/rstudio-copilot/

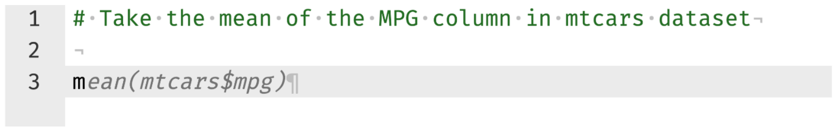
The talk highlights the differences between autocomplete and Github Copilot:

Autocomplete vs Copilot



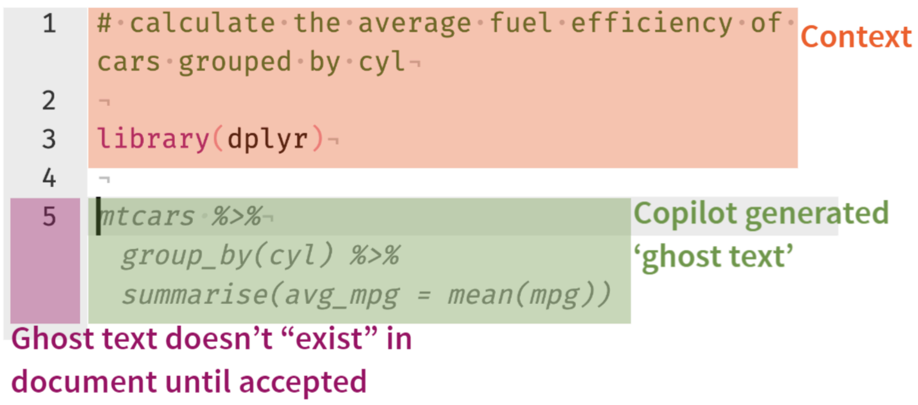
```
# Take the mean of the MPG column in mtcars dataset

mea
```

mean(x, ...)
Arithmetic Mean
Generic function for the (trimmed) arithmetic mean.
Press F1 for additional help



```
# Take the mean of the MPG column in mtcars dataset

mean(mtcars$mpg)
```

Here is an example of Copilot generated Code:

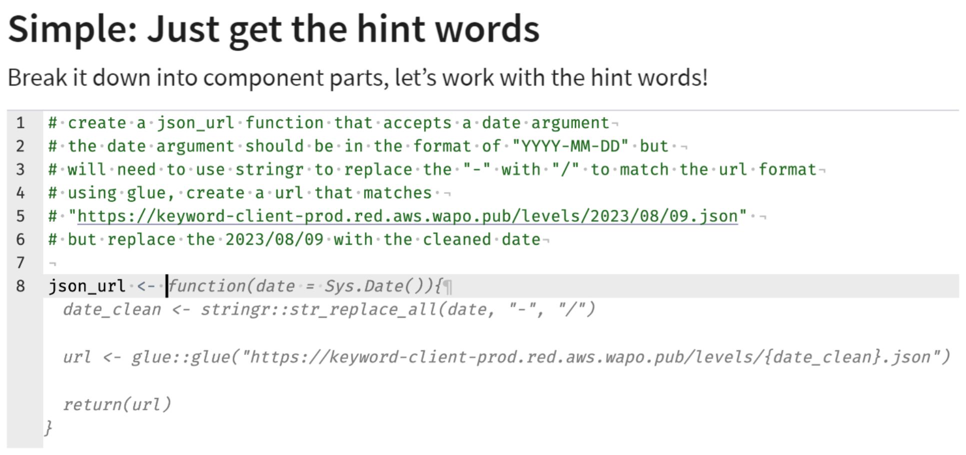


```
# calculate the average fuel efficiency of cars grouped by cyl

library(dplyr)

mtcars %>%
  group_by(cyl) %>%
  summarise(avg_mpg = mean(mpg))
```

Context
Copilot generated 'ghost text'
Ghost text doesn't "exist" in document until accepted

It helps in GenAI to provide context of the programming task. The description below of the objective will show an example. To help achieve better results, programmers will iterate on specific tasks. The talk "A hacker's guide to open source LLMs" provides additional information: https://www.youtube.com/watch?v=sYliwvml9Es

An example of context is below at the top of the code:



Simple: Just get the hint words

Break it down into component parts, let's work with the hint words!

```
# create a json_url function that accepts a date argument
# the date argument should be in the format of "YYYY-MM-DD" but
# will need to use stringr to replace the "-" with "/" to match the url format
# using glue, create a url that matches
# "https://keyword-client-prod.red.aws.wapo.pub/levels/2023/08/09.json"
# but replace the 2023/08/09 with the cleaned date

json_url <- function(date = Sys.Date()){
  date_clean <- stringr::str_replace_all(date, "-", "/")

  url <- glue::glue("https://keyword-client-prod.red.aws.wapo.pub/levels/{date_clean}.json")

  return(url)
}
```

GitHub Copilot Setup in RStudio

There are various versions of the RStudio IDE such as RStudio desktop, RStudio Server, and Posit Cloud, all of which support GitHub Copilot. The downloads for RStudio are here:
https://posit.co/downloads/

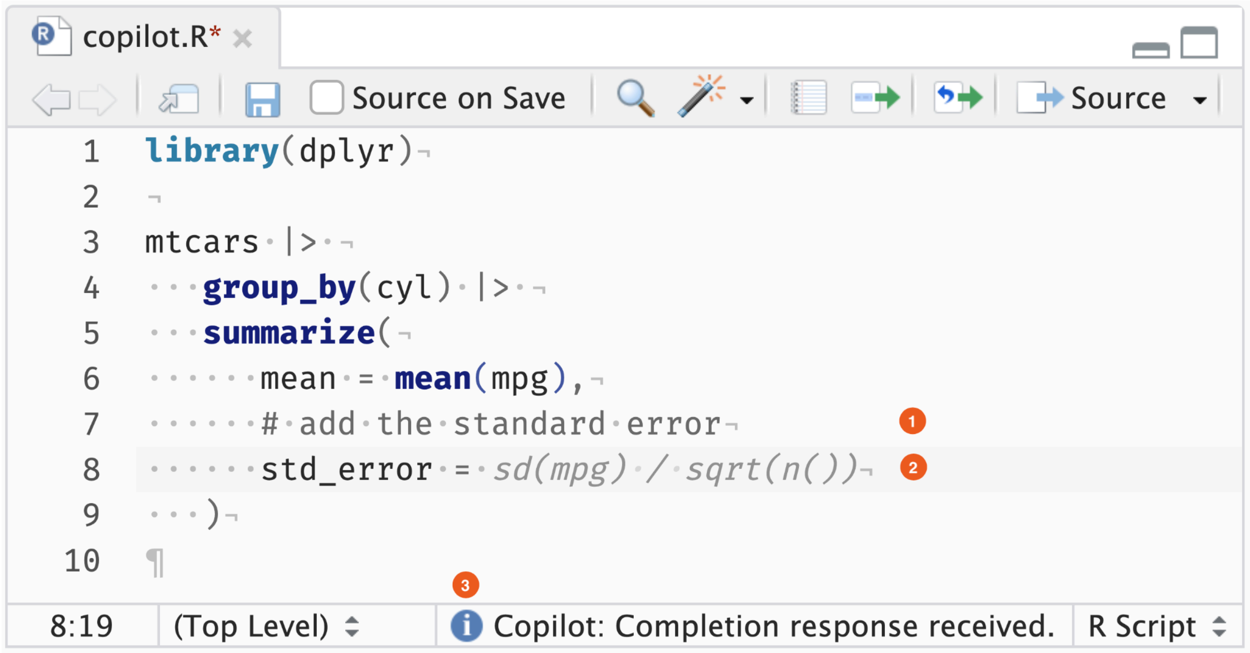
Posit Cloud is an online platform for learning R. The article below shows how to use Github Copilot in Posit Cloud:
https://posit.co/blog/github-copilot-on-posit-cloud/

Once a user has a Github Copilot account and subscription, they can configure the connection in RStudio using the following direction:
https://docs.posit.co/ide/user/ide/guide/tools/copilot.html

The link shows the following steps:

- Navigate to Tools > Global Options > Copilot.
- Check the box to "Enable GitHub Copilot".
- Download and install the Copilot Agent components.
- Click the "Sign In" button.
- In the "GitHub Copilot: Sign in" dialog, copy the Verification Code.
- GitHub Copilot: Sign in
- Navigate to or click on the link to https://github.com/login/device, paste the Verification Code and click "Continue".
- GitHub will request the necessary permissions for GitHub Copilot. To approve these permissions, click "Authorize GitHub Copilot Plugin".
- After the permissions have been approved, your RStudio IDE will indicate the currently signed in user.
- Close the Global Options dialogue, open a source file (.R, .py, .qmd, etc) and begin coding with Copilot!

Here is an example of Github Copilot in RStudio:



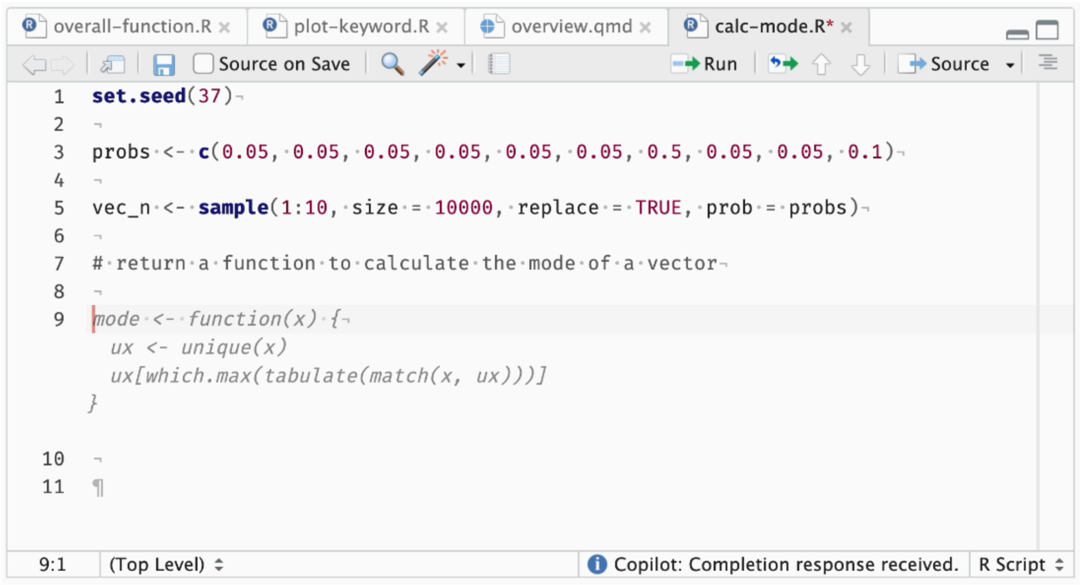
```r
library(dplyr)

mtcars |>
  group_by(cyl) |>
  summarize(
    mean = mean(mpg),
    # add the standard error
    std_error = sd(mpg) / sqrt(n())
  )

```

8:19 (Top Level) Copilot: Completion response received. R Script

The numbers correspond to:

1. A simple but specific comment providing additional context to Copilot.
2. Copilot's code suggestion, shown in light gray "ghost text".
3. The Copilot status bar, which indicates whether RStudio is waiting on a response to be generated, a completion response has been received, or no completions are available.

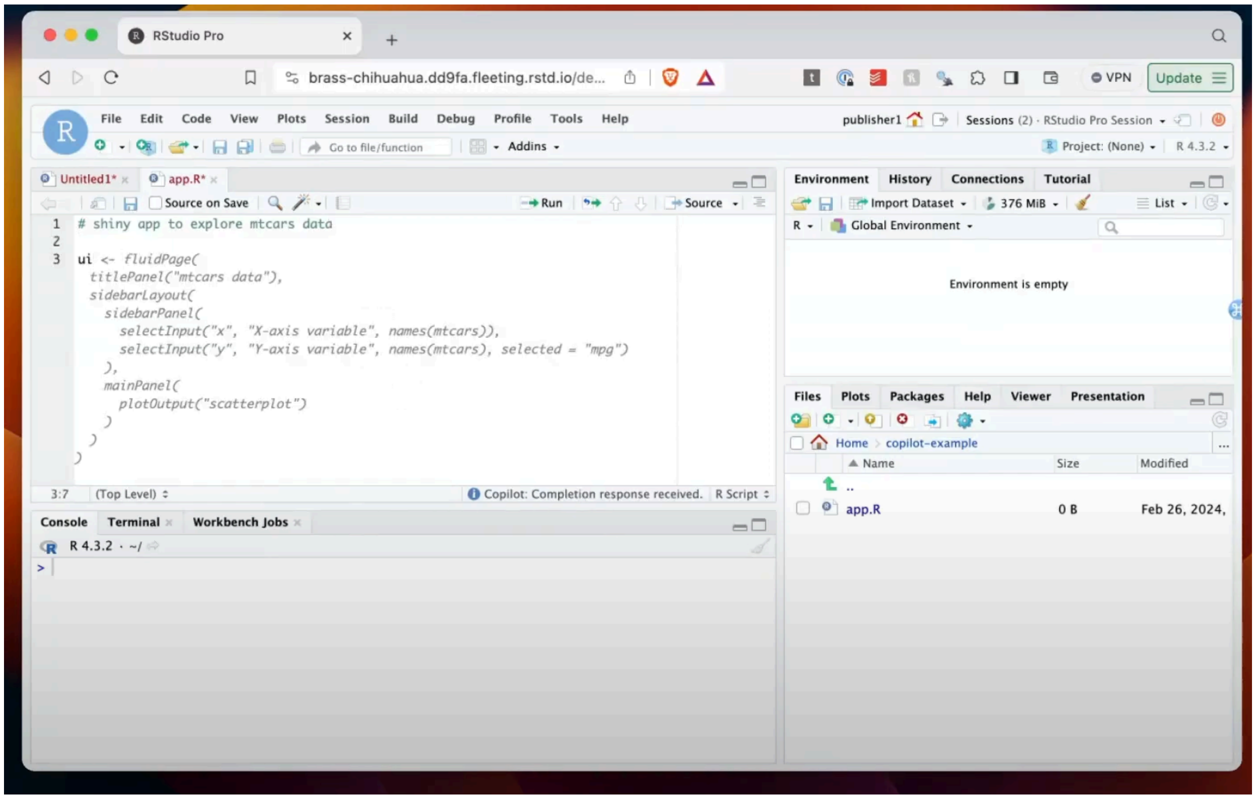
RStudio Copilot Example 1:
Below is a video showing an example of returning a function to calculate the mode of a vector where the function and output is generated from Github Copilot
https://colorado.posit.co/rsc/New_Innovations_OpenSource_Drug_Development/images/copilot-gif.gif



```r
set.seed(37)

probs <- c(0.05, 0.05, 0.05, 0.05, 0.05, 0.05, 0.5, 0.05, 0.05, 0.1)

vec_n <- sample(1:10, size = 10000, replace = TRUE, prob = probs)

# return a function to calculate the mode of a vector

mode <- function(x) {
  ux <- unique(x)
  ux[which.max(tabulate(match(x, ux)))]
}


```

9:1 (Top Level) Copilot: Completion response received. R Script

RStudio Copilot Example 2:
Below is a video showing an example of creating a Shiny app with Copilot. The gray code is generated by Github Copilot.
https://www.youtube.com/watch?v=40dTx5pEtyQ&t=7s



```
# shiny app to explore mtcars data

ui <- fluidPage(
  titlePanel("mtcars data"),
  sidebarLayout(
    sidebarPanel(
      selectInput("x", "X-axis variable", names(mtcars)),
      selectInput("y", "Y-axis variable", names(mtcars), selected = "mpg")
    ),
    mainPanel(
      plotOutput("scatterplot")
    )
  )
)
```

Copilot: Completion response received.

Environment is empty

Name	Size	Modified
..		
app.R	0 B	Feb 26, 2024,

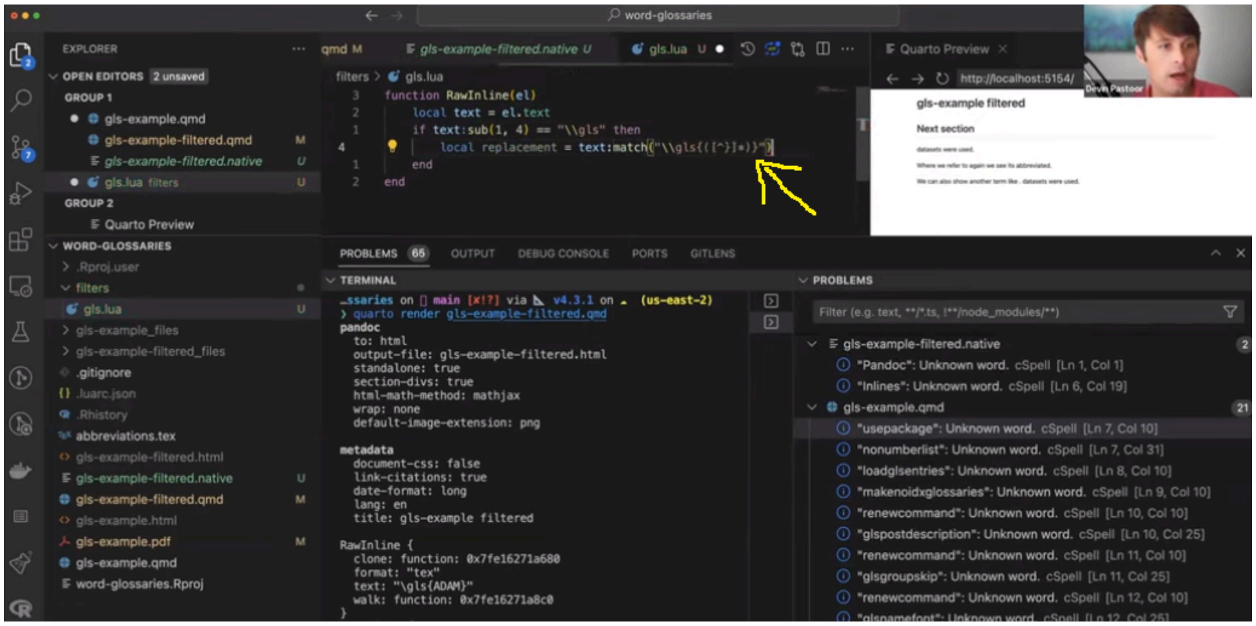
B. Copilot in VS Code

There are various versions of VS Code. VS Code for a desktop machine can be downloaded here:
https://code.visualstudio.com/

It is common to use Github Copilot in VS Code with the GitHub Copilot Extension:
https://marketplace.visualstudio.com/items?itemName=GitHub.copilot

Additional information about Github Copilot is below::
https://code.visualstudio.com/blogs/2023/03/30/vscode-copilot
https://code.visualstudio.com/docs/editor/github-copilot
https://docs.github.com/en/copilot/using-github-copilot/getting-started-with-github-copilot

Devin Pastoor provided the R in Pharma workshop Quarto All the Things! During the workshop, he uses Github Copilot to generate regular expressions within a clinical context as seen here:
https://youtu.be/k-dQ36sx4Rk?t=8276



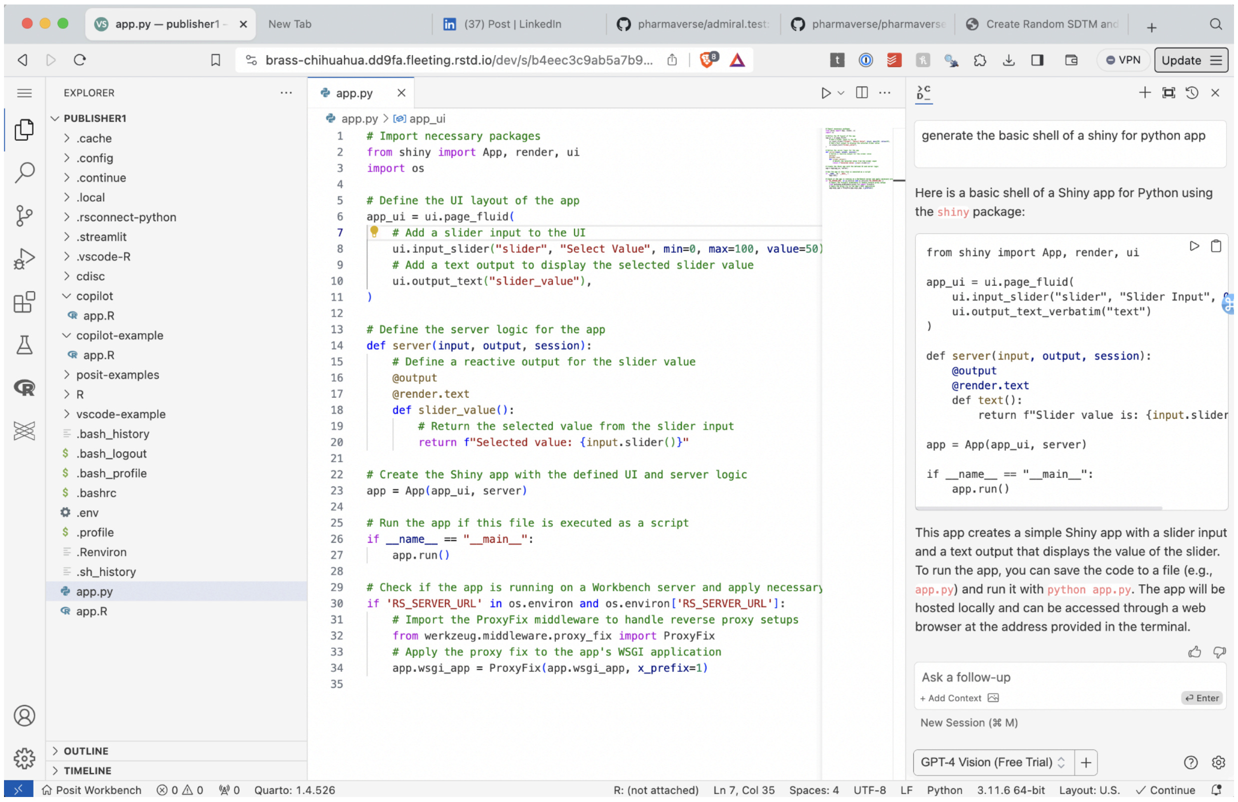
```lua
function RawInline(el)
    local text = el.text
    if text:sub(1, 4) == "\\gls" then
        local replacement = text:match("\\gls{([^}]*)}")
    end
end
```

```
…ssaries on  main [x!?] via ◺ v4.3.1 on ☁ (us-east-2)
❯ quarto render gls-example-filtered.qmd
pandoc
  to: html
  output-file: gls-example-filtered.html
  standalone: true
  section-divs: true
  html-math-method: mathjax
  wrap: none
  default-image-extension: png

metadata
  document-css: false
  link-citations: true
  date-format: long
  lang: en
  title: gls-example filtered

RawInline {
  clone: function: 0x7fe16271a680
  format: "tex"
  text: "\gls{ADAM}"
  walk: function: 0x7fe16271a8c0
}
```

```
gls-example-filtered.native
  "Pandoc": Unknown word. cSpell [Ln 1, Col 1]
  "Inlines": Unknown word. cSpell [Ln 6, Col 19]
gls-example.qmd
  "usepackage": Unknown word. cSpell [Ln 7, Col 10]
  "nonumberlist": Unknown word. cSpell [Ln 7, Col 31]
  "loadglsentries": Unknown word. cSpell [Ln 8, Col 10]
  "makenoidxglossaries": Unknown word. cSpell [Ln 9, Col 10]
  "renewcommand": Unknown word. cSpell [Ln 10, Col 10]
  "glspostdescription": Unknown word. cSpell [Ln 10, Col 25]
  "renewcommand": Unknown word. cSpell [Ln 11, Col 10]
  "glsgroupskip": Unknown word. cSpell [Ln 11, Col 25]
  "renewcommand": Unknown word. cSpell [Ln 12, Col 10]
```

VS Code Copilot Example 1:

Below is a screenshot showing an example of creating a Shiny for Python app with Copilot in VS Code:



```python
# Import necessary packages
from shiny import App, render, ui
import os

# Define the UI layout of the app
app_ui = ui.page_fluid(
    # Add a slider input to the UI
    ui.input_slider("slider", "Select Value", min=0, max=100, value=50)
    # Add a text output to display the selected slider value
    ui.output_text("slider_value"),
)

# Define the server logic for the app
def server(input, output, session):
    # Define a reactive output for the slider value
    @output
    @render.text
    def slider_value():
        # Return the selected value from the slider input
        return f"Selected value: {input.slider()}"

# Create the Shiny app with the defined UI and server logic
app = App(app_ui, server)

# Run the app if this file is executed as a script
if __name__ == "__main__":
    app.run()

# Check if the app is running on a Workbench server and apply necessary
if 'RS_SERVER_URL' in os.environ and os.environ['RS_SERVER_URL']:
    # Import the ProxyFix middleware to handle reverse proxy setups
    from werkzeug.middleware.proxy_fix import ProxyFix
    # Apply the proxy fix to the app's WSGI application
    app.wsgi_app = ProxyFix(app.wsgi_app, x_prefix=1)
```

generate the basic shell of a shiny for python app

Here is a basic shell of a Shiny app for Python using the shiny package:

```python
from shiny import App, render, ui

app_ui = ui.page_fluid(
    ui.input_slider("slider", "Slider Input",
    ui.output_text_verbatim("text")
)

def server(input, output, session):
    @output
    @render.text
    def text():
        return f"Slider value is: {input.slider

app = App(app_ui, server)

if __name__ == "__main__":
    app.run()
```

This app creates a simple Shiny app with a slider input and a text output that displays the value of the slider. To run the app, you can save the code to a file (e.g., app.py) and run it with python app.py. The app will be hosted locally and can be accessed through a web browser at the address provided in the terminal.

Other Python examples to test can be found here: https://github.com/sol-eng/python-examples

II. ChatGPT

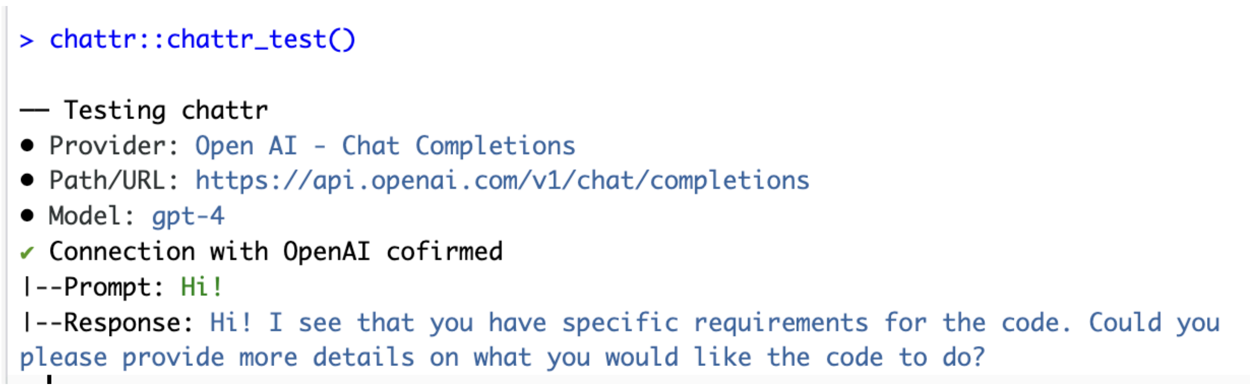
ChatGPT (Chat Generative Pre-Training Transformer) provides access to a public LLM via an interface. Below we will show tools that bring ChatGPT with an IDE. There are many extensions for using ChatGPT in VS Code. These can be found by doing a search in VS Code.

ChatGPT in RStudio with chattr R Package

Below we will feature how to access ChatGPT in RStudio. ChatGPT provides a REST API endpoint that can be accessed with a secret key as explained here:
https://platform.openai.com/account/api-keys

The chattr package will discover the secret key via the Environment Variable called OPENAI_API_KEY. The usethis R package can be used to set the variable with the code below: usethis::edit_r_environ()

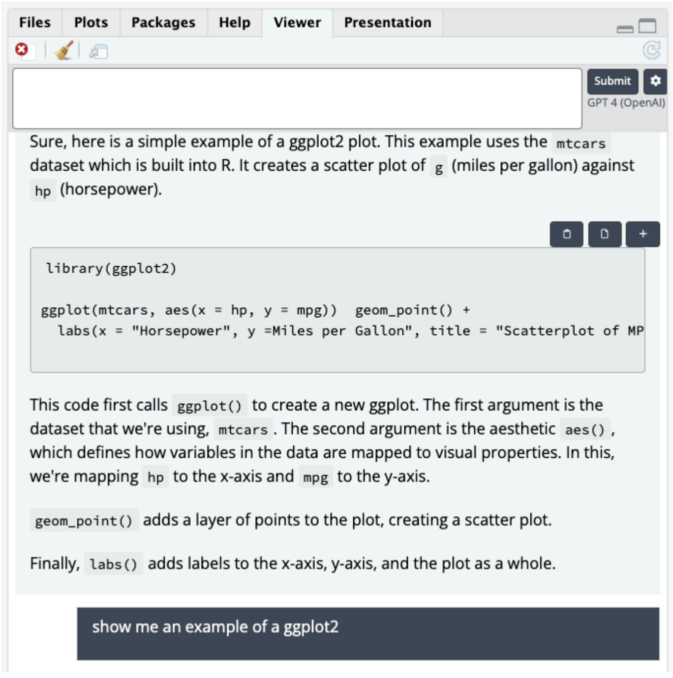
Create a OPENAI_API_KEY and set it equal to the OpenAI key. Save the file and restart R. Function chattr_test() will confirm that the connection works:



```
> chattr::chattr_test()

── Testing chattr
● Provider: Open AI - Chat Completions
● Path/URL: https://api.openai.com/v1/chat/completions
● Model: gpt-4
✔ Connection with OpenAI cofirmed
|--Prompt: Hi!
|--Response: Hi! I see that you have specific requirements for the code. Could you
please provide more details on what you would like the code to do?
```

chattr Example 1:

Below is an example of generating ggplot2 code with ChatGPT via chattr.
https://blogs.rstudio.com/ai/posts/2024-04-04-chat-with-llms-using-chattr/



Sure, here is a simple example of a ggplot2 plot. This example uses the `mtcars` dataset which is built into R. It creates a scatter plot of `g` (miles per gallon) against `hp` (horsepower).

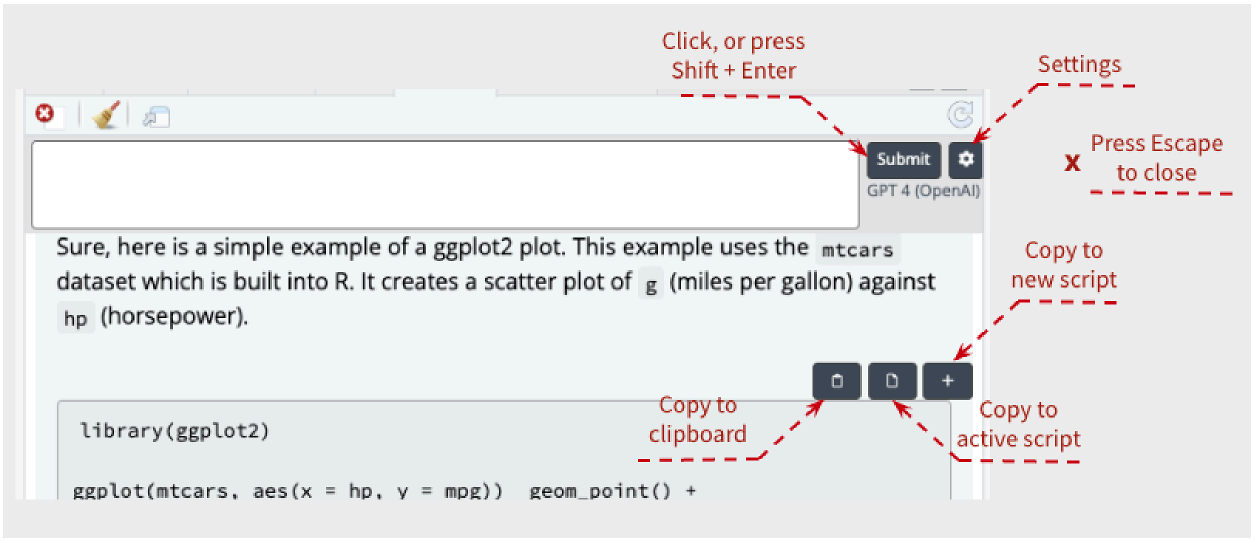
```
library(ggplot2)

ggplot(mtcars, aes(x = hp, y = mpg))  geom_point() +
  labs(x = "Horsepower", y =Miles per Gallon", title = "Scatterplot of MP
```

This code first calls `ggplot()` to create a new ggplot. The first argument is the dataset that we're using, `mtcars`. The second argument is the aesthetic `aes()`, which defines how variables in the data are mapped to visual properties. In this, we're mapping `hp` to the x-axis and `mpg` to the y-axis.

`geom_point()` adds a layer of points to the plot, creating a scatter plot.

Finally, `labs()` adds labels to the x-axis, y-axis, and the plot as a whole.

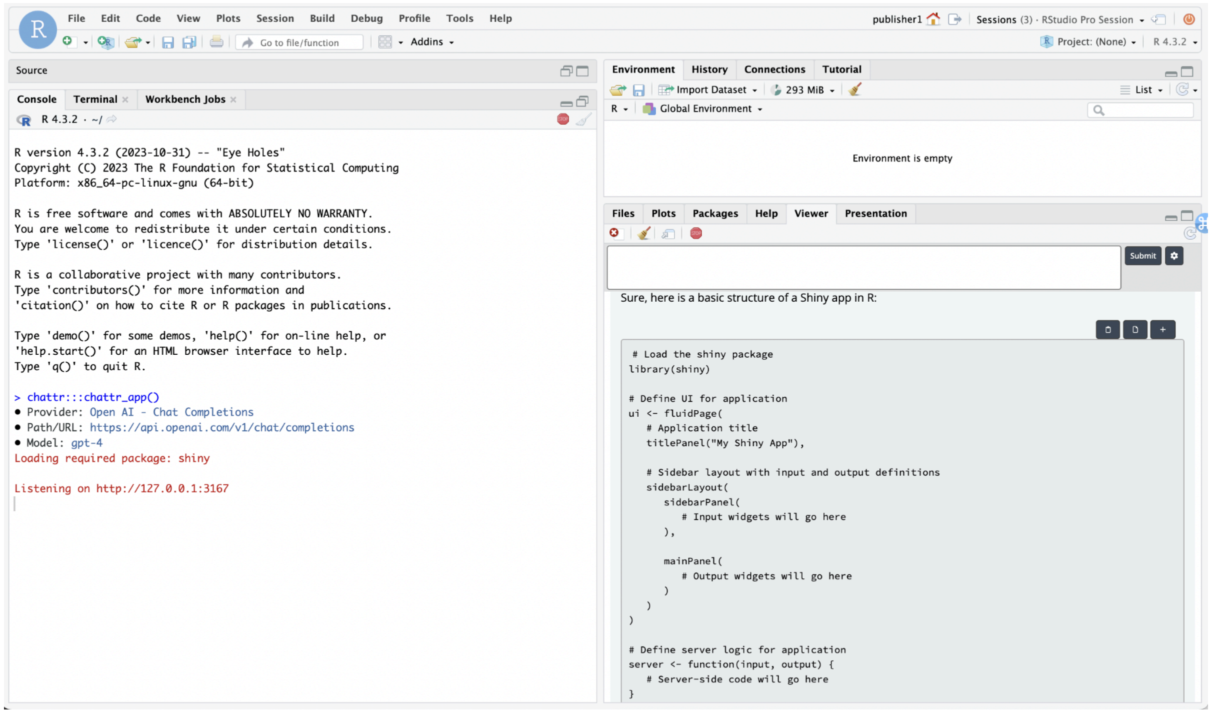
show me an example of a ggplot2



Sure, here is a simple example of a ggplot2 plot. This example uses the `mtcars` dataset which is built into R. It creates a scatter plot of `g` (miles per gallon) against `hp` (horsepower).

```
library(ggplot2)

ggplot(mtcars, aes(x = hp, y = mpg))  geom_point() +
```

chattr Example 2:
Below is an example of generating a Shiny app with ChatGPT via chattr.



```
R 4.3.2 · ~/

R version 4.3.2 (2023-10-31) -- "Eye Holes"
Copyright (C) 2023 The R Foundation for Statistical Computing
Platform: x86_64-pc-linux-gnu (64-bit)

R is free software and comes with ABSOLUTELY NO WARRANTY.
You are welcome to redistribute it under certain conditions.
Type 'license()' or 'licence()' for distribution details.

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.

> chattr:::chattr_app()
● Provider: Open AI - Chat Completions
● Path/URL: https://api.openai.com/v1/chat/completions
● Model: gpt-4
Loading required package: shiny

Listening on http://127.0.0.1:3167
```

Sure, here is a basic structure of a Shiny app in R:

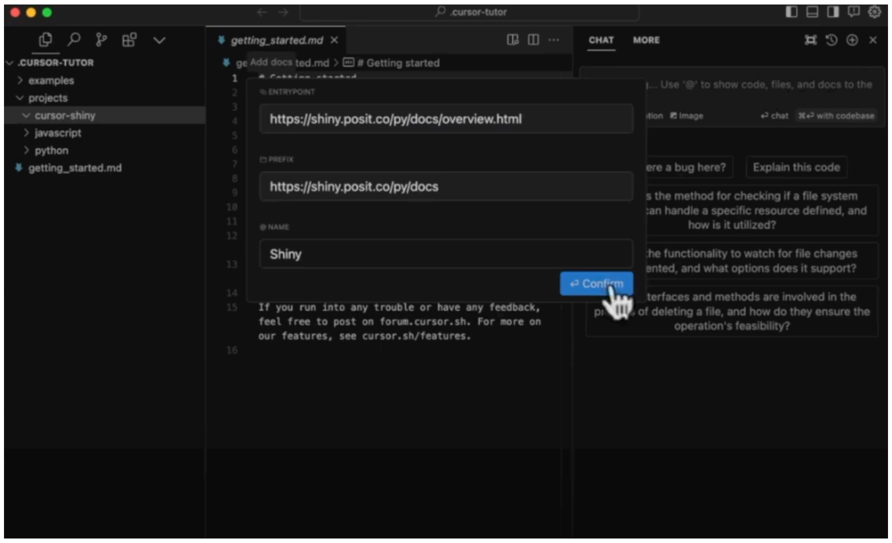
```
 # Load the shiny package
library(shiny)

# Define UI for application
ui <- fluidPage(
   # Application title
   titlePanel("My Shiny App"),

   # Sidebar layout with input and output definitions
   sidebarLayout(
      sidebarPanel(
         # Input widgets will go here
      ),

      mainPanel(
         # Output widgets will go here
      )
   )
)

# Define server logic for application
server <- function(input, output) {
   # Server-side code will go here
}
```

III. Example with cursor.sh

cursor.sh is an IDE built for pair-programming. It supports **RAG** (Retrieval Augmented Generation). **RAG** was introduced by Meta in 2020 here: https://arxiv.org/abs/2005.11401v4

cursor.sh can be supplemented with current documentation and additional context via RAG for enhancing the programming experience as seen below.

Cursor Example 1:
Below is a video of using Cursor.sh with RAG. The video shows the user asking for Shiny for Python examples yet the LLM was trained before it existed and references another tool. Then the user loads the documentation in for Shiny for Python and is able to get an example!
https://www.youtube.com/watch?v=megC3ulM1Wo&t=14s



ENTRYPOINT
https://shiny.posit.co/py/docs/overview.html

PREFIX
https://shiny.posit.co/py/docs

NAME
Shiny

```
If you run into any trouble or have any feedback,
feel free to post on forum.cursor.sh. For more on
our features, see cursor.sh/features.
```

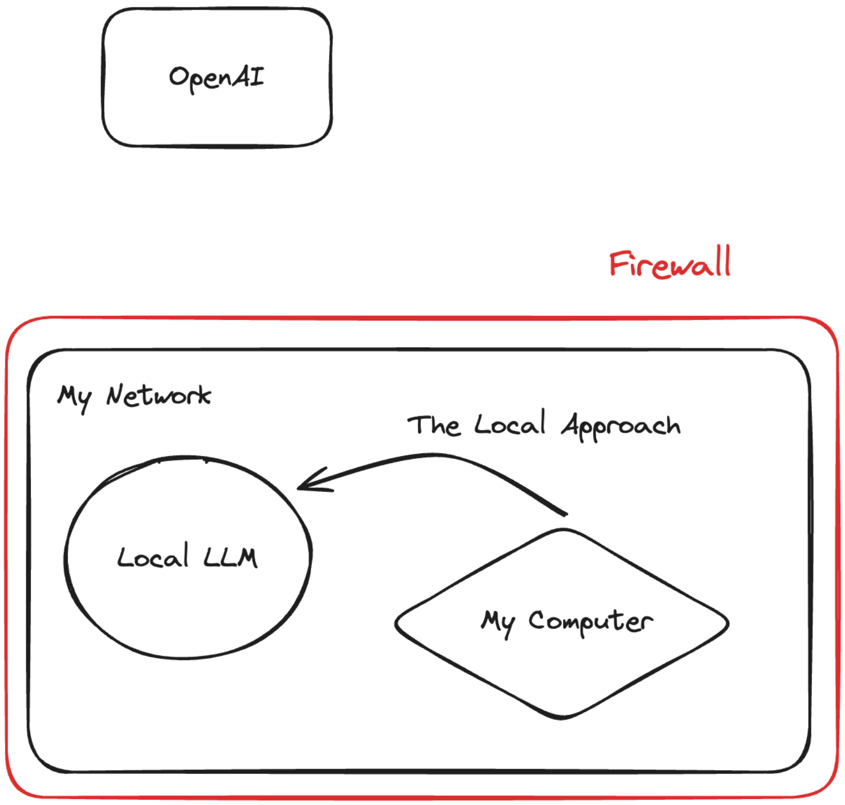
s the method for checking if a file system can handle a specific resource defined, and how is it utilized?

he functionality to watch for file changes ented, and what options does it support?

terfaces and methods are involved in the pr of deleting a file, and how do they ensure the operation's feasibility?

IV. Local Models with chattr
Sharon Machlis covers "5 easy ways to run an LLM locally" that discusses various ways to deploy a LLM on a local system: https://www.infoworld.com/article/3705035/5-easy-ways-to-run-an-llm-locally.html

Below is an example diagram showing the internal nature of local LLMs:



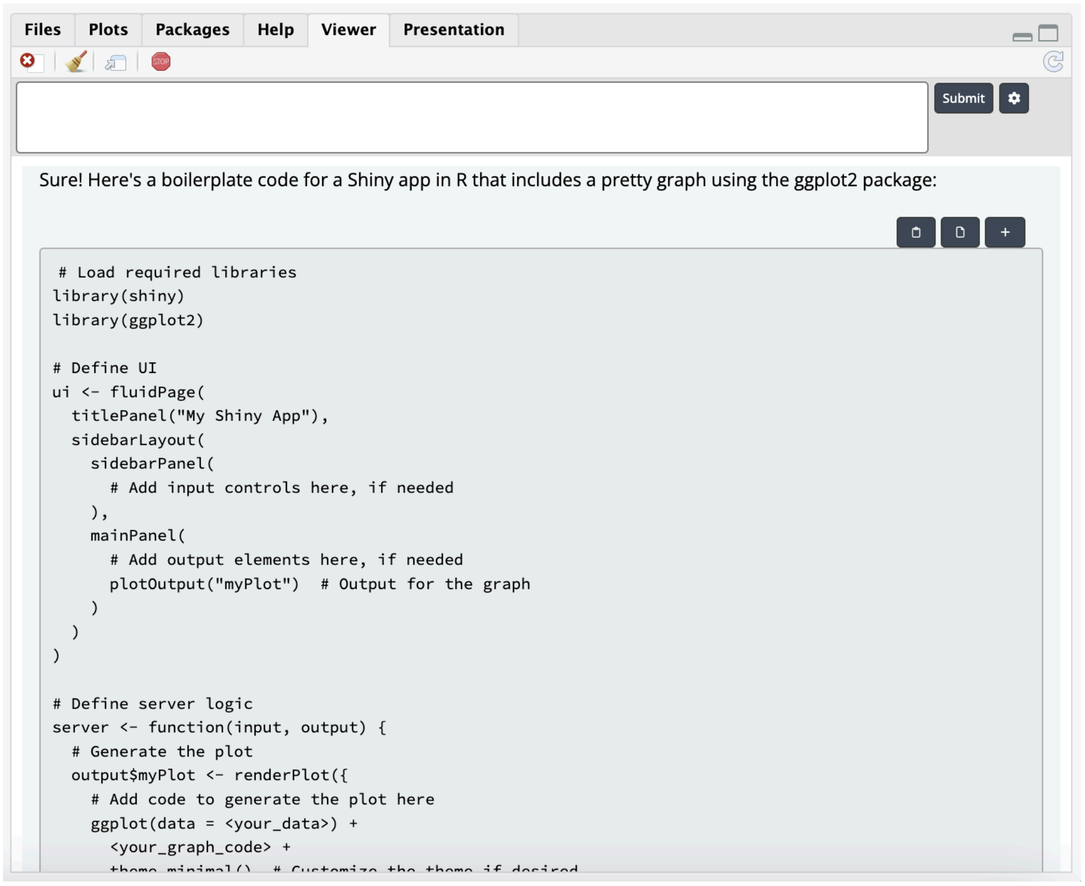
chattr will work with local models on a local machine (laptop, server etc.) with LlamaGPTJ-chat. The local models supported as April 2024 are:

- GPT-J (ggml and gpt4all models)
- LLaMA (ggml Vicuna models from Meta)
- Mosaic Pretrained Transformers (MPT)

.

chattr Example 2:

Below is an example showing chattr working with a local LLM on the same server as Posit Workbench:

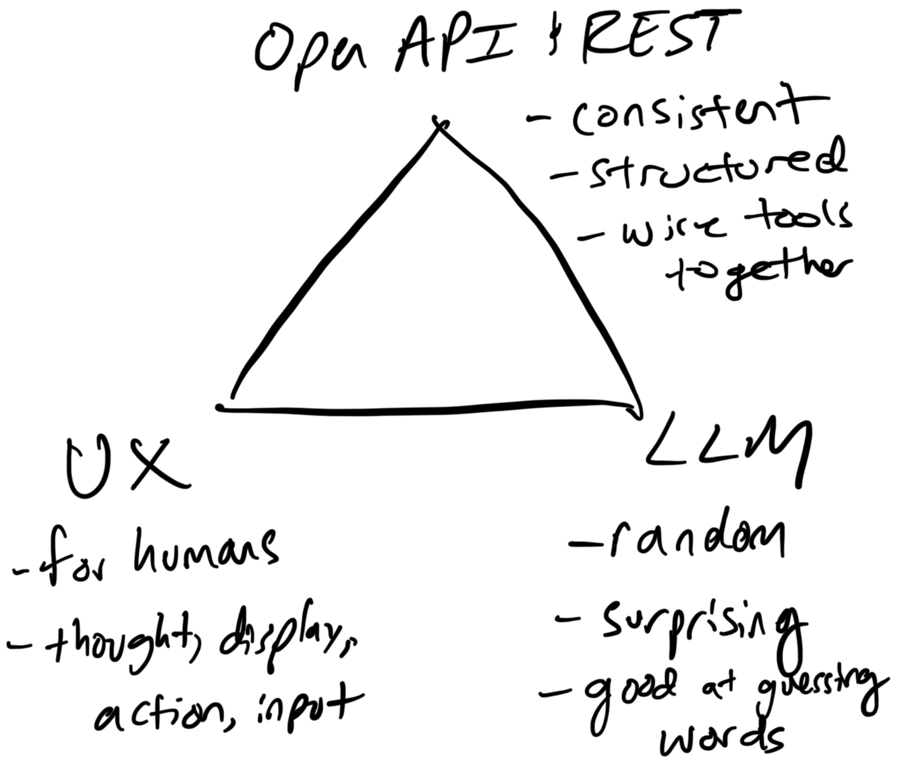


Sure! Here's a boilerplate code for a Shiny app in R that includes a pretty graph using the ggplot2 package:

```
 # Load required libraries
library(shiny)
library(ggplot2)

# Define UI
ui <- fluidPage(
  titlePanel("My Shiny App"),
  sidebarLayout(
    sidebarPanel(
      # Add input controls here, if needed
    ),
    mainPanel(
      # Add output elements here, if needed
      plotOutput("myPlot")  # Output for the graph
    )
  )
)

# Define server logic
server <- function(input, output) {
  # Generate the plot
  output$myPlot <- renderPlot({
    # Add code to generate the plot here
    ggplot(data = <your_data>) +
      <your_graph_code> +
      theme_minimal()  # Customize the theme if desired
```

While we believe that software development and data science happens in an IDE, there are many valid reasons to host an interface for non-coding professionals. The focus here is on optimizing tools such as:

- Computing (OpenAPI & Rest)
- User Experience (UX)
- Large Language Models (LLM)



If an organization has an internal local model (LLM), API (plumber), Shiny or other interface (Streamlit etc.) can be used to share it with non-programming professionals via an OpenAI type of interface. With Shiny, there are various options for surfacing the local LLM such as Shiny via R, Shiny with Reticulate, or Shiny for Python. Below is an example using Shiny for Python.
https://github.com/wch/chatstream/blob/main/examples/doc_query/app.py

Conclusion

This paper was to highlight practical examples of how statistical programmers can use GenAI for learning open source for statistical programming tasks. As the use of AI increases by clinical statistical programmers, it is important organizations understand the tools to provide to users. We discussed common tools for using GenAI as well as how to access local models. The information outlined in this paper highlights the quickly changing space of GenAI and examples of how it can be used today by statistical programmers to learn open source.

CONTACT INFORMATION

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

Phil Bowsher
phil@posit.co