PHUSE 2026 - Paper ML32

Voice-Driven Agent Assisted Open-Source Data Science

Phil Bowsher, Posit/RStudio PBC

Abstract

As featured at GenAI Day (youtube.com/@RinPharma/), many pharmaceutical companies are using AI to assist with late-stage development tasks. These use-cases span from integrating AI with Shiny, an AI assistant to help programmers write ADaM (Analysis Data Model) code, automating repetitive data tasks, and converting plain English instructions into CDISC-compliant code. As AI becomes more integrated into the software and tooling for drug development, features such as voice-driven tasks completion and conversation enabled capabilities will likely be more widely available. There are many tools used for such engagement today, like Gemini (Google), Siri (Apple), Alexa (Amazon), Microsoft Copilot Voice and ChatGPT (OpenAI). This paper will explore the latter and how to use it for driving R based data visualizations with ggplot2 as an experiment in the possibility of this tool in drug development.

Introduction

ggplot2 was released over 20 years ago, and it changed R forever. ggplot2 brought a package to R that enabled statisticians and data scientists a way to elegantly create data visualizations with a simple grammar. ggplot2 is downloaded approximately 1 million times per week and is the most popular R package. This paper will introduce *ggbot2* as a voice assistant for ggplot2. ggbot2 is a small R package and getting up and running is pretty quick once the API key is obtained from openAI as detailed below.

Please note that LLMs make errors. Domain expertise and programming knowledge is critical when working with AI coding assistants and tools like ggbot2. Posit PBC champions *code-first open source data science* as a key part of analytical computing. Working with code is critical for reproducibility and the ability for AI to generate code is a central focus for Posit's AI tools, not analysis as explained by Joe Cheng in the following video: :https://www.youtube.com/watch?v=owDd1CJ17uQ&t=878s

Further reading can be found below on these topics:

https://wesmckinney.com/blog/llms-arithmetic/

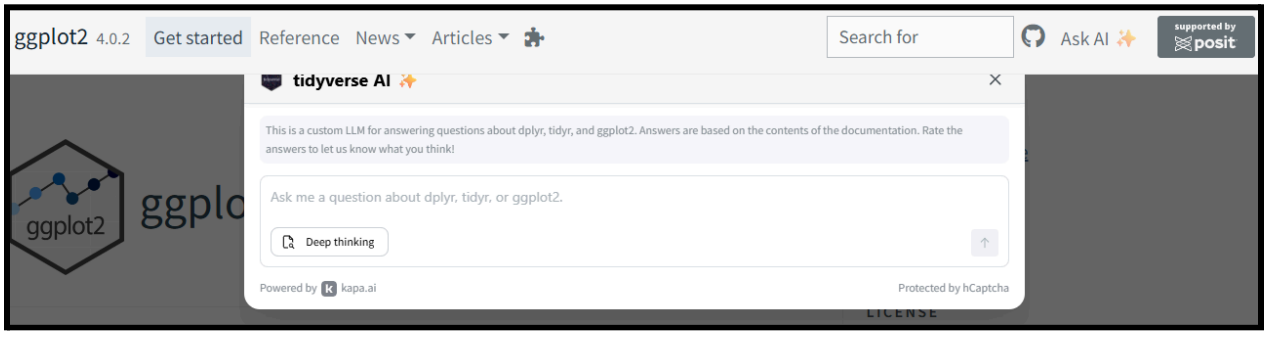
https://posit.co/blog/databot-is-not-a-flotation-device/

https://posit.co/blog/positron-ai-product-announcement-aug-2025/

ggplot2

This paper will not provide an introduction to ggplot2. There are many amazing resources available such as the info page found here: https://ggplot2.tidyverse.org/

New users can explore the "Ask AI" kapa.ai integrations to learn and prompt about ggplot2:



ggbot2 - Setup

ggbot2 was released in 2025. The main page can be found here: https://github.com/tidyverse/ggbot2

Currently, ggbot2 requires an OpenAI API key. With the API key, users will need to set it as an environment variable named *OPENAI_API_KEY*. Below we will detail the steps in Positron.

Users will need to store the API key securely in the environment. Posit PBC recommends saving an environment

variable rather than using the API key directly in users code. In R, users can use the R package usethis, to set up the API keys in Positron. Using usethis, add your API key(s) to your .Renviron file. You can open your .Renviron for editing with **usethis::edit_r_environ() or via Ctrl Shift P in Positron and File: Open File**. Add the API key(s) to your .Renviron, for example:

OPENAI_API_KEY=my-api-key-openai-xyz

Then restart your R session. After the API key is loaded into the .Renviron file, we will need to install the ggbot2 R package:

pak::pak("tidyverse/ggbot2")

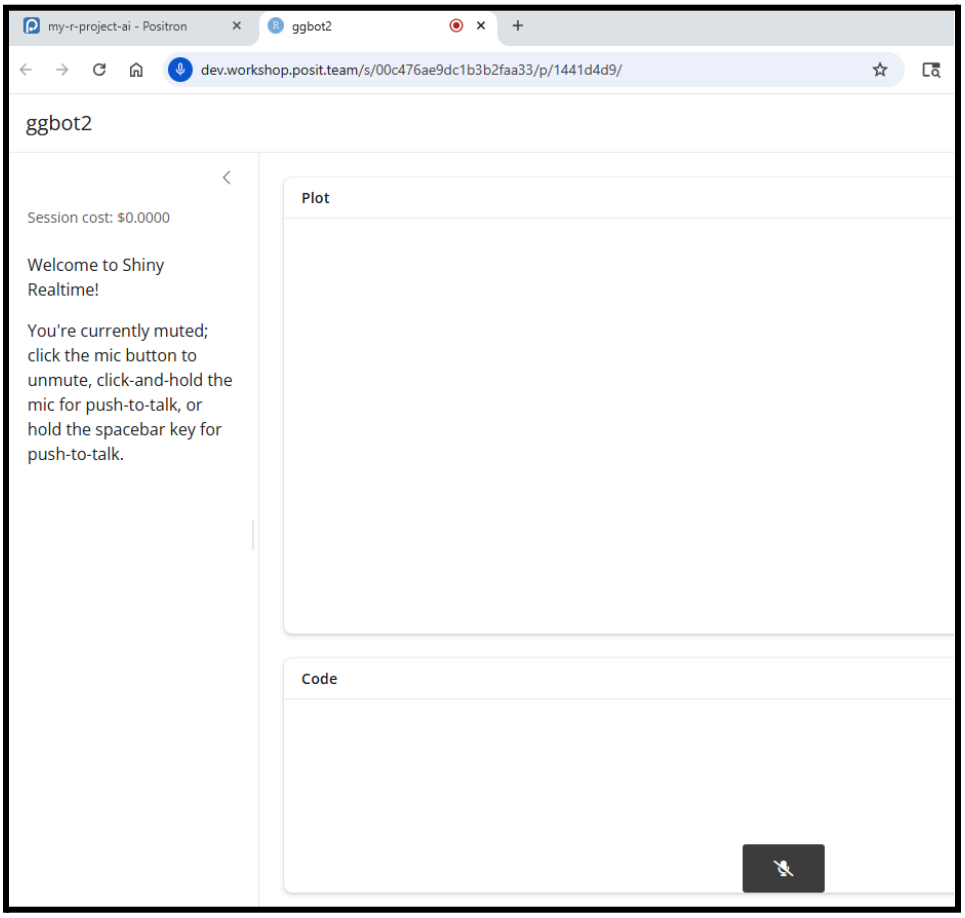
Now the package will be installed and ready to use.

Using ggbot2

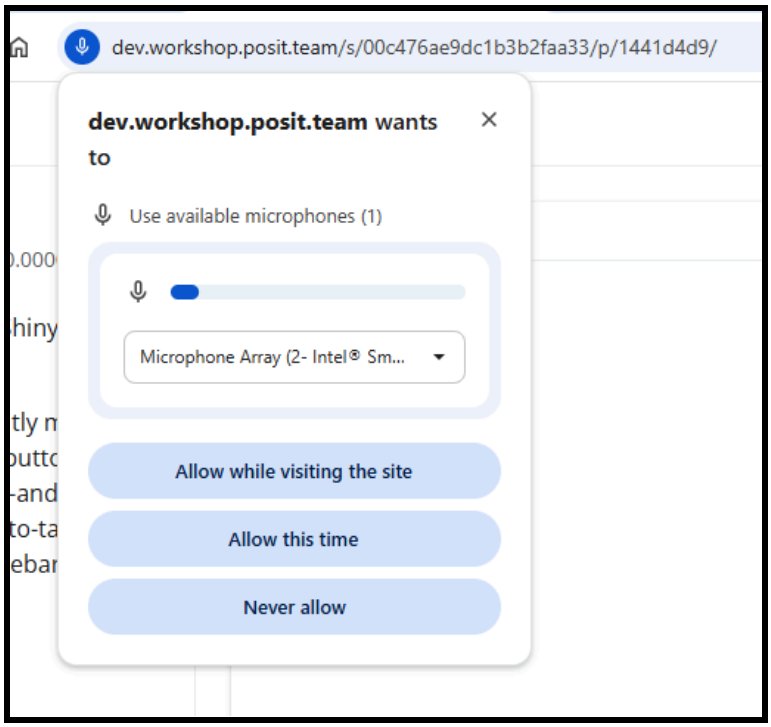
To start using ggbot2, simple run the following command:

ggbot2::ggbot(mtcars)

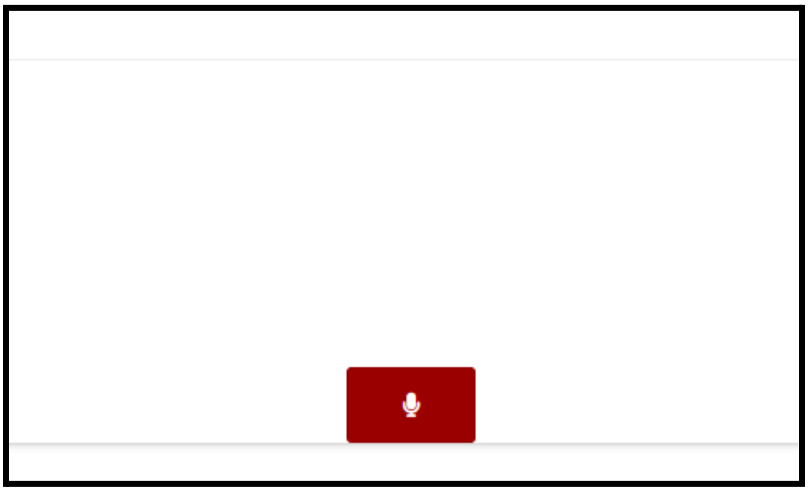
ggbot2 uses Shiny as the interface for users to generate ggplot2 code with voice commands. The function will launch a pre-built (bundled) Shiny app in a separate tab such as:



Next, users will need to permit a microphone to be used such as:



After the microphone is enabled, users will need to un-mute the microphone within the Shiny app in the lower middle area. Users will see a message such as “You're currently muted; click the mic button to unmute, click-and-hold the mic for push-to-talk, or hold the spacebar key for push-to-talk.”:



The mic feature is implemented with the *shinyrealtime* R package, a package for integrating OpenAI's Realtime API with Shiny applications in both R and Python: https://github.com/posit-dev/shinyrealtime

While ggbot2 uses shinyrealtime under the hood, it could be used separately for creating custom tools similar to ggbot2. For a review of Shiny and AI, please see the paper below, which covers tools like shinychat, querychat and other use-cases:
Shiny & LLMs: Landscape and Applications in Pharma

https://github.com/philbowsher/Shiny-LLMs-Landscape-and-Applications-in-Pharma/blob/main/Shiny%20%26%20LLMs_%20Landscape%20and%20Applications%20in%20Pharma.pdf

Pharma Example: ggbot2 with ADAM data

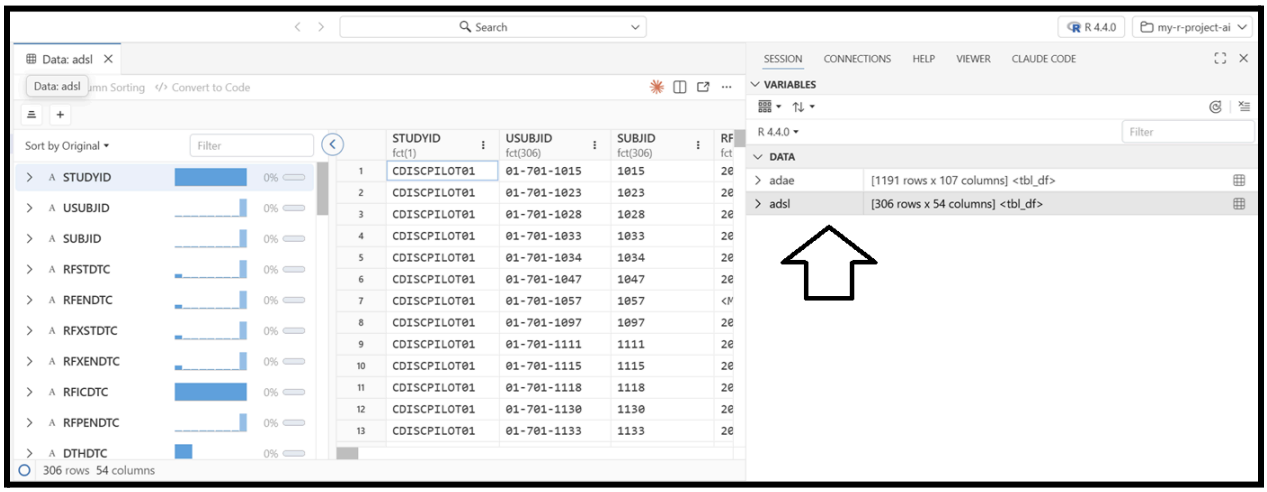
For this example, users will need to load some **adsl and or adae data** into the R sessions environment such as:

```
library(pharmaverseadam)
library(tern)
library(dplyr)

adsl <- adsl %>%
  df_explicit_na()

adae <- adae %>%
  df_explicit_na()
```

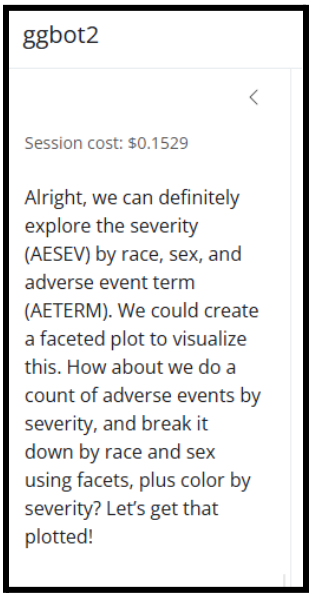
This will load the 2 datasets into the Session such as:



Now we will run ggbot2 with the clinical data:

ggbot2::ggbot(adae)

After the microphone is un-muted, say “Hello” to start the session with ggbot2. Users can use natural language - no need to change the tone or mode of speaking. Users can speak naturally, like they would to a person engaged in conversation. ggbot2 will speak back and even suggest ideas or ask for next steps. Feel free to interrupt ggbot2, there is no need to wait for it to complete the thought. ggbot2 will notice the change and listen for a new direction. Here is an example of its spoken text as displayed in the app:



ggbot2 only needs a general sense of the visualizations goal. It may make a mistake or generate something users don't like - but it can quickly fix it with a new request or provide feedback like one would in conversation.

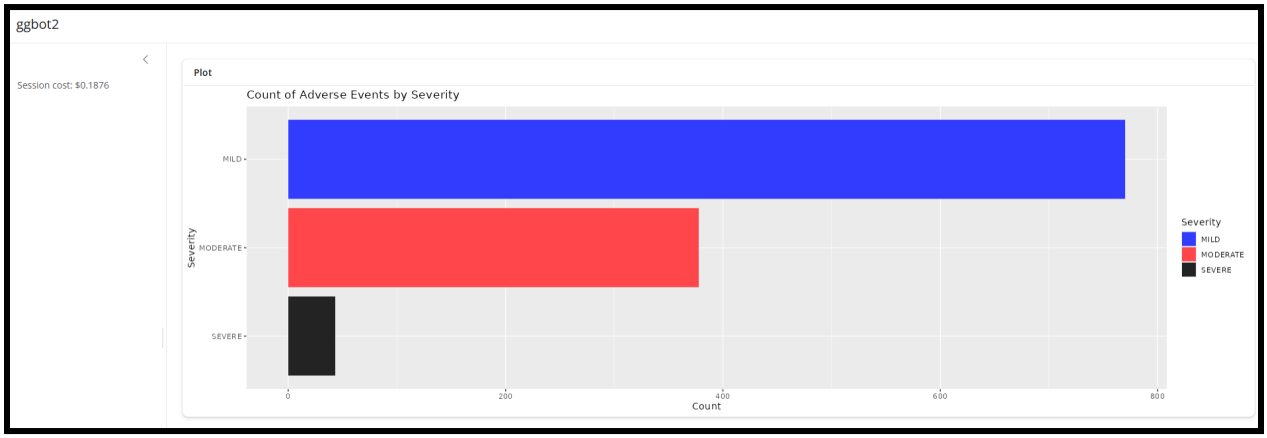
Below are some visualizations made with ggbot2 using test ADaM data from the pharmaverseadam R package. Example spoken prompts used.

- "Using the adverse events data, create a bar chart of severity by age"
- "Make the bars colorful using blue"
- "Add the study name to the title to the bar chart"
- "The bar chart is too busy, let's remove the age groups"
- "Add to the bar chart the top 5/10 most common adverse events"
- "Add an annotation that calls out the year"
- "Make the plot/bar chart look less busy"
- "Let's go back to the last version"

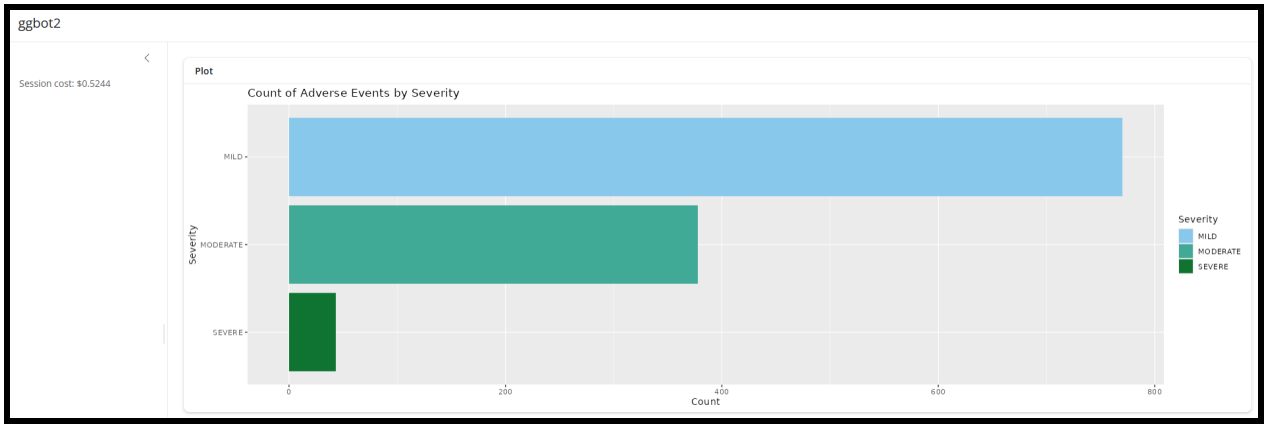
Below are the outputs of such spoken prompts:

Simple Bar chart of severity:

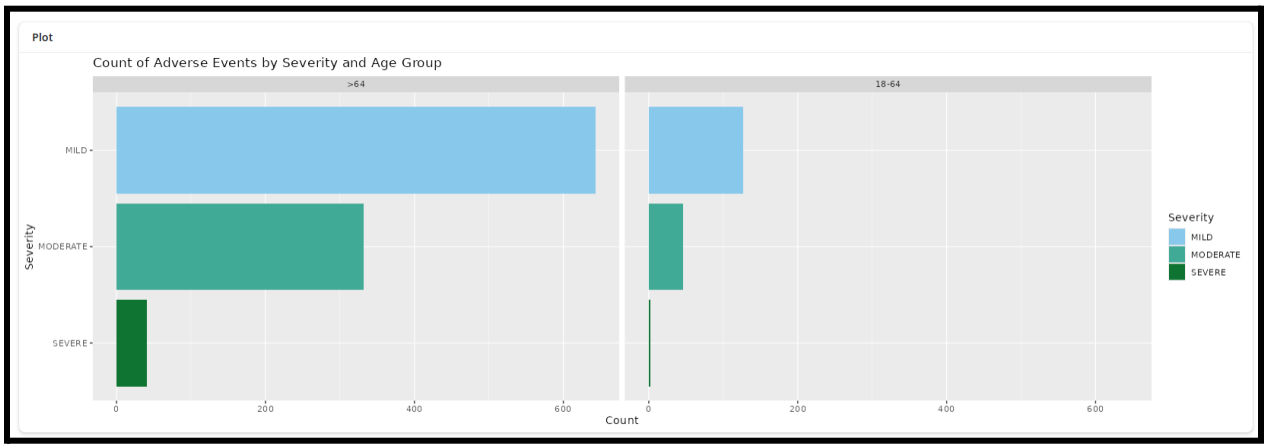
Simple Bar chart of added color:



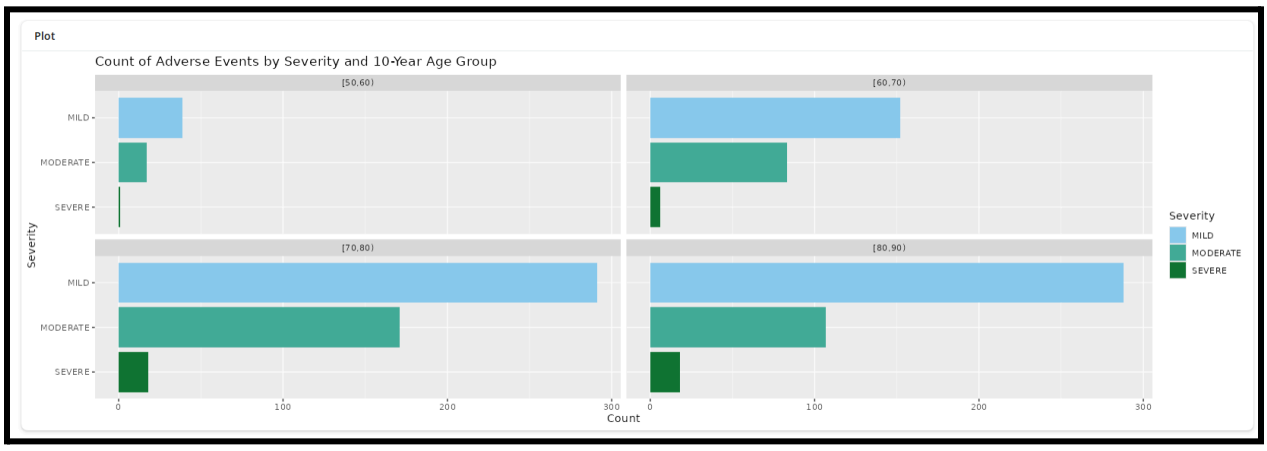
Simple Bar chart with changes to color:



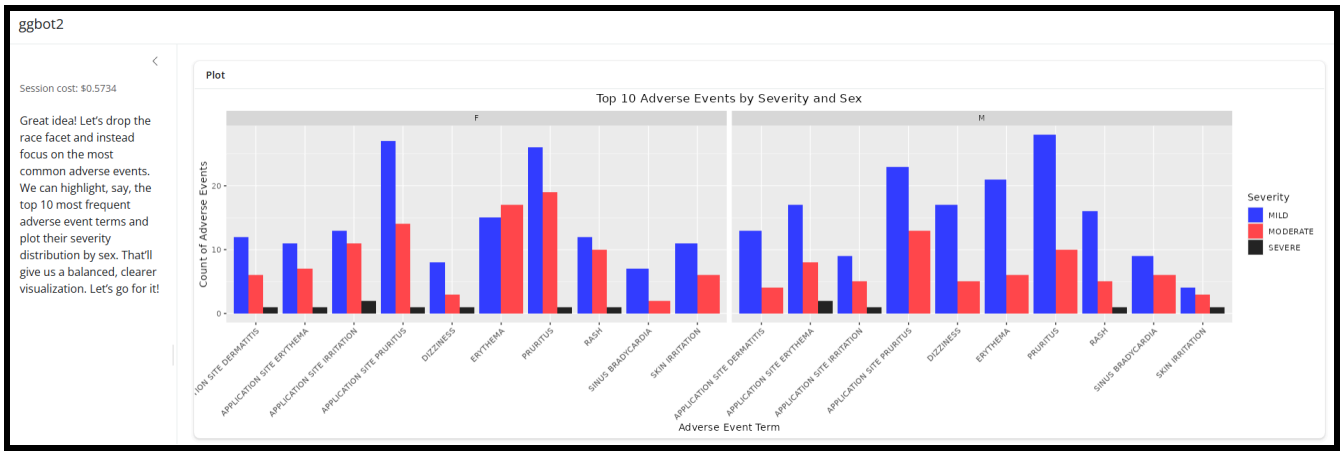
Simple Bar chart with changes to Age:



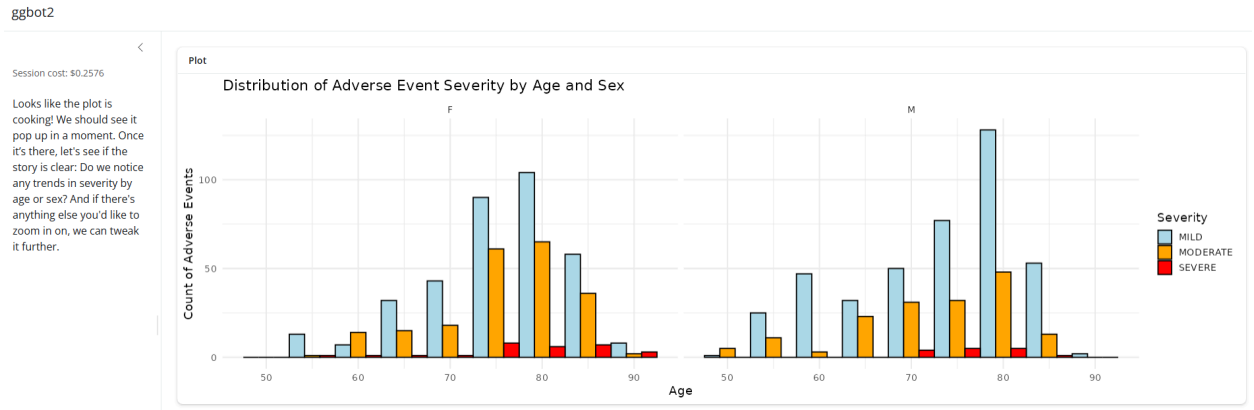
Simple Bar chart with changes to Age groups:



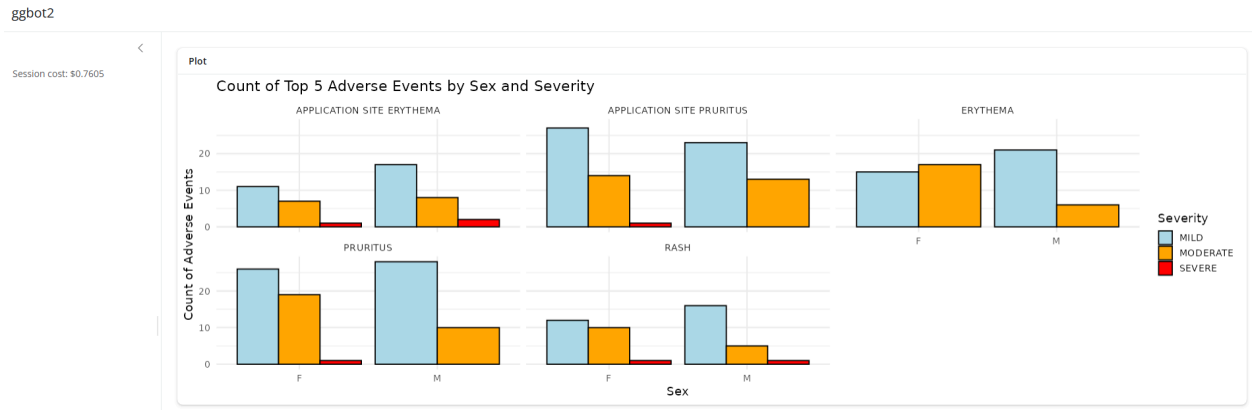
Bar chart with adverse events shown:



Bar chart with severity broken out by age:



Bar chart with severity broken out by age, sex and top 5 adverse events:



Notes on ggbot2 Usage

Below are some notes on using gggbot2.

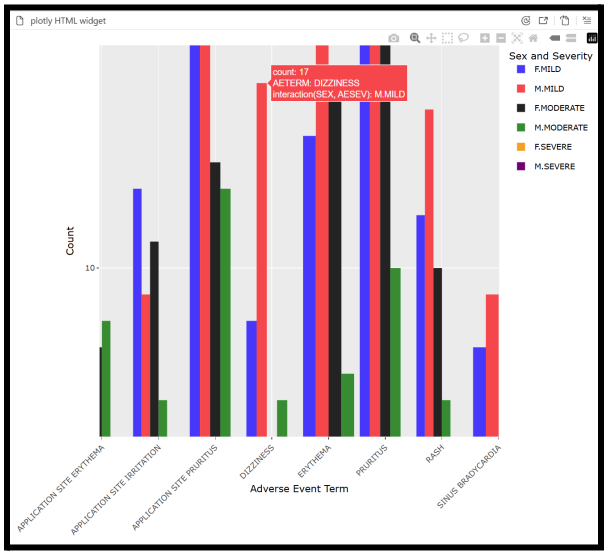
Errors - as with AI, ggbot2 can make mistakes. Users can ask the assistant to go back and start over or try to fix the error. While writing this paper, I found error resolution tricky and found it was almost always better to start over or go back. A few errors I encountered were:

A. ggplot2 errors such as "`stat_density()` requires an x or y aesthetic."
B. R or Shiny code errors such as "Error in rep_len: invalid 'length.out' value"
C. Hallucinations such as "Warning: Error in count: Must group by variables found in `.data`. ✖ Column `AGE_GROUP_10YR` is not found."

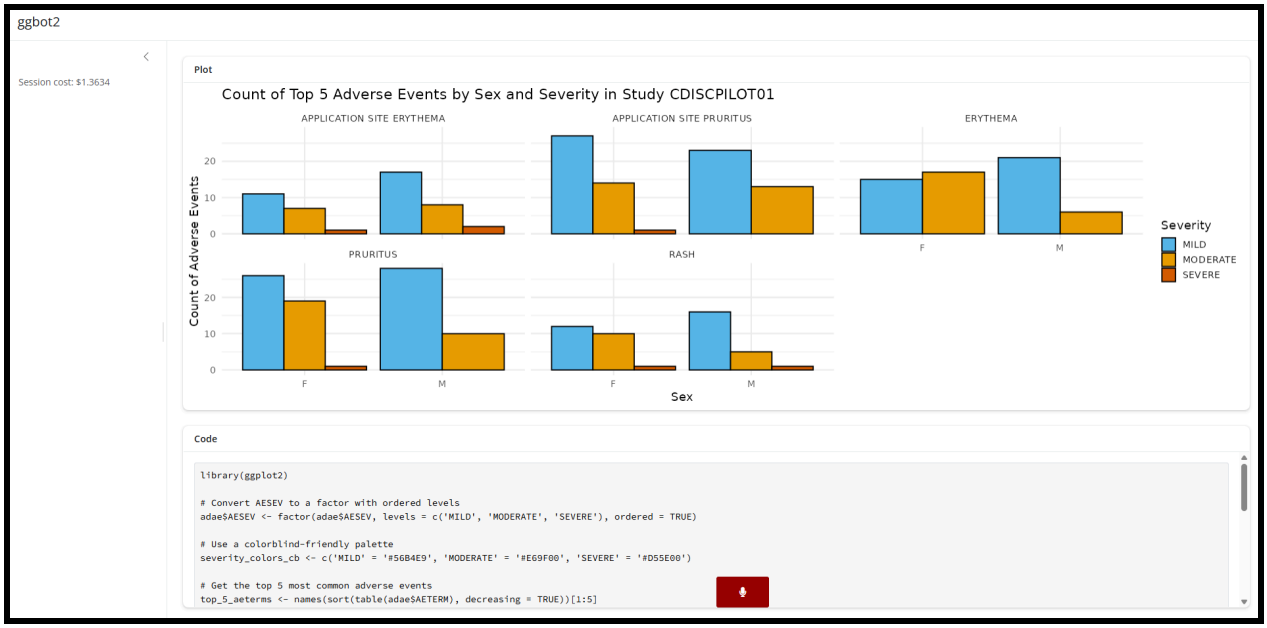
```
> ggbot2::ggbot(adae)
Warning: Error in count: Must group by variables found in `.data`.
✕ Column `AGE_GROUP_10YR` is not found.
  199: <Anonymous>
  198: stop
  197: value[[3L]]
  196: tryCatchOne
  195: tryCatchList
```

Missing capabilities - since ggbot2 is still under development, some features may be missing such as:

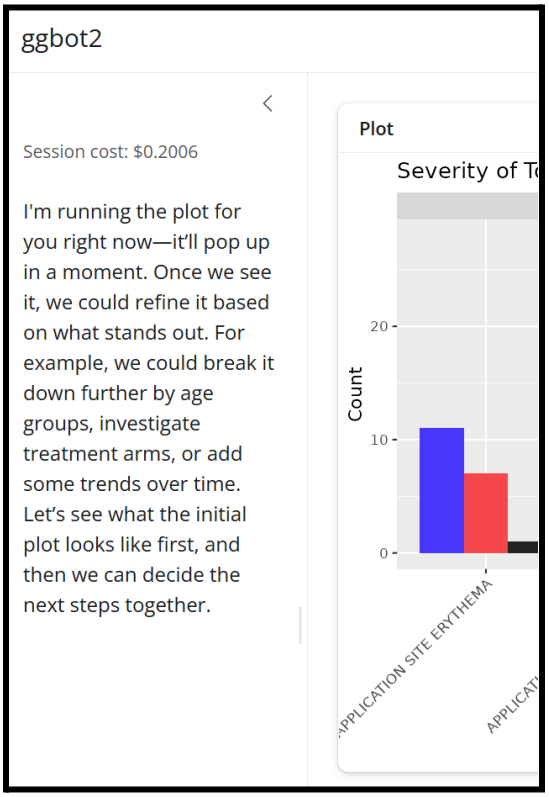
A. Plotly - while I was able to get it to write and wrap the ggplot2 code in plotly, it was unable to show an interactive visualization in the Shiny app itself. I did find however, the plotly code worked well when run independently back in Positron generated by ggbot2.



R Code Reproducibility - ggbot2 will produce code underneath each visualization created. This allows users to save or copy the data into a Quarto document or other location for running independently. The application does not preserve states and once closed, it does not seem possible to retrieve code. Below shows the code generated:



Cost - the ggbot2 Shiny application will show the cost of each session. The session will start at $0.00 and use funds as the code is written. Even saying “hello” seems to cost around $.08 cents while code prompts can cost around the same or a little more or less:



Packages - when I first asked ggbot2 to use plotly, it pushed back and said it was built for ggplot2. I again prompted it and it did indeed try but failed as it did not have plotly in the session as an R package available and gave the error: Warning: Error in library: there is no package called ‘plotly’

This seems to highlight that ggbot2 will try to use packages it does not have available in the session which requires the session to end, package installed and then application restarted. Therefore, be sure to have the dependencies installed prior to engagement with the tool. I thought this was interesting as other AI tools, like Databot in Positron, will indeed see and even fix missing packages by asking or attempting the install.

Python - Python has a similar package to R's ggplot2 called plotnine (https://plotnine.org/). I tried to get ggbot2 to write python code with plotnine but it refused in the Shiny app even though Python was installed on my server. I did however get it to write the plotnine code in the output on the left, to compare or copy as needed.

Future Notes

The information above develops further the exciting use-case of voice driven data science and code generation. It seems clear that in the future, it will be more common to use tools/apps like *Spokenly* to get speech-to-text in various open-source packages and tools. As an OS level app, it could be used to talk into Slack, a browser etc. and could certainly be used as a workaround if an R/Python tool/app doesn't have a native voice feature.

https://spokenly.app/

Tools like ggbot2 could be designed with a simple input control to use for example tools such as Spokenly. The difference would be to speak into Spokenly, which would translate the voice to text and then it would submit the text to an app. The future state could be one where a native voice feature would be available for tools like ggbot2, where the assistant is engaged in conversation and can interrupt, essentially talking to the assistant in real time like one would a human. Developers today use spokenly to interact with Claude Code - providing the experience of being able to speak in real time and then revise the text to be edited and completed as needed.

Conclusion & Contact

This paper highlights ggbot2, a voice driven R package for helping to write ggplot2 code. As voice engagement with AI becomes more common, it seems likely that more open-source tools will have similar capabilities used in drug development to help modernize clinical processes with innovative new features and capabilities such as ggbot2 detailed here. Moreover, ggbot2 can help lower the barrier of entry for writing R and Python code, helping users get content created faster or assisting in writing code for new users, who are already accustomed to voice driven AI engagement with promoting.

Phil Bowsher

phil@posit.co

https://github.com/philbowsher