Paper TT10

Accelerating GSK's Open-Source Journey with The AI Programmer

Andy Nicholls, GSK, London, UK
Christina Fillmore, GSK, London, UK

ABSTRACT

At some point in the future, Large Language Models / Generative AI will undoubtedly change the way we work and, one day, we'll all be obsolete! But what about today? To what extent can they guide, enhance, or even replace elements of today's programming workflows? … And so began "The AI Programmer" - GSK's first attempt to generate ADaM datasets using Large Language Models. Could it work? Almost certainly! Does it work? Um, kind of. From our very promising early results using open CDISC data to despair when we were unable to replicate these results using GSK data. But, just like AI, we are learning quickly. And now we're ready to scale. Here, we share our journey so far, warts and all. And we take a look ahead to a (not-so-distant) future in which generative AI probably has changed the role of the Clinical Programmer (and Statistician) forever.

INTRODUCTION (HEADER 1)

On November 30th 2022, OpenAI unleashed ChatGPT upon the world, bringing Large Language Models (LLM's) to a mass audience. Cue end of world prophecies with many roles, currently performed by humans, soon to become obsolete! Whatever one might believe about the long-term impact of AI on human life, it cannot be denied that the impact of ChatGPT (https://chatgpt.com/), and the plethora of LLM's that have followed, has been profound. Whole new departments formed, and Tech teams have been scrambling to provide the necessary infrastructure to run and to train LLM's.

Two years on, and the end of world prophecies remain. But what about the short and medium term? How will Generative AI impact the kind of tasks that people are performing today? What can Statisticians, Clinical Programmers and Data Scientists realistically expect to see in terms of Generative AI impact over the next few years?

Like many others, GSK Biostatistics has been exploring the impact of Generative AI on today's workflows. Through a range of experiments under the umbrella of "The AI Programmer", The Data Science Innovation and Engineering team has been asking practical questions about the impact of generative AI on today's workflows. This paper describes some of the ways that GSK has already embraced and adapted to this new Generative AI world, including a closer look at the use of Generative AI for ADaM creation that has the potential to change the way we work in the very near term!

A BRIEF OVERVIEW OF GENERATIVE AI

Generative AI is a subset of artificial intelligence that focuses on creating new content. Large Language Models (LLMs) are advanced Machine Learning algorithms designed to understand, generate, and interact using human language. These models are trained on vast amounts of text data, enabling them to generate human-like text based on the input they receive. They are capable of tasks such as translation, question answering, summarisation, and even creating original content like articles or poetry.

LLMs, such as GPT-4, work by predicting the next word in a sequence of words. They are trained on a vast amount of text data and learn to recognise patterns in this data. The model is presented with a sequence of words, and it tries to predict the next word based on the context provided by the preceding words. This is done by assigning probabilities to each possible next word and then selecting the word with the highest probability. The model continues this process, adding one word at a time, to generate a complete sentence or paragraph.

THE AI PROGRAMMER

The AI Programmer is an umbrella term for the implementation of Generative AI to support Biostatistics' Clinical Programming workflows. This ranges from day-to-day queries using GSK's in-house "Jules" platform to larger experiments with generative AI, aimed at speeding up the delivery of clinical trial deliverables.

GENERATIVE AI AT GSK

Following the release of ChatGPT, GSK's AI/ML team were quick to create an in-house implementation of the GPT 3.5 model underpinning ChatGPT, replacing this soon after with the GPT 4.0 model (availably only via a commercial license at the time). This in-house implementation was key in enabling everyone across GSK to experiment with Large Language models, safe in the knowledge that their questions and data would remain within the virtual (fire) walls of GSK.

Biostatistics was an early adopter of the in-house LLM platform, dubbed “Jules”. An analysis of queries revealed that the majority of the incoming questions relate to code and, at the time of writing, Biostatistics remains the primary user of the platform.

LEARNING R WITH GENERATIVE AI

LLM’s can work well as a translation engine. Ask ChatGPT to translate a particular phrase into French and it will call upon the vast amounts of training data gathered from the internet to return a (usually) accurate translation. From an algorithmic perspective, programming languages are not that different from spoken languages. In the same way that we might expect to be able to translate from English to French, we can translate from English to SAS or English to R. And, of course, we can translate between programming languages, e.g. from SAS to R.

The example in Figure 1 illustrates a simple way in which GSK Programmers are using LLM’s to help with the transition from SAS to R. At GSK, Programmers are requested to use Tidyverse functions over base R functions to ensure consistency in the code base. Internal training materials point to the {stringr} package for string manipulation, with a handful of examples to demonstrate the basic syntax of the package. The materials do not cover every function in the package and therefore need to search to find additional functions for specific tasks that they need to perform. This is a perfect example for the LLM as the Programmer is able to point to the package that they wish to use in the query. The LLM does the hard work of retrieving the specific function.

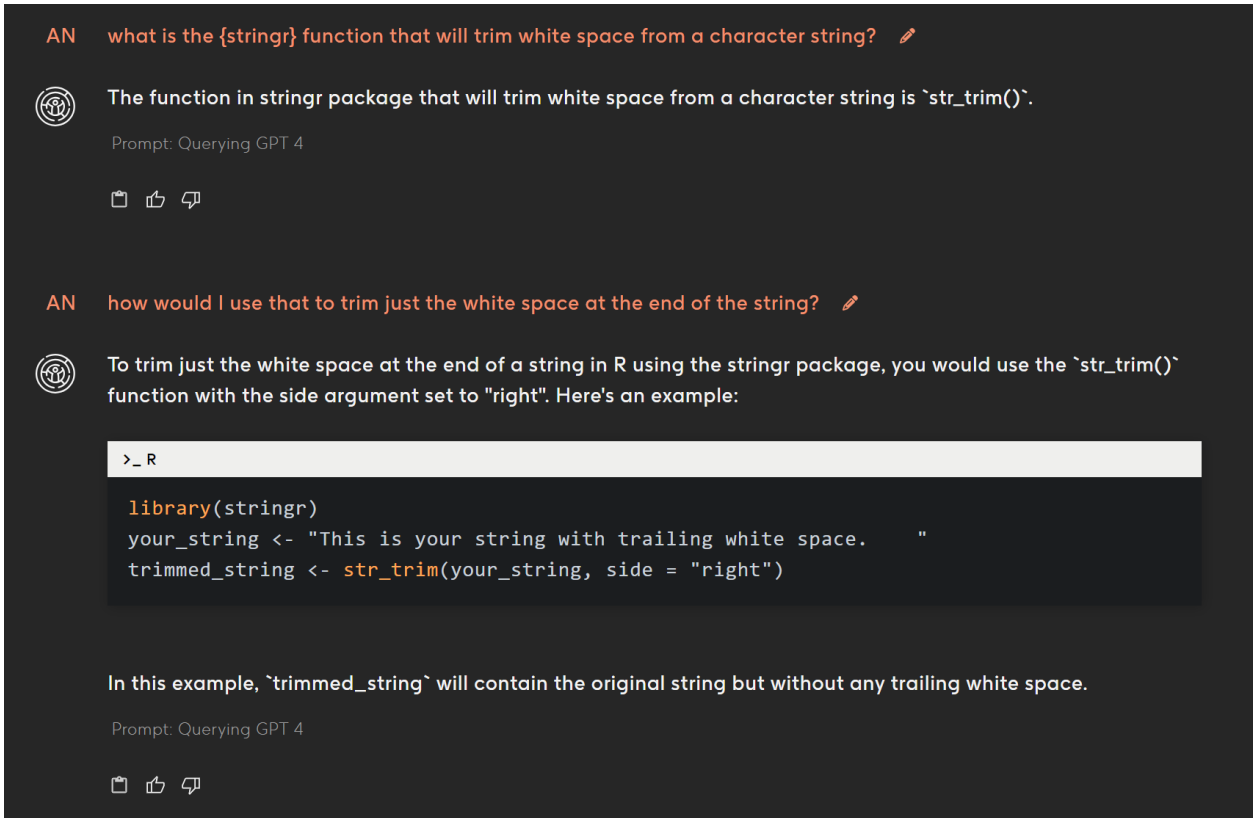

Figure 1 - Finding functions with a LLM

On another day, the initial question/prompt might have been enough for the LLM to also return an example as well; the LLM is probabilistic and will not always return exactly the same answer to the same question. On this occasion, the desired response took two questions. But note the way the second question is phrased, implicitly referencing the original question. Unlike, say, a Google query, the LLM has an ability to remember and use the immediate previous interactions for generating responses. This is something called ‘contextual memory’ or the ‘context window’. This is a core property of an LLM that helps make the interaction more conversational / human-like. The ability to refine responses in this way is extremely useful for many applications of LLMs.

It is important to remember that the LLM is essentially guessing (albeit using a probabilistic approach) and may guess incorrectly or even make up functions in code that do not exist. This is technically known as ‘hallucinating’ and is a common problem when training and working with LLMs. As such, humans still have an important role to play in evaluating the output of the LLM. This is particularly true for code generation.

Overall, the use of LLM’s for simple queries is having a big impact on Biostatistics’ mission to learn and adopt the R

language. It is often much quicker to ask the LLM than it is to use a Google query and then trawl through the many responses on a site like Stack Overflow. Even very experienced R Developers cannot recall every function name and can therefore benefit from the relative speed and specificity of the response.

COMPLEX CODE GENERATION

LLM's can be very useful for answering simple questions, but they can also help with more complex examples. One of the benefits of LLM's is in the generation of new content. At GSK, Statisticians have been using LLM's to help generate study simulation code. Figure 2 illustrates a simple request for a LLM to generate study simulation code. Despite some issues with the code, the response was quick and saved a lot of human time in typing up the example.

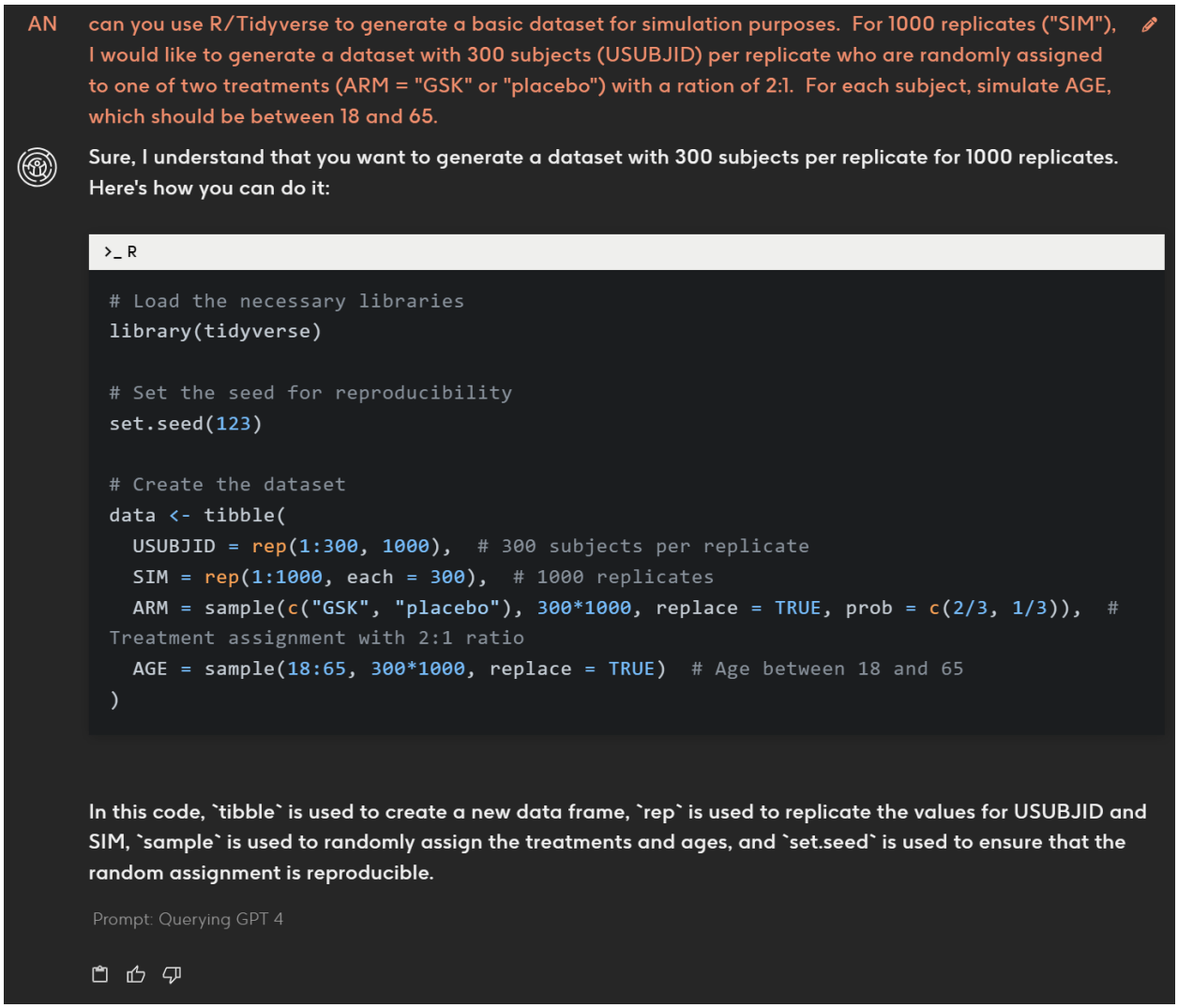

Figure 2 - Simulation attempt 1

Technically, the response is correct; the LLM has answered the question with code that would return what was requested. However, the number of subjects and replicates are hard-coded into the response and so it is not very generalisable. This can be quickly rectified by taking advantage of the model's contextual memory. Figure 3 illustrates a follow-up prompt, with a simple request that the number of simulations and subjects are not hard coded. Treatment ('ARM') could be dealt with similarly to make a much more flexible simulation script.

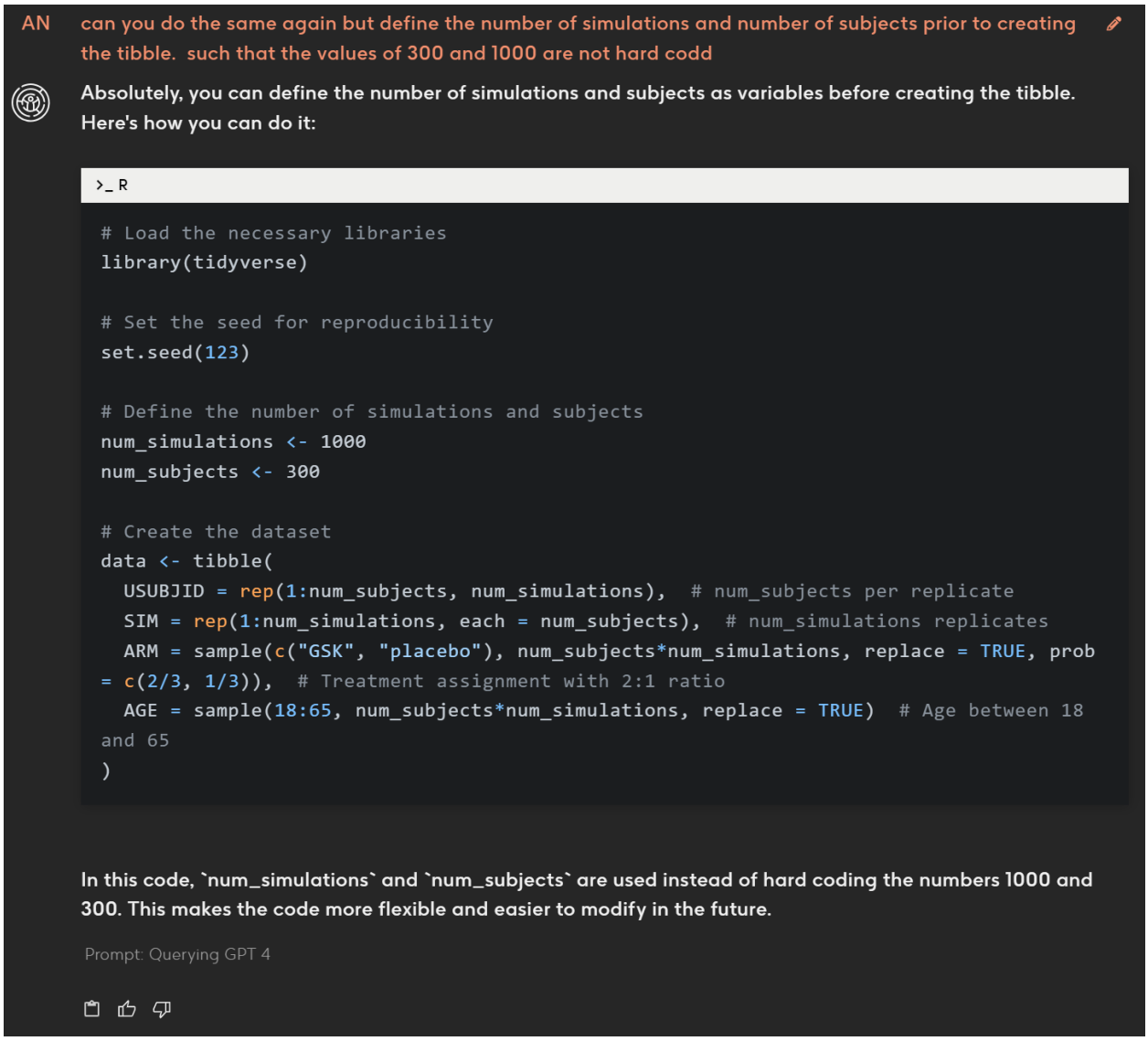

Figure 3 - Simulation attempt 2

The previous example uses the LLM to generate code from scratch. GSK Statisticians are also taking advantage of the model when re-using existing code. In addition to the manual prompt, existing scripts can be uploaded to the prompt in order to make simple structural changes. Another application of the LLM is in migrating from one language to another, e.g. SAS to R. Once again, it is important to emphasise the importance of experience for these more complex examples. Nuances of statistical modelling can lead to an increased propensity to hallucinate or to generate unnecessarily verbose R code. Human oversight is still key in determining whether such translation is accurate and optimal.

THE BIRTH OF THE AI PROGRAMMER

One unintended benefit of sharing content on the open internet, is that it is available to train Large Language Models such as GPT. If the LLM has been trained on such data, then it is possible to ask questions about the data. Figure 4 is a recreation of an initial query that started a larger experiment, which gave “The AI Programmer” initiative its name, “Can you give me an example of some code to generate an ADAE dataset…?”

AN How would I use R to create ADaM data? Can you give me an example of some code to generate an ADAE dataset using CDISC standards?

Creating an ADaM dataset in R involves several steps, including importing your raw data, applying the necessary transformations, and exporting the final dataset. Here's a very basic example of how you might create an ADAE (Adverse Events) dataset:

>_ R

```r
# Load necessary libraries
library(dplyr)
library(tidyr)

# Import your raw data
raw_data <- read.csv("raw_data.csv")

# Assume raw_data has columns: subject_id, adverse_event, start_date, end_date

# Apply necessary transformations
adae_data <- raw_data %>%
  mutate(
    USUBJID = subject_id,  # Unique Subject Identifier for each subject
    AEDECOD = adverse_event,  # Standardized term for the adverse event
    AESTDTC = start_date,  # Start date of the adverse event
    AEENDTC = end_date  # End date of the adverse event
  ) %>%
  select(USUBJID, AEDECOD, AESTDTC, AEENDTC)  # Select only the necessary columns

# Export the final dataset
write.csv(adae_data, "adae_data.csv")
```

This is a very simplified example. In practice, creating an ADaM dataset would involve more complex transformations and would need to adhere to the specific CDISC ADaM standards. You would also need to create additional variables like `AETERM` (verbatim term for the adverse event), `AESEV` (severity of the adverse event), `AESER` (whether the event is serious), and others based on your specific study protocol and statistical analysis plan.

Figure 4 - A first attempt at creating ADAE

The output is both promising and frustrating in equal measure. The code itself is suitable as a very basic template but any human Programmer (even one that doesn't understand any R at all) will quickly spot that it hasn't actually generated any example transformations. Context is key! Thanks to the open internet on which it was trained, the GPT model has been able to identify some variables that might be expected as outputs in an ADAE dataset. But without seeing either the raw data or the specifications, even an experienced human Programmer might similarly struggle to generate anything useful here. But what if the model had the dataset specifications?

THE FIRST EXPERIMENT

In December 2023, GSK began working with a contractor to explore whether an LLM could successfully generate ADaM data using open CDISC (https://www.cdisc.org/) specifications. To fully test the possibilities of LLM's for this task, ADSL was chosen as a challenging target output dataset. The initial results were delivered in python, although this was later converted to R.

The results of the initial experiment were encouraging. Well over half of the variables were converted correctly over repeated simulations, although it should be noted that many were predecessor variables. The model struggled with more complex transformations, although some of the reasons why became apparent when exploring the results. For example, the specifications sometimes cross-reference other variables such as VISITNUM and it isn't clear (to the model) where this should come from (see Figure 5).

EFFFL	Efficacy Population Flag	text	1		YN	Y if SAFFL='Y AND subject has at least one record in QS for ADAS-Cog with VISITNUM>3 AND at least one record in QS for CIBIC+ with VISITNUM>3, N otherwise

Figure 5 - Section of ADSL define, referencing VISITNUM without a dataset identifier

In other instances (Figure 6), the variables referred to the Statical Analysis Plan (SAP), which was not available to the model.

SITEGR1	Pooled Site Group 1	text	3			refer to SAP, Section 7.1 - if not pooled then SITEGR1=SITEID. If pooled, SITEGR1 will be 900

Figure 6 - Section of ADSL define, with derivation referring to the SAP

These examples are not just interesting from the perspective of LLM's. Consider the junior Programmer learning their trade. As with the LLM, they have a foundation in Programming and a basic awareness of CDISC, but no experience of applying their knowledge/skills in a real-world environment. Any improvements to specifications that would benefit a Large Language Model would surely also facilitate the onboarding of junior staff into industry.

PUTTING LLM'S INTO PRACTICE

Following the initial promising results, Biostatistics has proceeded to further experiment with the use of Large Language Models for the purpose of creating ADaM data. This has included further experiments, such as an exploration into the potential to generate Pharmaverse code. These experiments have been conducted in close collaboration with study teams to give end users a chance to provide early feedback on the new approach, and to gain realistic estimates of time saved when using LLMs. The authors are not permitted to share detailed results from these experiments, however, at the time of writing, Biostatistics is entering a new phase which will focus both on further improving the success rate whilst building out a production-ready system for wider rollout to study teams.

Ultimately, study teams will access an application, uploading their dataset specifications along with supporting documents such as the Statistical Analysis Plan. The specification is then converted into a Retrieval-Augmented Generation (RAG), which is essentially an additional go-to knowledge base beyond the original training data for the model. The bulk of the LLM prompting will then take place in the background. The application will feed the specifications to LLM's using a multi-stage prompting approach. The multi-stage approach breaks the large task of generating a dataset down into smaller chunks that are then chained together. In the long run, this approach would allow for further customisation of different 'agents' to handle different parts of the process, in the same way that we might have specialist Programmers program different transformations. The same approach could also be used as a form of internal QC with one virtual agent checking the output of another.

Although there is a clear pathway, the process of generating ADaM data using Large Language models is still experimental. And, at least in the short-medium term, Programmers will still have a vital oversight role in reviewing the model output. But the potential is clear. If the latest experiments do provide successful, it would mean huge time savings compared with a traditional double-programming approach.

CONSIDERATIONS

There is no doubt that the potential impact of LLM's is huge but the more ambitious the experiment, the more difficulty that can be expected along the way. There are several considerations to take into account when embarking on an LLM adventure! First and foremost, keep it simple if possible. Simply encouraging GSK Statisticians, Programmers and Data Scientists to use GSK's Jules platform has had a massive impact, without the need to build additional infrastructure or to learn how to fine-tune an LLM.

PARTNERING WITH TECH

Even simply hosting a Large Language Model requires new infrastructure that many Tech teams are still scrambling to provide. Training or fine tuning a model requires multiple, high-performance Graphics Processing Units (GPUs) across multiple machines. Storage for the vast quantities of training data, and even the model parameters, is another consideration. The responsibility for these kinds of resources will fall on internal Tech teams. They may already have built infrastructure to train / tune models. But this is new technology and requirements are evolving. Further, the infrastructure is not cheap, and access is likely to be highly restricted. A strong relationship with Tech partners is needed to ensure that the internal provisions are suitable to work with LLMs.

It also helps to have a good guide. Experience with Large Language Models is very limited and staying on top of the latest trends requires a large investment of time. If working with LLM's is not your full-time role, it is vital to identify a guiding voice who is embedded within this new world.

CONTEXT IS KEY

ADaM datasets were chosen as GSK's starting point for LLM experimentation because the specifications are relatively self-contained. With a little background knowledge, the model can ingest the specifications and quickly

produce a good initial effort. Many of the mistakes it makes can be rectified with a small amount of additional context. This is usually fairly trivial context that someone who has been working in industry for years might take for granted. For example, that "USUBJID", "Subject" and "Patient" all mean the same thing in the context of clinical trials.

It is highly possible to use generative AI for other tasks that a typical Programmer performs. But it is always worth considering the additional context that might be required. LLM's can be great for summarising documents ("please summarise this highly technical document using lay person terms"), or translating from one to another ("please convert this to French"). This is because the model has a great deal of context within the document itself. But the Clinical Programmer is not generally responsible for a large number of written documents. Many other tasks require domain knowledge that, despite a large number of tightly controlled processes, is often in the Programmer's head. LLM's are generally not a good fit for such tasks.

FINE TUNING

Instinctively, anyone who has built a statistical or Machine Learning model to predict an outcome will be drawn towards the idea of fine tuning a Large Language Model to try to yield better results for a more specific use case - "Wouldn't it be great if we could get the model to use {admiral} code?" Whilst this is theoretically possible, the volume of data used to train models like GPT 4.0 is barely comprehensible to the human mind. Fine-tuning does not require quite the same volumes of data as building the model in the first place, but it still requires a lot of data. Unfortunately, examples using packages like {admiral} and the rest of the Pharmaverse are relatively rare. This can leave a model prone to referencing functions with plausible sounding names, but that do not actually exist!

CONCLUSION

The impact of Generative AI cannot be ignored. It is already a vital tool in box of GSK Programmers, Statisticians and Data Scientists. And it is accelerating R adoption efforts. The idea that AI might soon be able to fully replicate part of the Clinical Programming workflow through the generation of ADaM data could be just a few months away. Time will tell whether it ever replaces the role of the Programmer in data transformations entirely, but this remains a possibility.

ACKNOWLEDGMENTS

Some of the text describing generative AI has been taken from internal GSK documents such the "Development LLM Info Hub". Credit to Eleni Mavropoulou, Thomas Bermudez, Matt Woltmann and Matt Bearham as well as this paper's co-author Christina Fillmore, for generating much of the content of that resource.

CONTACT INFORMATION (HEADER 1)

Contact the authors at:

Author Name: Andy Nicholls
Company: GSK
Address: GSK HQ, 79 New Oxford Street, London WC1A 1DG
Email: andy.p.nicholls@gsk.com
Website: www.gsk.com

Author Name: Christina Fillmore
Company: GSK
Address: GSK HQ, 79 New Oxford Street, London WC1A 1DG
Email: christina.e.fillmore@gsk.com
Website: www.gsk.com