Paper ET09

Navigating the Safe Integration of AI (ChatGPT) into Clinical Trials Programming Workflow

Mykyta Vysotskyi, Intego Group, Warsaw, Poland

ABSTRACT

Working toward the constant push to shorten the time and effort required to process clinical trial data, the industry is diligently trying to replace the current statistical programming process with artificial intelligence (AI). This is a complex and expensive endeavor that does not seem to give much consideration to the tools currently available that could be helpful in the present moment. This article goes against this trend by discussing immediate, free, and safe methods for easily and effectively bringing AI into the regular work of a statistical programmer.

In this presentation, practical ideas for smoothly integrating AI (using ChatGPT) into the everyday routine tasks of clinical trial programmers will be introduced. Statistical programming is an area which continually looks for new ideas within traditional methods. This article explains strong methods that use the synergy between human expertise and AI capabilities, showing a practical way to make use of AI's potential in clinical trials for improving efficiency and productivity.

INTRODUCTION

In the world of programming, where complex tasks and a high degree of precision are integral, the question arises of how to improve and optimize processes. In this context, the use of advanced technologies and artificial intelligence (AI) becomes relevant. This article introduces a revolutionary tool - ChatGPT 3.5, developed by OpenAI. With its remarkable ability to generate human-like text and code in various programming languages, this artificial intelligence can serve as an assistant in the field of programming, particularly in clinical research. This work explores the experience of using ChatGPT on the GPT-3.5 architecture, as this platform is completely free and most commonly used among similar ones, emphasizing its ability to safely generate code in SAS and R programming languages.

WHAT IS CHATGPT 3.5

ChatGPT, or Generative Pre-trained Transformer, is a deep learning model based on the transformer architecture, which enables efficient processing and generation of natural language. The GPT-3.5 model was trained on diverse internet sources until January 2022 using supervised learning and reinforcement learning methods, allowing it to improve its responses based on feedback. This implies more accurate text generation and an enhanced ability to adapt to the context of queries.

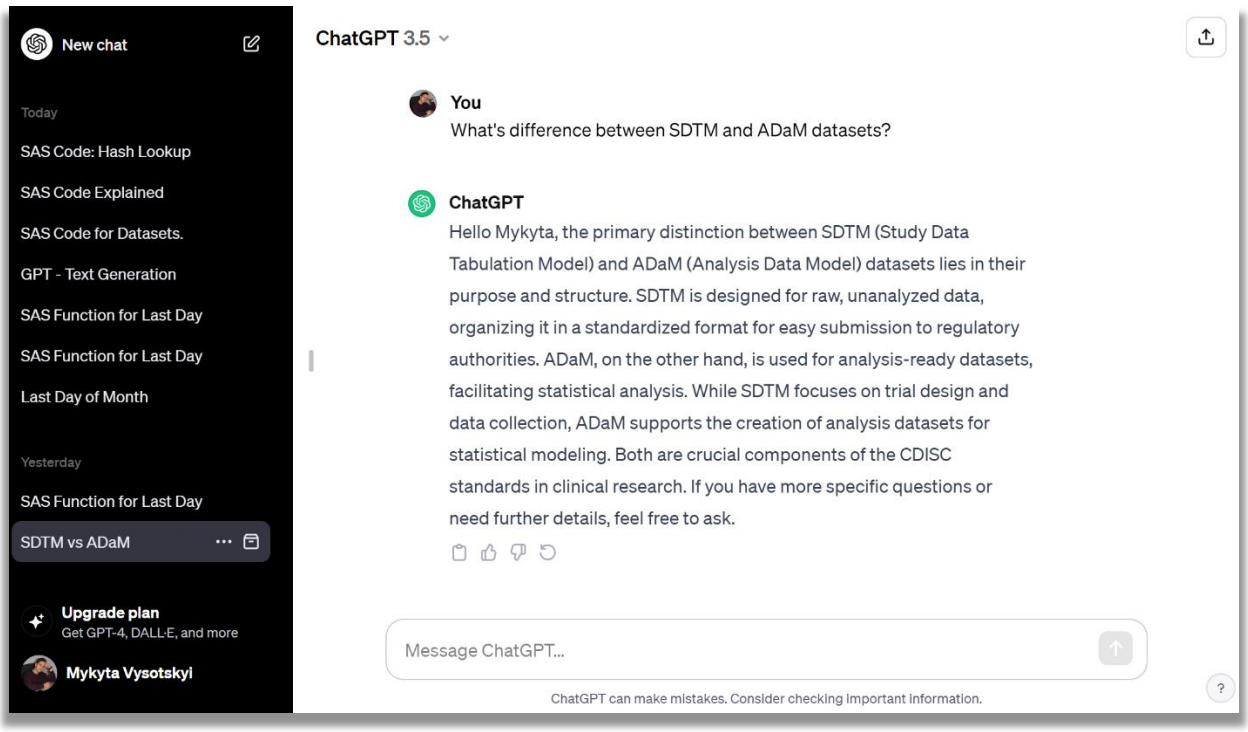
The system uses machine learning to analyze and process large volumes of data, enabling it to recognize, interpret, and generate human language. It can perform various tasks, ranging from answering common questions to generating code in different programming languages. An important feature of the GPT-3.5 model is its ability to undergo continuous learning, allowing it to constantly expand its knowledge and improve the quality of generated content.

ChatGPT does not possess consciousness or subjective opinions; it relies solely on the information acquired during training and lacks the capability for independent thinking or learning without external input.

HOW TO USE CHATGPT 3.5

It may sound obvious, but to start working with the chatbot, you need to go to the website (www.chat.openai.com) and register using one of the provided methods. You should use ChatGPT just like a regular chat. To get started, simply enter a message in the main page's input field. Instead of a live person, you'll have artificial intelligence as your conversation partner. The bot interprets your messages as queries and responds to them, usually within a few seconds. You can initiate a conversation with a simple greeting like 'Hello. Tell me a joke,' or you can make a specialized query like 'What's the difference between SDTM and ADaM datasets?' For your convenience, it's better not to mix different types of queries and create new chats. All chats will be saved on the left side of the interface, and you can switch between them.

An example of the ChatGPT interface with the request 'What's the difference between SDTM and ADaM datasets?':



As you can see, the response 'Hello Mykyta, the primary distinction between SDTM (Study Data Tabulation Model) and ADaM (Analysis Data Model) datasets lies in their purpose and structure. SDTM is designed for raw, unanalyzed data, organizing it in a standardized format for easy submission to regulatory authorities. ADaM, on the other hand, is used for analysis-ready datasets, facilitating statistical analysis. While SDTM focuses on trial design and data collection, ADaM supports the creation of analysis datasets for statistical modeling. Both are crucial components of the CDISC standards in clinical research. If you have more specific questions or need further details, feel free to ask.' is quite comprehensible.

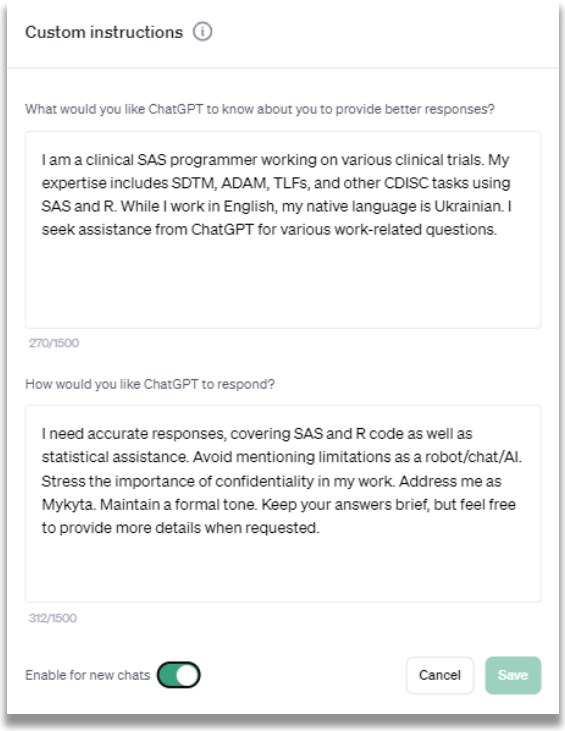
If the received answer doesn't satisfy you, you can manually regenerate the response. To do this, click the 'Regenerate Response' button or ask ChatGPT to do so in a dialogue format. The second method is better because it will be easier for the bot to understand what it did wrong, and this will help it learn. However, try not to use too many attempts for response regeneration, as the number of requests is limited. There may come a moment when ChatGPT disables the use of the service and asks you to come back in an hour - this is done to avoid excessive loads. Nevertheless, don't worry if your dialogue is interrupted - the message history is saved.

It's important to understand that the work of the chatbot is based on the analysis of a vast amount of textual data. ChatGPT processes this data in search of the answer that it believes best fits your message. For this reason, the bot can make mistakes and provide irrelevant or even incoherent information. But the advantage of artificial intelligence is that it constantly learns by processing new information and interacting with people. You can see how the bot learns even in one chat. To do this, argue with it and prove the correct point of view.

ChatGPT possesses skills in SAS and R programming languages, making it a valuable tool in supporting clinical programmers in clinical trials. In SAS, the model can handle clinical programming tasks, including assisting in creating SDTM, ADAM, and TLFs. It can provide code and statistical assistance in data analysis. In the R programming language, ChatGPT also demonstrates its expertise. It can generate code, perform statistical analyses, and assist in various data processing and graph creation tasks. It is important to emphasize that these skills are based on the analysis of extensive data and information obtained during the training.

CUSTOM INSTRUCTIONS

To improve the accuracy of AI chatbot responses, I recommend filling out custom instructions, which include two forms: Personal Information and Response Preferences. You can open this form by clicking on your name (user name) in the lower left corner of the interface, then go to 'Custom Instructions.' The form looks as follows:



In the Personal Information section, I recommend providing details about your occupation and any specific characteristics about yourself. I use the following text: 'I am a clinical SAS programmer working on various clinical trials. My expertise includes SDTM, ADAM, TLFs, and other CDISC tasks using SAS and R. While I work in English, my native language is Ukrainian. I seek assistance from ChatGPT for various work-related questions.'

Response preferences can include various instructions for the chatbot on how you want it to respond to you. It all depends on your specific preferences, but I use the following preset: 'I need accurate responses, covering SAS and R code as well as statistical assistance. Avoid mentioning limitations as a robot/chat/AI. Stress the importance of confidentiality in my work. Address me as Mykyta. Maintain a formal tone. Keep your answers brief, but feel free to provide more details when requested.

SECURITY AND CONFIDENTIALITY

In the context of clinical trials, the safety of using ChatGPT remains the foremost concern. While the creators of the chatbot claim that it does not store chat information, it 'learns' from each conversation. And, of course, users 'communicate' with the chatbot on the internet, where there are no guarantees of security. More information about the confidentiality of ChatGPT 3.5 can be found on the official OpenAI.

In the context of programming in clinical trials, the key thing to understand when using ChatGPT 3.5 is adherence to the principles of Good Clinical Practice (GCP) and the Confidentiality Policy of your company. In a general sense, these principles can be summarized as follows:

- Data Confidentiality: Emphasis on the absolute confidentiality of all data related to the clinical trial, including patient information, trial protocols, and any programming code or algorithms developed.
- Regulatory Compliance: Commitment to adhering to all applicable regulatory requirements and standards, such as Good Clinical Practice (GCP), ensuring that programming activities align with industry guidelines for data security and confidentiality.
- Access Controls: Implementation of strict access controls and permissions for any databases, files, or systems containing sensitive clinical trial data. Only authorized personnel should have access to such information.
- Encryption: Use of encryption technologies for the transmission and storage of sensitive data to prevent unauthorized access, especially when dealing with patient information and clinical trial results.
- Secure Coding Practices: Adherence to secure coding practices to mitigate the risk of vulnerabilities and ensure that any software or programs developed maintain the confidentiality and integrity of the data they handle.
- Non-Disclosure Agreements (NDAs): Requirement for all personnel, including programmers, to sign NDAs, underscoring their commitment to maintaining the confidentiality of clinical trial information and preventing any unauthorized disclosure.
- Data Handling Procedures: Detailed procedures for the handling, processing, and storage of clinical trial data, specifying the steps to be taken to prevent data breaches or leaks during programming tasks.
- Documentation: Thorough documentation of all confidentiality measures implemented, including the details of access controls, encryption methods, and any changes made to programming code to enhance security.

Based on the above information, let's consider some considerations for the safe use of ChatGPT 3.5 in the context of programming in clinical trials:

- Data and Patient Data Confidentiality: Avoid providing confidential medical information or data that can be used to identify patients. Maintain high standards of confidentiality in accordance with clinical trial requirements.
- Business Information Security: Do not use ChatGPT to discuss details of clinical projects; make sure you do not disclose confidential business data. Discuss general principles and tasks, avoiding specific project details if

necessary. Avoid disclosing the names of pharmaceutical and other companies, project names, and the names of drugs and molecules.

- Data Evaluation and Verification: ChatGPT generates responses based on a vast amount of data and may not always ensure absolute accuracy. Validate and verify the information received, especially when high precision is required in the interpretation of clinical trial data.
- Use for Consultations: ChatGPT can be useful for general consultations and discussions of methodologies and standards in clinical trials. However, do not rely on it for making critical decisions regarding patient safety or research data.
- Ensuring Ethical Interaction: When interacting with ChatGPT in the context of clinical trials, ensure compliance with ethical standards, especially regarding patient confidentiality.
- Working with Files and Data in ChatGPT 4: A more advanced version of the chatbot, namely ChatGPT 4, is capable of handling datasets, images, and text documents. Never upload any types of files containing confidential or sensitive information into ChatGPT 4.

PRACTICAL APPLICATION

Let's move directly to the practical application of ChatGPT 3.5. You can assign almost any task related to text and code to the chatbot. Of course, this doesn't mean it will be executed perfectly, but the bot will definitely take it on. Over time, artificial intelligence will improve and perform your tasks more effectively. I have chosen three tasks with which ChatGPT handles well and which demonstrate its potential in the field of programming in clinical trials. It's important to note that the following or any other tasks you assign to ChatGPT should be implemented strictly according to the principles described in the Security and Confidentiality section.

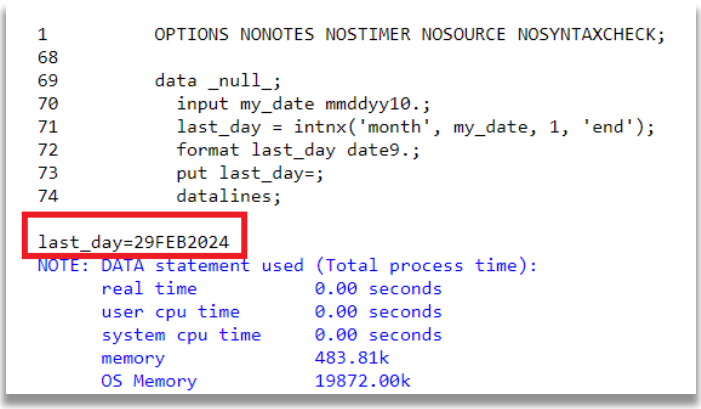
SMART SAS REFERENCE GUIDE

ChatGPT can be used as a reference guide; it excels in finding the SAS functions you need based on textual descriptions of the function. Let's consider an example of searching for a rarely used and interesting SAS function. Suppose we need to obtain the last day of the month based on the month and year of the input date. To search for the required function, I turned to ChatGPT with the following query: *'Please find me a function in SAS that will return the last day of the month based on the input date?'* and received the following information in response:
"*Certainly, Mykyta. In SAS, you can use the INTNX function to find the last day of the month based on an input date. Here's an example code:*

```
data _null_;
  input my_date mmddyy10.;
  last_day = intnx('month', my_date, 1, 'end');
  format last_day date9.;
  put last_day=;
  datalines;
01/15/2024
;
run;
```

Replace '01/15/2024' with your input date. The INTNX function is used with the 'month' interval and 'end' modifier to get the last day of the month. The result will be displayed in the 'last_day' variable.
Please let me know if you need further clarification or assistance."

Then I checked the functionality of the provided SAS code. The test date suggested by ChatGPT was '01JAN2024,' and according to the intention, we should obtain the number '31.' When I executed the provided code, I obtained the following result:



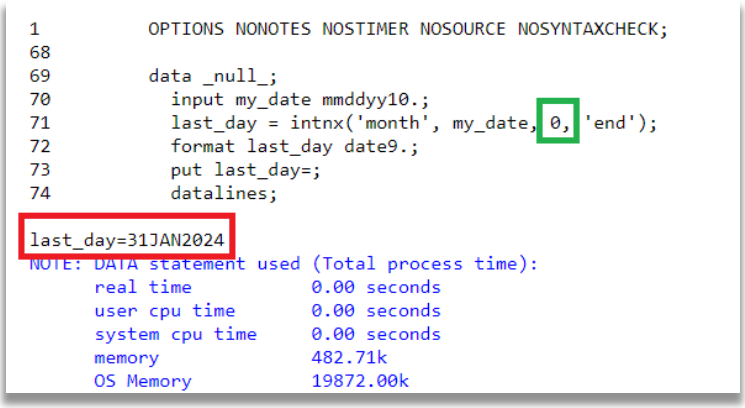
```
1          OPTIONS NONOTES NOSTIMER NOSOURCE NOSYNTAXCHECK;
68
69         data _null_;
70           input my_date mmddyy10.;
71           last_day = intnx('month', my_date, 1, 'end');
72           format last_day date9.;
73           put last_day=;
74           datalines;

last_day=29FEB2024
NOTE: DATA statement used (Total process time):
      real time           0.00 seconds
      user cpu time       0.00 seconds
      system cpu time     0.00 seconds
      memory              483.81k
      OS Memory           19872.00k
```

The variable 'last_day' contains the date of the last day of the subsequent month from the input test date, as seen in the log in the red box (29FEB2024). While we received a close result, as the last day of February 2024 is indeed the 29th, but not the exact result we needed, we continue the conversation with the chatbot and point out the error made. The clarifying query should include the essence of the mistake and a request for correction. We submit a clarifying query in the following form: *'Using your code I get the latest date of the NEXT month, when I need the latest day of a month based on the input date. Please correct your code.'* And we receive the following response:
"I apologize for the oversight, Mykyta. You're correct, and I appreciate your patience. Here is the corrected SAS code to obtain the last day of the month based on the input date:

```
data _null_;
  input my_date mmddyy10.;
  last_day = intnx('month', my_date, 0, 'end');
  format last_day date9.;
  put last_day=;
  datalines;
01/15/2024
;
run;
```

By setting the third argument of the INTNX function to 0, we ensure that it returns the last day of the current month. Please test it with your input date, and let me know if you have any further questions."

We execute the corrected code:

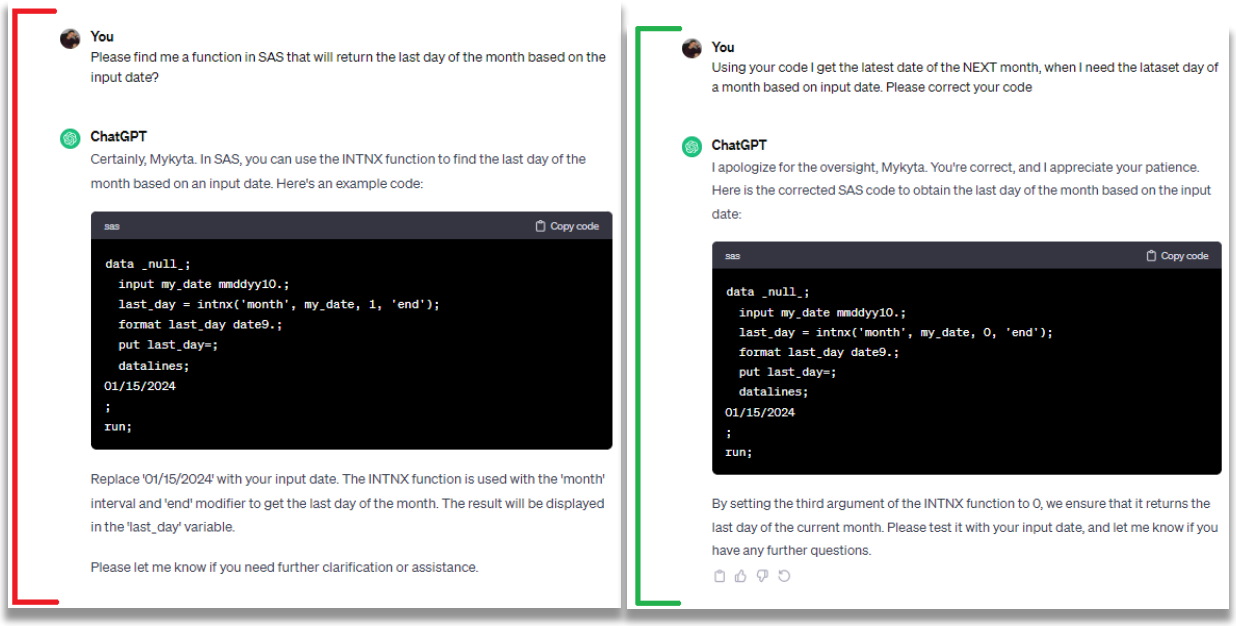


```
1          OPTIONS NONOTES NOSTIMER NOSOURCE NOSYNTAXCHECK;
68
69         data _null_;
70           input my_date mmddyy10.;
71           last_day = intnx('month', my_date, 0, 'end');
72           format last_day date9.;
73           put last_day=;
74           datalines;

last_day=31JAN2024
NOTE: DATA statement used (Total process time):
      real time           0.00 seconds
      user cpu time       0.00 seconds
      system cpu time     0.00 seconds
      memory              482.71k
      OS Memory           19872.00k
```

After running the corrected code provided by the chatbot, we obtain the desired result '31JAN2024' (red box). ChatGPT modified the increment argument of the INTNX() function from 1 to 0 (green box), allowing it to process the input date directly, rather than the next month of the input date.

It's worth noting that the query was focused on obtaining the result in the form of a day, and the obtained result is not absolutely precise since it provides the complete date. However, ChatGPT provided a functional code that executed correctly and provided a complete, and most importantly, correct full date. This is sufficient for further use, either by applying the less common DAY() function to extract the day or by refining the query until you obtain the exact day. This is a good example that illustrates the importance of verifying the result provided by the chatbot when working with ChatGPT. In turn, thoughtful interaction between the specialist and the AI chat is capable of achieving a lot.

In the general view, the chat screenshot is inserted below, and it's worth noting that ChatGPT provides code in a special box, making it very convenient for copying code. The initial query is highlighted in red, and the correction with the final and correct result is highlighted in green.



ASSISTANCE WITH UNFAMILIAR SYNTAX

ChatGPT 3.5 excels at explaining specific or unfamiliar syntax, which can accelerate your work with complex programs. Let's consider an example in which ChatGPT step by step explains SAS code with hash objects, with which I was not familiar until the time of writing this work. We'll take an SAS code that is unclear to us and ask ChatGPT to explain it step by step using the following query: *'Please explain me this part of SAS code step by step:*

```
data hash_code;
   if 0 then set work.CountryCapitals;
   if _n_ = 1 then do;
      declare hash hh (dataset: "work.CountryCapitals");
      hh.definedata("Country");
      hh.definekey("Capital");
      hh.definedone();
   end;
   set capitals;
   rc =hh.find();
   if rc = 0 then capital = capital;
run;'
```

In response, we receive the following code explanation: *"Certainly, Mykyta. Let's break down the provided SAS code step by step:*

1. Data Step Declaration: `data hash_code;`

This line initiates a new data step named ***hash_code.***

2. Conditional Set Statement: `if 0 then set work.CountryCapitals;`

This statement is a conditional set statement. Since ***if 0*** *is always false, this line effectively prevents the data set* ***work.CountryCapitals*** *from being read during the data step.*

3. Initialization of Hash Object:

```
if _n_ = 1 then do;
  declare hash hh (dataset: "work.CountryCapitals");
  hh.definedata("Country");
  hh.definekey("Capital");
  hh.definedone();
end;
```

This block of code initializes a hash object named ***hh*** *when the first observation (****_n_ = 1****) is processed. It specifies that the hash object will use the data from the* ***work.CountryCapitals*** *dataset, with "Country" as data and "Capital" as the key.*

4. Reading and Updating Data:

```
set capitals;
rc = hh.find();
if rc = 0 then capital = capital;
```

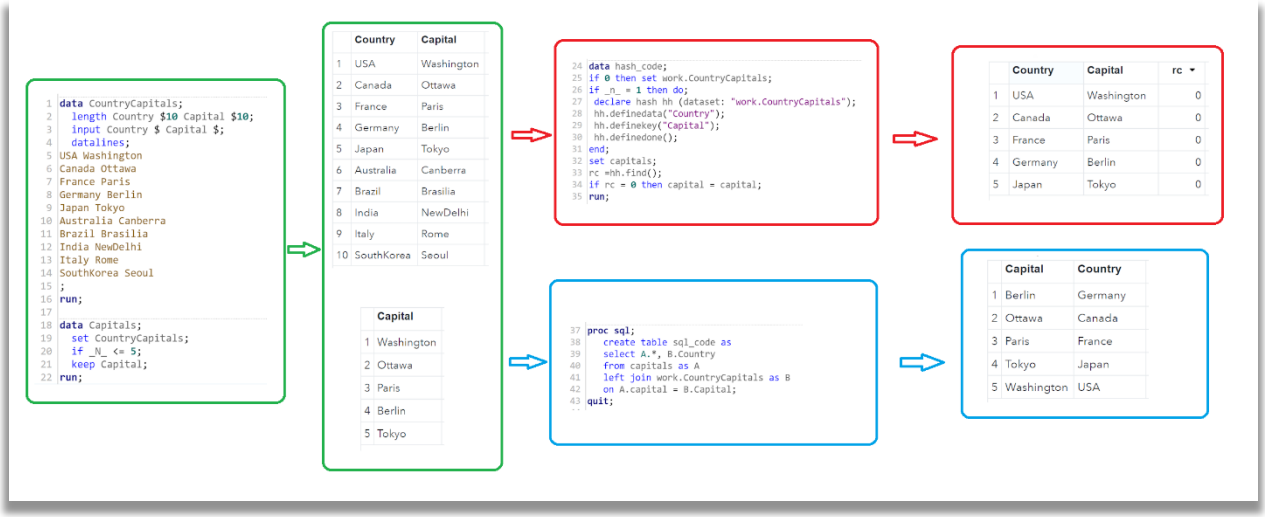
*This part of the code reads an observation from the **capitals** dataset and searches for a corresponding record in the hash object **hh**. If a match is found (**rc = 0**), it updates the variable **capital** with its current value.*
5. Closing the Data Step: `run;`
This line concludes the data step.
*In summary, this code uses a hash object to efficiently look up values in the **work.CountryCapitals** dataset based on the "Capital" key. The **capitals** dataset is then processed, and if a match is found in the hash object, the "capital" variable is updated.*
If you have specific questions or need further clarification on any part, feel free to ask."

I find this response comprehensive. ChatGPT step by step explained the provided code, correctly outlining the logical steps and making a conclusion about the outcome of executing this code. However, let's verify its logic. To do this, I request from the chatbot code with similar functionality and syntax that I understand, in the form of the following message: *'Provide me alternative SAS code using proc sql.'* In response to this request, we receive SAS code that utilizes PROC SQL with some explanations: *"Certainly, Mykyta. Here's an alternative SAS code using **proc sql** to achieve a similar result:*

```
proc sql;
   create table hash_code as
   select A.*, B.Country
   from capitals as A
   left join work.CountryCapitals as B
   on A.capital = B.Capital;
quit;
```

*This **proc sql** code performs a left join between the **capitals** dataset (aliased as A) and the **work.CountryCapitals** dataset (aliased as B) based on the "capital" and "capital" variables, respectively. The result is stored in a new table named **hash_code**, which includes all columns from the capitals dataset along with the "Country" variable from the **work.CountryCapitals** dataset.*
This SQL-based approach achieves a similar outcome to the hash object method, combining information from both datasets based on the common key. If you have any specific requirements or questions about this alternative, please let me know."

To perform the verification, we will prepare two datasets, 'CountryCapitals' and 'Capitals.' It's worth mentioning that I also assigned this task to ChatGPT to save time, and it kindly provided code for creating a dataset with 10 countries and their capitals, as well as a dataset with a subset of 5 capitals. We will run the original code and the alternative code on these datasets and observe the results.



As you can see, the contents of the obtained datasets are the same; the difference lies only in the sorting and the presence of the 'RC' variable in the dataset generated with the hash code. From this, we can conclude that ChatGPT 3.5 correctly interpreted the original code with hash syntax and proposed a correct alternative using PROC SQL.

CODE TRANSFORMATION FROM SAS TO R

Perhaps my favorite capability of ChatGPT 3.5 is its ability to transform SAS code into R code (and vice versa). Let's see how ChatGPT handles this task. We will consider a quite basic code for creating an adverse event table by SOC and Preferred Terms for treatment groups, and also create test datasets AE and SL. We will formulate a request with some settings that will help the chatbot process the code more accurately: *'Could you please convert the following*

SAS code to R code? Please prioritize the usage of dplyr and tidyr libraries. If you use any library, kindly initialize it at the top of the code. Remember that R is case-sensitive. Refer to the comments in my code; they should assist you.

```
/* Reading of CSV file AE */
data ae;
  infile "/home/vysnikita420/AE.csv" delimiter=' ' firstobs=2 dsd;
  length SUBJID $3 AEDECOD $10 AEBODSYS $10 TRTP $1;
  input SUBJID AEDECOD AEBODSYS TRTP;
run;
/* Reading of CSV file SL */
data sl;
  infile "/home/vysnikita420/SL.csv" delimiter=' ' firstobs=2 dsd;
  length SUBJID $3 TRTP $1;
  input SUBJID TRTP;
run;
/* Calculation of number subjects by events by SOC by treatment group. Pay attention
on NODUPKEY statement!!! */
proc sort data = ae out = ae_sort nodupkey;
  by TRTP SUBJID AEDECOD;
run;
proc freq data = ae_sort noprint;
  table TRTP*AEBODSYS*AEDECOD/ out = ae_freq_out (drop = percent);
run;
/* Calculation of number of subject in treatment group (will be used as denominator
for percentage calculation) */
proc freq data = sl noprint;
  table TRTP/ out = sl_freq_out (drop = percent rename = (count = denom));
run;
/* Calculation of percentage */
data results;
  merge ae_freq_out
        sl_freq_out;
  by TRTP;
  length result $40;
  result = put(count,3.-r) || " (" || put(count/denom*100, 5.1) || ")";
run;
proc sort data = results;
  by AEBODSYS AEDECOD;
run;
/* Preparing final data structure */
proc transpose data = results out = results_tr;
  by AEBODSYS AEDECOD;
  id TRTP;
  var result;
run;
/* Final dataset. Filling missing results with "  0 (  0.0)", keep only needed
variables*/
data final_sas;
  set results_tr;
  if a = "" then a = "  0 (  0.0)";
  if b = "" then b = "  0 (  0.0)";
  keep AEBODSYS AEDECOD a b;
run;'
```

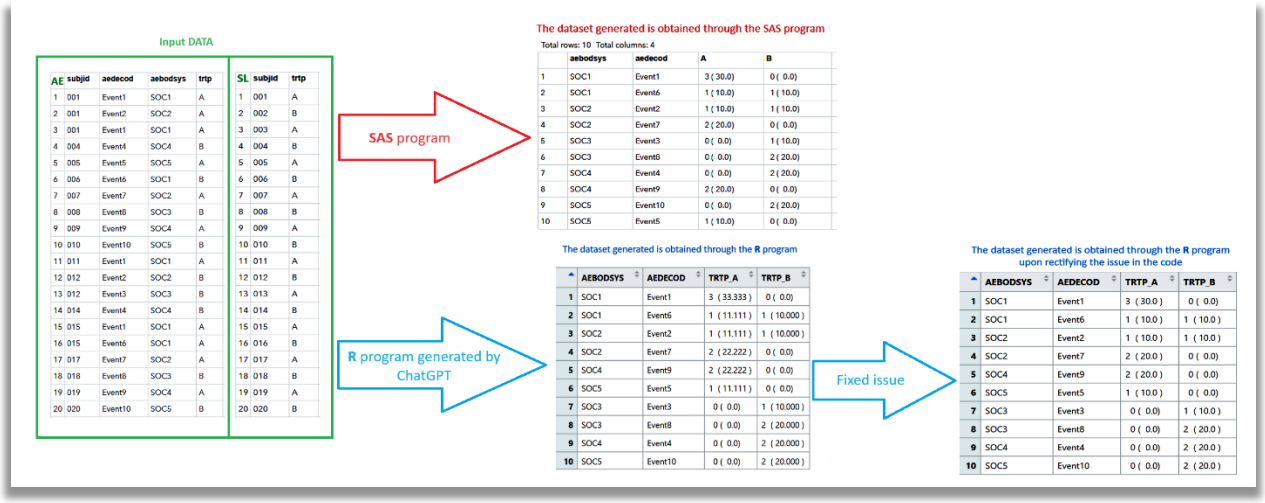
In response, we receive the following: *"Certainly, Mykyta. Below is the equivalent R code using the dplyr and tidyr libraries. Make sure to adjust the file paths accordingly.*

```
# Initialize libraries
library(dplyr)
library(tidyr)
# Reading of CSV file AE
ae <- read.table("/home/vysnikita420/AE.csv", sep = ' ', header = TRUE, skip = 1,
col.names = c("SUBJID", "AEDECOD", "AEBODSYS", "TRTP"))
# Reading of CSV file SL
sl <- read.table("/home/vysnikita420/SL.csv", sep = ' ', header = TRUE, skip = 1,
col.names = c("SUBJID", "TRTP"))
```

```
# Calculation of number subjects by events by SOC by treatment group
ae_sort <- ae %>%
  arrange(TRTP, SUBJID, AEDECOD) %>%
  distinct()
ae_freq_out <- ae_sort %>%
  group_by(TRTP, AEBODSYS, AEDECOD) %>%
  summarise(count = n()) %>%
  ungroup()
# Calculation of number of subjects in the treatment group
sl_freq_out <- sl %>%
  group_by(TRTP) %>%
  summarise(denom = n()) %>%
  ungroup()
# Calculation of percentage
results <- merge(ae_freq_out, sl_freq_out, by = "TRTP") %>%
  mutate(result = paste(format(count, digits = 3, scientific = FALSE), " (",
format(count/denom * 100, digits = 5, nsmall = 1), ")"))
# Preparing final data structure
results_tr <- results %>%
  select(-count, -denom) %>%
  pivot_wider(names_from = TRTP, values_from = result, names_prefix = "TRTP_")
# Final dataset. Filling missing results with "  0 (  0.0)", keep only needed
variables
final_r <- results_tr %>%
  mutate(across(starts_with("TRTP"), ~coalesce(., "  0 (  0.0)"))) %>%
  select(AEBODSYS, AEDECOD, starts_with("TRTP"))
```

Please adjust the file paths in the read.table functions to match your actual file locations. Let me know if you have any questions or need further assistance."

Great, let's verify the generated R code. To do this, we adjust the path to the input files as needed and execute the code. The code ran without a single error. Let's compare the obtained result with what we got when executing the original SAS code. Here, we can see that the result is different, specifically the percentages do not match. Upon closer examination of the R program's code, we find that the error lies in incorrect data reading. We change **'header = TRUE**' to **'header = FALSE**' in the **'read.table()**' function. After running the slightly corrected R code, we obtain a result that matches the content of what we obtained using the SAS code. Therefore, ChatGPT also handled this function. The results are provided in the image below.



Here are some recommendations for using this function:

- Formulate your request correctly, specifying preferred libraries.
- Write the original code as simply and concisely as possible.
- Include as many comments as possible and indicate their usefulness in the request.
- If necessary, send logical parts of the code in one chat instead of the entire code

CONCLUSION

Having explored the basics and the surface-level functionalities of ChatGPT 3.5, I am convinced that ChatGPT 3.5 is worth a try for anyone interested in discovering the capabilities of artificial intelligence. You can customize functions for the bot to perform tasks that will simplify your work as a programmer in clinical research. However, it is crucial to keep in mind that ChatGPT lacks its own consciousness and operates based on data provided by the user. When used in clinical research, it is essential to apply it with a high level of confidentiality and in accordance with security and ethical norms in this field. Furthermore, do not rely on ChatGPT to entirely replace human input - its responses should be reviewed, proofread, and adjusted as needed.

REFERENCES

1) Privacy policy - OpenAI
   URL: openai.com/policies/privacy-policy/

2) E6(R2) Good Clinical Practice: Integrated Addendum to ICH E6(R1) Guidance for Industry, 2018
   URL: fda.gov/media/93884/download

CONTACT INFORMATION

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

Mykyta Vysotskyi
Senior Statistical Programmer/Analyst
Intego Group, LLC
78 Grzybowska street
Warsaw, Poland / 02-677
Email: Mykyta.Vysotskyi@intego-group.com