PharmaSUG China 2025 - Paper AI-171

AI Application Attempt - Recognition and Conversion of Spec Files to SAS

Jinting Luo, Beijing Key Tech Statistical Consulting Co., Ltd;

ABSTRACT

In recent years, the clinical trial field continues to explore the opportunities and challenges AI large language models (LLM) bring. This paper aims to use the current-stage AI (DeepSeek R1) for real-time application, examining its potential to replace simple, repetitive tasks within the scope of statistical programmers' responsibilities (e.g., routine dataset programming for Finding-class Domain), shifting from manual creation to AI creation supplemented by human review, modification, and refinement.

Given the current-stage AI's reliance on precise instructions and deployment environments, it remains impractical to directly generate all CDISC datasets and TFLs in one step from trial design-related documents like SAP/SHELL in actual workflows. Therefore, this study focuses on Spec files with established mapping rules post-human processing, exploring the templatization and semi-automation of SDTM/ADaM programming.

INTRODUCTION

Reviewing recent case studies on AI applications in clinical trial data analysis reveals that integrating AI into work and learning tasks is inevitably becoming a major trend. Some have explored AI's ability to execute instructions based on known data mapping relationships like metadata, achieving tasks such as searching, filtering, and returning data results [1]. Others have investigated AI's capabilities in parsing SAS code and interactive scenarios, such as explaining function purposes, adding code comments, checking syntax errors, and understanding/writing macros [2]. Additionally, some have evaluated LLM performance in algorithm translation, code generation, and data visualization [3]. The findings are encouraging—AI can now preliminarily handle routine coding tasks—but its reliance on human verification, feedback, and continuous training for complex tasks remains high. In terms of efficiency, it still lags significantly behind seasoned statistical programmers proficient in SAS and clinical knowledge.

Undoubtedly, despite the rapid development of AI, current-stage AI functionalities still exhibit shortcomings when executing complex instructions, including semantic ambiguity, long-range dependencies, multi-step reasoning flaws, real-time limitations, and data access constraints. Thus, at this stage, AI can only serve as a tool requiring human guidance. On one hand, we must continue exploring AI's application scenarios; on the other hand, workflow and procedural adjustments are necessary to find a balance that truly delivers output.

FILES PREPARATION

LLM INSTRUCTIONS

The purpose of this part is to provide clear instructions to AI, categorizing the workflow from natural language processing of Spec files to the final output of CDISC dataset programs, forming structured, distributed, and example-driven instructions. This includes steps such as: acquiring external dynamic information, organizing target dataset creation rules, classifying different variables, modularizing code templates, providing reference approaches for complex variable creation, and standardizing code details.

Step1

Suppose you are a senior SAS programmer in the field of clinical trials.

Please process the relevant files in the order specified in this document and generate the corresponding code.

Different steps are separated by blank lines.

Step2

The attached file "01 Dataset Rules.xlsx" contains the rules for creating CDISC datasets for the project.

Please organize it into a table that fully includes the following columns: Dataset, Order, Variable, Label, Type, Length, Controlled_Terms_or_Format, Origin, Mapping_Rule, and Programmer_Notes.

At the same time, identify the domain to which each target dataset belongs based on the "Dataset" column.

If there are multiple sheets, perform the required steps for each sheet according to their names respectively.

Step3

Please classify the variables involved in the dataset according to the following rules and present them in a table.

At this time, the table only needs to list the serial number, variable name, and classification type. The classification basis is the Mapping Rule column.

It should be noted that the variables involved in the SUPP domain will appear in the parent domain dataset in the naming form specified in QNAM.

The following are the rules for identifying classifications:

Type 1: Variables that can be directly captured from the original dataset or other SDTM datasets and do not require additional processing.

Type 2: Simple variables that can be directly generated according to the mapping rules (such as character or numeric variables assigned directly).

Type 3: Variables derived from existing variables in this dataset and do not require additional processing.

Type 4: Other variables that require complex processing.

Type 5: SUPP variables.

Step4

The attachment "02 Program Template.txt" is the template used as a reference for writing CDISC programs.

In this template, [Domain_U] represents the uppercase CDISC domain name, and [Domain_L] represents the lowercase CDISC domain name. Please replace the contents in the template.

Step5

Identify different code sections based on the comment statements in the attachment "02 Program Template.txt".

Step6

Write the target variables involved in Type 1 into Section 4 of the template.

At this point, there is no need to retain the steps for key variables.

Note:

At this time, it is assumed that the program runtime environment configuration has been completed.

The current configuration logic library "raw" represents the logic library where the original data set is located, and "SDTM/ADaM" represents the logic library where the corresponding generated CDISC data set is located.

The "raw.aaa" in the template represents the original data set aaa (only as an example), "SDTM.DM" represents the SDTM data set DM.

Please write the correct data set according to the actual situation of the Mapping_Rule column.

Step7

Write the original dataset variables or other SDTM/ADaM dataset variables required for Type 4 into the section 1 of the template.

Note: This step only involves simple variable retention and merging operations.

The original dataset variables needed for the calculation of the variables in step four are retained using the "keep" statement.

No actual processing (such as: if the lbperf variable is the source of LBSTAT variable, then only this variable needs to be retained in the original dataset section) is performed here.

The specific complex calculations of the variables in step four will be added in the subsequent code.

Step8

Write the target variables involved in Categories 2 and 3 into the second section of the template.

Note: The length of character-type variables is uniformly specified as $900.

The variables in Type 3 can only appear here after the source variables have been processed in the previous code; otherwise, they will appear as complex variables in the next step.

At the same time, when assigning values to the variables involved in Type 3, they should be through the upper-level variables (that is, the standard variables that have been processed in this dataset).

Step9

Write the specific calculation rules for the target variables involved in classification four in the format specified in the template to the third section.

If there is a sequence of creation among the complex variables, then repeatedly write the code in the style of the third section multiple times.

Create new variables based on the requirements of the variables, after the existing [Domain_U]3 dataset, and increment the naming sequence number accordingly (for example: [Domain_U]4, [Domain_U]5).

Among them:

The SEQ variable is not included in the Type five range;

[Varlist] section is obtained based on the Varlist column in "01 Dataset Rules.xlsx";

[MergeVars] section is obtained based on the MergeVars column in "01 Dataset Rules.xlsx";

[Computational Rule] section is obtained based on the Computational Rule column in "01 Dataset Rules.xlsx".

If it is empty, then further search is conducted to see if there is a corresponding calculation rule in "03 Programming Rules.txt". If still empty, then no replacement will be performed;

Note: If there is no corresponding column in "01 Dataset Rules.xlsx", then it is necessary to infer and write based on other known information today, and add comments after the corresponding code to remind users to carefully check.

Step10

Output the SUPP variable that has already been processed in the parent domain dataset in the style specified in Section 5 of the template.

Step11

If the users do not provide the corresponding variables for KeyVars, then you need to create an appropriate KeyVars sequence based on the overall data set structure and variable situation in "01 Dataset Rules.xlsx", and add comments after the corresponding code to remind the users to carefully check.

Step12

Output a txt file containing the complete SAS code. The comment statements for dividing the sections, blank lines, and other statements that are not affected by variable creation remain unchanged.

If there are multiple target data sets, separate download buttons will be provided for each of them.

DATASET RULES

This part specifies the variable creation rules for the target dataset, derived from the project's Spec files. Considering the slight structural differences in Spec files across companies and to streamline the information AI needs to process when reading files, we convert the corresponding Excel sheet into a txt file, retaining only essential variables for dataset creation. Examples include: domain name, creation order, variable name, variable label, variable type, controlled terminology, and mapping rules.

Dataset	Order	Variable	Label	Type	Length	Controlled Terms or Format
LB	1	STUDYID	Study Identifier	Char	200	
LB	2	DOMAIN	Domain Abbreviation	Char	200	LB
LB	3	USUBJID	Unique Subject Identifier	Char	200	
LB	4	LBSEQ	Sequence Number	Num	8	
LB	8	LBTESTCD	Lab Test or Examination Short Name	Char	200	(LBTESTCD)
LB	9	LBTEST	Lab Test or Examination Name	Char	200	(LBTEST)
LB	13	LBCAT	Category for Lab Test	Char	200	
LB	15	LBORRES	Result or Finding in Original Units	Char	200	
LB	16	LBORRESU	Original Units	Char	200	(UNIT)
LB	20	LBORNRLO	Reference Range Lower Limit in Orig Unit	Char	200	
LB	21	LBORNRHI	Reference Range Upper Limit in Orig Unit	Char	200	
LB	22	LBLLOD	Lower Limit of Detection	Char	200	
LB	23	LBSTRESC	Character Result/Finding in Std Format	Char	200	(LBSTRESC)
LB	24	LBSTRESN	Numeric Result/Finding in Standard Units	Num	8	
LB	25	LBSTRESU	Standard Units	Char	200	(UNIT)
LB	26	LBSTNRLO	Reference Range Lower Limit-Std Units	Num	8	
LB	27	LBSTNRHI	Reference Range Upper Limit-Std Units	Num	8	
LB	29	LBNRIND	Reference Range Indicator	Char	200	(NRIND)
LB	30	LBSTAT	Completion Status	Char	200	(ND)
LB	31	LBREASND	Reason Test Not Done	Char	200	
LB	34	LBSPEC	Specimen Type	Char	200	(SPECTYPE)
LB	40	LBLOBXFL	Last Observation Before Exposure Flag	Char	200	(NY)
LB	47	VISITNUM	Visit Number	Num	8	
LB	48	VISIT	Visit Name	Char	200	

LB	49	VISITDY	Planned Study Day of Visit	Num	8	
LB	52	LBDTC	Date/Time of Specimen Collection	Char	200	ISO 8601 datetime or interval
LB	54	LBDY	Study Day of Specimen Collection	Num	8	
SUPPLB	63	RDOMAIN	Related Domain Abbreviation	Char	200	
SUPPLB	64	IDVAR	Identifying Variable	Char	200	
SUPPLB	65	IDVARVAL	Identifying Variable Value	Char	200	
SUPPLB	66	QNAM	Qualifier Variable Name	Char	200	
SUPPLB	67	QLABEL	Qualifier Variable Label	Char	200	
SUPPLB	68	QVAL	dat_rawa Value	Char	200	
SUPPLB	69	QORIG	Origin	Char	200	
SUPPLB	70	QEVAL	Evaluator	Char	200	

Order	Origin	Mapping Rule	Computational Rule	Varlist	MergeVars
1	Protocol(SDTM)	"XXXXXXXXXX"			
2	Assigned(SDTM)	"LB"			
3	Derived(SDTM)	Concatenate DM.STUDYID, DM.SITEID, and DM.SUBJID with "-"			
4	Derived(SDTM)	The sequence number obtained by sorting all observations for each subject in the domain by Key variables.			
8	Assigned(SDTM)	"HCG"			
9		"Pregnancy"			
13		`"Pregnancy test"`;			
15		raw.lburp.lbres_dec			
16		Null			
20		Null			
21		Null			

22		Null			
23	Derived(SDTM)	LBORRES			
24	Derived(SDTM)	LBORRES result converted based on standard unit LBSTRESU			
25	Assigned(SDTM)	LBORRESU			
26	Derived(SDTM)	LBORNRLO result converted based on standard unit LBSTRESU			
27	Derived(SDTM)	LBORNRHI result converted based on standard unit LBSTRESU			
29		Null			
30		If the raw.lburp.lbperf value is "No" or "Not Applicable" LBSTAT="Not Done".	If the raw.lburp.lbperf value is "No" or "Not Applicable" LBSTAT="Not Done".	SUBJID lbperf	SUBJID
31		If the raw.lburp.lbperf value is "No", LBREASND=raw.lburp.lbreasnd, if raw.lburp.lbperf value is "Not Applicable", LBREASND=raw.lburp.lbreasna_dec.	If the raw.lburp.lbperf value is "No", LBREASND=raw.lburp.lbreasnd, if raw.lburp.lbperf value is "Not Applicable", LBREASND=raw.lburp.lbreasna_dec.	SUBJID lbperf lbreasnd lbreasna_dec	SUBJID
34		"Urine"			
40	Derived(SDTM)	"Y" if it is the last non-null test value before administration			
47	Assigned(SDTM)	For scheduled visits: use SDTM.TV linked by VISIT to obtain VISITNUM; For unscheduled visits: link via SDTM.SV by LBDTC to obtain VISITNUM.			
48	Derived(SDTM)	For scheduled visits: Retrieve SDTM.TV.VISIT based on visit name linkage; For unscheduled visits: Obtain VISIT by linking LBDTC with SV			
49	Derived(SDTM)	For scheduled visits, use TV linked by			

		VISIT to obtain VISITDY.			
52		raw.lburp.lbdat			
54	Derived(SDTM)	If neither is empty, subtract the date part of DM.RFSTDTC from the date part of LB.LBDTC, and if the result is greater than or equal to 0, add 1.		USUBJID LBDTC RFSTDTC	USUBJID
63	Assigned(SDTM)	"LB"			
64	Assigned(SDTM)	"LBSEQ"			
65	Assigned(SDTM)	LBSEQ			
66	Assigned(SDTM)	"LBREASO"			
67	Assigned(SDTM)	"Other reasons not applicable"			
68	Assigned(SDTM)	raw.lburp.lbreasno			
69	Assigned(SDTM)	"CRF"			
70	Assigned(SDTM)	Null			

PROGRAM TEMPLATE

This part specifies the fixed structure of the CDISC program that AI needs to produce. On one hand, it divides different sections according to program logic and submission standards for ease of manual reading and writing; on the other hand, when AI computing resources are limited, processing requirements by sections better aligns with its current capabilities.

```
************************************************************************

************************ Prepared by KTSTAT ***************************;

*-----------------------------------------------------------------------
Environment Configuration - Start
-----------------------------------------------------------------------;

%global batfullpath ……;
%macro currentroot();
……
%mend currentroot;

%currentroot();
%include "&batcurrpath.\init.sas";

*-----------------------------------------------------------------------
Environment Configuration - End
-----------------------------------------------------------------------;
*======The following is the program for dataset processing==============*;
************************************************************************
Section 1: Variables Obtained Directly
************************************************************************;
data [Domain_U]1_1;
  set raw.aaa(keep=subject_id lbres_dec lbperf);
  length SUBJID  LBPERF $9000;
  SUBJID = subject_id;
 LBORRES = lbres_dec;
run;

proc sort data= [Domain_U]1_1; by SUBJID; run;
data [Domain_U]1_2;
  merge [Domain_U]1_1(in=OutFL) SDTM.DM(keep=SUBJID USUBJID RFXSTDTC
RFSTDTC);
  by SUBJID;
  if OutFL=1;
run;

data [Domain_U]1;
  set [Domain_U]1_2;
run;

************************************************************************
Section 2: Variables with Direct Assignment
************************************************************************;
data [Domain_U]2;
  set [Domain_U]1;
  length DOMAIN $900;
  DOMAIN='';
```

```
run;

*******************************************************************************
Section 3: Variables requiring complex calculations
*******************************************************************************;
**[Domain_U]LOBXFL;
data [Domain_U]3_[Domain_U]LOBXFL;
  set [Domain_U]2(keep=[Varlist]);
  [Computational Rule]
run;

**[Domain_U]STAT;
data [Domain_U]3_[Domain_U]STAT;
  set [Domain_U]2(keep=[Varlist]);
  [Computational Rule]
run;

proc sort data= [Domain_U]3_[Domain_U]LOBXFL; by [MergeVars]; run;
proc sort data= [Domain_U]3_[Domain_U]STAT; by [MergeVars]; run;
data [Domain_U]3;
  merge [Domain_U]2(in=OutFL)
        [Domain_U]3_[Domain_U]LOBXFL(keep=[MergeVars] [Domain_U]LOBXFL)
        [Domain_U]3_[Domain_U]STAT(keep=[MergeVars] [Domain_U]STAT);
  by [MergeVars];
  if OutFL=1;
run;

*******************************************************************************
Section 4: Sorting
*******************************************************************************;
proc sort data=[Domain_U]3 out=[Domain_U]
sortseq=linguistic(numeric_collation=on);
  by [KeyVars];
run;

proc sort data=[Domain_U] out=[Domain_U]_Check nouniquekey;
  by [KeyVars];
run;

data [Domain_U];
  set [Domain_U];
  by [KeyVars];
  retain [Domain_U]SEQ 0;
  if first.USUBJID then [Domain_U]SEQ=1;
  else [Domain_U]SEQ=[Domain_U]SEQ+1;
run;

*******************************************************************************
Section 5: Generating the SUPP[Domain_U] dataset
*******************************************************************************;
data SUPP[Domain_U];
  set [Domain_U];
  length RDOMAIN IDVAR IDVARVAL QNAM QLABEL QVAL QORIG QEVAL $900;
   RDOMAIN = "[Domain_U]"; IDVAR = "MBSEQ"; IDVARVAL = put(MBSEQ, best.);
   QORIG = "Collected";  QEVAL = "";

  * MBALL;
```

```
  QNAM = "MBALL"; QLABEL = "检查结果"; QVAL = lbres_desc; output;
  keep STUDYID RDOMAIN USUBJID IDVAR IDVARVAL
       QNAM QLABEL QVAL QORIG QEVAL;
run;

data SUPP[Domain_U];
  set SUPP[Domain_U];
  if QVAL~='';
run;

*======End of dataset processing program================================*;

*------------------------------------------------------------------------
Properties configuration program - Start
------------------------------------------------------------------------;
…
*------------------------------------------------------------------------
Properties configuration program - End
------------------------------------------------------------------------;
```

PROGRAMMING RULES

This part provides AI with learning materials for complex variables, facilitating their code writing process by following the summarized approaches based on statistical programmer' experience.

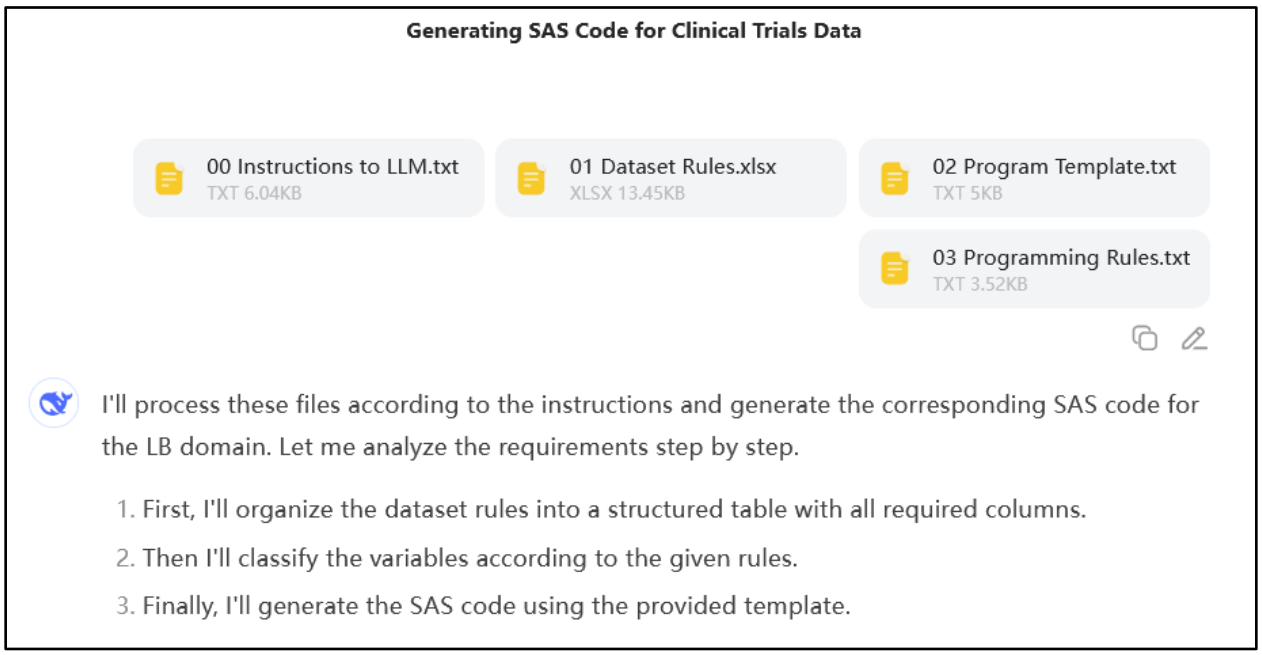
```
*************************************************************************;
**SEQ variable;
proc sort data=ADEG6 out=ADEG_output
sortseq=linguistic(numeric_collation=on);
  by USUBJID PARAMN EGDTC VISITNUM AVISITN; /* key vars */
run;
proc sort data=ADEG_output out=ADEG_check nouniquekey;
  by USUBJID PARAMN EGDTC VISITNUM AVISITN;
run;
data ADEG;
  set ADEG_output;
  by USUBJID PARAMN EGDTC VISITNUM AVISITN;
  retain ASEQ 0;
  if first.USUBJID then ASEQ=1;
  else ASEQ=ASEQ+1;
run;
*************************************************************************;


*************************************************************************;
**Request for the last non-empty test before administration;
**Where the macro parameter [Dataset] represents the input dataset,
[Cond] represents the filtering conditions to be executed when inputting the
dataset,
[Output] represents the name of the output dataset,
[OrderVarList] represents the variables for sorting (with multiple variables
separated by vertical bars),
[LastVar] indicates the data level of the --LOBXFL variable to be
calculated,
[KeyVarList] represents the sequence of key variables for ensuring
uniqueness,
[ResultVar] represents the variable for identifying non-empty test values,
```

```
[DateVar] represents the test date, a
[STDTC] represents the administration date;

%LOBXFL(Dataset = raw_pe
      ,Cond =
      ,Output = LOBXFL
      ,OrderVarList = SUBJID|PETESTCD|PEDTC
      ,LastVar = PETESTCD
      ,KeyVarList = SUBJID| PETESTCD | PEDTC
      ,ResultVar = PEORRES
      ,DateVar = PEDTC
      ,STDTC = RFXSTDTC
      );

**Generate the target variable;
data PE_LOBXFL;
  set LOBXFL;
  PELOBXFL = LOBXFL;
run;

**Merge into the main dataset;
proc sort data=PE_LOBXFL; by SUBJID PETESTCD PEDTC; run;
proc sort data=PE2; by SUBJID PETESTCD PEDTC; run;
data PE3;
    merge PE2(in=OutFL)
          PE_LOBXFL(keep=SUBJID PETESTCD PEDTC PELOBXFL) ;
    by SUBJID PETESTCD PEDTC;
    if OutFL;
run;
*******************************************************************************;
```

OPERATIONAL WORKFLOW

For users, the method described here only requires inputting preparation files, initial role settings, and project parameters into the AI, followed by manual confirmation of AI's writing progress and appropriate guidance. This method records and consolidates AI's learning materials and instructions as external files, enabling rapid completion of AI's initialization learning process and transitioning it into the target role's working state. Ideally, importing external files alone as a training method could yield a stable AI assistant capable of semi-automatically producing templated CDISC code. However, given current limited computing power and the unconfirmed stability of AI's usage, subsequent steps in this document will intermittently verify whether AI should continue execution and incorporate manual guidance when necessary.

STEP 1: IMPORT PREPARATION FILES AND SET INITIAL ROLES



STEP 2: INPUT PROJECT PARAMETERS AND CHECK VARIABLE CLASSIFICATION

Some information not provided in the preparation files requires manual specification, such as project name and programmer name. Theoretically, these details could also be included in the dataset rules file; here, they serve as an experiment with external input parameters.

After parameters are confirmed, the AI assistant will organize the "01 Dataset Rules.xlsx" file and classify variables according to the "00 Instructions to LLM.txt". At this point, DeepSeek has already determined the target domain as LB based on the data rules file.

The variable classification rules are as follows:

Type 1	Variables that can be directly extracted from the raw dataset or other SDTM datasets without requiring additional processing.
Type 2	Simple variables that can be directly generated based on mapping rules.
Type 3	Variables derived from existing variables within this dataset that do not require additional processing.
Type 4	Other variables requiring complex processing.
Type 5	Variables that need to be placed in the SUPP domain.

Classification results are as follows (partial examples):

#	Variable	Classification Type
1	STUDYID	Type 1 (Directly extracted from SDTM)
2	DOMAIN	Type 2 (Directly assigned)
3	USUBJID	Type 4 (Derived from SDTM.DM)
4	LBSEQ	Type 4 (Derived from sorting KeyVars)
5	LBTESTCD	Type 2 (Directly assigned)
6	LBTEST	Type 2 (Directly assigned)
7	LBCAT	Type 2 (Directly assigned)
8	LBORRES	Type 1 (Directly from raw.lburp.lbres_dec)
9	LBORRESU	Type 2 (Assigned as Null)

STEP 3: SIMPLE VARIABLES CODING

First, replace the placeholders in the template according to the replacement relationships identified in the previous step. Next, DeepSeek organizes variables related to Type 1 (variables directly captured from the original dataset or other SDTM datasets), Type 2 (directly assigned variables), and Type 3 (simple derived variables) and writes them into the corresponding sections as per the instructions.

The processing results for **Type 2** and **Type 3** are as follows:

Key features:
Type 2 Variables (Direct assignment):
DOMAIN, LBTESTCD, LBTEST, LBCAT, LBSPEC assigned fixed values
Null assignments for variables like LBORRESU, LBORNRLO
Type 3 Variables (Derived from existing):
LBSTRESC directly copies LBORRES
LBSTRESU copies LBORRESU (though currently null)
Added appropriate lengths for character variables
Maintained proper CDISC variable naming conventions

```
*************************************************************************
Section 2: Directly assigned variables
*************************************************************************;
data LB2;
  set LB1;
  length DOMAIN $2 LBTESTCD $8 LBTEST $40 LBCAT $20 LBSPEC $10 LBSTRESC $200
LBSTRESU $20;

  /* Type 2 Variables (Direct Assignment) */
  DOMAIN = "LB";
  LBTESTCD = "HCG";
  LBTEST = "Pregnancy";
  LBCAT = "Pregnancy test";
  LBSPEC = "Urine";
  LBORRESU = "";
```

```
  LBORNRLO = "";
  LBORNRHI = "";
  LBLLOD = "";
  LBNRIND = "";

  /* Type 3 Variables (Derived from existing variables) */
  LBSTRESC = LBORRES;
  LBSTRESU = LBORRESU;
run;
```

The processing results for **Type 1** are as follows:

Key changes made:
Changed dataset names from [Domain_U] to LB
Updated source dataset from raw.aaa to raw.lburp based on mapping rules
Added extraction of lbdat, lbreasnd, and lbreasna_dec for later calculations
Kept necessary variables for subsequent steps
Maintained merge with SDTM.DM for required variables

```
*************************************************************************
Section 1: DIRECTLY EXTRACTED VARIABLES
*************************************************************************;
DATA LB1_1;
  SET RAW.LBURP(KEEP=SUBJECT_ID LBRES_DEC LBPERF LBDAT LBREASND LBREASNA_DEC
                     LBREASNO);
  SUBJID = SUBJECT_ID;
  LBORRES = LBRES_DEC;
  LBDTC = LBDAT;
  LBREASO = LBREASNO;**SUPP Variables;
RUN;

PROC SORT DATA=LB1_1; BY SUBJID; RUN;

DATA LB1_2;
  MERGE LB1_1(IN=OUTFL)
        SDTM.DM(KEEP=STUDYID USUBJID SUBJID RFSTDTC);
  BY SUBJID;
  IF OUTFL=1;
RUN;

DATA LB1;
  SET LB1_2;
  /* RETAIN VARIABLES FOR LATER CALCULATIONS */
  KEEP STUDYID USUBJID SUBJID LBORRES LBDTC RFSTDTC LBPERF LBREASND
LBREASNA_DEC;
RUN;
```

STEP 4: COMPLEX VARIABLE CALCULATION SECTIONS

For variables in Type 4 that require complex calculations, DEEPSEEK first refers to the variable creation logic demonstrated in the program writing rules. When no reference calculation method is provided, it flexibly writes code by identifying mapping rules in the dataset rules.

The code for variables with reference calculation methods is as follows:

```
proc sort data=LB3 out=LB sortseq=linguistic(numeric_collation=on);
  by USUBJID LBDTC LBTESTCD;
run;

proc sort data=LB out=LB_Check nouniquekey;
  by USUBJID LBDTC LBTESTCD;
run;

data LB;
  set LB;
  by USUBJID;
  retain LBSEQ 0;
  if first.USUBJID then LBSEQ=1;
  else LBSEQ=LBSEQ+1;
run;
```

The code for variables without references and of lower difficulty is as follows:

```
*-- LBSTAT and LBREASND --*;
data LB3_LBSTAT_LBREASND;
  set LB2(keep=USUBJID lbperf lbreasnd lbreasna_dec);
  length LBSTAT $8 LBREASND $200;
  if lbperf in ("No", "Not Applicable") then LBSTAT = "Not Done";
  if lbperf = "No" then LBREASND = lbreasnd;
  else if lbperf = "Not Applicable" then LBREASND = lbreasna_dec;
run;
```

STEP 5: OUTPUT THE COMPLETE CODE

```
*******************************************************************************

************************ Prepared by KTSTAT ***************************;
*-----------------------------------------------------------------------
Environment Configuration - Start
-----------------------------------------------------------------------;

%global batfullpath ……;
%macro currentroot();
……
%mend currentroot;

%currentroot();
%include "&batcurrpath.\init.sas";
*-----------------------------------------------------------------------
Environment Configuration - End
-----------------------------------------------------------------------;

*======The following is the program for dataset processing==============*;
*******************************************************************************
```

```
Section 1: Directly extracted variables
*********************************************************************;
data LB1_1;
  set RAW.LBURP(KEEP=SUBJECT_ID LBRES_DEC LBPERF LBDAT LBREASND
                     LBREASNA_DEC LBREASNO);
  SUBJID = SUBJECT_ID;
  LBORRES = LBRES_DEC;
  LBDTC = LBDAT;
  LBREASO = LBREASNO;**SUPP Variables;
run;

proc sort data=LB1_1; by SUBJID; run;

data LB1;
  merge LB1_1(in=OutFL)
        SDTM.DM(keep=STUDYID USUBJID SUBJID RFSTDTC);
  by SUBJID;
  if OutFL=1;
run;

*********************************************************************
Section 2: Directly assigned variables
*********************************************************************;
data LB2;
  set LB1;
  length DOMAIN $2 LBTESTCD $8 LBTEST $40 LBCAT $20 LBSPEC $10 LBSTRESC
$200 LBSTRESU $20;

  /* Type 2 Variables (Direct Assignment) */
  DOMAIN = "LB";
  LBTESTCD = "HCG";
  LBTEST = "Pregnancy";
  LBCAT = "Pregnancy test";
  LBSPEC = "Urine";
  LBORRESU = "";
  LBORNRLO = "";
  LBORNRHI = "";
  LBLLOD = "";
  LBNRIND = "";

  /* Type 4 Variables (Derived from existing variables) */
  LBSTRESC = LBORRES;
  LBSTRESU = LBORRESU;
run;

*********************************************************************
Section 3: Variables requiring complex calculations
*********************************************************************;
*-- LBSTAT and LBREASND --*;
data LB3_LBSTAT_LBREASND;
  set LB2(keep=USUBJID lbperf lbreasnd lbreasna_dec);
  length LBSTAT $8 LBREASND $200;

  if lbperf in ("No", "Not Applicable") then LBSTAT = "Not Done";
  if lbperf = "No" then LBREASND = lbreasnd;
  else if lbperf = "Not Applicable" then LBREASND = lbreasna_dec;
run;
```

```
*-- LBDY (Study Day Calculation) --*;
data LB3_LBDY;
  set LB2(keep=USUBJID LBDTC RFSTDTC);
  length LBDY 8;

  if not missing(LBDTC) and not missing(RFSTDTC) then do;
    LBDTC_N = input(scan(LBDTC,1,'T'),yymmdd10.);
    RFSTDTC_N = input(scan(RFSTDTC,1,'T'),yymmdd10.);

    if LBDTC_N ne . and RFSTDTC_N ne . then do;
      LBDY = LBDTC_N - RFSTDTC_N;
      if LBDY >= 0 then LBDY = LBDY + 1;
    end;
  end;
run;

**LBLOBXFL - Using macro approach from programming rules;
data raw_lb1;
  set LB2(keep=SUBJID LBTEST LBDTC LBORRES RFXSTDTC);
  where not missing(LBORRES);
run;

%LOBXFL(Dataset = raw_lb1
    ,Cond =
    ,Output = LOBXFL
    ,OrderVarList = SUBJID|LBTEST|LBDTC
    ,LastVar = LBTEST
    ,KeyVarList = SUBJID|LBTEST|LBDTC
    ,ResultVar = LBORRES
    ,DateVar = LBDTC
    ,STDTC = RFXSTDTC
    );

data LB_LOBXFL;
  set LOBXFL;
  LBLOBXFL = LOBXFL;
run;

proc sort data= LB3_LBSTAT_LBREASND; by SUBJID LBTEST LBDTC; run;
proc sort data=LB_LOBXFL; by SUBJID LBTEST LBDTC; run;
proc sort data=LB2; by SUBJID LBTEST LBDTC; run;

**Merge all calculated variables;
data LB3;
  merge LB2(in=OutFL)
        LB3_LBSTAT_LBREASND (keep=SUBJID LBTEST LBDTC LBSTAT LBREASND)
        LB3_LBDY(keep=USUBJID LBTEST LBDTC LBDY)
        LB_LOBXFL(keep=SUBJID LBTEST LBDTC LBLOBXFL);
  by SUBJID LBTEST LBDTC;
  if OutFL=1;
run;

*************************************************************************
Section 4: Sort and finalize dataset
*************************************************************************;
proc sort data=LB3 out=LB sortseq=linguistic(numeric_collation=on);
```

```
  by STUDYID USUBJID LBCAT LBTESTCD LBDTC;
run;

data LB;
  set LB;
  by STUDYID USUBJID LBCAT LBTESTCD LBDTC;
  retain LBSEQ 0;
  if first.USUBJID then LBSEQ=1;
  else LBSEQ=LBSEQ+1;
run;


*******************************************************************************
Section 5: Generating the SUPPLB dataset
*******************************************************************************;
data SUPPLB;
  set LB;
  length RDOMAIN IDVAR IDVARVAL QNAM QLABEL QVAL QORIG QEVAL $900;

  RDOMAIN = "LB"; IDVAR = "LBSEQ"; IDVARVAL = put(LBSEQ, best.);
  QORIG = "CRF"; QEVAL = "";

  *LBREASO;
   QNAM = "LBREASO"; QLABEL = "Other reasons not applicable";
   QVAL = LBREASO; output;

  keep STUDYID RDOMAIN USUBJID IDVAR IDVARVAL QNAM
       QLABEL QVAL QORIG QEVAL;
run;

data SUPPLB;
  set SUPPLB;
  if QVAL~='';
run;

*======End of dataset processing program================================*;

*-------------------------------------------------------------------------------
Properties configuration program - Start
-------------------------------------------------------------------------------;
…
*-------------------------------------------------------------------------------
Properties configuration program - End
-------------------------------------------------------------------------------;
```

EVALUATION OF IMPLEMENTATION RESULTS

ADVANTAGES

For simpler and clearer requirements, DeepSeek performs better. For instance, it excels in tasks such as variable classification, template section division, placeholder substitution, and direct assignment or extraction of simple variables. When dealing with variable creation demands that have clear mapping relationships and relatively straightforward logic, DeepSeek's processing is also quite stable. Judgments and transcriptions of statements like IF-ELSE can be used directly.

Additionally, while writing code, it can automatically adjust the variable range covered by the keep statement according to the previous and subsequent requirements, and give the length in advance when assigning variables, which enhances program efficiency and stability. Furthermore, after providing programming rules, DeepSeek demonstrates notable capabilities in learning, dissecting, and applying code. Judging by the results in the examples, its application outcomes are correct and standardized.

DISADVANTAGES

As initially anticipated, DeepSeek performs poorly in coding for more complex variables with generalized calculation rules, making it nearly unusable in practical work. For example, in the creation of the LBLOBXFL variable, even though we can clearly discern logic like "first check non-empty pre-dose test values, then take the last result as the baseline" in its code, it fails to recognize the logical flaws of this method at the structural level of the actual dataset. Therefore, for these more complex variables, given the current execution performance of AI, we must either articulate the calculation rules more clearly, meticulously, and accurately to ensure step-by-step execution for correctness or directly provide the complete calculation rules in the programming rule file for its learning reference.

Moreover, solely in terms of workflow, since the AI cannot access external project information in real-time, preparatory tasks such as file and data preparation remain essential. Additionally, without importing the original dataset, the AI cannot identify information like Key Vars in the dataset, requiring manual judgment.

FUTURE DIRECTIONS AND CHALLENGES

Based on the findings of this study, it appears that the current level of AI may not yet replace a senior or intermediate statistical programmer, but it already possesses the capability to serve as an assistant to statistical programmers. What we need to do is issue more modular and procedural precise instructions, communicate our requirements to AI, and provide it with appropriate learning materials.

Since the project experience and industry understanding accumulated by senior statistical programmers are currently areas where AI is lacking—especially in handling calculation rules and code for complex variables—future efforts could focus on continuous improvement through programming rule training and learning. This aims to train the AI to become a personalized assistant and code template manager for each statistical programmer, saving us from some simple repetitive tasks.

CONCLUSION

This paper explores the recognition and conversion of Spec documents into SAS programs in the clinical trial domain using the AI large language model (LLM). By refining and breaking down LLM instructions, supplementing program target dataset rules, programming rules, and frameworks, it investigates the simple repetitive tasks that AI can accomplish under current limited computational resources, while also highlighting its potential for continuous learning and optimization. Future research may further deepen AI's learning of rules for handling complex variables and explore more natural language format specifications tailored to AI's reading instructions, leveraging AI to accelerate the automation process.

We hope the ideas presented here will inspire future applications of the AI large language model (LLM) in assisting statistical programmers. Happy coding!

REFERENCES

[1]Yu Peng; Michael Wang. 2024. “Trial Agent: An Innovative and Intelligent LLM based Agent for Autonomous Clinical Trial Data Analysis.“ Proceedings of the PharmaSUG China 2024 Conference, Beijing, CN: PharmaSUG China.

[2]Guangyong Li; Junkai Zhao. 2024. “Enhancing SAS Programming Efficiency with Large Language Models (LLMs).” Proceedings of the PharmaSUG China 2024 Conference, Beijing, CN: PharmaSUG China.

[3]Hao (Harry) Chen. 2023. “AI HI: Accelerating Biometrics Activities with the Power of Large Language Models (LLMs).” Proceedings of the PharmaSUG China 2023 Conference, Nanjing, CN: PharmaSUG China.

ACKNOWLEDGMENTS

We want to thank Nan Li, Dawei Jiao, Yingwei Si and Zhiwei Jiang for the support on this work.

CONTACT INFORMATION

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

Jinting Luo
Beijing Key Tech Statistical Consulting Co., Ltd
jin.ting.luo@ktstat.com