PharmaSUG China 2024 - Paper AA-10057

An End-to-end Digital Platform for TFL Automation -- From Metadata to TFL Outputs

Yanan Sui, Jiapeng He, Mingliang Fan, Xuejie Zhou, Lin Jiang, Tingting Zeng, and Jinling Li, BeiGene

ABSTRACT

As a statistical programmer, generating tables, figures, and listings (TFL) makes up the most of our daily work, and traditional programming becomes heavier and more time-consuming when TFL scope rapidly grows in clinical trials. To ensure a high-quality and timely TFL delivery, a digital platform built in BeiGene to provide the end-to-end automation solution including TFL metadata, programs, and outputs automation.

Taking table automation as an example, to automatically stabilize the metadata based on various study mock shells, tables in mock shell will be parsed into title/footnote, column header, block and value level and digitalized into a centralized database. With AI techniques applied (e.g., text fuzzy match and sematic analysis), SDTM/ADaM annotations will be attached into corresponding levels to complete the metadata. Once metadata is initialized, a real-time checking system and visualized annotated shell will be provided for users to easily review and finalize the metadata. At last, dynamic, well-structured, and submission-ready SAS programs will be auto-generated based on the metadata. This approach saves most of manual programming work by 80-90%, and coding free can be expected as the growing and recyclable metadata library being referred in future studies.

BACKGROUND

The process and guidance of statistical programming team deliverables have been setup and well followed in BeiGene, along with the implementation of standard ADaM specifications, TFL shells and macros have already saved some time on programs generation. Besides, we have a mature, perfect Client/Server architecture platform where programmers can perform regular programming work. What's more, the AI technology is rapidly growing and with more widespread application in the industry, it came to us that it was necessary to build an intelligent tool -- Clinical Data Analysis and Reporting System (CDARS), which aims to provide an end-to-end TFL automation solution and help programmers reduce repetitive coding work, raise productivity and ensure high quality of TFL deliverables to get well prepared for upcoming wave of changes, which include both tremendous studies and creative technology.

Since CDARS piloted in some deliverables and reduced manual coding work significantly, this paper introduces how CDARS is built, how you may use, furthermore, to discuss current limitations, challenges, and future expectations.

HOW CDARS IS BUILT AND WHAT CAN CDARS DO

As mentioned above, the CDARS was developed with a back-end using the Flask framework and a front-end built with Vue.js. The back-end performs complex algorithm to conduct business logic, including parsing docx TFL shell as digitalized metadata, performing metadata checking, etc. The front-end provides user interface to allow users' interactions, including uploading TFL shells, reviewing and editing TFL metadata, etc.

The Figure 1 demonstrates the overall process of CDARS, and followings introduce key features:

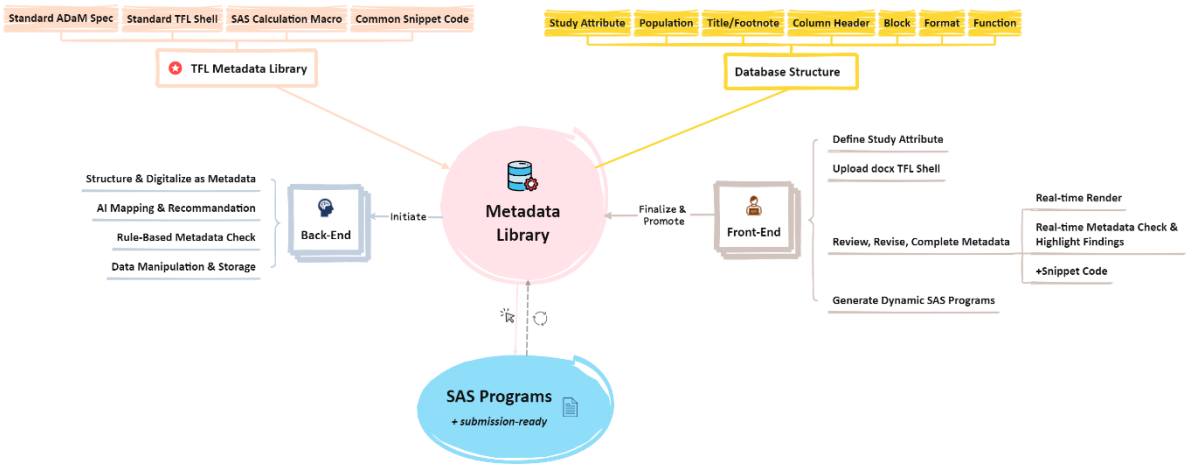

Figure 1. CDARS overall process

1. PARSE STANDARD TFL SHELL & INITIATE METADATA LIBRARY

The standard TFL metadata library is made up by digitalizing TFL shells, attaching appropriate value level and reporting related annotations, which are the foundation of the CDARS. The key concept here is to de-structure TFL shells as minimum components, which are easy for subsequent reference and assemble.

The TFL metadata library is initialized with the company standard TFL shell and ADaM specifications, take the most familiar demographic table to programmer as an example (Figure 2), the table is de-structured as title/footnote, column header and block three layers, each layer consists of information parsed directly from shell and further necessary value level annotations that are attached according to standard ADaM specifications, details stated below:

- Title/Footnote:
  - Information directly from shell, like title, footnotes, population, etc.
  - Value level annotations, like criteria applied for population, parameters regarding RTF formatting (landscape or portrait) etc.
- Column Header:
  - Information directly from shell, like number and label of column headers.
  - Value level annotations, like the ADaM dataset and criteria applied to calculate big N of each column header, parameters regarding RTF formatting (i.e., alignment, width of each column), etc.
- Block:
  - Figure 3 shows the block metadata structure. Information captured based on the shell contexts, including block label, analysis type, indent, etc. Among these parameters, the analysis type is critical and key to the analysis, which is pre-defined and summarized based on standard table shells, and a set of SAS macros are developed to meet regular analysis needs. As it shows in the figure, relevant macro will be called after identifying the specific analysis type of the block.
  - In addition to the analysis type, figure 3 also lists some value level parameters varying according to the analysis type. In general, they are summarized as 4 categories: denominator, analysis, statistics related and other specific parameters, which are used as either macro input or dataset pre or post processing. For example, for descriptive

analysis (i.e., SUM), ADaM, variable for analysis, and decimal for statistics shall be clarified.

In addition to the title/footnote, column header and block metadata databases, CDARS also maintains some other databases regarding study attribute, population, format, and common SAS function. The study attributes database stores key study features, as well as population, format and function databases hold metadata that are either referred from reference or customized per study needs. These databases are designed for centralized management and further easily reference.

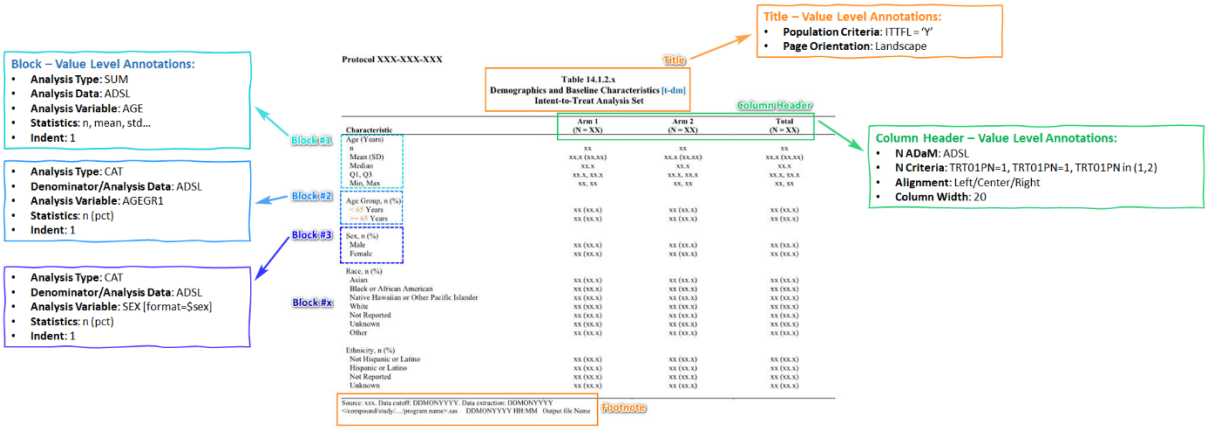

Figure 2. de-structured table and value level annotations

Block	Block Label	Analysis Type	Denominator			Analysis			Statistics				Others		Macro
			Data	Filter	Variable	Data	Filter	Variable	Type[a]	Statistics[b]	Label	Decimal	Sorting	RTF Indent	
Block#1	Block Label#1	SUM	X			X	X	X		X	X	X		X	%m_tfl_calculate_sum
Block#2	Block Label#2	CRIT	X	X		X	X			X				X	%m_tfl_calculate_crit
Block#3	Block Label#3	CAT	X	X		X	X	X		X			X	X	%m_tfl_calculate_cat
Block#4	Block Label#4	EVE	X	X		X	X	X		X			X	X	%m_tfl_calculate_eve
Block#5	Block Label#5	KM				X	X	X	X	X	X	X		X	%m_tfl_calculate_km
Block#6	Block Label#6	COX				X	X	X	X	X		X		X	%m_tfl_calculate_cox
Block#7	Block Label#7	LOGRANK				X	X	X	X	X				X	%m_tfl_calculate_logrank
…	…	…													…

X stands for options can be used for customizing analysis.

[a] customize parameters such as alpha, timelist, strata…

[b] including n, std, mean, q1, q3, min, max, geomean, geocv, hr, ci, pvalue…

Figure 3. block level metadata

2. APPLY AI ALGORITHM TO CONVERT A NEW TFL SHELL AS DIGITALIZED METADATA

So, what if a new study level TFL shell comes to the CDARS? First, CDARS applies algorithm to de-structure and digitalize shell contexts as metadata. Furthermore, CDARS compares study level title and block metadata with the reference metadata library, which one or more reference chose to refer. For cases that are exact match, value level annotations from metadata library will be applied to the study; for cases that are not exact match, CDARS conducts fuzzy match by using python fuzzywuzzy package and based on Levenshtein Distance algorithm to get similarity scores, complete value level annotations and provide recommendations. Below are detailed and optimized steps regarding how to perform titles and blocks fuzzy match.

1) Pre-processing of contexts to improve fuzzy match performance, for example removing punctuations words (e.g., ‘,’, ‘.’, ‘;’, ‘(’, ‘)’, etc.), stop words (e.g., ‘the’, ‘a’, ‘shoud’, ‘by’, etc.) and duplicated words.
2) Re-assign subgroups for each TFL by identifying key words from the section and group as Baseline, AE, Efficacy, Lab, etc.

3) Within the same subgroup defined in the above step, conducts title fuzzy match with reference and get similarity scores. All records with scores above the threshold are kept, which is set at 60.
4) Conducts blocks fuzzy match according to the title fuzzy match results and finally keep records with similarity scores above 60:
    a) For blocks having titles with similarity score >= 60, perform algorithm within blocks from matched titles.
    b) For blocks having titles with similarity score < 60, perform algorithm within blocks from same subgroup re-assigned in step 2).

In general, utilizing both exact match and fuzzy match algorithm improves the match rate to 80% - 90% comparing to the about 40% match rate by exact match algorithm only.

3. HOW CDARS GENERATES SAS PROGRAMS

As above introduction, CDARS digitalizes TFL shell as metadata, which also facilitates to combine metadata as dynamic, well-structured, and elegant SAS programs. Which means once metadata is complete, plenty of readable and executable SAS programs can be generated by one-button click. CDARS can generate dynamic & submission-ready TFL programs in SAS, extract/insert user customized snippet codes, check and archive previous programs.
It is worthy to mention that SAS programs are designed in programmer style for easy review and debug, Figure 4 demonstrates programs created for demographic table.

- Proper structure, consisting of program header, format, data handling, header/big N/block calculation, reporting phase, clear comments and placeholder for user snippet codes.
- Wrap as macro for cases that multiple outputs can be generated by one program to make code tidy and clear. This approach also considers distinguishing centralized processing from macro parameters.
- Provides flexibility to add snippet codes. Snippet codes that are defined at the specific location could be identified, captured, and inserted in the afterwards SAS programs generation.
- Make your choice on whether to obtain submission-ready programs, create PDF outputs, archive or replace programs.

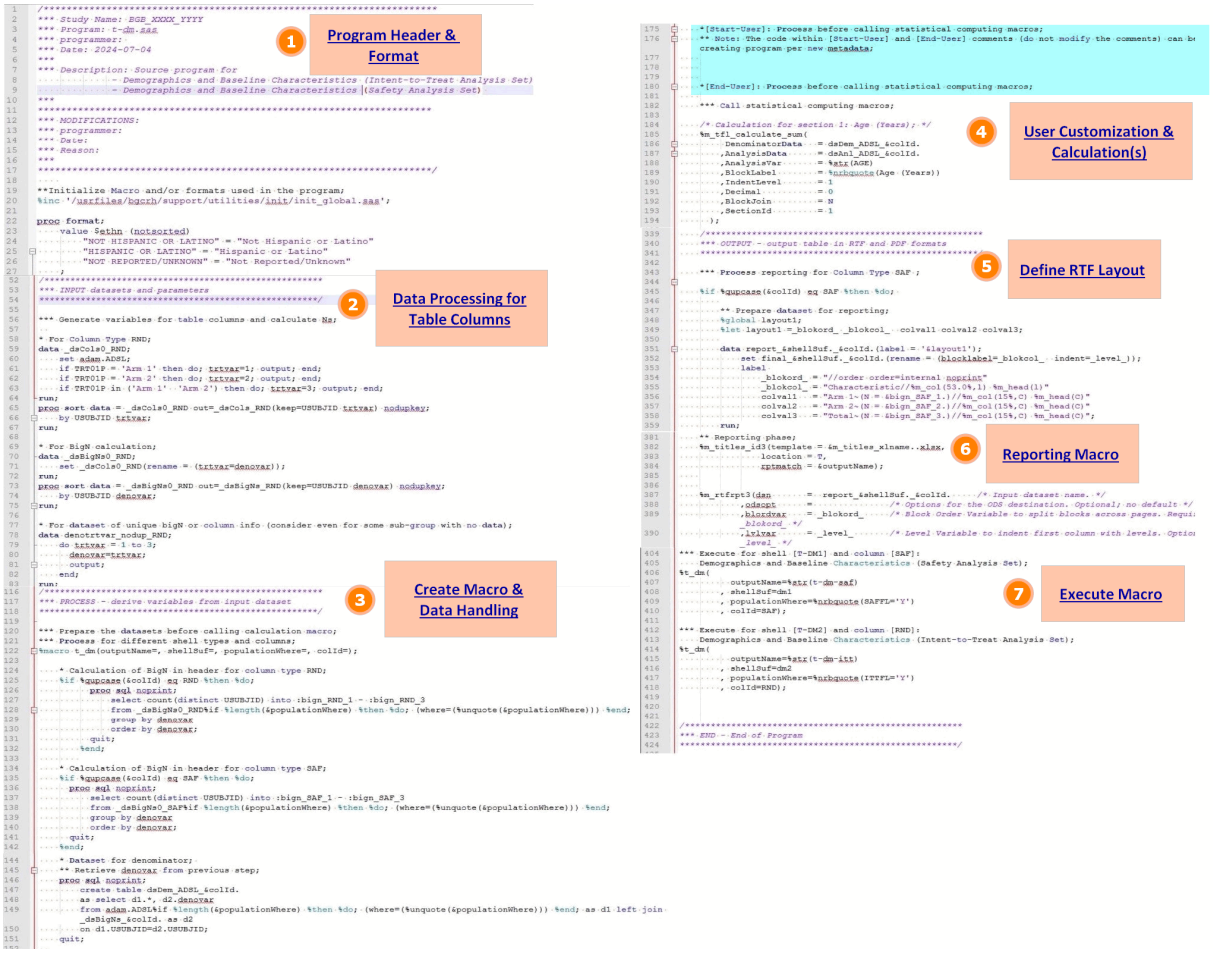

Figure 4. dynamic table program

HOW YOU USE CDARS

4. WHAT YOU NEED TO DO

Basically, you shall input some specific study attributes, upload TFL shell to create metadata and ensure metadata accuracy.

- Study attributes shall be input manually for now because they are not obtained either from the metadata library or AI recommendation. Relevant study attributes include monotherapy or combination therapy, oral or/and infusion study drug(s), etc. For example, table shells and ADaM variables are generally different between monotherapy and combination therapy. Clarifying it helps conduct metadata matching with corresponding TFL metadata library and fetch appropriate ADaM variable annotations.
- You shall upload TFL shell with docx format to create digitalized metadata and get initialized value level annotations.
- You should ensure metadata are appropriate and with correct annotations, especially for annotations which are mapped through AI algorithm. Reviewing, revising, and completing metadata are critical to generate correct SAS programs.

5. HOW CDARS FACILITATES PROCESS AND REDUCES MANUAL WORK

As mentioned above, metadata is the foundation of CDARS to generate SAS programs, but the purpose of CDARS is not to transfer programmers' work from coding to creating metadata. What CDARS does want to achieve is making metadata automated and accurate, reducing manual work, facilitating metadata

generation, matching, and finally programs. Apart from applying AI algorithm to improve metadata and annotations mapping, CDARS does develop a user-friendly front-end interface to provide convenient operations and important hints.

- **Real-time render**
  CDARS provides render for both column header and table. The main idea of render is what you see is what you will get, as such you can use it with less effort to figure out metadata is correct or not. For example, when you are editing the block metadata, a real-time table render displays at the right panel (Figure 5), which highlights current block under editing and demonstrates how changes you've made in metadata affect the table.

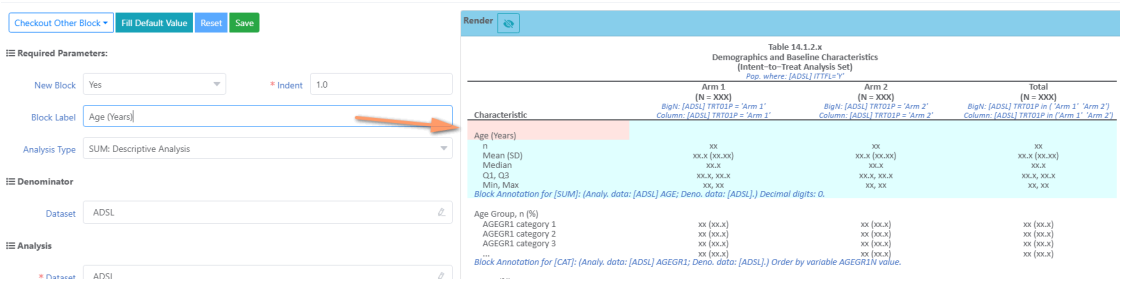

Figure 5. real-time render

- **Real-time check**
  CDARS has a rule-based cross-checking system to make sure accuracy of title/footnote, column header and block metadata. For instance, the block analysis type is always checked if it is filled in properly because of its importance for analysis. A real-time checking is also applied whenever metadata finishes editing. If the checking system finds any issues, then CDARS will highlight background of relevant cells as red immediately to alert. You may get detailed specifications by double clicking red cells (Figure 6).

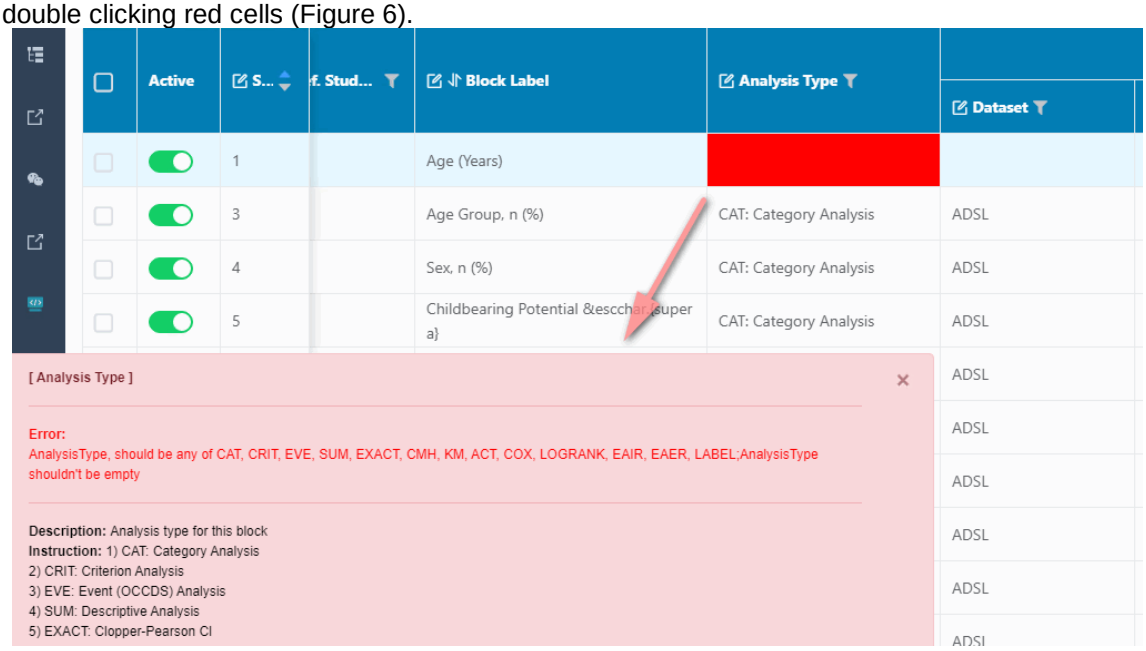

Figure 6. real-time check

- **Friendly User Interface for metadata update**
  Besides getting instructions by double clicking, CDARS provides more functions to reduce manual work and enhance user experiences. Such as considering the block metadata may

contain several analysis types and following different parameters maybe required, CDARS is designed to display the default value for each field of value level annotation. You could click to fill all empty fields as default values instead of input one by one. In addition, each field is pre-loaded with some common use examples, which are provided to help better figure out how to fill in it. You may quick select from the drop-down list and modify based on the selected example. All value level instructions, default values and examples vary according to the block analysis type. Figure 7 demonstrates such features.

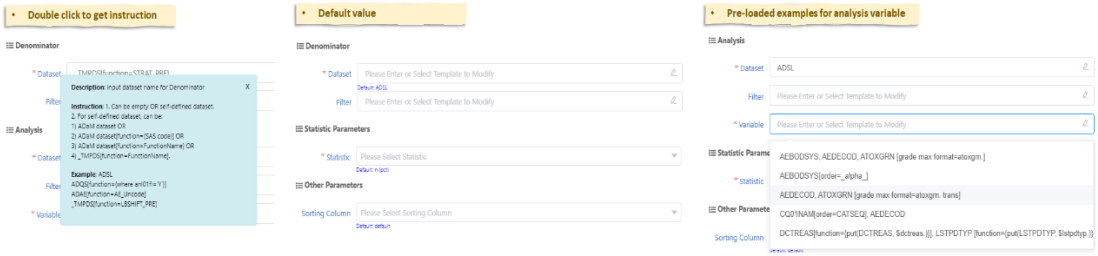

Figure 7. user interface for instruction, default value and examples

- **Recyclable metadata library**
  CDARS not merely keeps all your inputs and adjustments for metadata in the system, but also has another key feature -- recycling metadata, which means that metadata could be promoted to the digitalized metadata library and involved in future studies' reference once they are fully validated. With the growth of metadata library, less manual work can be expected as more reference options are provided in CDARS system.

LIMITATION & FUTURE EXPECTATION

- **Two-way sync between TFL metadata and program:** Current generating SAS programs are based on metadata, although CDARS provides flexibilities to customize programs, what if users just open and modify SAS programs? The challenge is how to identify and capture changes, sync back and update the metadata database.
- **Enhanced AI application:** With the development and application of AI technology, CDARS also plans to explore and utilize more AI models to optimize back-end algorithm and enhance the capability, like large language model. Taking advantages of AI to allow programmers focus on something more valuable.
- **Shell-TFL automation integration:** Parsing and de-structuring TFL shell are the prerequisites of using CDARS, which sort of relies on the structured TFL shell and may encounter risks to parse 'non-standard' shells. Therefore, TFL shell creation is also a part of CDARS future functionalities. This feature is users could design a shell within CDARS and meanwhile all relevant metadata are initialized. Advantages include omitting the TFL shell uploading step, thereby accelerating the subsequent digitalizing shell and mapping annotations steps.
- **SDTM/ADaM-TFL metadata flow and crosscheck:** Further to enrich the database, not only referring to the TFL metadata library, but also the upstream metadata, i.e., study level SDTM/ADaM variables information. Inappropriate metadata annotations which are either from reference or AI recommendation can be replaced with the specific and correct study SDTM/ADaM variables by utilizing study level specifications. Furthermore, study attributes can also be captured instead of manual input.

CONCLUSION

This paper provides the end-to-end TFL automation solution implemented in BeiGene, introduces concept and product, discusses advantages and limitations of CDARS.

Current CDARS is founded with basic functions and requires some necessary user interaction. With further exploration, development of AI methods and utilizing novel technology, CDARS will continuously iterate and optimize to become more user-friendly, powerful, automated, and intelligent.

ACKNOWLEDGMENTS

I want to emphasize that it would not happen without all CDARS developers' contribution, cooperation, as well as all users' positive and valuable feedback. Especially, I would like to express my sincere gratitude to our CTO, Jinling Li and Tingting Zeng, both CDARS and ourselves grew and improved rapidly under their oversee, guidance and solid technique support. Allow me to introduce our creative developers and perfect teamwork more: Tingting designed and generated SAS programs by JavaScript. Jiapeng He, the CDARS great back-end engineer who digitalized TFL shells as metadata. Mingliang Fan, the owner of plenty of SAS calculation macros and standard TFL metadata. Xuejie Zhou leaded the AI exploration on TFL metadata automation to continuously optimize and enhance the algorithm. Lin Jiang performed complex metadata checking algorithm to ensure metadata quality and make findings traceable. I took this opportunity to be a front-end developer for User Interface design and development.

CONTACT INFORMATION

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

Yanan Sui
BeiGene
yanan.sui@beigene.com