PharmaSUG 2025 - Paper DS- 338

Efficient CDISC Controlled Terminology Mapping: An R-Based Automation Solution

Yunsheng Wang and Chao Wang

ABSTRACT

This paper presents a method for automating the generation of SDTM Controlled Terminology (CT) terms using an R Shiny application. The system connects customized CT library to map EDC raw terms to their corresponding SDTM CT terms. Using similarity-based matching algorithms, the system ensures that all EDC raw terms are auto mapped to corresponding SDTM CT terms. When exact matches are not found, the web-based interface allows users to review and modify the suggested CT terms to align with sponsor requirements. The tool supports the export of terms such as datasets, Excel files, or program code. The paper demonstrates methodology, system architecture, and practical applications, highlighting the tool's potential to improve clinical data management, data visualization, and data standardization in clinical trials.

INTRODUCTION

The R Shiny Interface for CDISC CT Mapping Automation is a powerful tool designed to simplify and streamline the process of mapping raw terms from EDC systems to their corresponding CDISC CT terms. This application enhances the efficiency and accuracy of mapping by automating the process and providing users with an interactive interface to easily navigate through the necessary steps.

APPLICATION DETAILS

As shown in Figure 1, the left-side panel of the application allows users to browse through their raw data and easily select specific variable they want to work with. Once a raw variable is selected, users can input the corresponding **target SDTM variable** (SDTMVAR) defined in SDTM IG. The application then gets the relevant **CODELIST** value from the study specified SDTM IG Metadata based on the selected **SDTMVAR**.

Once the **CODELIST** is identified, the application retrieves the corresponding **CDISC Submission Value** (the final format required for SDTM submission) from the selected CT version based on the **CODELIST** value.

For example, if the user selects the raw variable CMDOSFRQ from raw cm dataset, the **SDTMVAR** will be defined as CMDOSFRQ. In the SDTM IG, the CMDOSFRQ variable is associated with a **CODELIST** value of FREQ. This **CODELIST** value, FREQ, is then used to

filter out all **CDISC Submission Value** from CDISC CT. Following this, the automation process will perform a series of matching steps on the raw terms (RawString) to automatically identify and map them to the correct CDISC Submission Value.

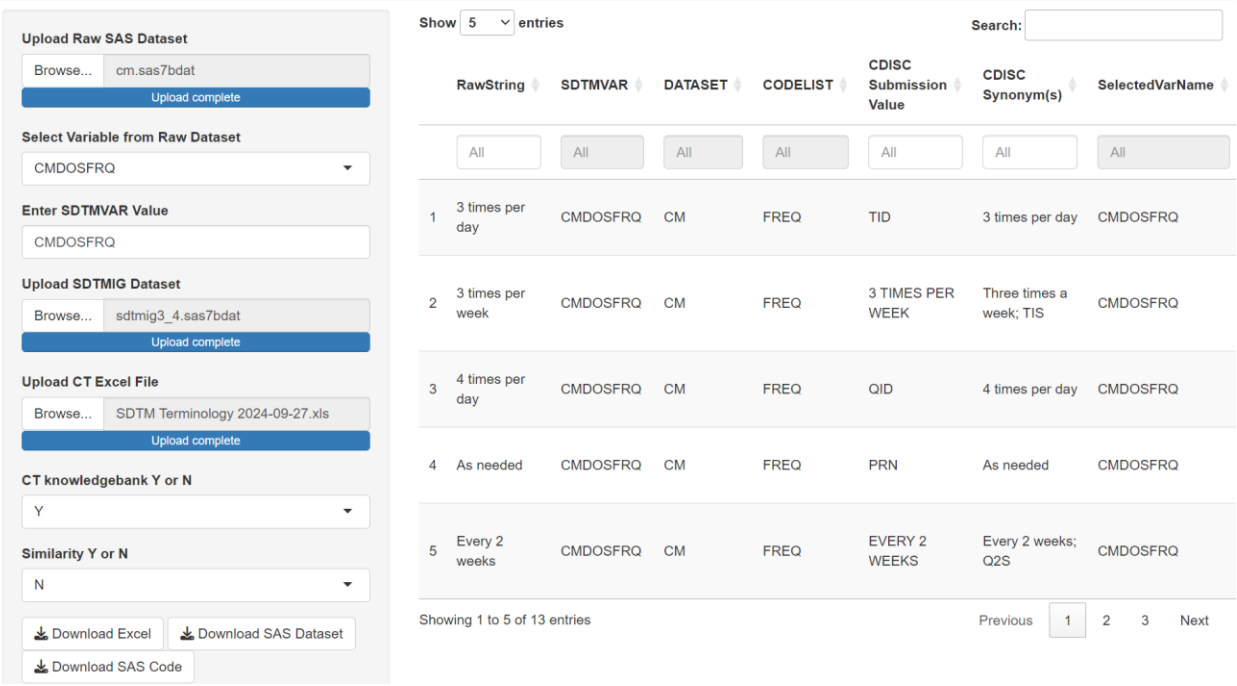


Figure 1. R Shiny Interface for CDISC CT Mapping Automation

The final mapping results are displayed in the right-side main panel, where users can review and make any necessary updates. To improve the accuracy of the matching process, two options are available during automation:

1. **CT Knowledge Bank**: This option is used when sponsor-specific mappings or mappings collected from previously submitted studies are available.
2. **Similarity Process**: This option automatically generates mappings using Levenshtein Distance similarity methods to compare raw terms with CDISC Submission Values, helping to identify additional matches.

Once all inputs are completed, and the mapping has been reviewed and updated, users have the option to download the mapping files. These files can be downloaded in the following formats:

1. **SAS Code**: Contains the SAS code for further programming and data manipulation (Figure 2 Right).
2. **Excel File**: Includes raw terms and the mapped CDISC Submission Value, useful for future reference and reporting (Figure 2 Left).

3. **SAS Dataset**: A ready-to-use dataset for creating formats in SAS.

This flexibility ensures users can export the results in a way that best suits their needs for SDTM programming and documentation.

RawString	SDTMVAR	DATASET	CODELIST	CDISC Submission Value
cap = Capsule	CMDOSU	CM	UNIT	CAPSULE
gtt = Drop	CMDOSU	CM	UNIT	DROP
g = Gram	CMDOSU	CM	UNIT	g
mcg = Microgram	CMDOSU	CM	UNIT	ug
mg = Milligram	CMDOSU	CM	UNIT	mg
mL = Milliliter	CMDOSU	CM	UNIT	mL
Other	CMDOSU	CM	UNIT	OTHER
Puff	CMDOSU	CM	UNIT	PUFF
Spray	CMDOSU	CM	UNIT	SPRAY
tab = Tablet	CMDOSU	CM	UNIT	TABLET
U = Unit	CMDOSU	CM	UNIT	U
tsp = Teaspoon	CMDOSU	CM	UNIT	tsp

```
if CMDOSU='cap = Capsule' then CMDOSU='CAPSULE';
if CMDOSU='gtt = Drop' then CMDOSU='DROP';
if CMDOSU='g = Gram' then CMDOSU='g';
if CMDOSU='mcg = Microgram' then CMDOSU='ug';
if CMDOSU='mg = Milligram' then CMDOSU='mg';
if CMDOSU='mL = Milliliter' then CMDOSU='mL';
if CMDOSU='Other' then CMDOSU='OTHER';
if CMDOSU='Puff' then CMDOSU='PUFF';
if CMDOSU='Spray' then CMDOSU='SPRAY';
if CMDOSU='tab = Tablet' then CMDOSU='TABLET';
if CMDOSU='U = Unit' then CMDOSU='U';
if CMDOSU='tsp = Teaspoon' then CMDOSU='tsp';
```

Figure 2. Auto-generated Excel File Output (Left) and SAS Code Output (Right)

PREPARATION AND DEVELOPMENT

To increase the number of matches, several key steps were implemented during the development process:

1. **Process Raw Data**
   To enhance accuracy, the raw terms (**RawString**) undergo a preprocessing step where they are split into a new intermediate string**, RawString_cleaned**, as illustrated in Figure 3. When **RawString** contains characters such as '=', '/', or ';', the string is split at these characters, and the resulting parts are separated into individual rows (Figure 3, right). For example, "gtt=Drop" would be split into two rows: "gtt" and "Drop", which are then stored in **RawString_cleaned**. This step is important because sometimes the right side, like "gtt", might not match any terms in CDISC CT, but the left side, "Drop", could have a match. By splitting the string, both parts are stored in **RawString_cleaned**, increasing the chances of finding a match for either value. If other unexpected characters are encountered, an additional process can be applied using an R block, as demonstrated below:

```r
#data clean process
#Split the 'RawString' column by '/', ';', or '=' and output to different rows
final1 <- final1 %>%
  mutate(values = RawString) %>%  # Create a 'values' column based on 'RawString'
  separate_rows(values, sep = "[/;=]") %>%  # Split values into separate rows
  mutate(RawString_cleaned = tolower(gsub("\\s+", " ", trimws(gsub("\r?\n|\r", " ", values))))) %>%
  select(RawString,SDTMVAR,DATASET, CODELIST,RawString_cleaned)

#more RawString Clean Process if necessary
```

	RawString	Freq	SDTMVAR
2	cap = Capsule	14	CMDOSU
3	g = Gram	22	CMDOSU
4	gtt = Drop	20	CMDOSU
5	mcg = Microgram	217	CMDOSU
6	mg = Milligram	2187	CMDOSU
7	mL = Milliliter	163	CMDOSU
8	Other	215	CMDOSU
9	Puff	6	CMDOSU
10	Spray	12	CMDOSU
11	tab = Tablet	92	CMDOSU
12	tsp = Teaspoon	3	CMDOSU
13	U = Unit	88	CMDOSU

RawString	SDTMVAR	DATASET	CODELIST	RawString_cleaned
cap = Capsule	CMDOSU	CM	UNIT	cap
cap = Capsule	CMDOSU	CM	UNIT	capsule
g = Gram	CMDOSU	CM	UNIT	g
g = Gram	CMDOSU	CM	UNIT	gram
gtt = Drop	CMDOSU	CM	UNIT	gtt
gtt = Drop	CMDOSU	CM	UNIT	drop
mcg = Microgram	CMDOSU	CM	UNIT	mcg
mcg = Microgram	CMDOSU	CM	UNIT	microgram
mg = Milligram	CMDOSU	CM	UNIT	mg
mg = Milligram	CMDOSU	CM	UNIT	milligram
mL = Milliliter	CMDOSU	CM	UNIT	ml
mL = Milliliter	CMDOSU	CM	UNIT	milliliter

Figure 3. Sample Frequency table of Concomitant Medications Unit (Right) Cleaned RawString (Left).

2. **Prepare CDISC CT**

   A preparation process is applied to the CDISC CT after users defined their source of CT. Since the way of we find match in CDISC CT is by search **RawString_cleaned** values in **CDISC Submission Value** first, if no matches are found we will search through **CDISC Synonym(s)**. However, **CDISC Synonym(s)** composite by multiple terms separated by ';'. If directly search through **CDISC Synonym(s)** by exact match it will not be able to find as many results as we want. Therefore, before matching a preparation process is applied to the CDISC CT. The **CDISC Synonym(s)** string is split by the semicolon (;), and the split terms are then combined with the **CDISC Submission Value** and stored in **CTString**, as shown in **Figure 4**. For example, if the **CDISC Submission Value** is "/month" and the associated **CDISC Synonym(s)** is "Every Month; Per Month" the **CTString** column will contain the terms "Every Month" and "Per Month" from the synonyms, along with "/month" from the CDISC Submission Value.

Code	Codelist Code	CDISC Submission Value	CDISC Synonym(s)	CTString
C176387	C71620	/MBP	/10^6 BP; /Mb; /Mbp; Per Megabase Pair	/MBP
C176387	C71620	/MBP	/10^6 BP; /Mb; /Mbp; Per Megabase Pair	/10^6 BP
C176387	C71620	/MBP	/10^6 BP; /Mb; /Mbp; Per Megabase Pair	/Mb
C176387	C71620	/MBP	/10^6 BP; /Mb; /Mbp; Per Megabase Pair	/Mbp
C176387	C71620	/MBP	/10^6 BP; /Mb; /Mbp; Per Megabase Pair	Per Megabase Pair
C66967	C71620	/min	*NA*	/min
C130188	C71620	/mm	*NA*	/mm
C122199	C71620	/mm2	*NA*	/mm2
C64498	C71620	/month	Every Month; Per Month	/month
C64498	C71620	/month	Every Month; Per Month	Every Month
C64498	C71620	/month	Every Month; Per Month	Per Month

Figure 4. Data Cleaning Process for CDISC Synonym(s) and Submission Value

3. **Data Integration**

   The **RawString** is generated after the user selects a specific variable they want to format into CDISC CT. This **RawString** is then integrated with the SDTM IG metadata using the corresponding SDTMVAR. The **CODELIST** value associated with that specific **SDTMVAR** in the SDTM IG metadata is used to retrieve the relevant values from the CDISC CT.

   Following the data processing steps outlined in Steps 1 and 2, the **RawString** is cleaned and stored as **RawString_cleaned**. This cleaned data is then used to search for matching terms in the processed CDISC CT terms (**CTString**) to determine the final **CDISC Submission Value** for submission, as shown in Figure 4.

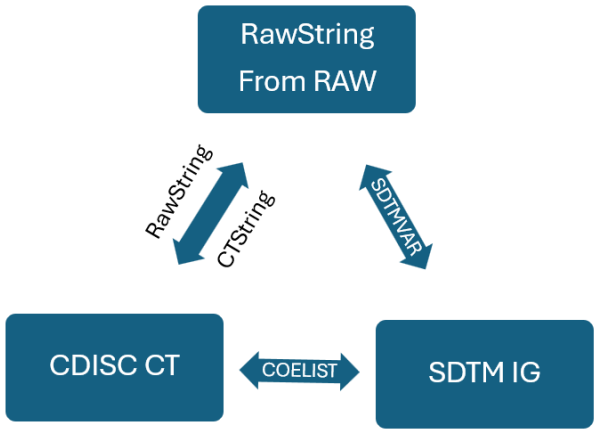
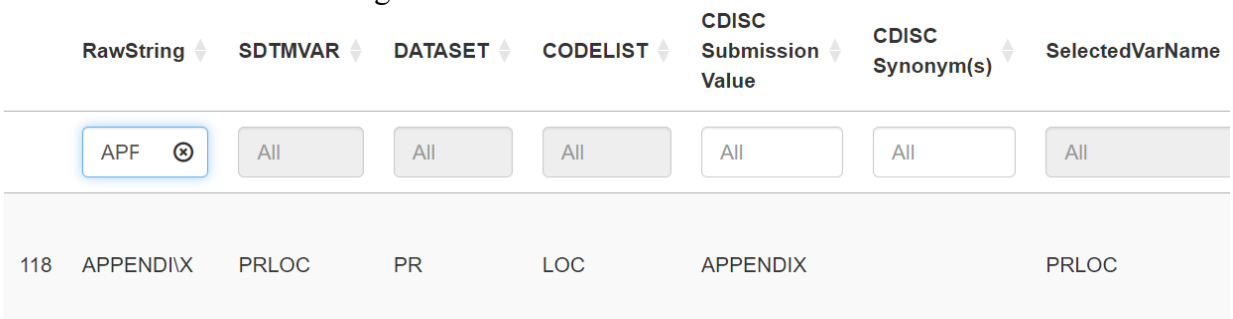


Figure 4. Data Integration Process

4. **String Matching Algorithm**
   To ensure accurate matches, two primary approaches are used:

   Exact Match
   The first approach involves performing an exact match between the **RawString_cleaned** and the **CTString**. If a customized **CT Knowledge Bank** is available, the application will first attempt an exact match with the mappings from this bank, which may include sponsor-defined mappings or previously defined mappings. This ensures the match meets sponsor requests. If no customized CT Knowledge Bank is available, the exact match will be performed between **RawString_cleaned** and the **CTString** from the CDISC CT. This approach ensures that the matched **CDISC Submission Value** meet CDISC standard.

   Similarity Based Match
   If no match is found using the exact match approach, the second method is employed. In this step, the **Levenshtein distance** similarity method is applied to identify the closest match between the **RawString** and **CTString**.
   The Levenshtein distance method is particularly useful for handling spelling errors, which are common in raw terms from EDC systems. It works by calculating the minimum number of single-character edits required to transform one string into another. These edits include insertion, deletion, or substitution of characters. This method allows the application to identify terms that are similar but not identical, increasing the chances of finding a valid match despite small spelling discrepancies, such as missing letters, extra spaces, or swapped characters.
   For example, if a raw term like "APPENDIX" is recorded as " APPENDI\X " in the EDC system due to a typographical error, the Levenshtein distance method can still recognize the two terms as closely related and suggest a match with the correct CDISC CT value for " APPENDIX " as shown in Figure 5.



	RawString	SDTMVAR	DATASET	CODELIST	CDISC Submission Value	CDISC Synonym(s)	SelectedVarName
	APF ⊗	All	All	All	All	All	All
118	APPENDI\X	PRLOC	PR	LOC	APPENDIX		PRLOC

Figure 5. Levenshtein Distance Method: Identifying Terms with Spelling Variations.

Levenshtein distance similarity method ensures that raw terms, even with minor discrepancies, are properly mapped to their corresponding controlled terms in the CDISC CT.

In similarity-based matching, the R similarity threshold plays a critical role in determining how closely the **RawString** must match the **CTString** to be considered a valid match. This threshold defines maximum allowable Levenshtein distance for considering two strings a match. For instance, a lower allowable Levenshtein distance threshold (e.g., 2) results in fewer matches (Figure 5), but they are more likely to be accurate. Conversely, higher Levenshtein distance threshold (e.g., 5) identifies more matches but may introduce false matches (Figure 6).

To manage this, users can manually adjust the results through the R Shiny portal, removing incorrect matches and retaining the correct ones. Given the importance of accuracy in CDISC CT Mapping, the Levenshtein distance is fixed at 2 in the current application. This setting balances the need for accurate matches while minimizing the risk of false positives.

By combining these two approaches, the application ensures that **RawString** values are mapped to the most accurate **CDISC Submission Value**.

	RawString	SDTMVAR	DATASET	CODELIST	CDISC Submission Value	CDISC Synonym(s)	SelectedVarName
	APF ⊗	All	All	All	All	All	All
151	APPENDI\X	PRLOC	PR	LOC	APPENDIX		PRLOC
152	APPENDI\X	PRLOC	PR	LOC	PENIS		PRLOC
153	APPENDI\X	PRLOC	PR	LOC	PHARYNGEAL TONSIL	Adenoid	PRLOC

Figure 6. CDISC CT Mapping Automation in R Shiny with higher threshold of 5

CONCLUSION

The R Shiny application for CDISC CT Mapping Automation offers an effective and efficient way to map raw data terms to the appropriate CDISC Submission Values. It uses a combination of exact matching and similarity-based matching techniques to ensure that the mappings are accurate and reliable, even when there are small variations in the raw data.

The application is flexible, allowing users to customize the mapping process, including the option to use a CT Knowledge Bank to align with both CDISC and sponsor submission requirements. With an easy-to-use interface and robust backend processing, users can quickly review, update, and export the results in various formats such as SAS code, Excel files, and SAS datasets.

The development and analysis of this application were carried out using R (version 4.4.2). Key packages used for this application include: shiny, haven, dplyr, writexl, stringdist , fuzzyjoin, readxl, tidyr, and DT.

Contact Information Your comments and questions are valued and encouraged.

Contact the author at:

Yunsheng Wang

yunsheng.wang@clinchoice.com

ClinChoice Inc

Chao Wang

chao.wang@clinchoice.com

ClinChoice Inc