Paper OS14

Effortless SDTM Up-versioning: Leveraging Open-Source R for Efficient Data Standardization

Ramya Gangadi, Ephicacy Lifesciences, Bangalore, India
Madhavi Gundu, Ephicacy Lifesciences, Bangalore, India

ABSTRACT:

CDISC has always recognized the importance of developing data standards to streamline the electronic submission process, benefiting the pharmaceutical industry. As part of this effort, the CDISC-SDS team will be regularly updating the implementation guide to introduce new domains and implementation rules. For any new studies submitted to regulatory agencies in 2023 and beyond, it is recommended to use SDTM-IG 3.3. Given the substantial updates, understanding these changes poses a significant challenge for the Pharma/Biotech industry.

To address this, we have developed a post-processing approach using R open-source technology. By leveraging a difference file from the CDISC Library, this approach allows for the seamless updating of SDTM datasets from version 3.2 to 3.3, or from 3.3 to 3.4, without affecting the standard macros and programs of an existing sponsor library. This paper demonstrates R programming techniques and provides examples to create SDTM 3.3-compliant datasets, using SDTM 3.2 datasets as input.

INTRODUCTION:

The pharmaceutical industry relies on CDISC standards for efficient and regulatory-compliant data submission. Regular updates to the SDTM Implementation Guide (IG) introduce new domains, variables, and validation rules, ensuring compatibility with evolving regulatory requirements. However, transitioning from one SDTM version to another poses challenges, including modifications to metadata structures and validation processes.

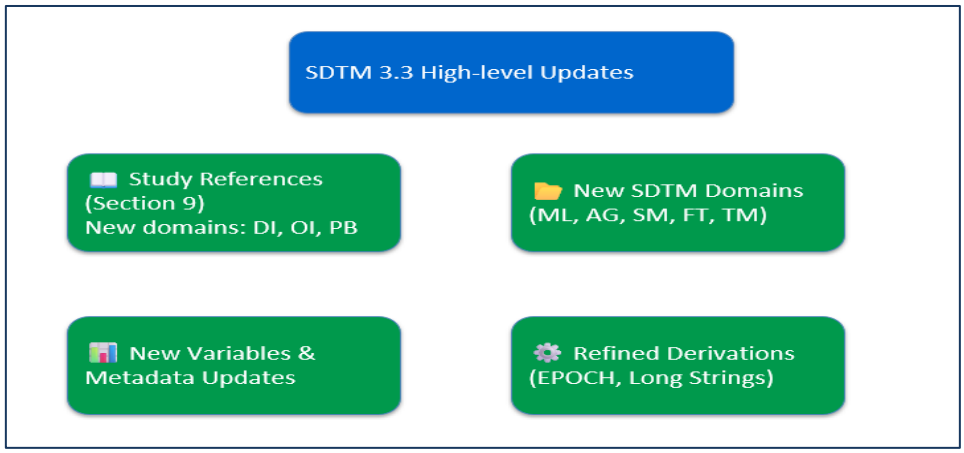
This paper introduces an automated post-processing approach using open-source R programming to facilitate SDTM up-versioning while maintaining the integrity of existing SDTM datasets. By utilizing difference files from the CDISC Library, this method ensures seamless migration without disrupting established workflows.

METHODOLOGY

This section details the approach used to upgrade SDTM datasets efficiently.

Understanding SDTM Version Changes:

- ✓ A new Section 9: Study References has been added to define study-specific terminology.
- ✓ New SDTM domains have been introduced across Interventions, Special Purpose, Findings, and Trial Design categories.
- ✓ Some existing domains now include new variables.
- ✓ Additional details clarify variable derivation, including EPOCH rules and handling long character strings.



To ensure compliance, it is essential to identify and integrate these changes systematically.

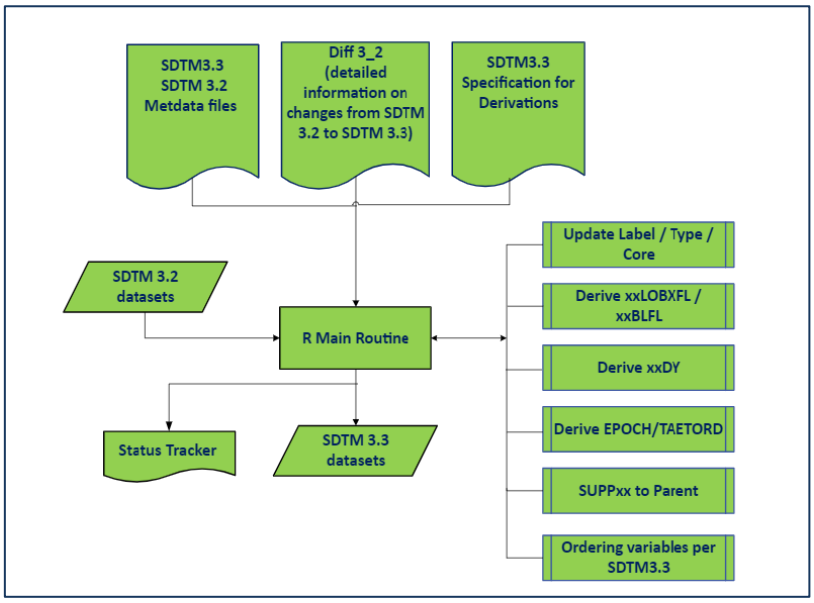
LEVERAGING R FOR AUTOMATED UP-VERSIONING

The open-source R programming language provides robust data processing capabilities, enabling the automation of SDTM version migration. The key steps include:

1. Extracting the difference file from the CDISC Library.
2. Mapping dataset variables between versions.
3. Updating metadata and applying transformation rules.
4. Creation and utilization of a specification file for addition of any methods/applying based on sponsor rules.
5. Creating status file based on modifications which will help the reviewers/client to oversee the changes.
6. Validating outputs against regulatory requirements.

The methodology ensures that existing sponsor macros and programs remain unaffected while upgrading datasets efficiently.

Process Flow:



Read SDTM Metadata files in R:
Read the SDTM 3.2 excel, SDTM 3.3 excel, and Difference excel file detailing the changes from 3.2 to 3.3 available from CDISC library.

```
# Read SDTM Meta excel files

setwd(data_path)
list_study <- list.files(data_path)
print(list_study)
sdtm3.2 <- read_excel(paste0(data_path,"SDTMIG_v3.2.xlsx"), sheet = "Variables")
sdtm3.3 <<- read_excel(paste0(data_path,"SDTMIG_v3.3.xlsx"), sheet = "Variables") %>%
  rename_with(~ gsub(" ", "_", .)) %>%  # Replace spaces in column names with '_'
  select(Dataset_Name,
         Variable_Name,
         Variable_Label,
         Variable_Order)
diff_sdtm <- read_excel(paste0(data_path,"Diff_3_2.xlsx"), sheet = "Basic")
```

Pre-processing the Difference file:

The automated approach for updating SDTM 3.2 datasets to **SDTM 3.3** using a **Difference File**.

Read the SDTM 3.2 datasets and apply metadata changes based on impact and Action from the Difference file.
For eg: for the impact ‘Variable Label’ and Action ‘Value Update’ which is a Major change based on level categories, read the 'Attribute (previous)' and 'Attribute (updated)'.

The figure below shows difference file for the changes needed for EG domain.

Updated Version	Previous Versi	Action	Impact	Change Level	Class	Dataset Na	Variable Name	Automation Needed (Y/N	Attribute (updated)	Attribute (previous)	SUPP_PARE
SDTMIG v3.3	SDTMIG v3.2	Value Update	Structure	Major	Findings	EG		Not Needed	One record per ECG observation per replicate per time point or one record per ECG observation per beat per visit per subject	One record per ECG observation per time point per visit per subject	
SDTMIG v3.3	SDTMIG v3.2	Value Update	Core	Major	Findings	EG	EGBLFL		Perm	Exp	
SDTMIG v3.3	SDTMIG v3.2	Value Update	Variable Label	Major	Findings	EG	EGMETHOD		Method of Test or Examination	Method of ECG Test	
SDTMIG v3.3	SDTMIG v3.2	Value Update	Variable Label	Major	Findings	EG	EGREASND		Reason ECG Not Done	Reason ECG Not Performed	
SDTMIG v3.3	SDTMIG v3.2	Add	Variable		Findings	EG	EGBEATNO		{ "_links": "{...}", "core": "Perm", "description": "A sequence number that identifies the beat within an ECG.", "label": "ECG Beat Number", "name": "EGBEATNO", "ordinal": "14", "role": "Variable Qualifier", "simpleDatatype": "Num" }		

Figure: Screenshot of Difference file

```r
# Process Diff_sdtm
metadata1 <- diff_sdtm %>%
     select ('Dataset Name' , 'Variable Name', 'Attribute (updated)',
               'Impact','Attribute (previous)','Action')

metadata2 <- metadata1 %>%
  mutate(
    Impact_no = case_when(
      Impact == 'Variable Label' ~ 1,
      Impact == 'Type' ~ 2,
      Impact == 'Core' ~ 3,
      Impact == "Variable" ~ 4,
      Impact == "Core" & is.na(`Attribute (previous)`) ~ 5,
      TRUE ~ 0
    )) %>%
    filter(Impact_no != 0)

metadata<- metadata2 %>%
  rename(
    dsetname=`Dataset Name`,
    varname=`Variable Name`,
    u_attr=`Attribute (updated)`,
    p_attr=`Attribute (previous)`
  )
```

Figure: Screenshot of Processing step for the Difference file.

Define Transformation Rules for SDTM3.3 Conversion

The methodology involves defining subroutines in R to perform key transformations, including:

- **Creating new variables** and updating the metadata as per 3.3(e.g., EPOCH, TAETORD, XXDY, XXLOBXFL).
- **Modifying variable labels** to align with SDTM 3.3 specifications.
- **Updating data types** where necessary to meet compliance standards.
- **Converting the variables** from Perm to Exp and vice-versa.

The figure below shows the R function to create the study day for a domain.

```r
# Study day calculation
# dataset <- Source dataset DS
# var1< - variable to be created DSDY

stddy <- function(dataset, var1)
{

  #Check if **DY exists in dataset
  if (var1 %in% colnames(dataset)) {
    print(paste0(var1, " already exists in source dataset"))
    status <<- rbind(status, data.frame(dataset_name = glb_dset, variable_name = var1,
                                        comment = "Variable exists in 3.2"))
  }
  else
  {
    # replace **DY with DTC to get the ***DTC source
    dt <- gsub("DY$", "DTC", var1)

    if (dt %in% colnames(dataset)) {
      print(paste0(var1, " is missing. Deriving ", var1, " from ", dt, "..."))

      # Combine dataset with DM to get RFSTDTC
      dm_path <- paste0(in_path, "dm.sas7bdat")
      dm<- read_sas(dm_path)
      merged_data <- dataset %>%
        left_join(dm %>% select(USUBJID, RFSTDTC), by = "USUBJID")

      # Convert dates to Date format
      dt <- as.Date(merged_data[[dt]])
      rfstdtc <- as.Date(merged_data$RFSTDTC)
```

Figure: Screenshot of function study day calculation.

```
      # Compute ***DY
      merged_data[[var1]] <- ifelse(
        dt < rfstdtc,
        as.numeric(dt-rfstdtc),
        as.numeric(dt-rfstdtc)+ 1 )

      dataset_dy <- merged_data %>% select(-RFSTDTC)

      #Add label variable created
      dataset_metadata <- sdtm3.3 %>%
        filter(Dataset_Name == toupper(glb_dset) & Variable_Name == var1)
      dataset_lbl<- dataset_metadata$Variable_Label
      attr(dataset_dy[[var1]], "label") <- dataset_lbl

      dataset<- rearrange(dataset_dy)

      status <<- rbind(status, data.frame(dataset_name = glb_dset, variable_name = var1,
                                          comment = "Variable added"))
      }

    else {
      print(paste0( dt, " does not exist in dataset. No action taken."))
      status <<- rbind(status, data.frame(dataset_name = glb_dset, variable_name = dt,
                comment = paste0("Variable ",dt," does not exist, cannot derive ",var1)))
      }
  }

  return(dataset)  # Return updated dataset
}
```

Read the Differences file & apply transformations:

The process involves analysing the specification file to create new variables like **EPOCH, TAETORD, XXDY, XXSTDY, and XXENDY** based on the "Differences" file. The differences file helps identify impacted variables and actions, such as adding new ones. The SDTM3.2 datasets are read and based on “Differences” file , the respective sub-routine is executed to generate the SDTM 3.3 datasets.

The specification file is reviewed to identify any existing derivation rules or to define new ones as needed. This process may involve modifying the current **SDTM mapping specification** or incorporating additional rules for derived variables.

The below figure shows the code to traverse through input SDTM 3.2 datasets

```
sdtm_mig<- function()
{

  files <- list.files(in_path,pattern = ".sas7bdat")

  for (file in files)
  {
        dset <- NULL
        file_path <- paste0(in_path, file)
        dset <- read_sas(file_path)
        glb_dset <<- tools::file_path_sans_ext(basename(file_path))
        print(paste0("Datset Name",glb_dset))

        # get the metadata for the domain parameter passed
        dset_meta <- metadata %>%
              dplyr::filter(dsetname == toupper(glb_dset))

       if (nrow(dset_meta) > 0) {

        print(paste0("Data changes for --",glb_dset))
        df_ref<- as.data.frame(dset_meta)
```

Program code with calls to respective functions based on the Difference file.

```
       # Add Variable
       if (impact_ == 4)     {
         print(paste0("Add Variable.................",var_name))
         # Check if the variable is **DY
         if (grepl("DY$", var_name)) {
           print(paste0(var_name, " end with 'DY' "))
           dset<- stddy(dset,var_name)
          }
         else if (grepl("LOBXFL$", var_name)) {
           print(paste0(var_name, " end with 'LOBXFL' "))
           dset<- bflg(dset,var_name,"RFSTDTC")
          }
         else if (grepl("BLFL$", var_name)) {
           print(paste0(var_name, " end with 'BLFL' "))
           dset<- bflg(dset,var_name,"RFSTDTC")
          }
         else if (var_name %in% c("EPOCH", "TAETORD"))
         {
           print(paste0(var_name, "In EPOCH sub-routine "))
           dset<- gen_epoch(dset,var_name)
         } }

      #If any Dataset variable exists in the Diff datasset
      else if(var_name %in% colnames(dset)) {
        # label change
          if (impact_ == 1)   {
            print("Changing Variable label...........")
            attr(dset[[var_name]], "label") <- chg_attr
            status <<- rbind(status, data.frame(dataset_name = glb_dset, variable_name = var_name,
          }
```

Program code to update the dataset and convert to SDTM3.3

The below figure updates the dataset and writes the changes made in the target location out_path.

```
          #Variable in Diff Metadata doesnot exist in SDTM3.2
          else
          {
            status <<- rbind(status, data.frame(dataset_name = glb_dset, variable_name = var_name,
                                    comment = "Variable does not exists in SDTM3.2 for changes"))
          }

        } # End of varibles from the Metadata for the given dataset.

        # Call to check for Perm/Exp updates
        core_update(toupper(glb_dset), dset)

        #Save the updated datast to target
        file_path <- file.path(out_path, paste0(file))
        write_sas(dset, file_path)
      }

  } # End of Loop through each source dataset

}
```

Update Status tracker:

For every change listed in the Difference File, a status record is created to track the modifications made. This record includes details such as the impacted variable, type of change (addition, modification, or deletion), and the corresponding action taken. It ensures transparency and traceability of all transformations applied.

Dataset_Name	Variable_Name	Status
ds	DSDY	Variable added
eg	EGMETHOD	Variable does not exists in SDTM3.2 for changes
eg	EGREASND	Variable does not exists in SDTM3.2 for changes
eg	EGLOBXFL	Variable added
eg	EPOCH	Variable added
eg	TAETORD	Variable added

Figure: Screenshot of status tracker.

VALIDATION:

The validation of SDTM 3.3 datasets is performed using Pinnacle 21, a widely adopted tool for ensuring compliance with CDISC standards. When assessing the transformation from SDTM 3.2 to 3.3, we can compare the P21 reports of both versions by analyzing the errors, warnings, and issues identified in each. This comparison helps verify that the transformed datasets not only adhere to the updated compliance rules and new validation checks introduced in SDTM 3.3 but also ensures that any discrepancies or new validation findings are addressed appropriately.

CONCLUSION:

The transition between different SDTM versions is crucial for ensuring compliance with evolving regulatory standards. The approach presented in this paper demonstrates an efficient, automated method for SDTM up-versioning using open-source R. By leveraging difference files from the CDISC Library, this method streamlines metadata updates, preserves existing sponsor macros, and minimizes manual intervention.

The automated workflow enables seamless dataset transformation from SDTM 3.2 to 3.3 (or higher), ensuring data integrity and reducing operational burden. Additionally, the creation of validation reports and status files enhances transparency, facilitating easier review and compliance verification.

By integrating R-based automation into clinical data workflows, pharmaceutical and biotech companies can improve efficiency, maintain regulatory adherence, and optimize their data standardization processes. This work highlights the power of open-source tools in the life sciences industry and underscores the benefits of automation in regulatory data management.

REFERENCES

Study Data Tabulation Model Implementation Guide: Human Clinical Trials. Clinical Data Interchange Standards Consortium (CDISC) Submission Data Standards (SDS) Team. Version 3.3. November 2018

CDISC: Standards: CDISC Library https://library.cdisc.org/browser/#/mdr/sdtmig/3-3/classes/GeneralObservations

ACKNOWLEDGMENTS

We would like to express our warm gratitude to the management of the organization – “Ephicacy” for the encouragement and support in helping with all the necessary facilities to write this paper. Our sincere thanks to our families and team for their unwavering support and encouragement.
I sincerely thank my co-author Madhavi. Gundu and my colleagues Vaibhav Dewangan and Utkarsh Pathak for their extended support and valuable contributions to this work. Your guidance and collaboration are truly appreciated.

CONTACT INFORMATION

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

Author Name: Ramya Gangadi
Company: Ephicacy Lifesciences
Address: Bangalore, India
Email: ramya.gangadi@ephicacy.com
Website: https://www.ephicacy.com/

Co - Author Name: Madhavi Gundu
Company: Ephicacy Lifesciences
Address: Bangalore, India
Email: madhavi.gundu@ephicacy.com
Website: https://www.ephicacy.com/