International Journal of Scientific Research in Engineering and Management (IJSREM)
Volume: 08 Issue: 09 | Sept - 2024 SJIF Rating: 8.448 ISSN: 2582-3930

A Comprehensive Guide to Generating SDTM Demographic Datasets Using R in Clinical Trials

Arvind Uttiramerur
Programmer Analyst at Thermofisher Scientific, USA

ABSTRACT

R is a programming language widely used for statistical analysis and data visualization, offering a flexible and interactive environment supported by various packages for data cleaning, tidying, and analysis. It is particularly relevant for professionals in mathematics and statistics, including biostatisticians and programmers in the pharmaceutical and biotech industries. R provides a robust array of user-developed packages that can efficiently manipulate complex datasets, such as those based on the Study Data Tabulation Model (SDTM). The popularity of R in data-related fields has surged exponentially over the past decade due to its open-source nature, powerful statistical capabilities, and advanced visualization tools.

In this paper, we demonstrate a step-by-step approach to generating an SDTM Demographic (DM) dataset using R. The process leverages R packages such as sas7bdat, tidyverse, haven, parsedate, dplyr, tidyr, and Hmisc. We also provide a detailed procedure for setting up the R environment required for this process. While R has been extensively used for exploratory analysis in the pharmaceutical and biotech industries, its application in creating and analyzing clinical trial datasets, such as SDTM, has been limited. Traditionally, SAS® has been the preferred tool for generating clinical trial datasets. This paper explores R's potential as a viable alternative, offering enhanced flexibility and cost-effectiveness in clinical trial.

INTRODUCTION

R is increasingly being used across various industries for data analysis and visualization, offering critical insights through its powerful capabilities. Although R has been employed for exploratory analysis within the pharmaceutical and biotech industries for many years, it has not yet been widely adopted for the creation and analysis of clinical trial datasets. In the highly regulated Pharma/Biotech sector, validated systems that have undergone the rigor of the Software Development Life Cycle (SDLC) are typically recommended. While open-source tools like R are often utilized for checking the quality of clinical trial datasets, they have not traditionally been used for analyzing clinical trial data, particularly for regulatory submissions.

Despite the existence of some user-developed R packages that have not gone through a formal SDLC process, this paper investigates the technical feasibility of using R to generate SDTM (Study Data Tabulation Model) and ADaM (Analysis Data Model) datasets. We explore how the functionalities offered by this cost-effective and open-source software can be leveraged in the context of clinical trial data management.

R is a language and environment designed for statistical computing and graphics, making it a viable alternative to SAS for generating specialized clinical trial datasets, tables, listings, and figures. Freely available and supported by the R Foundation for Statistical Computing, R offers a broad range of specific packages for the design, monitoring, and analysis of clinical trial datasets. These packages include sas7bdat for reading SAS files, dplyr for data manipulation, tidyr for data tidying, and Hmisc for attaching labels to variables, among others.

STUDY DATA TABULATION MODEL (SDTM) OVERVIEW

The Study Data Tabulation Model (SDTM) is a standardized framework developed by the Clinical Data Interchange Standards Consortium (CDISC) for organizing and formatting data collected during clinical trials. SDTM is designed to facilitate the submission of clinical data to regulatory authorities, such as the U.S. Food and Drug Administration (FDA) and the European Medicines Agency (EMA). The model ensures that data is presented in a consistent format, making it easier for regulatory reviewers to understand and analyze the data.

The SDTM standard is divided into various domains, each corresponding to different types of data collected during a clinical trial. These domains include, but are not limited to, Demographics (DM), Adverse Events (AE), Laboratory Test Results (LB), and Medical History (MH). Each domain is represented by a dataset, where variables are organized into columns, and observations (data points) are organized into rows.

DEMOGRAPHICS (DM)

Importance of the Demographic (DM) Dataset

The Demographic (DM) dataset is a core component of the SDTM standard. It contains key demographic information about the subjects who participated in the clinical trial, such as their age, sex, race, and ethnicity. This dataset serves as the foundation for analyzing and interpreting other clinical data collected during the trial, as demographic variables often influence the outcomes and interpretations of the study.

The DM dataset typically includes variables such as:

- *STUDYID: Unique identifier for the study.*
- *USUBJID: Unique subject identifier.*
- *SUBJID: Subject identifier for the study.*
- *RFSTDTC: Reference start date/time of the study.*
- *RFENDTC: Reference end date/time of the study.*
- *BRTHDTC: Date of birth.*
- *AGE: Age of the subject at the time of informed consent.*
- *AGEU: Unit of age (e.g., years, months).*
- *SEX: Sex of the subject.*
- *RACE: Race of the subject.*
- *ETHNIC: Ethnicity of the subject.*
- *ARM: Description of the subject's treatment group.*

Relevance of the DM Dataset in Clinical Trial Analysis

The DM dataset is essential for stratifying subjects into relevant subgroups during the analysis of clinical trial data. This stratification helps in identifying trends, differences, and potential biases within the study population. For example, differences in treatment efficacy or safety across different age groups, sexes, or ethnicities can be explored using the DM dataset.

Additionally, demographic data are often used to ensure that the study population is representative of the target population, which is crucial for the generalizability of the study results.

R PACKAGES AND R SYNTAX

R can be used for clinical trial data manipulation and creation of CDISC: SDTM/ADaM data sets. The following R packages and syntax were used during this evaluation.

R- Packages	R-syntax
Base : Base R functions	Select
Dplyr : Designed for data transformation and merge dataset	Mutate
Tidyr: Transpose dataset	group by
Lubridate : Date functions	filter
Haven : Loads and exports SAS files	arrange
Sas7bdat : Loads and exports SAS files	summarize
Hmisc: Check attributes	spread/gather
Hmisc: Attach Label	left/right/inner/full joins
Bridge: Generates infile SAS® program and CSV file for a data set	bind_cols/bind_rows

R CODE FOR GENERATING DATASET

R packages and libraries are installed, the R-environment is ready to read and process the input dataset's used to generate.

Read sas dataset files:

```
DM<- read.sas.7bdat (“c:/sdtm/dm.sas7bdat”)

Supdm<- read.sas.7bdat (“c:/sdtm/suppdm.sas7bdat”)

Ds<- read.sas.7bdat (“c:/sdtm/ds.sas7bdat”)

Suppds<- read.sas.7bdat (“c:/sdtm/suppds.sas7bdat”)

Ex<- read.sas.7bdat (“c:/sdtm/ex.sas7bdat”)

Suppex<- read.sas.7bdat (“c:/sdtm/suppex.sas7bdat”)

Dm <- dm[order(subjid, ]

Suppdm<- suppdm[order(subjid, ]
```

Library setup:

```
sdtm <- "//product/study/analysis/data/sdtm" # assign dir to object named sdtm

out <- "//product/study/analysis/data/adam"
```

GENERATION OF THE DM DATASET USING R

In the context of this paper, we will demonstrate the generation of an SDTM-compliant DM dataset using R, a powerful open-source software environment. We will leverage various R packages such as sas7bdat, tidyverse, haven, parsedate, dplyr, tidyr, and Hmisc to import, clean, and organize the raw data into the standardized SDTM format. The step-by-step approach will highlight how these packages can be effectively utilized to create a compliant DM dataset, offering a viable alternative to traditional tools like SAS.

Create a SDTM Demographics (DM) domain program in R:

***Load Libraries**: Use dplyr for data manipulation, tidyr for tidying, and lubridate for date operations.*

```
# Install the packages if not already installed
 install.packages("dplyr")
 install.packages("tidyr")
install.packages("lubridate")
# Load the libraries
library(dplyr)
library(tidyr)
library(lubridate)
```

***Load Data:** Import raw data from a CSV file. Modify this according to your data format. To import raw data from a CSV file in R, you can use the read.csv() function. This function reads the contents of a CSV file into a data frame, which is a type of table structure commonly used for storing and manipulating data in R.*

Sample Code to Load Data from a CSV File

```
# Load the necessary libraries (dplyr, tidyr, lubridate)
# Load the raw data from a CSV file
raw_data <- read.csv("path_to_your_file/raw_data.csv", stringsAsFactors = FALSE)
# View the first few rows of the loaded data
head(raw_data)
```

If your CSV file uses a different delimiter (e.g., a semicolon ; instead of a comma ,), you can use the read.delim() function or specify the delimiter using the sep argument in read.csv() like so:

```
raw_data <- read.csv("path_to_your_file/raw_data.csv", sep = ";", stringsAsFactors = FALSE)
```

If your file has a header row (the first row with column names), read.csv() will automatically recognize it. If your CSV file does not have a header, you can specify header = FALSE and provide custom column names.

Convert Dates: Ensure dates are properly formatted using as.Date.

To ensure that dates are properly formatted in your dataset, you can use the as.Date() function in R. This function converts character vectors to Date objects. Here's how you can convert dates in your data frame:

If your data also includes time (e.g., 2024-09-01 14:30:00), you can use the lubridate package functions like

ymd_hms():

```
raw_data <- raw_data %>%
  mutate(
    DateTimeColumn = ymd_hms(DateTimeColumn)
  )
```

Derive Age: *Calculate age in years based on the difference between start_date and birth_date.*

To calculate age in years based on the difference between a start date and a birth date, you can use the lubridate package in R. The interval() function can help you compute the difference between two dates, and as.period() can convert this difference into years.

Here's how you can derive age in years:

```
# Assuming your raw data has columns named 'start_date' and 'birth_date'
# Replace these names with the actual column names in your data
# Convert date columns to Date type if not already done
raw_data <- raw_data %>%
mutate( start_date = as.Date(start_date, format = "%Y-%m-%d"),
birth_date = as.Date(birth_date, format = "%Y-%m-%d") )
# Calculate age in years
raw_data <- raw_data %>%
mutate( age = as.numeric(interval(birth_date, start_date) / years(1)) )
# View the first few rows of the data to check the new age column head(raw_data)
raw_data <- raw_data %>%
  mutate(
    age = interval(birth_date, start_date) %/% years(1) +
        as.numeric(interval(birth_date, start_date) %% years(1) / months(1)) / 12
  )
```

Select and Rename Columns*: Align columns with the SDTM DM standard and rename as needed.*

To align columns with the SDTM Demographic (DM) dataset standard and rename them in R, you can use the dplyr package's select() and rename() functions. Here's a step-by-step guide:

```
# Load package   and Assuming you've already loaded your data. Example of loading data from a CSV file
raw_data <- read.csv("path/to/your/data.csv")
# Define file paths for existing SDTM datasets
sdtm_dm_path <- "path/to/DM.sas7bdat"  # Update this path
# Read in the available SDTM DM dataset
sdtm_dm <- read_sas(sdtm
```

Code for DM using R Programming: Main program using R

```
dm <- sdtm_dm %>%
Mutate (  STUDYID = "STUDY123",  # Simulated Study Identifier
DOMAIN = "DM",   # Domain Abbreviation
SITEID = as.character(substr(SUBJID,1,3)),
SUBJID1 = as.character(SUBJID),
USUBJID = paste(STUDYID, SITEID, SUBJID, sep="-"),
RFSTDTC = as.Date("2023-01-01"),  # Simulated Reference Start Date
BRTHDTC = format_iso_8601(parse_iso_8601(BIRTHDT)),
BRTHDTC = substr(BRTHDTC, 1, regexpr("\\T", BRTHDTC)-1),# Simulated Birth Date
AGE = as.integer((as.Date("2023-01-01") - BRTHDTC)/365.25),  # Calculated Age
AGEU = "YEARS",  # Age Units
SEX1 = ifelse(SEX=="FEMALE","F",ifelse(SEX=="MALE","M","")), # Sex Male female
RACE1 = ifelse(RACE=="BLACK","BLACK OR AFRICAN AMERICAN",
ifelse(RACE=="CAUCASIAN","CAUCASIAN",ifelse(RACE=="ASIAN","ASIAN",""))),  # Simulated Race
ETHNIC = ifelse(RACE=="HISPANIC","HISPANIC OR LATINO",""),  # Simulated Ethnicity
ARMCD = ifelse(TRTGROUP=="Placebo","PBO",ifelse(TRTGROUP=="Active","ACT","")),
ARM = TRTGROUP,
ACTARMCD = ARMCD,
ACTARM = TRTGROUP,
ARMNRS = "",
ACTARMUD = "",
COUNTRY = "",
DMDTC = "",
DMDY = "") %>%
select (STUDYID, DOMAIN, USUBJID, SUBJID, SUBJID1, SITEID, RFICDTC, BRTHDTC, AGE, AGEU,
SEX1, RACE1, ETHNIC, TRTGROUP, ARMCD, ARM, ACTARMCD, ACTARM, ARMNRS, ACTARMUD,
COUNTRY, DMDTC, DMDY) %>%
arrange (SUBJID)
# Create variables as per the spec from rawdata DRUG
Drugtrt1 <- drug %>%
mutate(DOSEDTC = format_iso_8601(parse_iso_8601(DOSEDT)),
```

```
      DOSEDTC = substr(DOSEDTC, 1, regexpr("\\+", DOSEDTC)-1)) %>%
  select(SUBJID, DOSEDT, DOSEDTC) %>%
  arrange(SUBJID, DOSEDTC)
# Get first observation from data DOSEDTC
ft_dosedtc <- Drugtrt1 %>%
  filter(!is.na(DOSEDTC)) %>% #For non-missing values
  group_by(SUBJID) %>%
  slice_min(order_by = DOSEDT) %>% #For 1st observation
  mutate(RFSTDTC = DOSEDTC, RFXSTDTC = DOSEDTC)%>%
  select(SUBJID, RFSTDTC, RFXSTDTC)
# Unique returns a data table with duplicated rows removed
ft_dosedtc1 <- unique(ft_dosedtc, incomparables=FALSE)
# Get last observation from data DOSEDTC
lt_dosedtc <- Drugtrt1 %>%
  filter(!is.na(DOSEDTC)) %>%
  group_by(SUBJID) %>%
  slice_max(order_by = DOSEDT) %>%
  mutate(RFENDTC = DOSEDTC, RFXENDTC = DOSEDTC) %>%
  select(SUBJID, RFENDTC, RFXENDTC)
lt_dosedtc1 <- unique(lt_dosedtc, incomparables=FALSE)
# Create variables as per the spec from rawdata DRUG
term1 <- term %>%
  mutate(RFPENDTC = format_iso_8601(parse_iso_8601(TERMDT)),
      RFPENDTC = substr(RFPENDTC, 1, regexpr("\\T", RFPENDTC)-1)) %>%
  select(SUBJID, RFPENDTC) %>%
  arrange(SUBJID)
# Create variables as per the spec from rawdata DIED
died1 <- died %>%
  mutate(DTHDTC = format_iso_8601(parse_iso_8601(DEATHDT)),
      DTHDTC = substr(DTHDTC, 1, regexpr("\\T", DTHDTC)-1),
      DTHFL="Y") %>%
select(SUBJID, DTHDTC, DTHFL) %>%
```

```
  arrange(SUBJID)
# Merge all the datasets to create DM
dmfinall <- left_join(left_join(left_join(left_join(dm1, ft_dosedtc1, by="SUBJID"),
                                lt_dosedtc1, by="SUBJID"),
                        term1, by="SUBJID"),
                died1, by="SUBJID")
# Dropping a variable SUBJID
dmfinal<- within(dmfinal, rm(SUBJID))
# Rename variables
dmfinal <- dmfinal %>%
  rename(SUBJID=SUBJID1, SEX=SEX1, RACE=RACE1) #new_name = old_name
# Order and keep only required variables in final DM
dm <- dmfinal %>%
  select(STUDYID, DOMAIN, USUBJID, SUBJID, RFSTDTC, RFENDTC, RFXSTDTC, RFXENDTC,
         RFICDTC, RFPENDTC, DTHDTC, DTHFL, SITEID, BRTHDTC, AGE, AGEU, SEX, RACE, ETHNIC,
         ARMCD, ARM, ACTARMCD, ACTARM, ARMNRS, ACTARMUD, COUNTRY, DMDTC, DMDY)
#Apply Variables labels
label(DM)=as.list(Variables$Label)
#Apply Dataset label label(DM) = Datasets$Description
dm <- remove_all_labels(dm)
label(dm) <- "Demographics"
# Save the simulated DM domain dataset
```

Write Output: *Save the processed DM dataset to a CSV file and provide a summary to verify the output.*

```
write.csv(dm, "path/to/Simulated_DM_Domain.csv", row.names = FALSE,col.names=TRUE)
# Output DM.xpt
write.xport(dm, file = "xpt/dm.xpt")
```

Finally, check the structure of your new dataset to ensure that the columns have been correctly selected and renamed: This will give you an overview of the new dataset structure, confirming that it aligns with the SDTM DM standard. This will give you an overview of the new dataset structure, confirming that it aligns with the SDTM DM standard.

Filter Missing Values: Remove rows with missing values in required fields.

To remove rows with missing values in required fields from your dataset in R, you can use the filter() function from the dplyr package along with !is.na() to identify and exclude rows where specific fields are NA (missing). Here's how you can do it:

Make sure you've already loaded the dplyr package and Assume you want to remove rows where any of the required SDTM DM fields such as USUBJID, BRTHDTC, AGE, or SEX are missing. You can do it as follows:

```
dm_demo <- dm_demo %>%
  filter(!is.na(USUBJID) &
         !is.na(BRTHDTC) &
         !is.na(AGE) &
         !is.na(SEX))
```

CONCLUSION

This paper has demonstrated the viability of using R as an alternative to SAS for generating SDTM Demographics (DM) datasets in the pharmaceutical and biotech industries. By leveraging a combination of R packages, including sas7bdat, tidyverse, haven, parsedate, dplyr, tidyr, and Hmisc, we have shown that R can efficiently process raw clinical trial data to produce standardized SDTM-compliant datasets.

Our approach highlights the flexibility and cost-effectiveness of using R, especially for organizations looking to reduce dependence on proprietary software while still adhering to regulatory standards . The step-by-step guide provided serves as a practical resource for professionals aiming to integrate R into their clinical trial data management workflows.

However, the adoption of R for SDTM and ADaM dataset generation is still in its early stages, particularly in regulated environments where validated systems are crucial. While R offers robust capabilities for data manipulation and analysis , there are challenges related to its widespread adoption in clinical trial data management, including the need for formal validation processes and greater industry acceptance .

Future work could focus on further validating the R packages used in this paper, as well as expanding the use of R in other domains within the SDTM and ADaM frameworks . Additionally, developing user-friendly R packages specifically designed for clinical trial data management could help accelerate the adoption of R in this space.

In conclusion, R presents a promising alternative to SAS for generating SDTM datasets, offering enhanced flexibility, cost savings, and a broad range of statistical tools . As the pharmaceutical and biotech industries continue to evolve, the role of open-source tools like R is likely to expand, contributing to more efficient and accessible data management solutions in clinical research.

REFERENCE

1. https://sas-and-r.blogspot.com/p/simulation-examples.html
2. https://cran.r-project.org/doc/manuals/r-release/R-lang.pdf
3. *CRAN - PACKAGE DMIRALONCO(RSTUDIO.COM)(HTTPS://CRAN.RSTUDIO.COM/WEB/PACKAGES/ADMIRALONCO)*
4. *SDTM IN R ASSET LIBRARY • ADMIRAL (PHARMAVERSE.GITHUB.IO)*
5. *SDTM IN BUSINESS INTELLIGENCE, COLLINSON, PHUSE 2014*
6. *6.CLINICAL DATA IN BUSINESS INTELLIGENCE, COLLINSON, PHUSE 2016*