PharmaSUG 2026 - Paper HT-369
Virtual Data, Real Standards: Leveraging Data Simulation for Smarter Clinical Trials

Sangeeta Shabadi, Jazz Pharmaceuticals;
Nitesh R Patil, Cytel;
Jonathan Henshaw, Jazz Pharmaceuticals.

ABSTRACT

The Study Data Tabulation Model (SDTM) is central to standardized clinical trial data submissions across global regulatory agencies. While traditionally focused on data organization and compliance, SDTM emerged as a strategic tool for simulation in clinical trial analysis. As clinical data grows in volume and complexity, automation becomes critical. SDTM simulation delivers fast, flexible creation of CDISC-compliant synthetic datasets; customized for any study design or therapeutic area. This paper explores how data simulation in SDTMs involves generating synthetic SDTM-like datasets using random sampling and domain-specific rules to mimic real clinical data. Simulation techniques were used to create realistic subject profiles, dosing patterns, adverse events, and lab values, enforcing cross-domain logic and SDTM structural requirements. This allowed us to develop and test SDTM- 'ADaM- 'TFL workflows early, validate derivations, surface edge-case issues, and improve code robustness without waiting for actual trial data. This approach is increasingly valuable in modern statistical programming because it allows development, testing, and validation before real data is available. Our paper demonstrates how R and SAS simulate SDTM domains by empowering faster, more reliable, and more innovative programming by enabling early code development, stronger validation, and risk-free experimentation without dependence on real clinical data showcasing seamless integration and compliance with CDISC standards.

INTRODUCTION

Clinical trials generate large volumes of complex data that must be structured, standardized, and submission ready to meet global regulatory requirements.

Data simulation refers to the process of generating synthetic datasets that replicate the structure, behavior, and statistical characteristics of real clinical trial data without using actual patient information. In the context of clinical trials, simulation leverages study protocols, CRF designs, and domain specific rules to create realistic subject level data, including demographics, dosing patterns, adverse events, laboratory results, and efficacy endpoints.

Simulation has become increasingly necessary because, at the start of a study, real clinical data is typically unavailable or limited to a small number of enrolled subjects. During this early phase, programming teams are unable to fully develop or validate SDTM mappings, ADaM derivations, or TFL outputs due to insufficient data volume and variability. Waiting for substantial enrollment delays development, compresses timelines, and increases the risk of late-stage issues by providing immediate access to realistic data, allowing teams to begin development activities well before meaningful trial data exists.

In this paper, we describe a practical approach for simulating SDTM compliant datasets using SAS and R. Simulation is not just a tactical solution to early data unavailability; it represents a broader transformation in clinical data strategy. As the industry moves toward automation, standardization, and

intelligent analytics, early access to realistic, structured data becomes a critical enabler. Simulated SDTM datasets allow teams to design and stress test end to end pipelines independent of enrollment progress, reducing downstream risk and rework.

Looking ahead, data simulation will play a key role in metadata driven frameworks, continuous integration models (including modern pipeline testing methods), and AI assisted programming. Organizations can unlock faster development cycles, stronger validation, and more adaptive clinical trial operations.

By creating synthetic raw data sets leading to standard structured SDTM like datasets. early in the workflow, teams can fully build and validate:

- SDTM mapping macros
- ADaM derivation rules
- TLF templates
- End to end programming pipelines
- CDISC compliance checks
- Automation frameworks

This approach significantly accelerates the readiness of the programming package and reduces downstream rework.

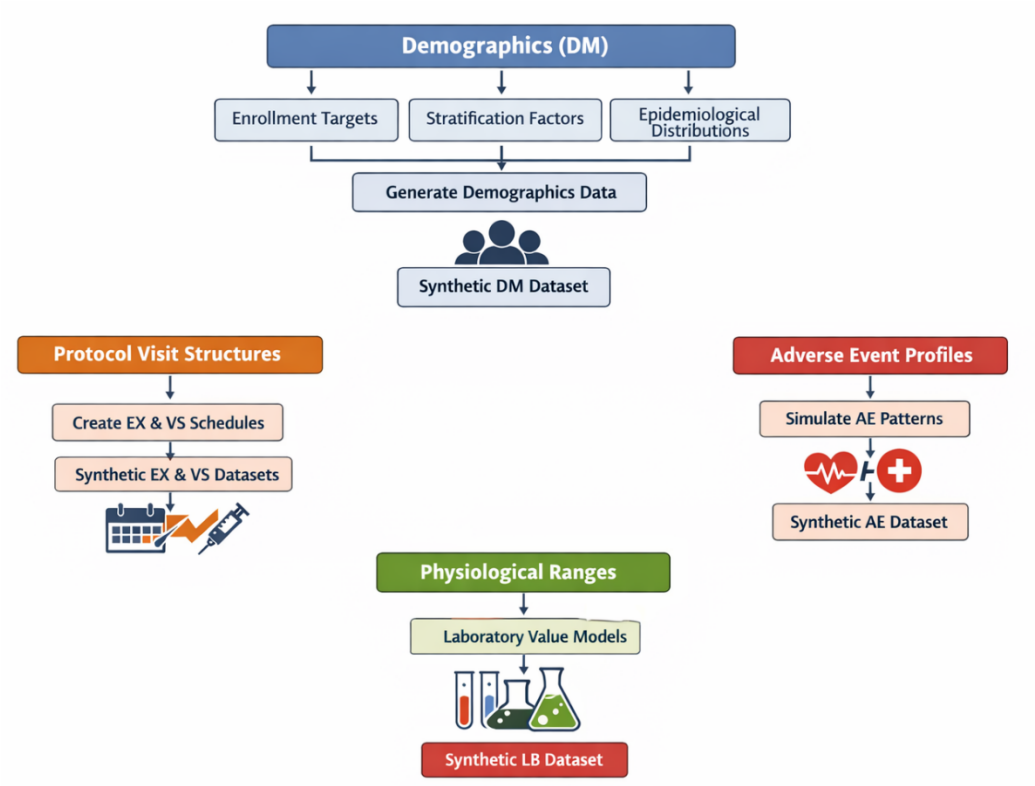
In our work - Synthetic data is generated using random sampling techniques combined with domain specific rules derived from the study protocol and CRFs. Cross domain logic and SDTM structural requirements were enforced to ensure internal consistency. The resulting datasets were used to build and test complete SDTM → ADaM → TFL workflows, validate derivation logic, identify edge case scenarios, and improve code robustness without dependency on live clinical data.

Simulation ensures structural and logical consistency with real world clinical data by adhering to protocol defined study designs, CRF structures, and CDISC SDTM standards. Domain specific rules and cross domain relationships are enforced so that simulated subjects, visits, and assessments behave in ways consistent with actual clinical trial conduct. At the same time, all values are synthetically generated and do not originate from real patients, ensuring that no sensitive or identifiable information is included. This allows unrestricted use of the data for development, testing, and validation purposes. As a result, programming workflows can be exercised realistically while fully eliminating privacy, confidentiality, and compliance risks.

It uses the study protocol, CRF design, and planned visit structure as the primary source of truth. Variables such as visit windows, dosing schedules, assessment frequency, endpoint definitions, adverse event capture, and expected laboratory panels can all be derived from study metadata. Controlled terminology requirements and SDTM implementation rules further guide the generation process. For example:

- DM (demographics) can be generated using enrollment targets, stratification factors, and epidemiological distributions.
- AE patterns can be simulated based on expected safety profiles of similar compounds or historical reference studies.
- LB values can be drawn from physiological distributions aligned with known ranges for the target population.
- EX and VS schedules can be generated directly from protocol visit structure

Figure 1: Design Defining Inputs for Synthetic Data



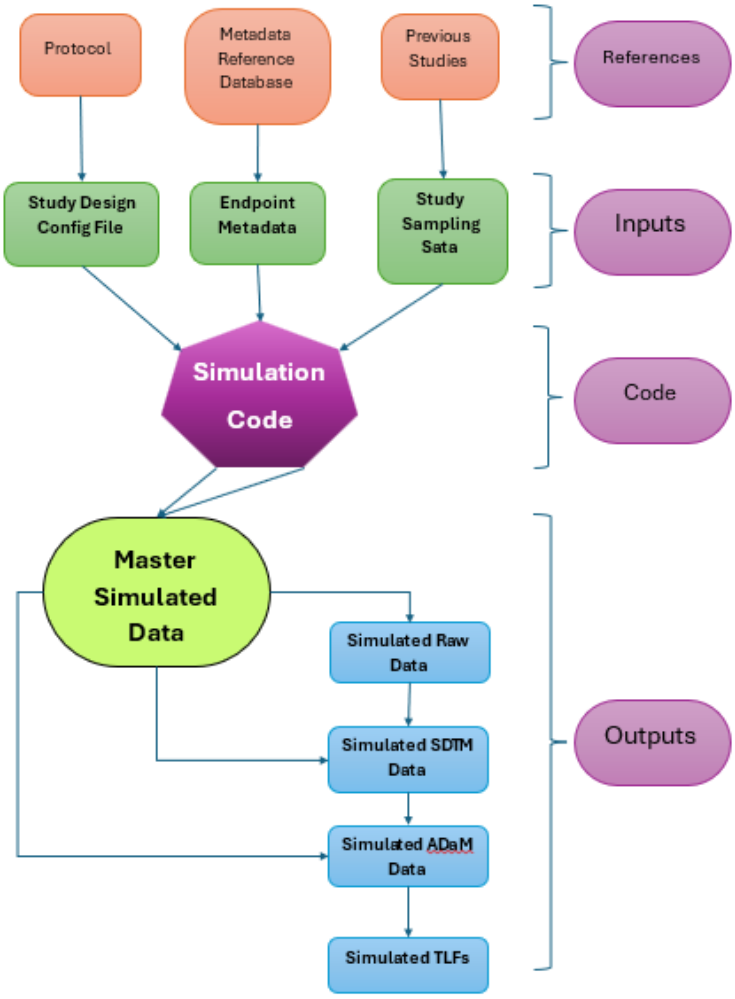
METHODS:

OVERALL WORKFLOW FOR THE SIMULATION FRAMEWORK

The simulation framework was designed to generate realistic, protocol aligned clinical trial data by systematically translating study design specifications into programmable structures. The workflow consists of five major steps:

1. Extracting key patient pathways, scenarios and outcomes from the study protocol,
2. Encoding these pathways into a structured Journey framework,
3. Representing all study level constants in a centralized configuration file and
4. Generating master subject level datasets
5. Rendering the master datasets into a usable “raw” data model that mirrors a study’s actual raw data structures (with capacity to also render directly to SDTM and ADaM if desired). This approach ensures that all components of the simulation remain traceable, reproducible, and consistent with protocol assumptions.

Figure 2: Overview of the SDTM Simulation Workflow



1. Protocol Review and Derivation of Journey Framework

The workflow begins with a detailed review of the clinical study protocol to identify the expected patterns of participant flow through the trial. The protocol specifies processes such as:

- Screening and rescreening
- Randomization
- On treatment visits and assessments
- Early termination scenarios (e.g., AE driven discontinuation, physician decision)
- Death and survival follow-up
- End of study rules

This protocol defines subject pathways and are codified into a Journey Framework, where each unique journey represents a realistic route, a subject may take through study. Journeys represent scenarios such as:

- Full study completion
- Treatment discontinuation due to progression Discontinuation due to adverse events
- Death during the study
- Different types of screen failures and rescreen cycles

Each journey determines which visits a subject receives, how long they remain on treatment, their end of study activities, and what type of adverse events may be generated. The strategy and goals of data simulation will determine whether journeys should closely reflect expected outcomes or ensure all data collection fields appear in the simulated data, or both. A well-formed Journey Framework becomes the backbone of all downstream simulated data derivation.

2. Encoding Protocol Parameters into JSON configuration

After defining journeys, all protocol controlled study parameters are centralized in a JSON configuration file. This file ensures that key inputs are stored in a format that is:

- Machine-readable
- Version controlled
- Reproducible
- Easy modifiable for scenario analyses
- The configuration file includes parameters such as:
- Planned sample size (plansub)
- Eligible age range (agemin, agemax)
- Randomization ratio and treatment names
- Demographic and regional distribution assumptions
- Visit structure, cycle patterns, and visit windows
- Allowed chemotherapy backbones

- Timing rules for survival follow up and End of study determination.

The simulation modules do not hardcode any protocol assumptions. Instead, they reference a central configuration file so that updating the protocol or testing alternative design scenarios requires no code changes—only configuration updates.

Program 1: Configuration JSON file

```
{
  "study": [
    { "study_id": "ABC123-303",
      "plansub": 780,
      "narms": 2,
      "agemin": 18,
      "agemax": 99

    }
  ],
  "endpoints": {
    "ecg_lead": [
      {"id":["PRAG01", "EGHRMNG01", "QRSAG01", "QTAG01", "QTCFAG01", "RR01", "EGINTP01"]}
      ],
    "ecmg": [
      {"id":["LVEF01"]}
      ],
    "lbchem_lbhema": [
      {"id":[
        "ALB01", "ALP01", "ALT01", "AST01", "BILDIR01", "BILI01", "BUN01", "CA01", "CREAT01", "GLU01", "K01", "LDH01", "SODIUM01",
        "BASO01", "BASOLE01", "EOS01", "EOSLE01", "HCT01", "HGB01", "LYM01", "LYMLE01", "MCH01", "MCHC01", "MCV01", "MONO01", "MONOLE01",
"NEUT01", "NEUTLE01", "NEUTB01", "NEUTBLE01",  "PLAT01", "RBC01", "WBC01"
      ]}
    ] ,

    "ecog": [
      {"id":["ECOG101"]}
      ],
    "lbpreg": [
      {"id":[
        "HCG01"
      ]}
    ] ,
    "lburin": [
      {"id":[
        "APPEARU01", "BACTU01", "BILIU01", "CASTSU01", "CLARITYU01", "COLORU01", "CRYSTALSU01", "EPICU01", "GLUU01", "HGBU01", "KETONESU01",
"LEUKASEU01", "NITRITEU01", "PH01", "PROTU01", "RBCU01", "SPGRAV01", "UROBILU01", "WBCU01", "YEASTU01"
```

3. Master Subject Level Dataset Generation

The next step uses the Journey Framework and configuration data to generate complete master datasets for each subject. These datasets serve as the foundation for all raw clinical data tables.

3.1. Subject Level Records (sim_subjects)

sim_subjects generates the master subject level dataset using protocol defined assumptions stored in the configuration file (config_json). The function simulates demographics, baseline characteristics, treatment allocation, country distribution, and unique identifiers. It assigns each subject to a protocol derived Journey, which determines their study pathway (e.g., full completion, early discontinuation, screen failure, rescreening). Reference start dates and treatment start dates are generated with a realistic timing jitter. This master table forms the foundation for all downstream simulations.

Program 2: sim_subject

```r
sim_subject <- function() {


  ########################################
  # Generate master subject level data set
  ########################################

  # Use core general function to generate subject records
  subjects1 <- fct_gen_subjects(
    subjnum = config_json$study$plansub,
    trtnum  = 2,
    trtcodes = c("T0" = "Apple",
                 "T1" = "Orange"),
    trtratio = c(1, 1),
    physc_list = c("Grapes",
                   "Peaches",
                   "Plums",
                   "Nectraines"),
    physc_prob = c(0.25,0.25,0.25,0.25),
    seed = NULL,
    sex_list = c("F", "M"),
    sex_prob = c(0.9, 0.1),
    race_list = c("AMERICAN INDIAN OR ALASKA NATIVE",
                  "ASIAN",
                  "BLACK OR AFRICAN AMERICAN",
                  "NATIVE HAWAIIAN OR OTHER PACIFIC ISLANDER",
                  "WHITE",
                  "DECLINED TO STATE"),
    race_prob = c(0.05, 0.2, 0.2, 0.04, 0.5, 0.01),
    gender_list = c("MALE (INCLUDING TRANS MAN)",
                    "FEMALE (INCLUDING TRANS WOMAN)",
                    "NON-BINARY",
                    "OTHER",
                    "DECLINED TO STATE"),
    gender_prob = c(0.0001, 0.9996, 0.0001, 0.0001, 0.0001),
    ethnic_list = c("HISPANIC OR LATINO", "NON HISPANIC OR LATINO", "DECLINED TO STATE"),
    ethnic_prob = c(0.2, 0.7, 0.1),
    country_list = c("GBR",
                     "USA"),
    country_prob = c(0.5, 0.5),
    age_min = config_json$study$agemin,
    age_max = config_json$study$agemax,
```

sim_subjects generate subject identifiers, demographics, baseline characteristics, treatment allocation, and reference/treatment start dates. Key behaviors include:

- Sampling demographic attributes based on distributions in the configuration file
- Constructing formatted IDs that align easily to CDISC (subjid,usubjid)
- Deriving regions from country assignments
- Generating ISO 8601–formatted date fields
- Assigning a journey to each subject
- Creating rescreen instances by linking secondary SUBJECT IDs back to the original (PRSUBJID)

The output, subjects, becomes the parent dataset for all other modules.

3.2. Study Visit Dates (sim_dates)

sim_dates constructs the full schedule of study visits and key clinical dates for each simulated subject. Using visit definitions from the protocol and journey specific visit eligibility from vis_template, the function anchors each planned visit day to the subject's reference date (RFSTDTC). It derives treatment dates, end of treatment (EOT), death related dates, and end of study assessments. Journey logic determines which visits occur, how far each subject progresses through the schedule, and whether screening, rescreening, or follow-up applies. The resulting subject level date matrix (subject_dates) provides the temporal backbone for exposure, safety, and efficacy simulations.

Program 3: **sim_dates**

```r
sim_dates <- function() {

  ##############################################
  # Read in required datasets
  ##############################################
  subjects <- readRDS(paste0(outdir_master,"subjects.Rda"))

  ##############################################
  # Generate simulated study visits data set
  ##############################################
  all_visits <- cross_join(subjects, vis_template) |>
    mutate(
      STDTM = case_when(!is.na(RFSTDTC) ~ as.POSIXct(as.POSIXct(RFSTDTC,
                                          format = "%Y-%m-%dT%H:%M", tz = "UTC") + (VISITDY * 86400)),
                        is.na(RFSTDTC) ~  as.POSIXct(as.POSIXct(sample(
                          seq(from = as.POSIXct("2020-08-01"), to =  as.POSIXct("2025-04-01"), by = "hour"),
                          dplyr::n(),
                          replace = TRUE
                        ), format = "%Y-%m-%dT%H:%M", tz = "UTC") + (VISITDY * 86400))
                        ),
      STDTC = format_ISO8601(STDTM, precision = "ymd")
    ) |>
    filter((JOURNEY == 1 & JOURNEY1 == "Y") |
           (JOURNEY == 2 & JOURNEY2 == "Y") |
           (JOURNEY == 3 & JOURNEY3 == "Y") |
           (JOURNEY == 4 & JOURNEY4 == "Y") |
           (JOURNEY == 5 & JOURNEY5 == "Y") |
           (JOURNEY == 901 & JOURNEY901 == "Y") |
           (JOURNEY == 911 & JOURNEY911 == "Y") |
           (JOURNEY == 912 & JOURNEY912 == "Y") |
           (JOURNEY == 921 & JOURNEY921 == "Y") |
           (JOURNEY == 931 & JOURNEY931 == "Y") |
           (JOURNEY == 941 & JOURNEY941 == "Y") |
           (JOURNEY == 942 & JOURNEY942 == "Y") |
           (JOURNEY == 951 & JOURNEY951 == "Y") |
           (JOURNEY == 952 & JOURNEY952 == "Y") |
           (JOURNEY == 961 & JOURNEY961 == "Y") |
           (JOURNEY == 971 & JOURNEY971 == "Y") |
           (JOURNEY == 972 & JOURNEY972 == "Y") |
           (JOURNEY == 981 & JOURNEY981 == "Y") |
           (JOURNEY == 982 & JOURNEY982 == "Y")
    )
```

sim_dates uses:

- Reference start date (RFSTDTC)
- Visit offsets and windows from vis_template
- Journey based visit eligibility
- EOT and death logic
- Survival follow-up timing patterns
- to generate:
- Cycle visit dates
- Screening/rescreen visit dates

- End of treatment
- Survival follow-up assessments
- End of study status and reason

The output subject_dates is a master timeline for each subject.

3.3. Exposure Data (sim_exposure)

sim_exposure simulates infusion level treatment exposure by generating dosing start and end times, cycle based dosing dates, dose adjustments, interruptions, infusion reactions, and other protocol driven administration patterns. Exposure records reflect real-world clinical scenarios such as dose reductions, skipped doses, and infusion related interruptions. These data inform AE timing and mechanistic links between treatment and safety outcomes.

Program 4: sim_exposure

```
sim_exposure <- function() {

  exposure_ip <- readRDS(paste0(outdir_master,"subject_visit_dates.Rda")) |>
    left_join(subjects |> select(SUBJECT, TRTGROUP, WEIGHTBL), by = "SUBJECT") |>
    left_join(subject_dates |> select(SUBJECT, DTHDTM) |> filter(!is.na(DTHDTM)), by = "SUBJECT") |>
    filter(str_starts(VISITCD, "C") & str_ends(VISITCD, "D1")) |>
    rowwise() |>
    mutate(ECCAT = fcase(VISITCD == "C1D1", "FIRST",
                         VISITCD != "C1D1", "SUBSEQUENT"),
           ECTRT = TRTGROUP,
           ECOCCUR  = "Yes",
           ECPLANDOS = fcase(
             TRTGROUP == "Apple", fifelse(WEIGHTBL < 70, 1800, 2400),
             TRTGROUP == "Orange", fifelse(ECCAT == "FIRST", (8 * WEIGHTBL), (6 * WEIGHTBL))
           ),
           ECPLANDOSU = "mg",

           ECPLANVOL = round(fcase(
             TRTGROUP == "Apple", ECPLANDOS / 50,
             TRTGROUP == "Orange", ECPLANDOS / 21
           )),
           ECPLANVOLU = "mL",
           ECSDTM = STDTM,
           ECEDTM = fcase(STDTM+9600 > DTHDTM & !is.na(DTHDTM) & !is.na(STDTM), DTHDTM,
                          STDTM+9600 > EOT & !is.na(EOT) & !is.na(STDTM) & is.na(DTHDTM), EOT,
                          STDTM+9600 <= DTHDTM & !is.na(DTHDTM) & !is.na(STDTM), STDTM+9600,
                          default = STDTM+9600
           ),
           ECINFINT  = fcase(!is.na(ECSDTM) & !is.na(ECPLANVOL) & ECPLANVOL > 0,
                             sample(c("N","Y"), n(), replace = TRUE, prob = c(0.9, 0.1))),
           ECINFINTREAS = fcase(ECINFINT == "Y", sample(c("Infusion Reaction","Other Adverse Event", "Other"),
                                                        n(), replace = TRUE, prob = c(0.7, 0.2, 0.1)),
                                default = NA_character_
           ),
           ECINFINTREASO = fifelse(ECINFINTREAS == "Other", "xxxxxxxxxx", NA_character_),
           ITRP_AEREL = fcase(ECINFINT == "Y", "Y"),
           ITRP_AEACN = fcase(ECINFINT == "Y", "DRUG INTERRUPTED"),
           ECINFCOMPL = fcase(ECINFINT == "Y", sample(c("N","Y"), n(), replace = TRUE)),
           ECINFCOMPREAS = fcase(ECINFCOMPL == "N", ECINFINTREAS,
                                 default = NA_character_
                                 ),
```

Exposure master data program includes exposure simulation that determines:

- Dosing dates and times
- Interruptions
- Dose reductions
- Infusion related flags (IRR, Other AE, ECADJ)

These exposure records become direct inputs for AE derivation.

3.4. Adverse Event Generation (sim_ae)

sim_ae generates a comprehensive set of adverse events by integrating subject journeys, treatment exposure, and AE specific sampling distributions. The module produces adverse event data sampled from prior studies in the same compound/indication. adverse events of special interest (e.g., infusion reactions, left ventricular dysfunction, pulmonary embolism), and AEs associated with protocol defined triggers such as dose adjustments, treatment interruptions, AE driven discontinuations, and death. Onset and end times are aligned with treatment periods and exposure records, and severity, seriousness, outcomes, and regulatory analysis flags are derived according to CDISC rules. The final AE dataset (adverse_events) is fully synchronized with subject dates and exposure history.

Program 5: sim_ae

```r
sim_ae <- function() {

  ## Read in subject dates
  subject_dates <- readRDS(paste0(outdir_master,"subject_dates.Rda"))

  ## Read in exposure
  master_ex <- readRDS(paste0(outdir_master,"exposure.Rda"))

  ## Generate AE records
  ae1 <- gen_ae(subjnum = number_of_subjects,
                samptype = "ONCOLOGY",
                aenumseq = c(4, 8, 8, 9, 9, 9, 10, 12, 25)) |>
    filter(AEDECOD != "Left ventricular dysfunction" & AEDECOD != "Infusion related reaction" &
             AEDECOD !="Pulmonary embolism")

  lvef <- gen_ae(subjnum = number_of_subjects,
                 samptype = "ONCOLOGY",
                 aenumseq = c(1, 3),
                 sampling_data = get_sampling_data("ae") |>
                   dplyr::filter(AEDECOD == "Left ventricular dysfunction")) |>
    slice_sample(n = sample(50:100, 1))

  nipt <- gen_ae(subjnum = number_of_subjects,
                 samptype = "ONCOLOGY",
                 aenumseq = c(1, 3),
                 sampling_data = get_sampling_data("ae") |>
                   dplyr::filter(AEDECOD == "Pulmonary embolism")) |>
    slice_sample(n = sample(50:100, 1))

  ## Generate IRR AE records
  ex_irr <- master_ex |>
    filter(ECINFINTREAS == "Infusion Reaction")

  ex_max_subj <- max(ex_irr$SUBJECT)

  irr <- gen_ae(subjnum = number_of_subjects,
                samptype = "ONCOLOGY",
                aenumseq = c(1),
                sampling_data = get_sampling_data("ae") |>
                  dplyr::filter(AEDECOD == "Infusion related reaction")) |>
    select(-SUBJECT) |>
    distinct(AETERM, AESEV, AESER, AEREL, AEOUT, AEACN, AEACNOTH, .keep_all = TRUE) |>
    mutate(count = ex_max_subj) |>
    uncount(count) |>
```

The sim_ae module synthesizes AE data by integrating:

- Exposure events
- Subject dates
- AE sampling distributions
- The logic produces realistic patterns of:
- Background indication-appropriate AEs
- AESIs (LVEF, pulmonary embolism, IRR)
- AEs associated with dose adjustments

- AEs responsible for treatment interruption
- AEs responsible for permanent discontinuation
- Death related AEs
- A rich set of CDISC aligned analysis variables is derived, including:
- Severity and grade
- Seriousness and seriousness components
- Action taken
- Relationship
- Outcome
- Onset/end datetimes
- Study day and duration
- Ongoing flags
- AESI flags

The final dataset, adverse_events, synchronizes temporally with exposure and subject timeline datasets.

3.5. Master Vital Signs Simulation

Unlike adverse events, which are event-driven and irregular, vital signs represent repeated, timepoint-driven continuous measurements collected at scheduled visits and sometimes multiple timepoints within a visit (e.g., predose/postdose). To demonstrate simulation across different clinical data types, we generated a master vital signs dataset using a metadata-driven approach. Timepoint definitions and visit rules were extracted from the protocol configuration (JSON) and combined with endpoint definitions of value spread, range and location from a standards library. A simulation engine based on the SIMSTUDY R package then generated baseline and longitudinal trends for each parameter (e.g., SYSBP, DIABP, PULSE, TEMP, HEIGHT, WEIGHT) across configured timepoints. The simulated results were merged with subject visit dates to ensure that only visits occurring in each subject's journey were retained, and date-time jitter was applied to distinguish predose and postdose collections. To mimic real-world variability in EDC data, controlled unit variability (e.g., inches vs cm, pounds vs kg, Fahrenheit vs Celsius) and "NOT DONE" patterns were introduced through targeted missingness and visit-level blanks. The resulting output was stored as a master VS file, which subsequently served as the source for CRF-like raw outputs and downstream SDTM mapping.

Program 6: sim_vital_signs

```r
sim_vital_signs <- function() {

## (1) Set-up time point metadata----
# Retrieve config data
timepoints_df <- config_json$timepoints$vs

# create the list of timepoints according to parameter
height <- unique(timepoints_df$timepoint[grep("screening", timepoints_df$timepoint, ignore.case = TRUE)])
weight <- unique(timepoints_df$timepoint[grep("screening|day 1 - predose|end of treatment",
                                              timepoints_df$timepoint, ignore.case = TRUE)])


# Assign metadata
timepointnum <- nrow(timepoints_df)
timepoint_vct <- paste0(c("TPT"), sprintf("%02d", 1:nrow(timepoints_df)))

## (2) Set-up trend metadata----
trend_vct <- rep(c("TPT01"), each=nrow(timepoints_df))
trend_vct[1] <- "NA"

## (3) Set-up end point metadata ----
# Retrieve config data
endpoints_vct <- unlist(config_json$endpoints$vs$id)

# Retrieve endpoint metadata from metadata library
endpoints_df <- read.csv(paste0(metadir,"endpoints_neo1.csv"),header=TRUE, sep = "|") |>
  dplyr::filter(endpointid %in% endpoints_vct) |>
  dplyr::arrange(by_group=endpointid) |>
  dplyr::mutate(formula = as.character(formula)) |>
  dplyr::mutate(variance = as.character(variance)) |>
  dplyr::mutate(distribution = case_when(distribution %in% c("categorical") ~ "normal", TRUE ~ distribution )) |>
  ## Assign trend generic template and store as a list
  dplyr::mutate(trend_list = list(trend_vct))

## (4) Create metadata process controlling variables, vectors and lists ----
#md <- fct_metadata()
md <- fct_metadata(inds=endpoints_df)

```

Other simulation modules follow the same structured approach: they use the configuration file, subject level timelines, and exposure records to produce domain specific raw datasets (e.g., labs, PK/PD, vital signs, efficacy endpoints). Each domain is generated with realistic between subject variability, visit based timing, and journey specific trimming to ensure alignment with protocol defined study conduct.

4. Raw Dataset Generation and Downstream Outputs

Master subject level datasets are first created to establish consistent subject identifiers, treatment assignments, visit schedules, and key study milestones. These master datasets served as the foundation for generating raw, CDISC like datasets representing core clinical domains.

Using protocol defined schedules, CRF structures, and domain specific rules, synthetic datasets were created by transforming the master data, for Adverse Events (AE), Exposure (EX), Laboratory (LB), Vital Signs (VS), and predefined efficacy endpoints. Each domain incorporated realistic record counts, visit timing, controlled terminology, and interdomain relationships to mirror expected clinical data behavior. This structured approach ensured internal consistency across domains and supported downstream SDTM mapping. The resulting raw datasets enable early development and validation of SDTM (which in turn can enable ADaM, and TFL workflows), significantly accelerating programming readiness prior to the availability of actual clinical data.

DM(Demographics) datasets are created using the master subject-level simulation file as per CRF. Derived values are mapped to CRF-standard encodings to ensure compatibility with downstream SDTM mapping. The resulting simulated raw.dm dataset mimics an EDC-like “Demographics” form output and serves as an input to SDTM.DM creation and validation.

AE datasets are generated from a master subject-level simulation layer and CRF-driven metadata. Domain-specific rules are applied to populate seriousness criteria, relationship, action taken, and outcome fields while maintaining internal consistency and controlled terminology. The final simulated raw.ae output is written in a CRF-like layout suitable for downstream simulated SDTM.AE mapping

Exposure records are then filtered to protocol-relevant scenarios (e.g., Cycle 1 Day 1 first IV dose) and enriched with CRF-style context fields (Instance, Folder, Page) to mimic EDC raw outputs. Key exposure attributes such as planned dose, dose volume, actual volume infused, start/end date-time, and infusion interruption/completion indicators are derived from the master exposure records and mapped into CRF-like variables, including standardized *_STD fields and character-based *_RAW representations. The resulting Simulated RAW.EX exposure dataset is exported in a structure suitable for downstream Simulated SDTM.EX mapping.

The simulated master data is designed to be flexible and can be converted directly into SDTM, ADaM, or TFL outputs depending on downstream needs, which remains an area for future development. In this work, a structured **master → raw → SDTM** pipeline was intentionally followed to reflect real world clinical data workflows. This approach ensured alignment with expected raw data structures and supported realistic mapping and validation. By mirroring production pipelines, the simulated framework enables a smooth transition when real clinical data becomes available.

Program 7: raw DM dataset

```r
create_raw_dm <- function() {

  subjects <- readRDS(paste0(outdir_master,"subject_dates.Rda"))

  raw <- subjects |>
    mutate(
      SITE=substr(SITEID, 2, nchar(SITEID)),
      CNTR_STE=gsub(" ", "", paste(COUNTRY, SITE)),
      SUB = as.character(sprintf("%04d",SUBJECT)),
      SUBJECT=paste(CNTR_STE,SUB, sep="-"),
      INSTANCENAME = "Screening",
      FOLDER = "SCRN",
      FOLDERNAME = "Screening",
      DATAPAGENAME = "Demographics",
      RECORDPOSITION = 0,
      BRTHDATY = NA,
      BRTHDATY_RAW =  as.character(as.numeric(format(Sys.Date(), "%Y")) - AGE),
      BRTHDATY_YYYY = as.numeric(format(Sys.Date(), "%Y")) - AGE,
      BRTHDATY_MM = NA,
      BRTHDATY_DD = NA,
      AGE = AGE,
      AGE_RAW = as.character(AGE),
      AGEU = "Years",
      AGEU_STD = "4",
      SEX_STD = SEX,
      SEX = ASEX,
      GENIDENT = case_when(ASEX == "Female" ~ "Female (including Trans Woman)",
                           ASEX == "Male" ~ "Male (including Trans Man)"),
      GENIDENT_STD = case_when(ASEX == "Female" ~ "2",
                               ASEX == "Male" ~ "1"),
      CHILDPOT_RPORRES = case_when(
        ASEX == "Female" ~ "Yes",
        ASEX == "Male" ~ "No"
        ),
      CHILDPOT_RPORRES_STD = case_when(
        ASEX == "Female" ~ "Y",
        ASEX == "Male" ~ "N"
      ),
      ETHNIC = AETHNIC,
      ETHNIC_STD = as.character(AETHNICN),
      RACEAMI  = if_else(RACE == "AMERICAN INDIAN OR ALASKA NATIVE", 1, 0),
      RACEAMI_RAW = as.character(RACEAMI),
```

Program 8: raw AE dataset

```
create_raw_ae <- function() {

  master_ae <- readRDS(paste0(outdir_master,"adverse_events.Rda"))

  master_ex <- readRDS(paste0(outdir_master,"exposure.Rda")) |>
    select(SUBJECT, ECTRT, VISITCD, ECSDTM, ECEDTM, ECACTVOL) |>
    filter(ECACTVOL > 0 & !is.na(ECACTVOL))

  ex_first <- master_ex |>
    arrange(SUBJECT, ECSDTM, VISITCD) |>
    group_by(SUBJECT) |>
    slice_head(n = 1) |>
    mutate(C1D1 = ECSDTM) |>
    select(SUBJECT, ECTRT, C1D1, VISITCD)

  ex_last <- master_ex |>
    filter(ECTRT == "Apple") |>
    arrange(SUBJECT, desc(ECSDTM), desc(VISITCD)) |>
    group_by(SUBJECT) |>
    slice_head(n = 1) |>
    mutate(EXEN = ECEDTM,
           VISEN = VISITCD) |>
    select(SUBJECT, ECTRT, EXEN, VISEN)

  ex <- left_join(ex_first, ex_last, by = c("SUBJECT", "ECTRT")) |>
    mutate(TRTGROUP = ECTRT)


  raw <- master_ae |>
    left_join(ex, by = c("SUBJECT", "TRTGROUP")) |>
    mutate(
      SUBJECT = paste(sep="-", paste0(COUNTRY,sprintf("%03d", as.integer(SITEID))), sprintf("%04d", SUBJECT)),
      INSTANCENAME = "Adverse Events",
      FOLDER = "AE",
      FOLDERNAME = "Adverse Events",
```

Program 9: raw exposure dataset program

```
create_raw_ec_iv1 <- function() {

  SUBJECTS <- readRDS(paste0(outdir_master,"subjects.Rda"))
  SUBJ <- SUBJECTS |>
    mutate(
      SITE=substr(SITEID, 2, nchar(SITEID)),
      CNTR_STE=gsub(" ", "", paste(COUNTRY, SITE)),
      SUB = as.character(sprintf("%04d",SUBJECT)),
      NEW_SUBJECT=paste(CNTR_STE,SUB, sep="-")
    ) |>
    select(
      NEW_SUBJECT, SUBJECT
    )


  exposure <- readRDS(paste0(outdir_master,"exposure.Rda"))
  exposure  <- dplyr::left_join(exposure, SUBJ, by = "SUBJECT")

  raw <- exposure |>
    filter(VISITCD == "C1D1" & ECTRT=="Apple" | ECTRT=="Orange") |>
    mutate(SUBJECT=NEW_SUBJECT,
      INSTANCENAME = "Cycle 1 Day 1 (1)",
      FOLDER = "C1D1",
      FOLDERNAME = "Cycle 1 Day 1",
      DATAPAGENAME = "First Dose",
      ECOCCUR_STD = case_when(ECOCCUR == "Yes" ~ "Y",
                              ECOCCUR == "No" ~ "N"),
      ECTIDOSE = as.character(ECPLANDOS),
      ECTIDOSEU = ECPLANDOSU,
      ECTIDOSEU_STD = "13",

      ECDOSVOL = ECPLANVOL,
      ECDOSVOL_RAW= as.character(ECPLANVOL),

      ECSTDAT = ECSDTM,
      ECSTTIM = format(ECSDTM,"%H:%M"),
      ECENDAT = ECEDTM,
      ECENTIM = format(ECEDTM,"%H:%M"),
```

RESULTS

Evaluation of Simulation Integrity

The integrity of the simulated datasets is evaluated through a combination of structural, logical, and distributional checks. All generated datasets conform to expected SDTM domain structures. Cross domain consistency is verified by confirming alignment between subject level data and dependent domains such as AE, EX, LB, and VS, ensuring that all records are linked to valid subjects and visits.

Logical checks confirm that:

- protocol-specified rules are upheld, including correct visit sequencing, appropriate treatment-exposure timing, and proper alignment of adverse-event onset with dosing.
- Distributional assessments verified that simulated demographics, laboratory measures, and adverse-event patterns fell within clinically plausible ranges.

Together, these evaluations demonstrate that the simulated dataset accurately reproduced expected clinical-trial behavior while remaining entirely synthetic and free of sensitive information.

Ultimately, success is defined by how seamlessly the statistical-programming pipeline transitions from simulated data to real clinical-trial data.

Benefits of Data Simulation

The use of simulated SDTM compliant data enabled early and uninterrupted development of SDTM, ADaM, and TFL workflows prior to the availability of real clinical data. Programming teams were able to validate mapping logic, test derivation rules, and develop reporting output without dependency on enrollment progress. This approach facilitated early identification of edge case scenarios, such as missing data patterns and atypical visit sequences, which improved code robustness and reduced downstream rework.

Additionally, simulated datasets support automation development, quality control activities, and team alignment across programming and analysis functions. Overall, simulation significantly accelerated development timelines, improved validation of readiness, and increased confidence in the end-to-end clinical programming pipeline once actual study data became available.

Limitations of Data Simulation

Data simulation provides significant advantages; it also has important limitations. Simulated data, by definition, cannot fully capture unpredictable patterns, protocol deviations, and data quality issues observed in real clinical trials. Rare safety events, complex missingness mechanisms, and site-specific data entry behaviors may be underrepresented or absent. Additionally, simulation quality is highly dependent on the accuracy and completeness of the protocol, CRF, and metadata specifications; incorrect assumptions at this stage may propagate through downstream outputs. Simulated datasets are therefore suitable for development and validation but cannot replace real data for statistical inference or regulatory decision making. Care must be taken to clearly distinguish simulated results from analyses based on actual clinical data.

There are also clear opportunities for the configuration of the data simulation to be automated, including the use of Artificial Intelligence, for example to identify a more complete matrix of patient journeys and to extract the clinical trial protocol into the simulation configuration. This could further accelerate the setup time of the data simulation and subsequently the statistical programming pipeline.

CONCLUSION

SDTM-driven simulation provides a powerful way to accelerate development, validate pipelines, and ensure robust and compliant workflows before real clinical data becomes available. As automation and AI become integral to biometrics, synthetic data will play a critical role in modern clinical trial programming

ACKNOWLEDGMENTS

We extend our appreciation to the creators and maintainers of the Simstudy R package (https://kgoldfeld.github.io/simstudy/index.html), whose tools enabled the simulation and methodological exploration underpinning this work. We would like to express our gratitude to Matt Travell, whose original conceptual framework served as the foundation for this work. We also acknowledge the statistical programming Data Science team at Jazz Pharmaceuticals for sharing best practices, perspectives, and constructive feedback that strengthened the overall framework and our collaborators at Cytel FSP. We are further grateful to the broader, data standards, and statistical programming communities for sharing best practices, perspectives, and constructive feedback that strengthened the overall framework.

REFERENCE

Goldfeld, K., & WujciakJens, J. (2020). *simstudy: Illuminating research methods through data generation.* **Journal of Open-Source Software**, 5(54), 2763. https://doi.org/10.21105/joss.02763 [kgoldfeld.github.io]

CONTACT INFORMATION

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

Jonathan Henshaw,
Email: jonathan.henshaw@jazzpharma.com

Sangeeta Shabadi,
Email: sangeeta.shabadi@gmail.com

Nitesh Patil,
Email: patilnitesh115@gmail.com