Paper SM10

Precision Timing - Implementing Epoch Calculations with R for Clinical Data Integrity

Sheik Ibrahim, Navitas Life Sciences, India
Ram Mohan Konda, Navitas Life Sciences, India

ABSTRACT

Epoch calculation is critical for defining treatment periods in clinical trials. Using R programming, we developed a pipeline to compute epochs in a cardiovascular study tracking medication interval. The workflow utilized a flowchart-driven logic to classify start and end dates accurately. Exceptional cases—such as overlapping visits and missing timestamps—were handled with custom R functions to ensure data integrity. Validation against manual review and audits showed over 99% concordance. Challenges included managing inconsistent timestamp formats and complex patient visit patterns. Future directions involve incorporating machine learning to predict missing data and developing interactive dashboards for real-time quality checks. This R-based method improves reproducibility and supports precise temporal data segmentation essential for robust efficacy and safety analyses.

INTRODUCTION

The R-based m_epoch() framework automates clinical trial epoch assignments by standardizing partial ISO-8601 dates and resolving complex period overlaps. This pipeline ensures high-quality, regulatory-grade data integrity while significantly reducing manual effort and improving reproducibility across cardiovascular datasets.

OBJECTIVES

- To develop an R-based framework for precise epoch assignment of clinical observations, ensuring consistency across regulatory-grade datasets.
- To harmonize varied and incomplete date formats into a standardized temporal structure for robust safety and efficacy analysis.

KEY CHALLENGES

- **Partial ISO-8601 Data:** Reconciling varying granularities (YYYY/MM) to maintain chronological accuracy.
- **Multi-Epoch Complexity:** Managing subject transitions across multiple study phases and medication intervals.
- **Boundary & Ordering Ambiguity:** Resolving incorrect epoch boundaries to ensure data traceability.

WHY AUTOMATION?

In a regulatory environment where precision is paramount, transitioning from manual epoch assignment to an automated R-based framework is essential to eliminate human error and ensure the scalability of complex clinical datasets.

Clinical Data Integrity: Ensures high-quality, error-free epoch attribution across cardiovascular datasets for submission readiness.

Reproducibility & Consistency: Standardizes the R-based pipeline to produce identical results across different study environments and programmers.

Complex Logic Management: Leverages automated algorithms to handle intricate visit patterns and medication intervals that manual review cannot scale.

Time Efficiency: Reduces manual data cleaning efforts, accelerating the delivery of regulatory-grade SDTM and ADaM datasets.

Compliance & Audits: Creates a transparent, traceable audit trail that meets rigorous regulatory standards for data transparency.

Real-time Monitoring: Facilitates immediate identification of epoch misalignment through interactive dashboards and automated quality checks.

CASE STUDY OVERVIEW – PRECISION TIMING WITH R IN CLINICAL TRIALS

Precise epoch classification is vital for trial safety, but manual methods lack scalability and accuracy. We developed a validated R pipeline to automate treatment and follow-up period assignments. The workflow leverages dplyr, lubridate, and stringr to transform complex visit data. A custom m_epoch() function resolves partial ISO-8601 dates. The logic successfully mitigates overlapping visit windows and boundary inconsistencies across subjects. This automation ensures regulatory-grade consistency while significantly reducing manual QC efforts.

OVERVIEW OF THE M_EPOCH ALGORITHM

Validate Inputs: Verify that all required data frames and variables exist and meet the necessary data types.
Validate Subject Element Metadata: Ensure the reference epoch data is complete, unique, and free of logical gaps.
Process for partial dates: Standardize or impute incomplete ISO-8601 strings to allow for accurate chronological comparisons.
Join in_data and se_in: Merge the clinical observations with the subject element metadata using the unique subject identifier.
Sort based on the requirements: Organize the merged records by subject, date, and sequence to resolve any temporal overlaps.
Pick the right needed EPOCH: Apply window-matching logic to assign the specific epoch label to each clinical observation.

COMPREHENSIVE LIST OF INPUT PARAMETERS

The m_epoch() function uses strict input parameters to manage data, logic, and debugging while maintaining clinical data integrity.

INPUT PARAMETER	TEST#	TEST	EXPECTED OUTCOME
in_data	1.01	Supply in_data name for analysis	Analysis performed using the specified in_data
	1.02	No in_data name is supplied	Relevant error message is written to log, and function stops
se_in	2.01	Supply se_in name for analysis	Analysis performed using the specified se_in
	2.02	No se_in name is supplied	Relevant error message is written to log, and function stops
dtcvar	3.01	Supply dtcvar name for analysis	Analysis performed using the specified dtcvar
	3.02	No dtcvar name is supplied	Relevant error message is written to log, and function stops
opt	4.01	No opt name is supplied	Default is "TRT_LATE"
	4.02	opt = "EARLY"	If multiple elements for the same date, it picks the earliest element
	4.03	opt = "LATE"	If multiple elements for the same date, it picks the latest element
	4.04	opt = "TRT_EARLY"	If multiple treatment elements for the same date, it picks the earliest treatment element
	4.05	opt = "TRT_LATE"	If multiple treatment elements for the same date, it picks the latest treatment element
partial	5.01	partial = "TRUE"	EPOCH is derived for partial dates
	5.02	partial = "FALSE"	EPOCH is not derived for partial dates
	5.03	No partial is supplied	Relevant error message is written to log, and function stops
debug	6.01	debug = "TRUE"	Intermediate datasets are kept for debugging
	6.02	debug = "FALSE"	Intermediate datasets are removed
	6.03	No debug is supplied	Default is "FALSE"

INPUT DATASETS FOR EPOCH CALCULATION

Successful epoch assignment relies on structured input data consisting of clinical observation dates and standardized subject element metadata.

in_data: Dates may be partial (YYYY or YYYY-MM) and will be standardized.

USUBJID	PRSTDTC	PRSEQ
01	2020-01-01	1
01	2020-01-05	2
01	2020-01-14	3
01	2020-01-18T11:30	4
02	2019-06	1
03	2018	1

SE domain (se_in): SE dataset consists of start (SESTDTC) and end (SEENDTC) dates for each element per subject.

USUBJID	TAETORD	EPOCH	SESTDTC	SEENDTC
01	1	SCREENING	2019-12-24	2020-01-01
01	2	TREATMENT PERIOD 1	2020-01-01	2020-01-05
01	3	TREATMENT PERIOD 2	2020-01-05	2020-01-15
01	4	FOLLOW-UP	2020-01-16	2020-01-25
02	1	SCREENING	2019-06-01	2019-06-30
02	2	TREATMENT PERIOD 1	2019-06-30	2019-08-21
03	1	SCREENING	2018-01-01	2018-12-31

OUTPUT DATASET AFTER EPOCH CALCULATION

EPOCH is mapped based on user's needs using OPT parameter,
m_epoch (in_data = IN_DATA, se_in = SE, dtcvar = "PRSTDTC", opt = "TRT_LATE", partial = TRUE,debug = FALSE)

USUBJID	PRSTDTC	PRSEQ	EPOCH
01	2020-01-01	1	TREATMENT PERIOD 1
01	2020-01-05	2	TREATMENT PERIOD 2
01	2020-01-14	3	TREATMENT PERIOD 2
01	2020-01-18T11:30	4	FOLLOW-UP
02	2019-06	1	TREATMENT PERIOD 1
03	2018	1	SCREENING

SUBJECT PROCEDURE SCHEDULE AND STUDY EPOCH MAPPING

The following table demonstrates the final output generated by the m_epoch() function, where clinical observations are successfully mapped to their respective study phases by applying the "TRT_LATE" priority logic to resolve temporal boundaries.

USUBJID	PRSTDTC	PRSEQ	EPOCH (OPT = "EARLY")	EPOCH (OPT = LATE")	EPOCH (OPT = "TRT_EARLY")	EPOCH (OPT = "TRT_LATE")
01	2020-01-01	1	SCREENING	TREATMENT PERIOD 1	TREATMENT PERIOD 1	TREATMENT PERIOD 1
01	2020-01-05	2	TREATMENT PERIOD 1		TREATMENT PERIOD 1	TREATMENT PERIOD 2
01	2020-01-14	3	TREATMENT PERIOD 2	TREATMENT PERIOD 2	TREATMENT PERIOD 2	TREATMENT PERIOD 2
01	2020-01-18T11:30	4	FOLLOW-UP	FOLLOW-UP	FOLLOW-UP	FOLLOW-UP
02	2019-06	1	SCREENING	TREATMENT PERIOD 1	TREATMENT PERIOD 1	TREATMENT PERIOD 1
03	2018	1	SCREENING	SCREENING	SCREENING	SCREENING

EXCEPTIONAL HANDLING

To address the complexities of clinical data, the m_epoch() framework incorporates robust exceptional handling to ensure every observation is accounted for and assigned with high integrity.

Handling Partial Dates: Standardizes incomplete ISO-8601 strings (e.g., YYYY or YYYY-MM) through imputation or specific logic to allow for accurate chronological comparison.

Overlapping Epoch Ranges: Resolves conflicts where a single date falls into multiple study phases by using the OPT parameter to prioritize specific treatment or timing windows.

Missing Start/End Dates in Epoch Dataset: Identifies gaps in the Subject Element (SE) metadata and applies fallback logic or error flags to prevent incorrect period attribution.

Invalid or Unexpected Date Lengths: Validates the structure of date-time strings to ensure they conform to expected formats, filtering out or flagging malformed data before processing.

Invalid 'opt' Parameter Values: Implements strict input validation that stops the function and logs a detailed error message if the user provides an unsupported sorting option.

Zero Observation or Empty Datasets: Detects empty input files early in the pipeline to avoid script crashes and provides a clean exit or notification for the user.

No Matching Epoch Found: Handles cases where an observation date falls outside of all defined study windows by

assigning a null value or a "NOT FOUND" label for manual review.

INPUT DATASET VALIDATION AND INTEGRITY CHECKS

This initial phase ensures that both the clinical observation and epoch reference datasets are present, correctly formatted as data frames, and contain the essential unique subject identifiers.

```
m_epoch <- function(in_data = NULL, se_in = NULL,
                    dtcvar, opt = "TRT_LATE", partial = TRUE, debug = FALSE) {
  if (missing(in_data)) {
    stop("ERROR: No input dataset (in_data) provided.") }
  if (!is.data.frame(in_data)) {
    stop("ERROR: 'in_data' is not a data frame.") }
  if (missing(se_in)) {
    stop("ERROR: No input dataset (se_in) provided.") }
  if (!is.data.frame(se_in)) {
    stop("ERROR: 'se_in' is not a data frame.") }
  if (!"USUBJID" %in% names(in_data)) {
    stop("ERROR: Input dataset does not contain required variable 'USUBJID'.") }
  if (!dtcvar %in% names(in_data)) {
    stop(paste0("ERROR: Input dataset does not contain the date variable '", dtcvar,
"'.") }
```

PARAMETER AND STRUCTURAL VALIDATION OF STUDY EPOCH DATA

The function verifies that the subject element metadata includes all required regulatory variables and validates the user-defined sorting priority (opt) to prevent logic errors during assignment.

```
 required_se_vars <- c("USUBJID", "TAETORD", "EPOCH","SESTDTC", "SEENDTC")
  missing_vars <- setdiff(required_se_vars, names(se_in))
  if (length(missing_vars) > 0) {
    warning(paste("WARNING: Study epoch dataset missing variable(s):",
paste(missing_vars, collapse = ", "))) }
  valid_opts <- c("EARLY", "LATE", "TRT_EARLY","TRT_LATE")
  opt <- toupper(opt)
  if (!(opt %in%  valid_opts)) {
    warning(paste0("Warning: 'opt' value '", opt, "' is invalid. Using default
'TRT_LATE'."))
    opt <- "TRT_LATE" }
  if (all(is.na(in_data[[dtcvar]]))) {
    stop(paste0("ERROR: Date variable '", dtcvar, "' contains only missing values."))
}
  if (nrow(in_data) == 0) stop("ERROR: Input dataset has zero observations.")
  if (nrow(se_in) == 0) stop("ERROR: Study epoch dataset has zero observations.")
```

INPUT DATA TEMPORAL STANDARDIZATION AND FILTERING

This step extracts the core date components from ISO-8601 strings and establishes filtering flags to handle or exclude records based on the user's preference for partial date inclusion.

```
  in_data <- in_data %>%
    mutate(
      obs_no = row_number(),
      # Extract the date part (YYYY-MM-DD or partial) before 'T'
      dtc_date_part = word(as.character(.data[[dtcvar]]), 1, sep = "T"),
      # Store length for filtering later
      dtc_len = nchar(dtc_date_part),
      # Flag to keep only full dates if partial=FALSE
      keep_dtc = if(partial) TRUE else (dtc_len == 10))
  all_dtc_lengths <- unique(in_data$dtc_len)
  se_in <- se_in %>%
    mutate(
```

```
      SESTDTC_PART = word(SESTDTC, 1, sep = "T"),
      SEENDTC_PART = word(SEENDTC, 1, sep = "T"))
```

TEMPORAL MATCHING AND PARTIAL DATE RANGE ALIGNMENT

The algorithm performs a complex join between observations and study phases, applying conditional logic to align full or partial dates (YYYY or YYYY-MM) within the appropriate start and end boundaries.

```
 in_data_filtered <- in_data %>%
    filter(keep_dtc)
   matched <- in_data_filtered %>%
    left_join(se_in, by = "USUBJID", relationship = "many-to-many") %>%
     rowwise() %>%
    filter(!is.na(dtc_date_part) & !is.na(SESTDTC_PART) & !is.na(SEENDTC_PART) &
     ( (dtc_len == 10 & (dtc_date_part >= SESTDTC_PART) & (dtc_date_part <=
SEENDTC_PART)) |
     (partial & dtc_len == 7 &
                    (dtc_date_part >= str_sub(SESTDTC_PART, 1, 7)) &
                    (dtc_date_part <= str_sub(SEENDTC_PART, 1, 7))) |


  (partial & dtc_len == 4 &
                    (dtc_date_part >= str_sub(SESTDTC_PART, 1, 4)) &
                    (dtc_date_part <= str_sub(SEENDTC_PART, 1, 4))))
    ) %>%
    ungroup() %>%
    select(-starts_with("DTC_"), -starts_with("SEST_"), -starts_with("SEEN_"))
```

POST-MATCH RECONCILIATION, TIE-BREAKING, AND FINAL EPOCH ASSIGNMENT

To resolve instances where an observation overlaps multiple phases, the pipeline applies a hierarchical sorting strategy—prioritizing treatment periods and specific timing windows—to isolate a single, definitive epoch.

```
 if (nrow(matched) == 0) {
    warning("WARNING: No matching records found in study epoch join.") }
  matched <-matched %>% mutate(trt_ord = ifelse( str_detect(toupper(EPOCH),
"TREATMENT"), 0, 1))
  if (opt =="EARLY") {
    matched <- matched %>% arrange(obs_no, TAETORD)
  } else if (opt == "LATE") {
    matched <- matched %>% arrange(obs_no,desc(TAETORD))
  } else if (opt == "TRT_EARLY") {
    matched <-matched %>% arrange(obs_no, trt_ord,TAETORD)
  } else if (opt =="TRT_LATE") {
    matched <- matched %>% arrange(obs_no,trt_ord, desc(TAETORD)) }
  final_matched <-matched %>%
    distinct(obs_no,.keep_all = TRUE) %>%
    select(obs_no,EPOCH_NEW = EPOCH)
  final <- in_data %>%left_join(final_matched, by = "obs_no") %>%
    # Rename EPOCH_NEW to EPOCH (overwriting any existing EPOCH)
    mutate(EPOCH = EPOCH_NEW) %>%
    # Select original columns + new EPOCH, removing all temp columns
    select(-dtc_date_part, -dtc_len, -obs_no, -keep_dtc, -EPOCH_NEW)
```

POST-PROCESSING VALIDATION, DIAGNOSTIC REPORTING, AND FINAL OUTPUT

The final stage flags any remaining date irregularities, provides optional transparency through a global debug mode, and delivers a cleaned dataset ready for regulatory submission.

```
invalid_len <-all_dtc_lengths[!(all_dtc_lengths %in% c(4, 7, 10))]
  if (length(invalid_len) > 0) {
```

```
    warning("Warning: Some date variable values have unexpected lengths (expected 4,
7, or 10 characters).") }
  if (debug) {
    final <<- final
    matched <<- matched
    final_matched <<- final_matched
    in_data_filtered <<- in_data_filtered }
  return(final) }
result <- m_epoch(in_data = in_data, se_in = study_epoch,dtcvar = "PRSTDTC",
 opt = "TRT_LATE", partial =TRUE, debug = TRUE)

 list2env(result, envir = .GlobalEnv)
 print(result)
```

CHALLENGES VS FUTURE DIRECTIONS

This section outlines the primary obstacles encountered during the development of the R-based epoch assignment tool and the strategic roadmap designed to enhance its predictive power and user accessibility.

CHALLENGES	FUTURE DIRECTIONS
Inconsistent Timestamp Formats	Machine Learning for Missing Dates
Partial & Missing Dates	Interactive Dashboards (Shiny, Plotly)
Overlapping Treatment Periods	Automated Error Detection System
Unaligned Visit Windows	Integration with CDISC Standards
Data Entry Errors	Automated Error Detection System
Time Zone Variations	Integration with CDISC Standards
Variable Naming Inconsistency	User-Friendly GUI Tool
Performance with Large Datasets	Parallel Processing Optimization

CONCLUSION

Achieving over 99% accuracy, this automated solution provides a transparent and audit-ready workflow that simplifies the delivery of safety and efficacy analyses. Future integration of machine learning and interactive dashboards will further enhance real-time quality checks and the handling of missing temporal data.

ACKNOWLEDGMENTS

I am deeply grateful to the team at Navitas Life Sciences, especially Sampada Bharat Angane and Motkar Dinesh Harishchandra, for their exceptional guidance and invaluable support throughout this project.

CONTACT INFORMATION

Author: Sheik Ibrahim
Co Author: Ram Mohan Konda
Company: Navitas Life Sciences
Work Phone: +91 9843597483 or +91 8179749729
Email: sheik.ibrahim@navitaslifesciences.com or
Rammohan.konda@navitaslifesciences.com
Website: www.navitaslifesciences.com