PharmaSUG 2026 – Paper ET-197

Automated Delta Detection: A Scalable R-Shiny Framework for Comparing Clinical Datasets

Prabhakara Rao burma, Ephicacy Consulting Group, Latha doanapati, Ephicacy Consulting Group

1 ABSTRACT

In the modern clinical trial landscape, data is characterized by its high velocity and increasing complexity. As studies move toward decentralized models and real-time Electronic Data Capture (EDC) integration, the ability to rapidly identify “DELTA” discrepancies between successive data transfers has become a mission-critical task for Data Monitoring Committees (DMC) and Statistical Programming teams. Traditional methodologies, primarily anchored in static SAS® PROC COMPARE outputs, often present a significant bottleneck; these text-heavy listings are difficult for non-programming stakeholders to interpret and lack the interactivity required for high-volume reconciliation.

This paper presents a scalable R-Shiny framework designed to modernize the delta detection process. The application utilizes a format-agnostic extraction layer to ingest clinical datasets (SAS7BDAT, CSV, and XLSX) and implements a robust “Delta Engine” powered by dplyr and tidyr. By leveraging reactive left_join logic with many-to-many relationship handling, the tool ensures precise subject-level alignment even across longitudinal domains. Key innovations include a javascript-driven “Visual Diff” layer providing multi-tiered feedback: **Green highlighting for new records** and **Red highlighting for modified values**. This dual approach significantly reduces the time-to-insight and provides a transparent audit trail from previous to current transfer, ultimately accelerating the path to database lock.

2 INTRODUCTION

In the highly regulated environment of clinical trial development, data is a living entity. From initial entry to final Database Lock (DBL), data undergoes a continuous cycle of updates, queries, and reconciliations. Statistical programmers are frequently tasked with identifying changes between successive data transfers. A process known as “Delta Detection.” Pinpointing exact modifications is paramount for ensuring data integrity and patient safety.

2.1 The Limitations of Traditional Methods

Historically, the pharmaceutical industry has relied on established SAS® procedures, such as PROC COMPARE. While robust, these tools produce voluminous, text-heavy outputs that are difficult for non-programming stakeholders to interpret. Scanning a static document for discrepancies is not only time-consuming but also prone to human error, particularly in high-volume domains like Laboratory Results (LB).

2.2 The Shift Toward Interactive Analytics

This paper introduces a specialized R-Shiny framework designed to bridge the gap between raw data comparison and intuitive review. By focusing a “fit-for-purpose” comparison between a **Previous** and a **Current** transfer, this tool automates detection at the cell level. It provides a dual-purpose solution: an interactive dashboard for visual inspection and an automated, audit-ready Excel change log for documentation.

3 TECHNICAL METHODOLOGY

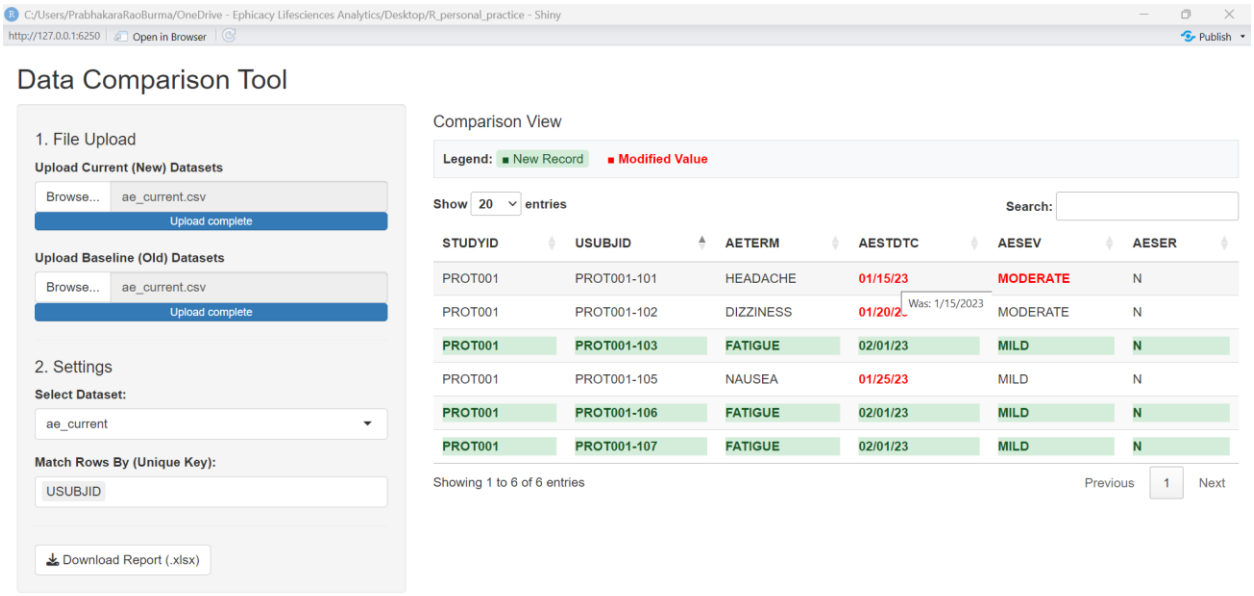
3.1 Data Sanitization and Normalization

To ensure high fidelity, the tool implements a sanitization layer. Leading and trailing whitespaces are removed during ingestion, and all variables are coerced to character strings. This eliminates “false positives” caused by data type mismatches (e.g., numeric 4.0 vs. Character "4").

3.2 The Sentinel Value Strategy

A challenge in browser-based review is the serialization of “missing” data. The tool utilizes a Sentinel Value Strategy where all NA values are replaced with a unique string literal ("EMPTY_VAL") before rendering. This ensures the JavaScript comparison logic remains stable across all browser types.

3.3 Multi-Tiered Visual Heuristics



The comparison logic is split into two layers:

1. **New Record Detection (Green):** The algorithm identifies records present in the current transfer but absent in the previous transfer (where the baseline key is "EMPTY_VAL"). These are highlighted with a green background.
2. **Modification Detection (Red):** For existing records, the tool performs a cell-by-cell comparison. Discrepancies trigger a red font and a **hover-over tooltip showing the previous value.**

3.4 The Join and Difference Logic

The application performs a left_join between the Current (D_c) and Previous (D_p) datasets:

$$D_{diff} = D_c \bowtie_{ID} D_p$$

To handle longitudinal domains, the tool implements many-to-many relationship handling to ensure alignment without data inflation.

4 AUDIT TRAIL AND REPORTING

A critical requirement for pharmaceutical validation is the “Change Log.” The tool automates this by pivoting data into a long format. This log is exported into a multi-tabbed Excel report using the openxlsx library, providing a permanent record of the data evolution.

4.1 Example Audit Trail Structure

Subject ID	Variable	Change Type	Previous Value	Current Value
101-001	AE_SEV	Modified	MILD	MODERATE
102-005	ALL	New Record	[N/A]	[New Data]

5 CODE

```
#libraries used for the delta report

library(shiny)

library(DT)

library(dplyr)

library(tidyr)

library(readxl)

library(haven)

library(openxlsx)

library(tools)
ui <- fluidPage(
  titlePanel("Data Comparison Tool"),

  sidebarLayout(
    sidebarPanel(
      h4("1. File Upload"),
      fileInput("files_new", "Upload Current (New) Datasets", multiple = TRUE
,
                accept = c(".csv", ".xlsx", ".xls", ".sas7bdat")),
      fileInput("files_old", "Upload Baseline (Old) Datasets", multiple=TRUE,
                accept = c(".csv", ".xlsx", ".xls", ".sas7bdat")),

      tags$hr(),
      h4("2. Settings"),
      uiOutput("domain_selector"),
      uiOutput("key_selector"),

      tags$hr(),
      downloadButton("downloadExcel", "Download Report (.xlsx)")
    ),

    mainPanel(
      h4("Comparison View"),
      div(style = "margin-bottom: 15px; padding: 10px; background: #f8f9fa; b
```

```
order: 1px solid #ddd;",
          span("Legend: ", style="font-weight:bold"),
          span("■ New Record", style="background-color:#d4edda; color:#155724
; padding: 2px 6px; margin-right:10px; border-radius:4px;"),
          span("■ Modified Value", style="color:red; font-weight:bold; paddin
g: 2px 6px;")
      ),
      DTOutput("contents")
    )
  )
)

server <- function(input, output, session) {

  # 1. File Reading
  read_multi_data <- function(file_infos) {
    req(file_infos)
    data_list <- list()
    for (i in 1:nrow(file_infos)) {
      name <- tolower(tools::file_path_sans_ext(file_infos$name[i]))
      ext <- tolower(tools::file_ext(file_infos$name[i]))

      df <- tryCatch({
        if(ext == "csv") read.csv(file_infos$datapath[i], stringsAsFactors =
FALSE)
        else if (ext %in% c("xlsx", "xls")) readxl::read_excel(file_infos$dat
apath[i])
        else if (ext == "sas7bdat") haven::read_sas(file_infos$datapath[i])
        else NULL
      }, error = function(e) return(NULL))

      if(!is.null(df)) data_list[[name]] <- df
    }
    return(data_list)
  }

  all_new_data <- reactive({ read_multi_data(input$files_new) })
  all_old_data <- reactive({ read_multi_data(input$files_old) })

  # 2. Dynamic Dropdowns
  output$domain_selector <- renderUI({
    req(all_new_data())
    common <- intersect(names(all_new_data()), names(all_old_data()))
    if(length(common) == 0) return(helpText("No matching filenames found."))
    selectInput("selected_domain", "Select Dataset:", choices = common)
  })

  output$key_selector <- renderUI({
    req(input$selected_domain, all_new_data())
    df <- all_new_data()[[input$selected_domain]]
    guesses <- intersect(c("Account", "USUBJID", "AESEQ", "LBSEQ", "Visit"),
names(df))
    if(length(guesses) == 0) guesses <- names(df)[1]

    selectInput("selected_keys", "Match Rows By (Unique Key):",
                choices = names(df), selected = guesses, multiple = TRUE)
  })
```

```
  # 3. Join Logic with IS_NEW Flag
  current_diff <- reactive({
    req(input$selected_domain, input$selected_keys, all_new_data(), all_old_d
ata())

    new_df <- all_new_data()[[input$selected_domain]]
    old_df <- all_old_data()[[input$selected_domain]]

    req(all(input$selected_keys %in% names(new_df)))

    # Identify which keys exist in the OLD dataset
    old_keys_only <- old_df %>%
      select(all_of(input$selected_keys)) %>%
      distinct() %>%
      mutate(FOUND_IN_OLD = 1)

    # Merge Data
    merged_df <- full_join(new_df, old_df,
                           by = input$selected_keys,
                           suffix = c("", "_Old"),
                           relationship = "many-to-many")

    # Determine IS_NEW status
    final_df <- left_join(merged_df, old_keys_only, by = input$selected_keys)
    final_df$IS_NEW <- ifelse(is.na(final_df$FOUND_IN_OLD), 1, 0)

    return(final_df)
  })

  # 4. Render Table (Fixed Logic for Keys)
  output$contents <- renderDT({
    df <- current_diff()
    req(df)

    visible_cols <- names(all_new_data()[[input$selected_domain]])

    # Locate the hidden IS_NEW column index for JS
    idx_is_new <- which(names(df) == "IS_NEW") - 1

    col_defs <- lapply(visible_cols, function(colname) {
      curr_idx <- which(names(df) == colname) - 1

      # FIX: Handle cases where the column is a KEY and thus has no "_Old" co
unterpart
      old_idx_raw <- which(names(df) == paste0(colname, "_Old"))

      # If old_idx_raw is empty (which happens for Keys like USUBJID), set it
to 0 so it becomes -1
      if(length(old_idx_raw) == 0) {
        old_idx <- -1
      } else {
        old_idx <- old_idx_raw - 1
      }

      list(targets = curr_idx, render = JS(sprintf(
        "function(data, type, row, meta) {
```

```
          if(type === 'display') {

            // PRIORITY 1: Check if the entire row is NEW
            var isNew = row[%d];
            if (isNew == 1) {
               return '<div style=\"background-color: #d4edda; color: #155724
; font-weight: bold; width: 100%%; height: 100%%;\">' + (data === null ? '' :
data) + '</div>';
            }

            // PRIORITY 2: Check for Modified Values (Red)
            // This only runs if isNew is NOT 1, and if an Old column exists
(oldIdx >= 0)
            var oldIdx = %d;
            if (oldIdx >= 0) {
              var oldVal = row[oldIdx];
              if (data !== null && String(data) !== String(oldVal)) {
                 return '<span title=\"Was: ' + oldVal + '\" style=\"color:re
d; font-weight:bold;\">' + data + '</span>';
              }
            }
          }
          return data;
        }", idx_is_new, old_idx)))
    })

    # Hide helper columns
    hide_cols <- c(grep("_Old$", names(df)), which(names(df) %in% c("FOUND_IN
_OLD", "IS_NEW")))
    hide_indices <- hide_cols - 1

    if(length(hide_indices) > 0) {
      col_defs[[length(col_defs) + 1]] <- list(targets = hide_indices, visibl
e = FALSE)
    }

    datatable(df,
              rownames = FALSE,
              escape = FALSE,
              options = list(
                scrollX = TRUE,
                columnDefs = col_defs,
                pageLength = 20
              ))
  })

  # 5. Excel Export
  output$downloadExcel <- downloadHandler(
    filename = function() { paste0("Delta_Report_", Sys.Date(), ".xlsx") },
    content = function(file) {
      req(all_new_data(), all_old_data())
      wb <- createWorkbook()

      common <- intersect(names(all_new_data()), names(all_old_data()))

      for(dom in common) {
        new_d <- all_new_data()[[dom]]
```

```
        old_d <- all_old_data()[[dom]]

        id <- intersect(c("Account", "USUBJID", "AESEQ"), names(new_d))
        if(length(id)==0) id <- names(new_d)[1]

        addWorksheet(wb, dom)
        writeData(wb, dom, new_d)

        new_l <- new_d %>% mutate(across(everything(), as.character)) %>% piv
ot_longer(-any_of(id))
        old_l <- old_d %>% mutate(across(everything(), as.character)) %>% piv
ot_longer(-any_of(id))

        log <- inner_join(new_l, old_l, by=c(id, "name"), suffix=c("_New","_O
ld"), relationship="many-to-many") %>%
          filter(value_New != value_Old)

        if(nrow(log) > 0) {
          log_name <- substr(paste0(dom, "_Log"), 1, 31)
          addWorksheet(wb, log_name)
          writeData(wb, log_name, log)
        }
      }
      saveWorkbook(wb, file, overwrite=TRUE)
    }
  )
}

shinyApp(ui, server)
```

6 CONCLUSION

By focusing on a visual-first approach, this R-Shiny tool eliminates the noise of traditional outputs and provides a clear, visual path to data reconciliation, ultimately accelerating the path to database lock.

7 REFERENCES

1. Wickham, H., & François, R. (2023). dplyr: A Grammar of Data Manipulation.
2. Xie, Y. (2023). Interactive Tables with the DT Package.
3. PharmaSUG (2024).

8 CONTACT DETAILS

Author 1: Prabhakara Rao BURMA
Associate Director Statistical Programming
Ephicacy Consulting Group LLC
e-mail: Prsas85@gmail.com
Author 2: Latha Donapati
Director Statistical Programming
Ephicacy Consulting Group LLC
e-mail: donapatilatha@gmail.com