PharmaSUG China 2024 - Paper PO-10087

Introduction of an R-Shiny Application for Adverse Event Data Verification

Keting Lv, Daiichi Sankyo (China) Holdings Co., Ltd.

ABSTRACT

Managing adverse event (AE) data can be a daunting task for Data Managers due to its high volume and real-time review requirements. To address this challenge, we introduce an R-Shiny application designed to support Data Managers in verifying AE data. The application covers various critical aspects, including data structure verification (detecting instances where the same subject has multiple adverse event records with overlapping dates), intra-table field relationships (identifying discrepancies such as an End Date occurring before the corresponding Start Date, ensuring proper selection of CTCAE Grade, and verifying that AE Outcome is appropriately labeled), cross-table verification (handling cases where adverse events occur before drug administration), and additional considerations addressing common mistakes and providing insights into data entry issues related to AE data. The application accepts input source data from CDISC datasets SDTM or non-standard formats, provided that minimal data mapping is supplied by the user.

Additionally, various types of interactive AE plots can be generated from this R-Shiny application to support AE data review.

In summary, our R-Shiny App offers valuable insights and streamlines the process of verifying AE data, ultimately improving data quality and review efficiency.

INTRODUCTION

Adverse Event data monitoring is a pivotal process in clinical trials. Although edit checks serve as the first line of defense against data entry errors, the complexity of different data sources and human error in database design often necessitate manual review or offline checks by data managers. This underscores the need for a support tool to aid data managers in their AE data review and verification.

The purpose of this paper is to present an R-Shiny based application designed to overcome the limitations of traditional SAS/Excel-based AE data check tools. Our goal is to create an interactive application that consolidates numerous outputs into a single dynamic dashboard, avoiding overwhelming lists. The application offers real-time updates and automatic data synchronization, making it user-friendly even for reviewers with no knowledge of R-Shiny programming.

The Shiny App comprises three components: the Overview tab, Utilities tab, and Graphics tab. The Overview tab allows users to upload datasets and provides instructions for various data check scenarios. It also enables users to review, filter, and download selected data. The Utilities tab displays data tables for different scenarios that require the attention of data managers to identify potential data issues. Data managers can easily download the data to a CSV file for further analysis. The Graphics tab generates visualizations of Adverse Events by subject ID for additional review.

Examples, dummy data, and R-code snippets are simplified or partially included in this paper for demonstration purposes. The real-world version of the app will include many different datasets and complex rules, which are difficult to illustrate comprehensively.

The example data used here is from SDTMCDISCP01: Example of SDTM datasets from the CDISC original Pilot 01, with some manually added data issues for demonstration purposes. The real-world check logic will be much more complex than the demo example.

METHODS

INFORMATION WORKFLOW

This flowchart provides a clear visual representation of how different types of data are processed and utilized within the R-Shiny application to generate meaningful insights and visualizations.

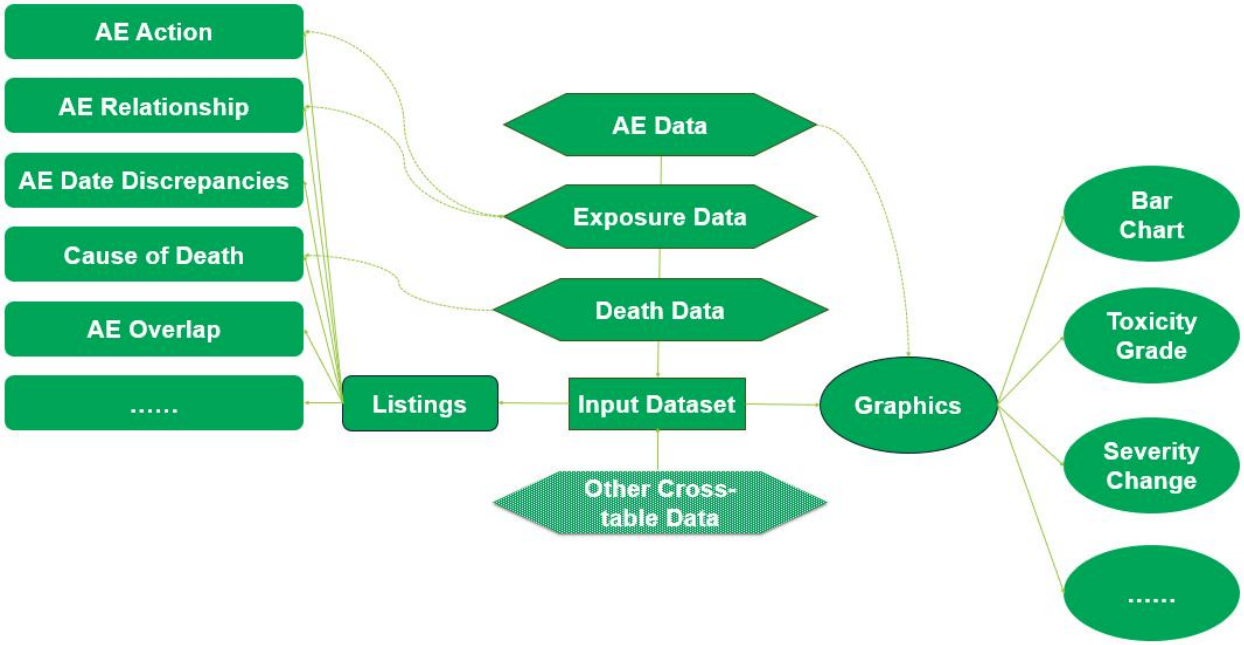


Figure 1. Flowchart of information

As the flowchart shows, this R-Shiny App is divided into three main sections: Input Dataset, Listings, and Graphics. Each section contains various elements (only demo examples are included here; the real-world App will have more elements). In this demo, the data inputs (AE Data, Exposure Data, Death Data) feed into the Input Dataset. The Listings element (AE Action, AE Relation, AE Date Discrepancies, Cause of Death, AE Overlap) aggregates data from various inputs or individual inputs. Also the Input Dataset is used to generate Graphics, which include Bar Charts, Toxicity Grades, and Severity Changes.

METHODS OF LISTINGS

Number	Domains	Report Title	Query Text
1	AE, Exposure	AE Action	AE start date are before the first administration or after the last administration, and the action taken with study treatment are not chosen as 'Not Applicable'
2	AE, Exposure	AE Relationship	AE occurs before the drug's first administration, its relation to the study treatment must not be 'unrelated'
3	AE	AE Date Discrepancies	AE end date is earlier than start date
4	AE, Death	Cause of Death	AE result in death, but the death is not recorded on the death form
5	AE	Overlap AE	AE that share the same LLT, and have overlapping start and end dates
6	…	…	…

Table 1. Example specifications of listings output

Domains indicate the relevant domains or categories that the listing pertains to. The report title provides a brief summary of each listing's focus. The query text describes the specific criteria or conditions that the listing is designed to check. Each listing specification helps ensure data accuracy and consistency by identifying potential issues or discrepancies in the AE data.

R code	SAS code

ae1 <- ae %>%
  mutate(
  USUBJID2 = lead(USUBJID),
  AELLT2 = lead(AELLT),
  AESEQ2 = lead(AESEQ),
  AESTDTC2 = lead(AESTDTC),
  AEENDTC2 = lead(AEENDTC)
) %>%
  filter(
  (USUBJID == USUBJID2 & AELLT == AELLT2 & is.na(AEENDTC)) |
  (USUBJID == USUBJID2 & AELLT == AELLT2 & AEENDTC > AESTDTC2)
) %>%
  select(STUDYID, USUBJID, AELLT, AESEQ, AESTDTC, AEENDTC, AESEQ2, AESTDTC2, AEENDTC2)
	data ae1;
	merge ae ae(firstobs=2 rename=(
	  USUBJID=USUBJID2
	  AELLT=AELLT2
	  AESEQ=AESEQ2
	  AESTDTC=AESTDTC2
	  AEENDTC=AEENDTC2))
	;
	if
	  (USUBJID= USUBJID2 and AELLT = AELLT2 and AEENDTC= .) or
	  (USUBJID= USUBJID2 and AELLT = AELLT2 and AEENDTC> AESTDTC2)
	;
	keep STUDYID USUBJID AELLT AESEQ AESTDTC AEENDTC AESEQ2 AESTDTC2 AEENDTC2;
	run;

Display 1. R and SAS code snippets to identify overlapping AEs

Both code snippets aim to identify overlapping AEs for the same subject and LLT, ensuring data accuracy and consistency in the AE start date and AE end date data.

METHODS OF VISUALIZATIONS

Purpose	R code
Filtering Data for Selected ID	filtered_ae <- reactive({ DAT_A()[DAT_A()$USUBJID == input$selected_usubjid,] })

Rendering the Interactive Plot	output$aeplot1 <- renderPlotly({
	  aeplot1 <- ggplot(filtered_ae(),
	  aes(x = AESTDY, y = AEDECOD,
	  text = paste("AEACN:", AEACN, "
	",
	  "AEREL:", AEREL, "
	",
	  "AESEV:", AESEV, "
	",
	  "AEOUT:", AEOUT))) +
	  geom_line(color = "black", linewidth = 1) +
	  geom_point(aes(color = AETOXGR), size = 3) +
	  labs(y = "", x = "Study Day",
	  title = "", color = "") +
	  guides(color = guide_legend(title = "Standard Toxicity Grade"))
	
	  ggplotly(aeplot1, tooltip = "text")
	})

Rendering the Data Table	output$gtable2 <- renderDataTable({
	  filtered_ae() %>%
	  select(STUDYID, USUBJID, AETERM, AEDECOD, AEACN, AEREL, AESEV, AEOUT, AETOXGR, AESTDY) %>%

Purpose	R code
	datatable() })

Display 2. Sample code of plot

This R-Shiny code is designed to create an interactive plot and a data table based on adverse event (AE) data filtered by a selected subject ID (USUBJID). The sample code uses ggplot2 for plotting and plotly for interactivity, along with DT for rendering the data table.

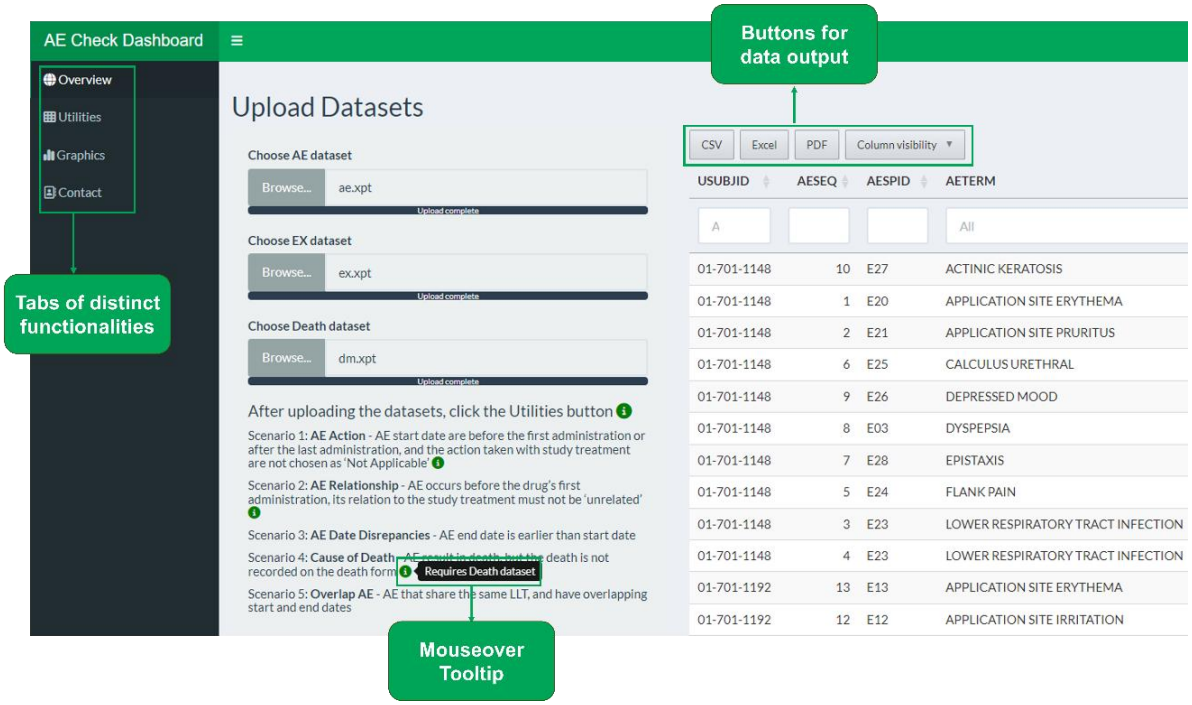
The reactive expression filters the dataset DAT_A() to include only the rows where the USUBJID matches the user-selected ID (input$selected_usubjid). The reactive function ensures that the filtered data updates automatically whenever input$selected_usubjid changes.

An interactive plot is created to visualize AE data over study days, with tooltips providing additional information, using ggplot2 and plotly. The aesthetics function maps AESTDY (study day) to the x-axis and AEDECOD (AE description) to the y-axis. Additional information (AEACN, AEREL, AESEV, AEOUT) is included in the tooltips. The geometries function adds a line (geom_line) and points (geom_point) to the plot. The points are colored based on AETOXGR (toxicity grade). The ggplot object is converted to a plotly object with tooltips enabled.

A data table is rendered to display detailed AE information for the selected subject. The selection function selects specific columns to display in the table. The datatable() function from the DT package renders the table filtered_ae() interactively.

RESULTS

OVERVIEW



Output 1. Overview Tab

The left sidebar contains tabs for different functions – uploading datasets, generating listings, and visualizations. The mouseover tooltip provides tips to let the user know if a specific dataset is needed for a specific check. The right side serves as a raw data-friendly review and output window.

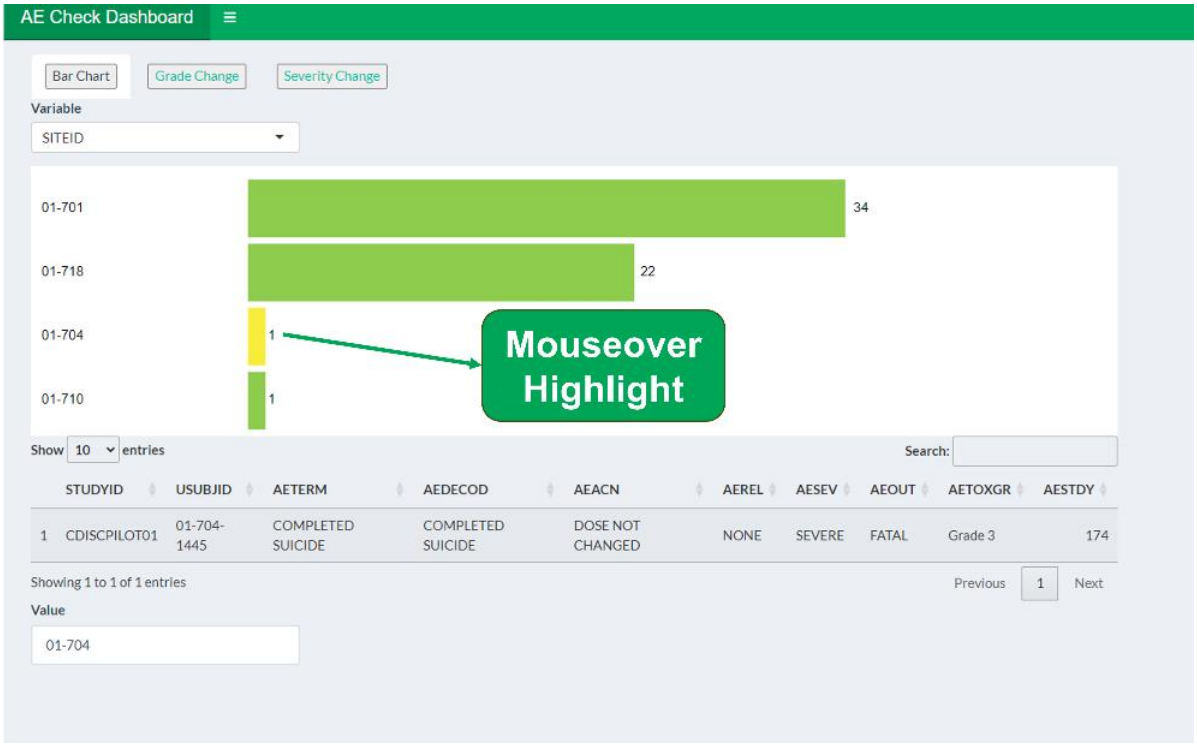
LISTINGS OUTPUT

Output 2. Listings Tab

Output 2 is a sample application of checks for AE date discrepancies. For example, when the AE end date is earlier than the AE start date, the violated records are listed here. The violated variables, AE start date and AE end date, are highlighted. There is also a “CSV” button for the listing output.

VISUALIZATIONS

Bar chart plot

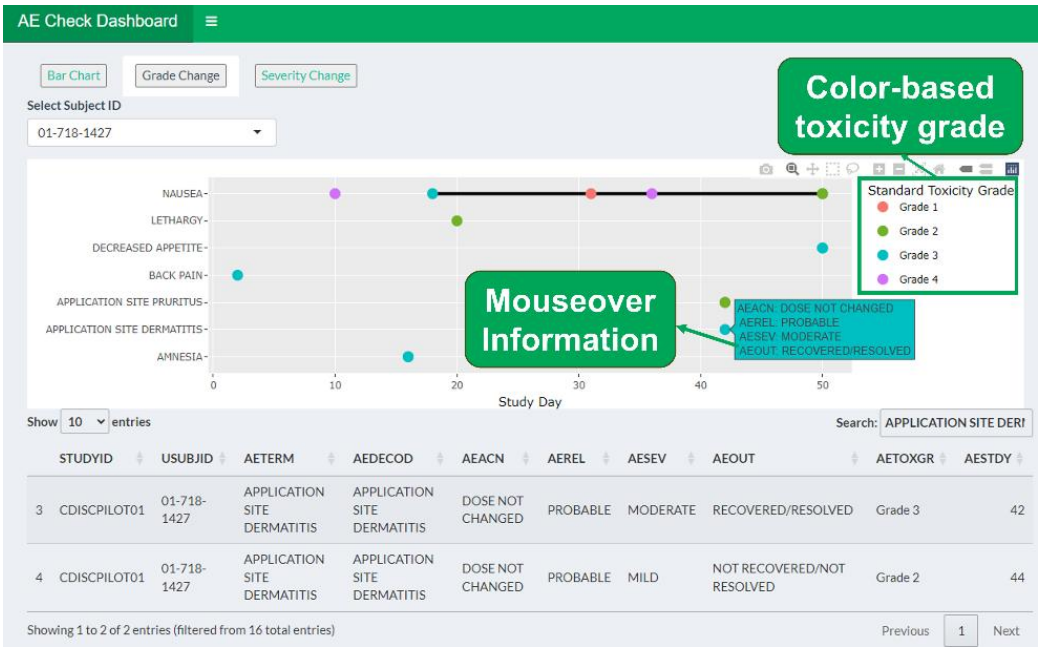


Output 3. Bar chart plot

This bar chart plot monitors the number of AE records at each site.

The mouseover highlight function in the bar chart is quite useful. When you hover over each bar, it displays detailed information about the corresponding AE records for that site. This feature allows users to quickly access and review specific data points without needing to navigate away from the chart. This functionality makes the dashboard more interactive and user-friendly, allowing for efficient data analysis and review.

Toxicity grade change over time plot

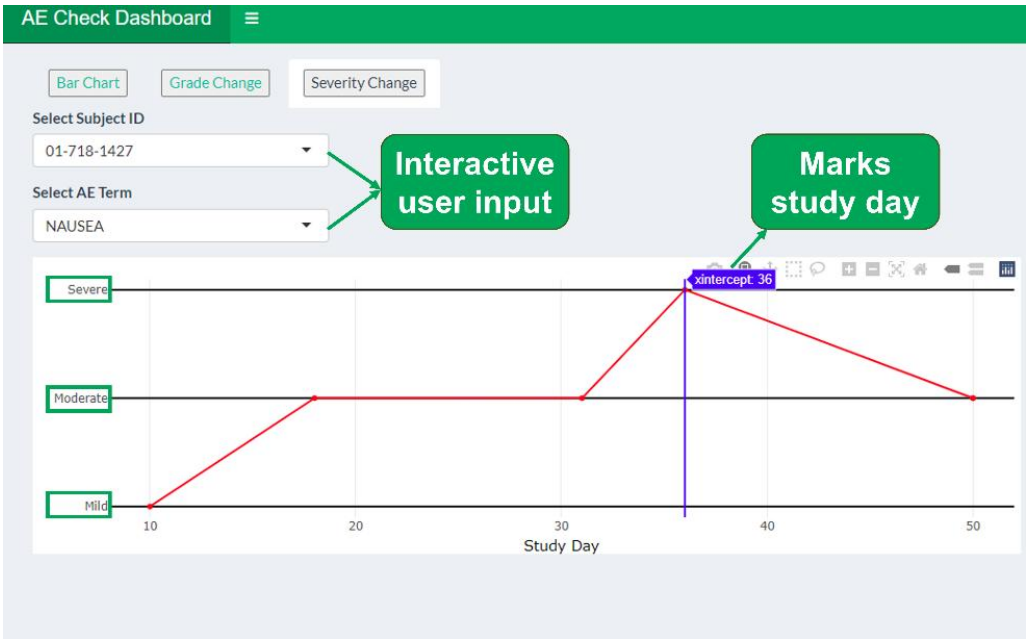


Output 4. Toxicity grade change over time plot

This plot visualizes the occurrence of adverse events over the course of the study. Each AE is color-coded based on its toxicity grade. The x-axis represents the study day, allowing you to see when each AE occurred. The y-axis lists different types of AEs, making it easy to identify which AE occurred at which time.

By tracking AEs over time, patterns or trends can be identified, providing a quick visual summary of the frequency and severity of AEs.

Severity change plot



Output 5. Toxicity grade over time plot

Output 5 is used for monitoring changes in the severity of an event over time. The select subject ID and AE term buttons allow for interactive user input to choose a specific event. The x-axis represents the timeline of the study. The red line tracks the severity of the condition over time. For example, it starts at Mild on day 10, peaks at Severe around day 36, and then returns to Moderate by day 50. The blue vertical reference line marks the study day when mouseover. Overall, this plot helps visualize how the severity of a condition changes over the study period, allowing for better analysis and decision-making

CONCLUSION

Overall, this Shiny app serves as a dashboard for monitoring and validating AE data in clinical trials, ensuring data quality and consistency with predefined scenarios. It provides an interactive platform for data managers to upload datasets, apply checks, and review results efficiently.

ACKNOWLEDGMENTS

The author would like to thank Daiichi Sankyo Biostatistics and Data Management Department tor reviewing the paper and providing valuable feedback.

CONTACT INFORMATION

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

Keting Lv
Daiichi Sankyo (China) Holdings Co., Ltd.
86-18106850068
lv.keting.kn@daiichisankyo.com.cn
https://bit.ly/Portfolio-klyu

APPENDIX I: DEMO R CODE

```
library(shiny)
library(shinythemes)
library(shinydashboard)
library(tidyverse)
library(DT)
library(shinyBS)
library(plotly)

ui <- fluidPage(theme = shinytheme("flatly"),
        dashboardPage(skin = "green",
                header = dashboardHeader(title="AE Check Dashboard"),
                sidebar = dashboardSidebar(
                  sidebarMenu(
                    menuItem("Overview", icon = icon("globe"), tabName = "overview"),
                    menuItem("Utilities", icon = icon("th"), tabName = "utilities"),
                    menuItem("Graphics", icon = icon("chart-simple"), tabName = "graphics"),
                    menuItem("Contact", icon = icon("address-book"), tabName = "contact")
                  )
                ),
                body = dashboardBody(tabItems(

                  tabItem(tabName = "overview",
                        titlePanel("Upload Datasets"),
                        sidebarLayout(
                          sidebarPanel(
                            style = "height: 80vh; overflow-y: auto;",
                            fileInput("sasdata1", "Choose AE dataset", multiple = FALSE, accept = ".xpt"),
                            fileInput("sasdata2", "Choose EX dataset", multiple = FALSE, accept = ".xpt"),
                            fileInput("sasdata3", "Choose Death dataset", multiple = FALSE, accept =
".xpt"),
                            h4("After uploading the datasets, click the Utilities button",
                              icon("info-circle", id = "info-icon-2", style = "color: green; cursor: pointer;")
                            ),
                            h5(HTML("Scenario 1: <b>AE Action</b> - AE start date are before the first
administration or after the last administration, and the action taken with study treatment are not chosen as
'Not Applicable'"),
                              icon("info-circle", id = "info-icon-3", style = "color: green; cursor: pointer;")
```

```
                    ),
                    h5(HTML("Scenario 2: <b>AE Relationship</b> - AE occurs before the drug's first administration, its relation to the study treatment must not be 'unrelated'"),
                      icon("info-circle", id = "info-icon-4", style = "color: green; cursor: pointer;")
                    ),
                    h5(HTML("Scenario 3: <b>AE Date Disrepancies</b> - AE end date is earlier than start date")),
                    h5(HTML("Scenario 4: <b>Cause of Death</b> - AE result in death, but the death is not recorded on the death form"),
                      icon("info-circle", id = "info-icon-5", style = "color: green; cursor: pointer;")
                    ),
                    h5(HTML("Scenario 5: <b>Overlap AE</b> - AE that share the same LLT, and have overlapping start and end dates"))
                  ),
                  mainPanel(
                    DT::DTOutput("alldata")
                  )
                ),
                bsTooltip(id = "info-icon-2", title = "Then select the scenarios listed below", placement = "right", trigger = "hover"),
                bsTooltip(id = "info-icon-3", title = "Requires EX dataset", placement = "right", trigger = "hover"),
                bsTooltip(id = "info-icon-4", title = "Requires EX dataset", placement = "right", trigger = "hover"),
                bsTooltip(id = "info-icon-5", title = "Requires Death dataset", placement = "right", trigger = "hover"),
              ),


              tabItem(tabName = "utilities",
                mainPanel(
                  tabsetPanel(
                    id = 'dataset',
                    tabPanel(
                      title = tags$button(
                        "AE Action",
                        id = "scenario1-btn",
                        title = "AE start date are before the first administration or after the last administration, and the action taken with study treatment are not chosen as 'Not Applicable'"
                      ),
                      DT::dataTableOutput("mytable1")
```

```
                    ),
                    tabPanel(
                      title = tags$button(
                        "AE Relationship",
                        id = "scenario1-btn",
                        title = "AE occurs before the drug’s first administration, its relation to the study treatment must not be ‘unrelated’"
                      ),
                      DT::dataTableOutput("mytable2")
                    ),
                    tabPanel(
                      title = tags$button(
                        "AE Date Disrepancies",
                        id = "scenario1-btn",
                        title = "AE end date is earlier than start date"
                      ),
                      DT::dataTableOutput("mytable3")
                    ),
                    tabPanel(
                      title = tags$button(
                        "Cause of Death",
                        id = "scenario1-btn",
                        title = "AE result in death, but the death is not recorded on the death form"
                      ),
                      DT::dataTableOutput("mytable4")
                    ),
                    tabPanel(
                      title = tags$button(
                        "Overlap AE",
                        id = "scenario1-btn",
                        title = "AE that share the same LLT, and have overlapping start and end dates"
                      ),
                      DT::dataTableOutput("mytable5")
                    )
                  )
                )),
```

```
                tabItem(tabName = "graphics",
                        mainPanel(
                          tabsetPanel(
                            id = 'graphic',
                            tabPanel(
                              title = tags$button(
                                "Bar Chart",
                                id = "scenario1-btn",
                                title = "Count of number of AE"
                              ),
                              selectInput("var", "Variable",
                                          list("AESEV", "AESER", "AEREL", "AEOUT", "AESCAN",
"AESCONG","SITEID"),
                                          selected = "SITEID"),
                              d3Output("d3", height = "300px"),
                              DT::dataTableOutput("gtable1"),
                              textInput("val", "Value")
                            ),
                            tabPanel(
                              title = tags$button(
                                "Grade Change",
                                id = "scenario1-btn",
                                title = "Select the subject, plot of AE Grade by Study Day will display"
                              ),
                              selectInput("selected_usubjid", "Select Subject ID", choices = NULL) ,
                              plotlyOutput('aeplot1', height = "300px"),
                              DT::dataTableOutput("gtable2"),
                            ),
                            tabPanel(
                              title = tags$button(
                                "Severity Change",
                                id = "scenario1-btn",
                                title = "Select the subject and Term, plot of Severity Change will display"
                              ),
                              selectInput("usubjid", "Select Subject ID", choices = NULL),
                              selectInput("aedecod", "Select AE Term", choices = NULL),
                              plotlyOutput('aeplot2')
```

```
                                    )
                                   )
                                  )),
                          tabItem(tabName = "contact",
                          )
)))
)
server <- function(input, output, session) {

  DAT_A <- reactive({
    req(input$sasdata1)
    inData1 <- input$sasdata1
    if (is.null(inData1)){ return(NULL) }
    mydata1 <- haven::read_xpt(inData1$datapath)
  })

  observeEvent(DAT_A(), {
    updateSelectInput(session, "selected_usubjid", choices = unique(DAT_A()$USUBJID))
  })

  DAT_B <- reactive({
    req(input$sasdata2)
    inData2 <- input$sasdata2
    if (is.null(inData2)){ return(NULL) }
    mydata2 <- haven::read_xpt(inData2$datapath)
  })

  DAT_C <- reactive({
    req(input$sasdata3)
    inData3 <- input$sasdata3
    if (is.null(inData3)){ return(NULL) }
    mydata3 <- haven::read_xpt(inData3$datapath)
  })

  output$alldata = DT::renderDataTable({

    DT::datatable(
```

```
    DAT_A(),
    filter = 'top', extensions = c('Buttons', 'Scroller'),
    options = list(scrollY = 650,
                   scrollX = 500,
                   deferRender = TRUE,
                   scroller = TRUE,
                   # paging = TRUE,
                   # pageLength = 25,
                   buttons = list('csv','excel','pdf',
                                  list(extend = 'colvis', targets = 0, visible = FALSE)),
                   dom = 'lBfrtip',
                   fixedColumns = TRUE),
    rownames = FALSE)

})

 output$mytable1 <- DT::renderDataTable({
  a_all <- DAT_A() %>%
    left_join(DAT_B() %>%
                group_by(STUDYID, USUBJID) %>%
                summarise(
                  TRTSDT = min(EXSTDTC, na.rm = TRUE),
                  TRTEDT = max(EXENDTC, na.rm = TRUE),
                  .groups = "drop"
                ),
              by = c("STUDYID", "USUBJID")) %>%
    filter(AESTDTC < TRTSDT | AESTDTC > TRTEDT) %>%
    filter(!AEACN %in% c("NOT APPLICABLE")) %>%
    select(STUDYID, USUBJID, AESEQ, AESPID, AETERM, AEACN, AESTDTC, AEENDTC, TRTSDT,
TRTEDT)


   DT::datatable(
    a_all,
    caption = tags$caption(
      style = "caption-side: bottom; text-align: left; margin: 8px 0;",
      "The AE start date are before the first administration or after the last administration, and the action
taken with study treatment are not chosen as ‘Not Applicable’"
    ),
```

```
    extensions = 'Buttons',
    options = list(
      dom = 'tB',
      paging = TRUE,
      searching = TRUE,
      fixedColumns = TRUE,
      autoWidth = TRUE,
      ordering = TRUE,
      buttons = list(
        list(extend = 'csv', filename = "AE Scenario 1")
      )
    )
  ) %>%
    formatStyle(
      columns = c('AEACN'),
      backgroundColor = 'lightgreen'
    )
})


 output$mytable2 <- DT::renderDataTable({
   b_all <- DAT_A() %>%
     filter(!AEREL %in% c("UNRELATED", "NOT RELATED", "NONE", "N")) %>%
     left_join(DAT_B() %>%
                 group_by(USUBJID) %>%
                 filter(EXSTDTC == min(EXSTDTC)),
               by = c("STUDYID", "USUBJID")) %>%
     filter(AESTDTC < EXSTDTC)%>%
     select(STUDYID, USUBJID, AESEQ, AESPID, AETERM, AEREL, AESTDTC, AEENDTC, EXSTDTC,
EXENDTC)
   DT::datatable(
     b_all,
     caption = tags$caption(
       style = "caption-side: bottom; text-align: left; margin: 8px 0;",
       "The AE occurred before the first administration of the drug, and its relationship to the study
treatment was not selected as ‘unrelated’."
     ),
     extensions = 'Buttons',
     options = list(
```

```
      dom = 'tB',
      paging = TRUE,
      searching = TRUE,
      fixedColumns = TRUE,
      autoWidth = TRUE,
      ordering = TRUE,
      buttons = list(
        list(extend = 'csv', filename = "AE Scenario 2")
      )
    )
  )%>%
    formatStyle(
      columns = c('AEREL'),
      backgroundColor = 'lightgreen'
    )
})

output$mytable3 <- DT::renderDataTable({
  c_all <- DAT_A() %>%
    filter(AESTDTC > AEENDTC & AESTDTC != "" & AEENDTC != "") %>%
    select(STUDYID, USUBJID, AESEQ, AESPID, AETERM, AESTDTC, AEENDTC)
  DT::datatable(
    c_all,
    caption = tags$caption(
      style = "caption-side: bottom; text-align: left; margin: 8px 0;",
      "The AE end date is earlier than start date."
    ),
    extensions = 'Buttons',
    options = list(
      dom = 'tB',
      paging = TRUE,
      searching = TRUE,
      fixedColumns = TRUE,
      autoWidth = TRUE,
      ordering = TRUE,
      buttons = list(
        list(extend = 'csv', filename = "AE Scenario 3")
```

```
    )
   )
  )%>%
   formatStyle(
    columns = c('AESTDTC','AEENDTC'),
    backgroundColor = 'lightgreen'
   )
})

output$mytable4 <- DT::renderDataTable({
  d_all <- DAT_A() %>%
   filter(AESDTH == "Y"|AEOUT == "FATAL") %>%
   left_join(DAT_C(),
             by = c("STUDYID", "USUBJID")) %>%
   filter(DTHFL !="Y")%>%
   select(STUDYID, USUBJID, AESEQ, AESPID, AETERM, AESDTH, AEOUT, DTHFL)
  DT::datatable(
   d_all,
   caption = tags$caption(
    style = "caption-side: bottom; text-align: left; margin: 8px 0;",
    "Adverse events result in death, but the death is not recorded on the death form."
   ),
   extensions = 'Buttons',
   options = list(
    dom = 'tB',
    paging = TRUE,
    searching = TRUE,
    fixedColumns = TRUE,
    autoWidth = TRUE,
    ordering = TRUE,
    buttons = list(
     list(extend = 'csv', filename = "AE Scenario 4")
    )
   )
  )%>%
   formatStyle(
    columns = c('DTHFL'),
```

```
      backgroundColor = 'lightgreen'
    )
})

output$mytable5 <- DT::renderDataTable({
  e_all <- DAT_A() %>%
    mutate(
      USUBJID2 = lead(USUBJID),
      AELLT2 = lead(AELLT),
      AESEQ2 = lead(AESEQ),
      AESTDTC2 = lead(AESTDTC),
      AEENDTC2 = lead(AEENDTC),
    ) %>%
    filter(
      (USUBJID == USUBJID2 & AELLT == AELLT2 & is.na(AEENDTC)) |
        (USUBJID == USUBJID2 & AELLT == AELLT2 & AEENDTC > AESTDTC2)
    ) %>%
    select(STUDYID,USUBJID,AELLT,AESEQ,AESTDTC,AEENDTC,AESEQ2,AESTDTC2,AEENDTC2)

  DT::datatable(
    e_all,
    caption = tags$caption(
      style = "caption-side: bottom; text-align: left; margin: 8px 0;",
      "Adverse events result in death, but the death is not recorded on the death form."
    ),
    extensions = 'Buttons',
    options = list(
      dom = 'tB',
      paging = TRUE,
      searching = TRUE,
      fixedColumns = TRUE,
      autoWidth = TRUE,
      ordering = TRUE,
      buttons = list(
        list(extend = 'csv', filename = "AE Scenario 5")
      )
    )
```

```
  )%>%
    formatStyle(
      columns = c('AEENDTC', "AESTDTC2"),
      backgroundColor = 'lightgreen'
    )
 })
 #Plot 2
 # Filter the data for the selected USUBJID
 filtered_ae <- reactive({
  DAT_A()[DAT_A()$USUBJID == input$selected_usubjid, ]
 })

 output$aeplot1 <- renderPlotly({
  # Create a ggplot object with text aesthetic for tooltips
  aeplot1 <- ggplot(filtered_ae(), aes(x = AESTDY, y = AEDECOD,
                        text = paste("AEACN:", AEACN, "<br>",
                                     "AEREL:", AEREL, "<br>",
                                     "AESEV:", AESEV, "<br>",
                                     "AEOUT:", AEOUT))) +
    geom_line(color = "black", linewidth = 1) +
    geom_point(aes(color = AETOXGR), size = 3) +
    labs(y = "", x = "Study Day",
         title = "", color = "") +
    guides(color = guide_legend(title = "Standard Toxicity Grade"))

  # Convert it to an interactive plotly object with tooltips
  ggplotly(aeplot1, tooltip = "text")
 })
 output$gtable2 <- renderDataTable({
  filtered_ae() %>%
    select(STUDYID, USUBJID, AETERM, AEDECOD, AEACN, AEREL, AESEV, AEOUT, AETOXGR,
AESTDY)%>%
    datatable()
 })

}
shinyApp(ui = ui, server = server)
```