PharmaSUG 2025 - Paper OS-107

Clinical Data Quality Assurance: An Interactive Application for Data Discrepancy Detection

Yushan Wang, Merck & Co., Inc., Rahway, NJ, USA

ABSTRACT

In today's data-driven clinical environment, ensuring the quality of data deliverable is essential for evaluating trial outcomes. This paper introduces an R Shiny application designed for statistical programmers to facilitate rigorous data discrepancy checks during programming and addresses frequent inquiries from a wide range of clinical stakeholders. The application streamlines the operational process by incorporating four key modules: Records with Missing Key Variables, Find Duplicates, Date Comparison, and Datasets Comparison.

By integrating these functionalities into an user-friendly interface and incorporating robust R programming techniques alongside with R shiny framework, this application empowers statistical programmers to systematically address common data inconsistency and stakeholder inquiries encountered during programming and analysis/modelling processes.

INTRODUCTION

Statistical programming teams often face challenges during the programming process and frequently receive questions and ad-hoc requests from various stakeholders, such as biomarker statisticians and pharmacokinetics-pharmacodynamics (PKPD) modelers, after datasets are generated. These challenges can be time-consuming and reduce the resources available for additional projects. To address these investigations and improve workflows, I have developed a user-friendly R Shiny application designed to automate key checks. This application effectively tackles common data discrepancies that can affect deliverable quality through four modules: Records with Missing Key Variables, Find Duplicates, Date Comparison, and Datasets Comparison. Each module plays a vital role in identifying data discrepancy. By using this intuitive platform, users will be able to effectively understand and manage their data, leading to faster decision-making and enhanced deliverable quality. This paper will review the purpose of each module, offer practical guidance on the app's structure and usage, and discuss coding strategy.

OVERVIEW OF THE APPLICATION

Upon accessing the home page, users will be prompted to upload a dataset they wish to examine. Once the dataset is successfully uploaded, users will be guided to navigate to the four data checking modules.

The first module, Records with Missing Key Variables, is developed to detect absent values in crucial fields that are vital for the analysis. After conducting this evaluation, the application provides a filtered dataset that highlights only those entries exhibiting missing values for the specified variables. This report holds significant importance because it enables users to quickly identify and address gaps in the data, ensuring the completeness of the dataset for subsequent analyses.

The next module, the Find Duplicates Module, is designed to identify any duplicate records within the dataset. By allowing users to select a specific set of variables, the application will present the duplicate entries corresponding to those chosen variables. This functionality is essential for programmers and stakeholders, as it helps ensure data accuracy, ultimately leading to more reliable results.

The third module, the Date Comparison module, enables users to compare two date variables from the uploaded dataset and choose a comparison method—such as greater than, less than, or equal to. The module then generates a dataset that reflects the results of the selected comparison, filtering and printing the records accordingly. This feature is essential because it helps verify the accuracy and validity of date entries within the dataset; by flagging instances according to the user's chosen criteria, it allows users to promptly address any date discrepancies and maintain the integrity of the data.

Finally, the Record Discrepancy module assesses discrepancies between two datasets. Users will be prompted to upload an additional dataset and select variables they would like to compare between the

newly uploaded dataset and the one they uploaded previously on the homepage. The system will compare the two datasets using the selected variables as identifier, and generate reports that include three components for the selected variables:

1. Matched records between two datasets
2. Records that exist only in the dataset uploaded on the home page.
3. Records that are present only in the newly uploaded dataset.

This comparison feature enables users to identify similarities and differences between two datasets based on the specified variables. For instance, it helps users pinpoint key variables essential for merging datasets effectively. Additionally, this feature enables users to share reports with stakeholders for the review of unmatched records that require data resolution. This enhances communication and clarifies data insights, making it easier for teams to collaborate and make informed decisions.

Each module offers feature to download the generated data reports, ensuring users can easily access their findings.

INTERFACE WALKTHROUGH

The application is designed to provide an intuitive interface for user to easily navigate. The following section provides a detailed demonstration of how to use the R Shiny application through each step.

HOME PAGE

Upon launching the app, users are greeted by the 'Home' page, where they are prompted to upload data file. The application supports multiple file formats, including Excel, which will prompt you to select a sheet upon uploading, CSV, and SAS. As illustrated in Figure 1, following a successful upload, a button appears to direct the user to the checker modules.

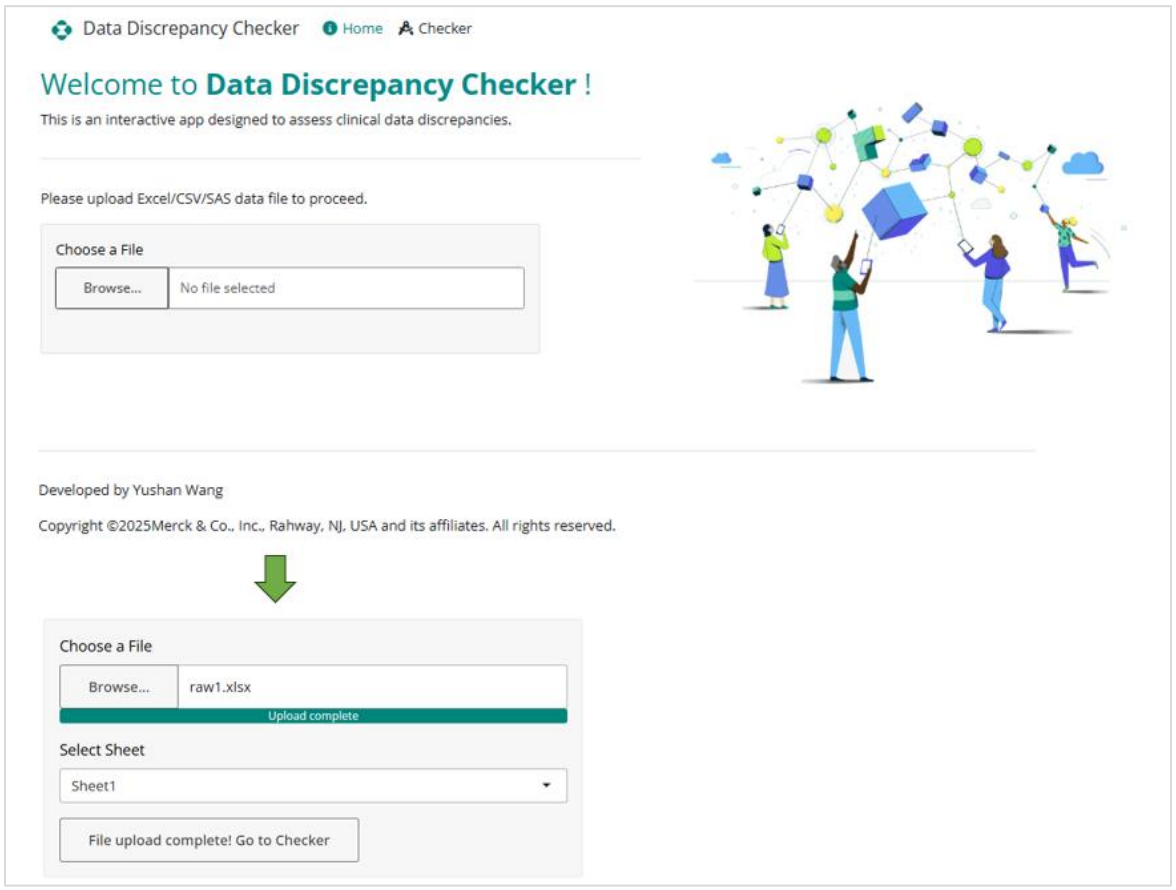

Figure 1. Home Page and File Upload

DATA CHECKING MODULES

Once the user enters the checker, they will find four data checking subtabs presented:

1. Records with Missing Key Variables
2. Find Duplicates
3. Date Comparison
4. Datasets Comparison.

At the bottom of each of the four modules, there are two download buttons that allow users to download the resulting dataset in the corresponding formats of XLSX and CSV for further examination on their local machines.

In the following sections which provide walkthrough of each module, a mock dataset is utilized for demonstration purposes.

Records with Missing Key Variables

The first tab is titled "Records with Missing Key Variables." It prompts the user to select variables from the uploaded dataset and then displays all the records that have missing entries for any of the selected variables, as shown in Figure 2.

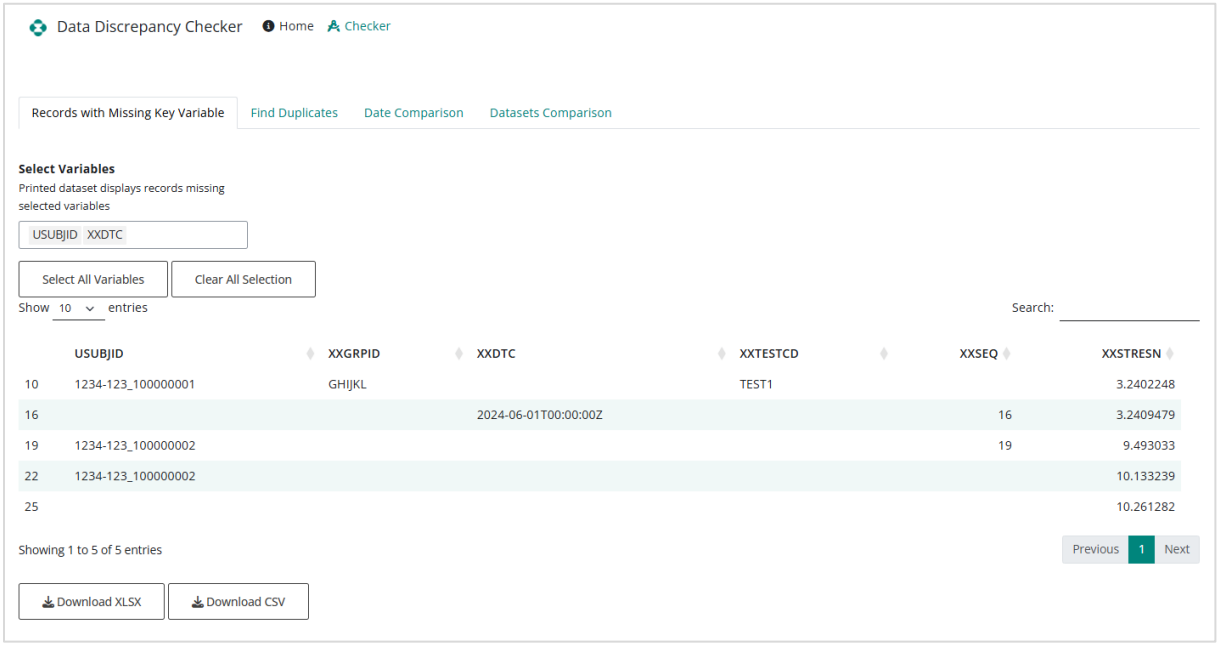

Figure 2. Records with Missing Key Variables

Additionally, there are two buttons positioned together beneath the variable selection box. One button allows users to select all variables, enabling them to examine the entire dataset for any missing values, while the other button provides a convenient option to clear any selections within the box.

Find Duplicates

The second tab is labeled "Find Duplicates." As shown in Figure 3, this module allows users to select variables from the uploaded dataset and then presents all records containing duplicate entries based on the chosen set of variables. It is important to note that this module identifies and displays records that are duplicated across all selected variables rather than showing duplicates for each individual variable separately. Beneath the variable selection box, there are two buttons: one that allows users to select all

variables for check of records that are identical across all variables in the dataset, and another that easily clears all selections.

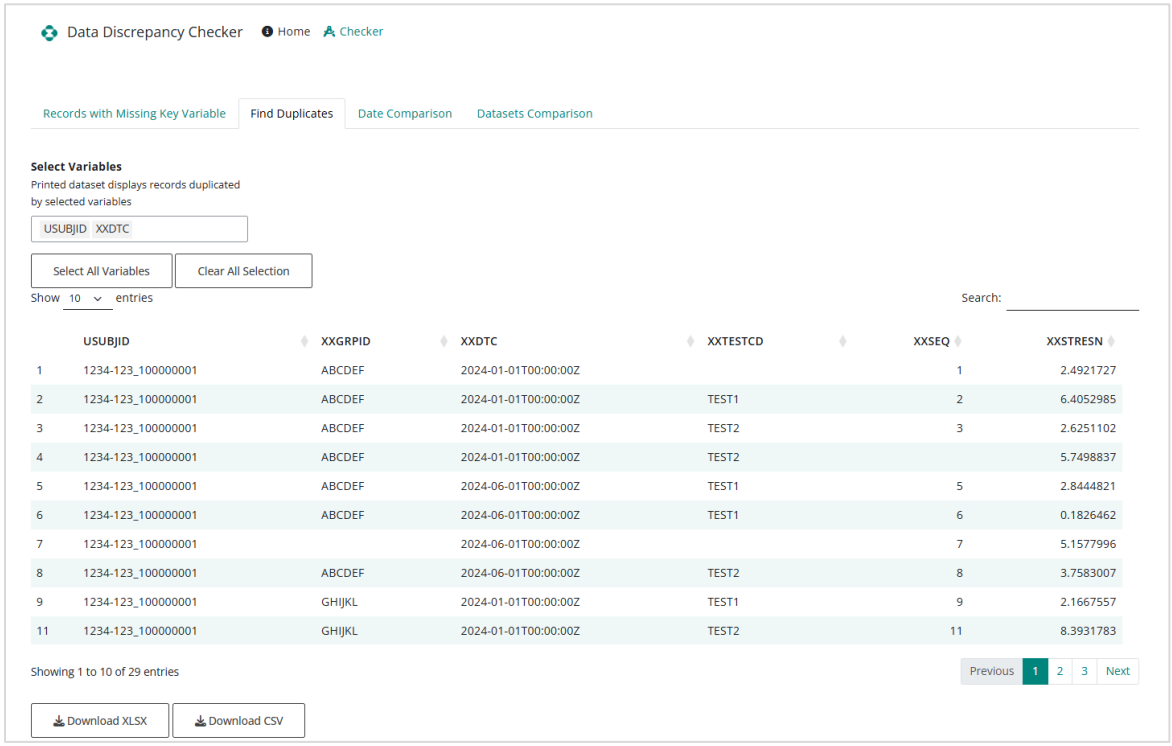

Figure 3. Find Duplicates

Date Comparison

The third tab provides the comparison between two date variables from the uploaded dataset and displays records that match the user's selection. When users access the variable selection option, warning messages will be triggered, as shown in Figure 4, if a chosen variable is not of date type.

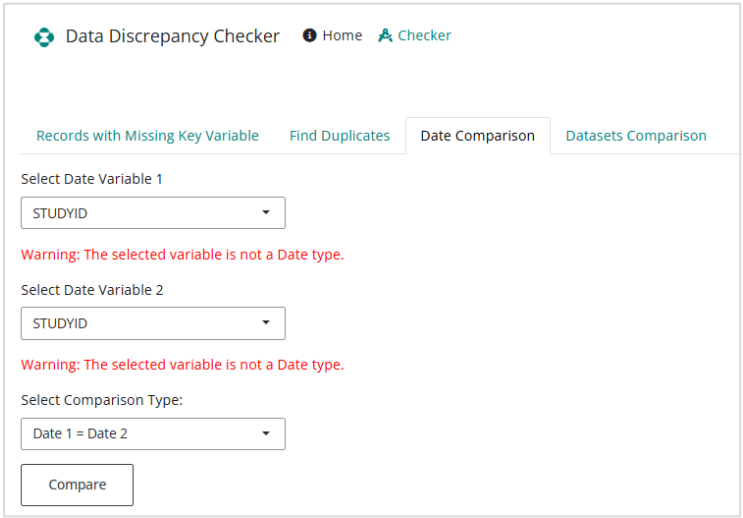

Figure 4. Warning Message in the UI for Date Comparison Module

After the user specifies the date variables and selects the comparison type (greater than, less than, or equal to), followed by selecting ‘Compare,’ the module generates outputs that meet the specified criteria. As illustrated in Figure 5, the selected Date Variable 1 and 2 are RFSTDTC and RFENDTC respectively,

with the chosen comparison criteria being Date 1 > Date 2, represented as RFSTDTC > RFENDTC. No records are returned, indicating that all entries do not satisfy the condition RFSTDTC > RFENDTC. This outcome is logically consistent, as the Subject Reference Start Date/Time (RFSTDTC) should precede the Subject Reference End Date/Time (RFENDTC).

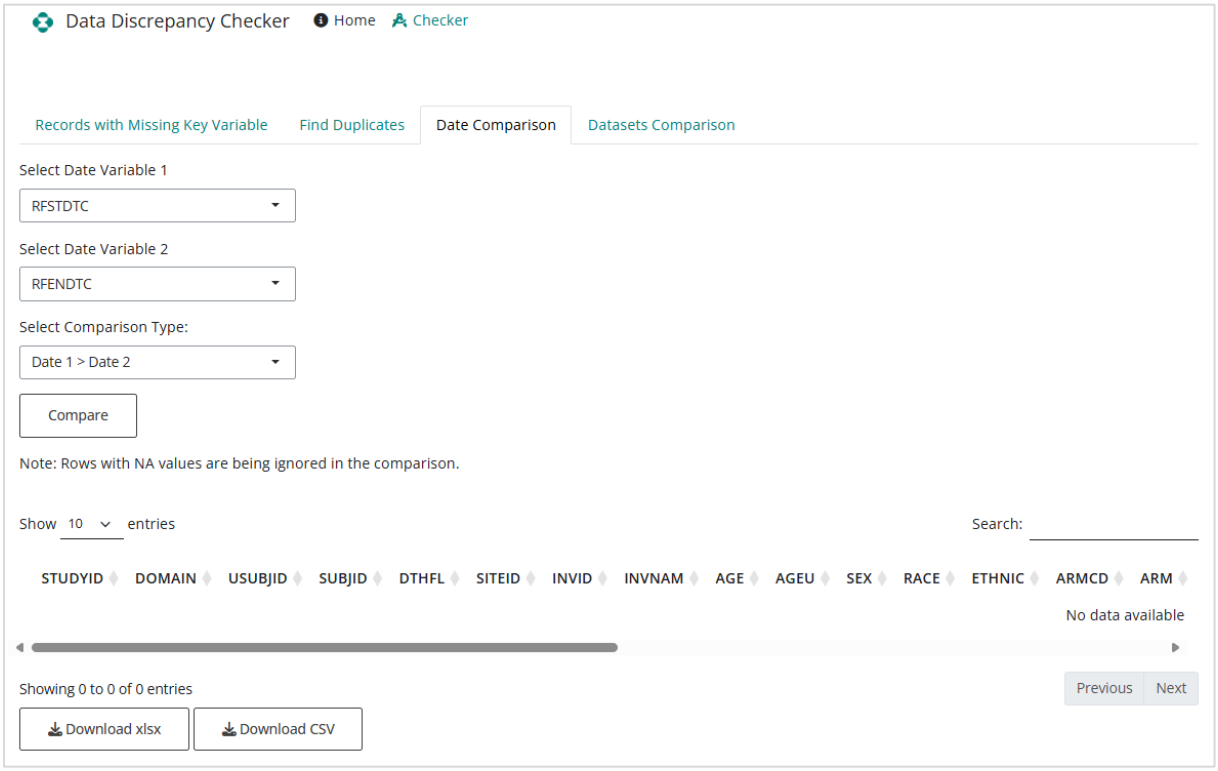

Figure 5. Date Comparison

Datasets Comparison

The final tab provides functionality for comparing two datasets. Users are prompted to upload additional Excel, CSV, or SAS files beyond the dataset initially uploaded on the homepage. They can then select specific variables from each dataset for comparison. Additionally, there are prompts to guide users in selecting target variables from both datasets in the same order to ensure accurate comparisons. For instance, as shown in Figure 6, users might select “SUBJID”, “XXREFID”, and “XXDTC” from one dataset and “Subject#”, “Refid”, and “Date” from the other, as these variables correspond to each other.

Subsequently, the module displays the comparison results and includes a selection dropdown that allows users to choose which of the three categories they wish to view. These categories are: Common records that are present in both datasets; Records that exist only in the dataset uploaded previously on the homepage; Records that exist solely in the dataset uploaded under this tab.

Note that the application only evaluates and prints records for the selected variables, rather than all variables in the datasets.

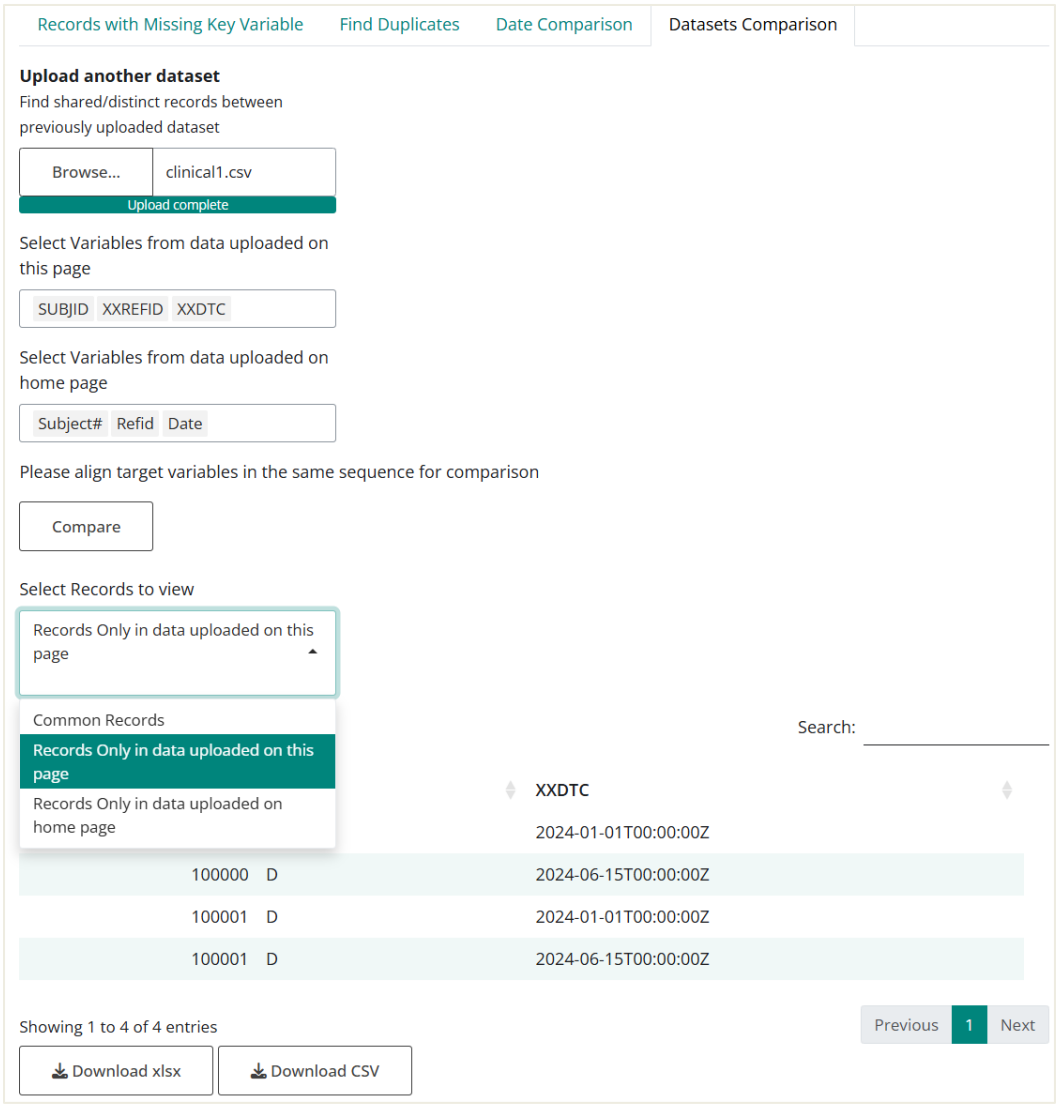

Figure 6. Datasets Comparison

CODING STRATEGY

The effectiveness of the application is based on its use of the R Shiny framework alongside specialized R functions. Each feature of the application has been developed as a separate function, which improves maintainability and flexibility. This design makes it easier to update individual features and supports function testing. Changes to one function do not unintentionally affect the entire application, reducing the risk of errors throughout the codebase.

Each function is seamlessly integrated with the Shiny server code. For instance, I have created a function called *compare_dat* that generates the datasets comparison results, which is stored in a separate file location. Rather than placing the complex comparison logic directly within the Shiny reactive expression, I encapsulate the function within a reactive context as outlined in Program 1:

```
comp_result <- reactive({
  req(input$homevar)
  req(input$compvar)
  if(input$compareButton > 0) {
    compare_dat(compdat(), dat(), input$compvar, input$homevar)
  }
})
```

```
output$comptb <- renderDT({
  req(comp_result())
  req(input$selectView)

  compdat <- NULL
  if (input$selectView == "Common Records") {
    compdat <- comp_result()$common
  } else if (input$selectView == "Records Only in data uploaded on this page") {
    compdat <- comp_result()$only_in_data_uploaded_thispage
  } else if (input$selectView == "Records Only in data uploaded on home page") {
    compdat <- comp_result()$only_in_data_uploaded_homepage
  }

  return(compdat)

})
```

Program 1. Embedding Function into Reactive

The code defines a reactive expression called *comp_result*, which returns the comparison result of two datasets based on user inputs. It first verifies that the necessary inputs are available and not NULL using the req() function, which stops execution if any input is invalid. Next, it checks if the compare button has been clicked at least once (i.e., its value is greater than 0). If the button has been clicked, it calls the *compare_dat* function with the specified parameters. The list of results calculated by *compare_dat* function are stored in *comp_result* as a reactive expression, allowing seamless transition to subsequent processing.

The *output$comptb* section in Program 1 utilizes the *renderDT* function to create a dynamic data table based on user input. It first checks for the necessary inputs. After that, the code identifies the appropriate subset of results to display from the reactive *comp_result()*, determined by the user's selection as indicated by *input$selectView* .This selection includes options for "Common Records," "Records Only in Data Uploaded on This Page," and "Records Only in Data Uploaded on Homepage".

The comp_result() function generates three distinct data frames calculated by compare_dat, using the user's selected variables for comparison and displaying only the columns corresponding to those selected variables.

1. If "Common Records" is selected, the application shows the data frame containing common records between two datasets.
2. If "Records Only in data uploaded on this page" is chosen, the application prints the data frame containing records solely in the newly uploaded data.
3. If "Records Only in data uploaded on home page" is selected, the application displays the data frame containing records only in the data uploaded at home page.

This approach ensures that the appropriate result is displayed in the data table, thereby allowing users to easily navigate through different categories of records while maintaining an efficient code flow.

Additionally, a significant challenge arises after users upload their datasets, as the variables imported into R may not be in the desired format. It is crucial to address these discrepancies before proceeding with dataset evaluation. For example, date and time values are frequently imported as character strings in R, which hinders proper date and time comparisons. To address this issue, I developed three functions specifically designed to facilitate the importing and formatting of data for Excel, CSV, and SAS file formats.
It is important to note that the SAS function I developed solely supports date representations in ISO format (YYYY-MM-DDThh:mm:ss.ddd) due to dependencies on the Admiral package, which is designed to streamline the analysis of clinical trial data in compliance with regulatory standards.

When a user uploads a dataset, the application automatically detects the dataset type and directs it to the appropriate processing function. These functions validate the data entries and convert them into the correct format if necessary, ensuring that the dataset is ready for the next stage of evaluation.
This function-based coding approach enables a clearer separation of the code flow, as the complex logic for evaluation is now encapsulated within its own function rather than being interwoven directly into the main application logic. In addition, the application uses reactive expressions to embed these functions and their results, allowing for additional processing as needed. This organization not only enhances clarity but also promotes the reusability of the functions in different contexts, should the need arise, while maintaining a coherent flow of code. By structuring the code in this manner, the application becomes more robust and easier to debug.

CONCLUSION

This R Shiny application serves as a valuable tool for statistical programmers, streamlining the process of data discrepancy detection. By automating essential checks, it enhances programming and communication efficiency, improves data understanding, and contributes a higher quality of deliverables. As clinical trial data become increasingly complex, leveraging innovative tools like this application is crucial to ensure reliable outcomes.

ACKNOWLEDGMENTS

I would like to express my sincere appreciation to Jeff Cheng and Sandeep Meesala at Merck & Co., Inc., Rahway, NJ, USA for their exceptional leadership and guidance throughout the app development process. Additionally, I am immensely thankful to my colleague Ryan Hernandez-Cancela at Merck & Co., Inc., Rahway, NJ, USA for generating mock data to facilitate application evaluation.

CONTACT INFORMATION

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

Yushan Wang
Scientist, Statistical Programming, Merck & Co., Inc., Rahway, NJ, USA
yushan.wang@merck.com
Yushan Wang - Scientist, Statistical Programming - Merck | LinkedIn