Paper DV03

Enhancing PK Data Exploration and Visualization by using a R Shiny Application

Runcheng Li, Hanyang Sun, Yiyuan Zeng, Jiannan Kang
Merck & Co., Inc, Rahway, NJ, USA

ABSTRACT

This paper introduces an R Shiny web application for PK data exploration and visualization. This tool offers a range of powerful features, including rapid data review, checklist-driven validation, dynamic plot generation, and downloadable reports for future use. By leveraging R Shiny's interactive capabilities, the tool streamlines workflows, enables real-time data interaction, provides comprehensive and efficient review, checking, and visualization for PK analysis. The user-friendly interface and dynamic features empower researchers in fostering informed decision-making and expediting research outcomes without the need of complex programming scripts.

INTRODUCTION

Pharmacokinetic (PK) analysis is one of the key objectives along with drug safety analysis and efficacy analysis in clinical trial for drug development. As the complexity and volume of PK data rises, the demand for an efficient, accurate, and user-friendly tool to explore and visualize data becomes increasingly critical. This paper introduces an innovative solution to this challenge: a R Shiny web application designed specifically for PK data review. With features such as quick data review, dynamic plotting, and checklist-driven validation, this tool not only accelerates the analysis process but also enhances the precision of its outcomes.

There are three sections in this paper. Firstly, it discusses the initiation of the application with focus on the synergy between identifying a business need and collaborating with PK scientists. This partnership ensures that the tool not only addresses the complex nature of PK data but also aligns with the real-world needs and user workflows. Secondly, the paper explains the app's architecture and benefits, illustrating how its user-centric design and powerful features can make PK data reviewing more insightful. Finally, it delves into the technical details behind the app by showcasing how R Shiny's capabilities are leveraged to create a business-driven application.

PROJECT INITIATION

The accuracy of PK data is always important in the success of the analysis. There are data issues that might not surface during routine data cleaning and reconciliation processes. To effectively identify these data issues, thorough understanding of PK concepts is required. For example, reviewing an individual subject's PK profile might reveal anomalies that are not apparent through standard data checks. These could include unexpected absorption patterns, clearance rates, or other pharmacokinetic parameters that are critical to the drug profile but can be overlooked due to lack of specialized PK knowledge. However, there could be some challenges when reviewing the ADPC data. Firstly, being unfamiliar of CDSIC standard can be challenge for PK scientists to fully understand the data structure and standard requirement. Secondly, PK scientists often find themselves in the position of having to create their own programming scripts to perform data checks. This means that they need to take time and effort to write and debug code to compare the analysis dataset ADPC against the source data. All these challenges underscore the need for a tool that is both intuitive and tailored to the specific needs of PK data review.

From the discussion with PK scientists, they conveyed their essential requirements for the application:

- Data Upload and Review: The ability to upload various datasets such as ADPC, ADSL, SDTM. They need a data table view with options to filter data and add filters as needed, coupled with the capacity for data browsing to visualize different variables in a dataset.
- Data Validation and Checks: The generation of a downloadable ADPC checklist to validate the data against the input source datasets.
- Visualization and Reporting: The application should be able to provide dynamic data plotting capabilities, including correlation plots and PK over time plots, with variable selection for subgroup stratification, additionally, the ability to zoom in/out, adjust coordinates, and export these plots and reports.

In response to these requirements our application was structured into three distinct modules: Data, Check, and Plot.

Data Module:

- Users can upload the necessary datasets including ADSL, ADPC, and SDTM datasets.
- For each uploaded dataset, a data table view allows for comprehensive data review and customized filters can be added as needed.

- Data browsing to visualize user-selected variables in the dataset.

Check Module:

- Generate ADPC checklist and allow user to download the report.
- Dynamically select the checks as needed.

Plot Module:

- Correlation and PK over time plots can be created, providing insightful visualizations of the PK data.
- Customized selection and filters could be applied based on the analysis need.

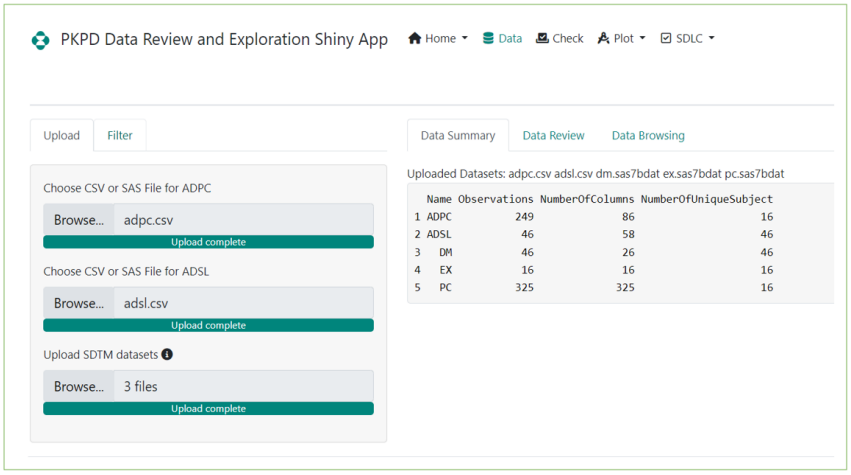
In creating these modules, we ensured that the application not only fulfilled the expressed needs but also offered a streamlined and user-friendly experience. This collaborative effort between programmers and PK scientists resulted in a tool that not only elevates data analysis but also embodies the specific demands and workflows of its users.

INTERFACE AND DEMOSTRATION

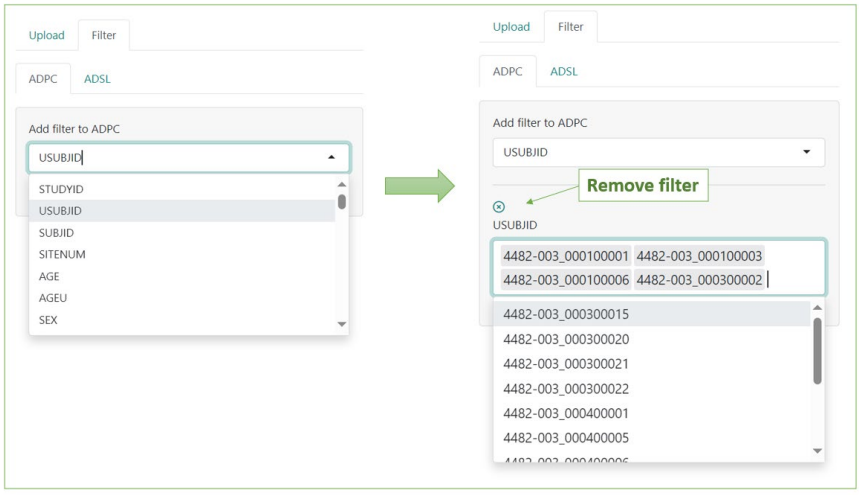
In this section, we will take a closer look at how each part of our R shiny App works.

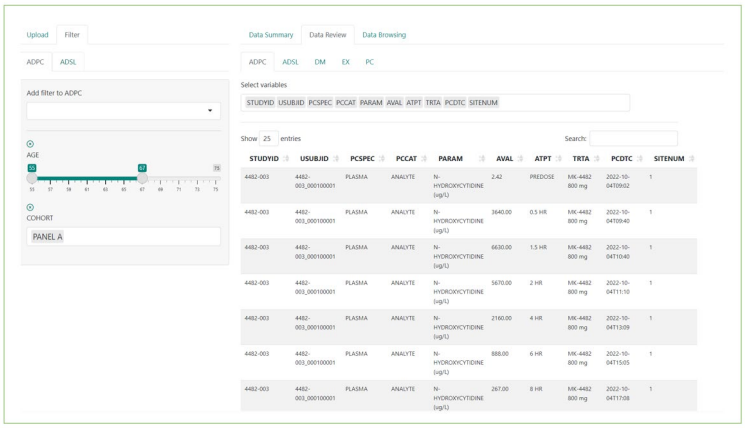
DATA MODULE

In the 'Data' section, users can easily upload the datasets, which may include ADPC, ADSL, and SDTM datasets. Once uploaded, the app provides a concise summary on the right side of the screen to verify the successful import of the data. The app is designed to accommodate with both SAS and CSV formats.

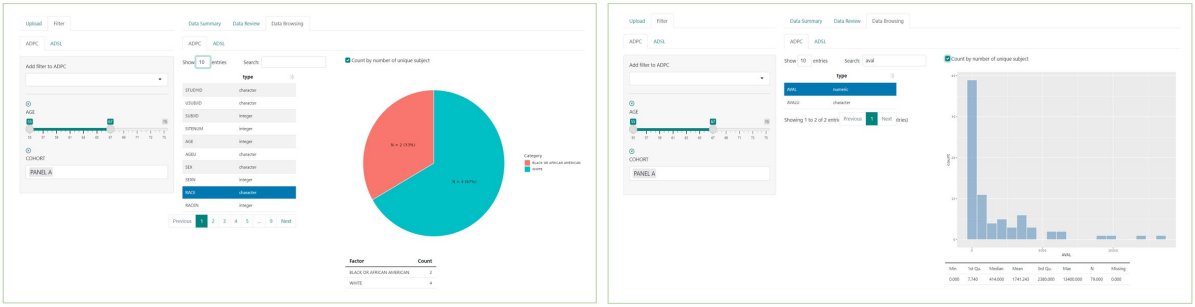


Following the upload, users can navigate to the 'Data Review' tab for a data table review of the datasets. This feature allows for the customization of the displayed variables, ensuring that users can focus specifically on the information most relevant to their interest. Additionally, users have the option to apply filters to their selected dataset, enhancing the specificity of their review. Adding a filter is straightforward: simply choose the variable name, and a dropdown list will appear for character variables, or a slider will be presented for numeric variables. Multiple filters can be added simultaneously. When an adjustment is needed, each filter has an 'X' button, allowing for quick and easy removal.



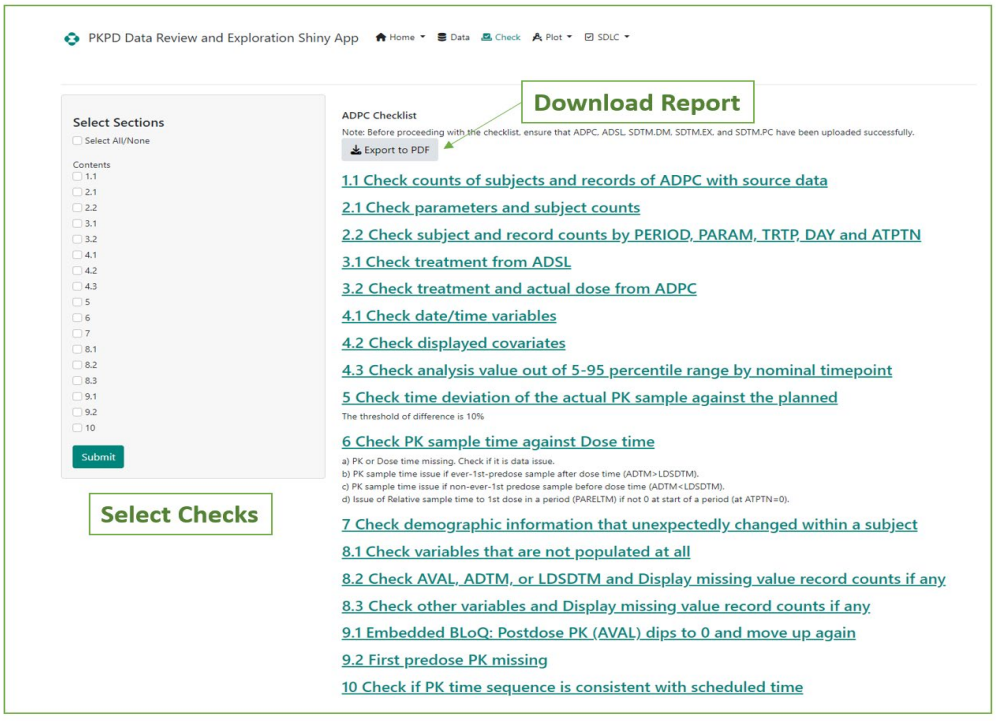


An alternative method for variable review is available under the ‘Data Browsing’ tab. Here, users can select a specific variable to observe the distribution of its values. The App will display a pie chart for categorical variables and a histogram for numeric ones, providing a clear visual representation of the data. Alongside these plots, a summary table presents key statistics for a more comprehensive understanding of the variable. The filter feature remains accessible in this tab, allowing users to refine their view of the data. Modifications to these filters can also be made easily, ensuring a flexible and insightful data browsing experience.



CHECK MODULE

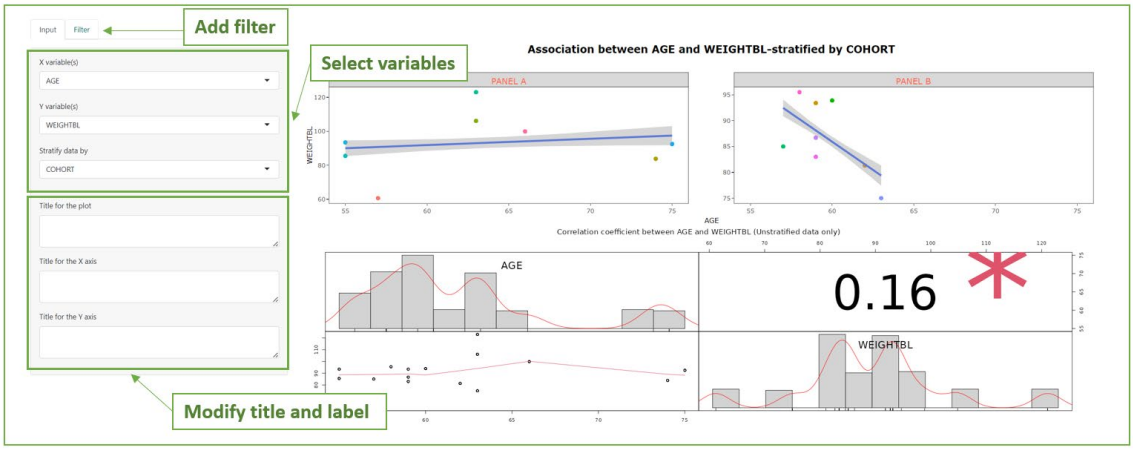
In the ‘Check’ tab, users are empowered with a feature to swiftly verify the ADPC against the uploaded datasets. The primary function of this checklist is to detect potential programming errors or data issues, the checking results could be helpful for modelers to have a preliminary review of the ADPC dataset. Users also have the flexibility to choose specific checks they wish to apply. Moreover, the results can be exported into a PDF file, facilitating an easy and accessible way to document and share findings.



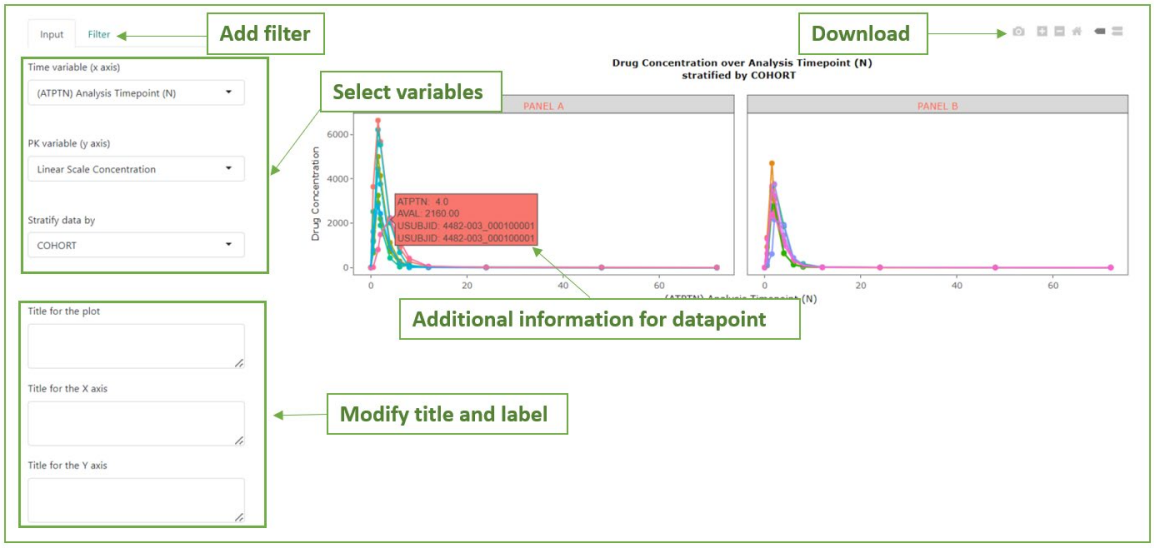
PLOT MODULE

Data visualization is a critical feature of our tool, and in response to feedback from PK scientists, we currently create two types of plots: the correlation plot and the PK over time plot.

The correlation plot is useful for users who are interested in examining the relationship between two variables. Within this feature, users can select any two numeric variables from the ADPC dataset to explore their inter dependencies. For a more nuanced analysis, the tool also includes a stratification function, allowing users to generate multiple plots based on different groupings or conditions. Additionally, an independent filter function is available, providing users the flexibility to refine the data further according to their specific analytical requirements. Here is an example of the correlation plot for variable AGE and WEIGHTBL stratified by COHORT.



The 'PK over time' plot is another helpful graph, offering modelers a clear view of the PK profile for selected subjects. Within this feature, users can select time variable from the ADPC dataset as the x-axis and use AVAL as the concentration variable on y-axis. To accommodate different data perspectives, there is an option to switch between linear and log scales for plotting. Users can also use a stratification variable to segment the data further. Like the correlation plot, it also includes an independent filter function, allowing for the application of customized filters to refine the data. By incorporating Plotly [1], a R package for generating interactive graphs, the user experience is significantly improved. Users can interact with the plot in real time, zooming in on details, panning across the plot, hovering over data points to review additional information, and downloading the plot in png format. Here is an example of the concentration vs analysis timepoint plot in linear scale and stratified by cohort.



R PROGRAMMING BEHIND THE APP

R shiny is a powerful tool, and throughout our development, we identified two critical elements to meet business objectives. First, the capability to upload data and adapt the user interface according to the datasets is crucial. This is where R Shiny's reactive values come to play, allowing us to create dynamic UI with flexibility. Secondly, the demand for sophisticated graphing solutions to accommodate the specific needs of PK scientists bring us to Plotly. This package enhances the application by providing advanced interactive plotting features, thereby greatly improve the user experience and meet the requirements of PK data visualization.

DYNAMIC UI WITH SHINY

Rather than predefine the UI, we need to use uiOutput (You can find more details in Mastering Shiny, Chapter 10 Dynamic UI [2]). We can write the UI part on the server side to make sure the UI is rendered correctly with the input dataset. Let's use the data filtering module as an example.

```
DataFilterModuleUI <- function(id){
  ns <- NS(id)
  tagList(
    uiOutput(ns("AddFilter")),
    tags$hr(),
    uiOutput(ns("ActiveFilter"))
  )
}
```

On the server side, things get much more complex as we need to both generate and continually update the UI in response to user actions. Initially, we create the 'AddFilter' component, which is designed to generate a dropdown list for every variable present in the uploaded dataset.

```
output$AddFilter <-renderUI({
    req(datain())
    selectInput(inputId = ns("add_filter"),
                label = FilterLabel,
                choices = c("",variables_in_filter()),
                selected = "",
                multiple = FALSE
    )
})
```

Next, we need to establish active filters for each selected variable. In fact, we create a UI component for all variables but keeping them hidden initially; only the ones chosen by the user will be shown. We can accomplish this by using 'shinyjs' to toggle the visibility of each UI components. Additionally, it's necessary to determine whether to use 'slideInput' or 'selectizeInput' for the input variables, based on certain conditions.

```
output$ActiveFilter <- renderUI({
    filter_list <- lapply(variables_in_filter(), function(x){
      if (is.numeric(datain()[[x]]) & !(x %in% AsCharVarList)) {
        rng <- range(datain()[[x]], na.rm = TRUE)
        div(
         shinyjs::hidden
          #Add removal button
          actionLink(ns(paste0(x, "_remove")), label="",
                     icon =icon("circle-xmark", lib = "font-awesome", class = "remove pull-right")),
          sliderInput(ns(x), x, min = rng[1], max = rng[2], value = rng)
         )
        )
      } else {
        div(
         shinyjs::hidden(
          #Add removal button
          actionLink(ns(paste0(x, "_remove")), label="",
                     icon =icon("circle-xmark", lib = "font-awesome", class = "remove pull-right")),
          selectizeInput(ns(x), x, choices = NULL, multiple = TRUE)
         )
        )
      }
    })
    do.call(tagList, filter_list)
})
```

The final piece involves updating the dataset when a filter is applied and ensuring any changes are reversed when a filter is removed. This is where functions like 'updateSelectizeInput' and 'updateSliderInput' come into play, allowing us to create a responsive and interactive UI within the server.

```
 #hide/display filter
 observe({
   req(datain())
   lapply(variables_in_filter(), function(x) {
     val <- datain()[[x]]
     observeEvent(req(input[["add_filter"]] == x),{
       shinyjs::show(x)
       shinyjs::show(paste0(x,"_remove"))
       if (is.character(val)|(x %in% AsCharVarList)){
         levs <- levels(as.factor(val))
         updateSelectizeInput(session, x, choices = levs, server = TRUE)
```

```
      }
    })
    observeEvent(input[[paste0(x,"_remove")]], {
      shinyjs::hide(paste0(x))
      shinyjs::hide(paste0(x,"_remove"))
      if (is.numeric(val) & !(x %in% AsCharVarList)) {
        rng <- range(val, na.rm = TRUE)
        updateSliderInput(inputId = x, min = rng[1], max = rng[2], value = rng)
      }
      else {
        levs <- levels(as.factor(val))
        updateSelectizeInput(inputId = x, choices = levs, selected = NULL, server= TRUE)
      }
    })
  })
})
```

By combing all those pieces together, we can generate a useful module to display the dynamic UI for filtering function. We use the similar idea to complete all other parts in our Shiny App to make sure the UI will be rendered correctly based on the uploaded data and user activities.

USE PLOTLY FOR DATA VISUALIZATION

In our Shiny App, we combine the 'ggplot2' with 'plotly' to create an interactive visualization of PK data. Here is an example of PK vs time plots.

```
p <- ggplot(adpc_df.new.pk(), aes(x=UQ(x.var), y=UQ(y.var)))+
            geom_point(aes(color=USUBJID), na.rm = T)+
            geom_line(aes(group=USUBJID, color=USUBJID), na.rm = T)+
            facet_wrap(vars(UQ(strata.var)), nrow = n.row,
                       scales = "free_y")+
            labs(title=title.main.1,
                 x=title.x,
                 y=title.y,
                 caption = input$footnote_pk,
                 color="",
                 shape='',
                 group='')+
            theme_bw()+
            theme(plot.title = element_text(face='bold',hjust = 0.5,
                                            margin = margin(6,0,30,0),
                                            size=10),
                  axis.title.x = element_text(size = 10),
                  axis.title.y = element_text(size = 10,
                                              margin=margin(r=10)),
                  panel.grid.major = element_blank(),
                  panel.grid.minor = element_blank(),
                  legend.text = element_blank(),
                  legend.position = 'none',
                  strip.text.x=element_text(size=10,color="tomato",face="bold.italic"),
                  strip.text.y = element_text(size=15),
                  axis.ticks.x=element_blank(),
                  axis.text.x = element_blank(),
                  plot.caption = element_text(size=8, hjust=0))
          p <- ggplotly(p) %>%
            config(modeBarButtonsToRemove=
                     c("zoom2d","pan2d","select2d","lasso2d",
                       "autoScale2d","toggleSpikelines"),
                   displaylogo=F) %>%
            layout(margin=list(t=130))
```

First, we initiate the base plot using ggplot2. This involves creating a ggplot object with 'ggplot()', then adding scatter points and lines using 'geom_point()' and 'geom_line(). We use reactive dataset as input to ensure our plot dynamically responds to data changes. Stratified plotting is achieved with 'facet_wrp()', which separates the data into distinct facets based on levels of a specified variable. Then, we enhance the plot's aesthetic appeal and clarity. The 'labs()' is used to define the plot's labels and titles, while 'theme_bw()' applies a clean, professional theme as a starting point. Further refinements are made using 'theme(…)', allowing us to modify various elements like text size, panel grids, legend, etc. The last step is converting to an interactive Plotly object. Utilizing 'config()' and 'layout()', we customize the plot's interactive features and optimize its overall layout for a superior visualization.

It is designed to produce a detailed and interactive plot that can be highly customized and be responsive to user inputs, making it suitable for dynamic presentation of PK data.

CONCLUSION

This paper has presented a comprehensive exploration of our R shiny web application for PK data exploration and visualization. By incorporating a flexible data upload system, dynamic user interface, and advanced visualization techniques, we have ensured that the application not only meets the diverse needs of PK scientists but also provides an intuitive, user-friendly experience. Moreover, the applications' development highlights the importance of understanding and addressing the unique challenges faced by PK scientist.

As we conclude, the fusion of R Shiny's interactive capabilities with understanding of real business can greatly improve the efficiency and accuracy in our work. Moving forward, we intend to delve deeper into the specific requirements of PK related analysis and continuously enhance our application with more advanced features and functionalities.

REFERENCES

[1] Plotly Open Source Graphing Libraries (https://plotly.com/graphing-libraries/)

[2] Mastering Shiny (https://mastering-shiny.org/index.html)

ACKNOWLEDGMENTS

We would like to thank Greg Zhou, Bela Patel for business endorsement and Jing Su, Brittany Walker for business leadership. We also would like to thank all team members who provided valuable comments and user test feedbacks.

CONTACT INFORMATION

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

Author Name: Runcheng Li

Company: Merck & Co., Inc

Email: runcheng.li@merck.com