PharmaSUG 2022 - Paper DV-82

Building Dashboards for Data Review and Visualization using R Shiny

John Shaik, Seagen Inc., Bothell WA
Sreeram Kundoor, Kite Pharma Inc., Bothell WA

ABSTRACT

In the pharmaceutical and biotech industry, SAS® is a widely used software for clinical trial data analysis and to prepare CDISC-compliant data sets for FDA submission. Though SAS® is a robust and feature-rich programming language, access to features that enable development of interactive data exploration tools may be beyond the scope of the license of some companies. In scenarios where ad-hoc or exploratory analyses are desired with quick access to data, R, as an open-source software, is suitable and freely available with large community support behind it.

Such quick access by cross-functional teams can be especially beneficial for efficacy data in single-arm clinical trials, for which interactive dashboards can easily be built using the Shiny package in R. These dashboards, also known as Shiny web apps, can be tailored to the specific needs of the study team. In this paper, a brief introduction to R programming and building dashboards using the R Shiny package to explore and visualize efficacy data in oncology clinical trials – including summary statistics and graphs such as waterfall, KM, and box plots – will be discussed in more detail.

INTRODUCTION

Often statistical programming teams or individuals receive numerous ad-hoc requests from cross-functional teams for various analysis. Every so often these requests are repetitive, tedious and time consuming. To overcome that tedious process and facilitate the cross-functional team's request, we have come up with an R dashboard which does the job for the users. Using the interactive R dashboard, the user can visualize the data and the summaries as soon as the data is available instead of waiting for the programmer to generate them. This interactive dashboard can help visualize various commonly used plots in clinical research and especially the oncology therapeutic area like scatter plot, box plot, KM plot and waterfall plot. This paper will provide a brief explanation of various components of the dashboard, how to use it, limitations, and the relevant code.

R AND R SHINY BASICS

R is a popular programming language used for statistical computing and graphical presentation. Its most common use is to analyze and visualize data. It is becoming more popular in both academia and industry as it is an open-source, free software that works on all platforms. RStudio® is the most popular IDE to write R programs and develop dashboards. R libraries are a collection of functions to perform certain tasks that can be readily reused again. R has many libraries built-in and then there are many user-created libraries readily available for download from the CRAN website.

Some basic operations using the R language:

```
> a <- 5          # assign 5 as the value of a
> print(a)        # print the value of a
[1] 5
> b <- a + 5      # addition
> print(b)
[1] 10
> c <- a+b*3      # addition and multiplication
> print(c)
[1] 35
> d <- c(1,2,3) # assign multiple integer values to a vector
> print(d)
[1] 1 2 3
> e <- c(3,4,5)
```

```
> de <- d+e     # vector addition
> print(de)
[1] 4 6 8
> colors <- c('red','blue','green') # assign multiple character values to a vecor
> colors        # print values in vector colors
[1] "red"   "blue"  "green"
```

R Shiny is merely a package in R to build dashboards. This package uses the R language to build web interfaces instead of HTML or JAVA. It provides a relatively quick and efficient method of creating dashboards for data analysis and visualization. To demonstrate its capabilities, we have built a demo dashboard to import data in different file formats such as CSV, TAB, and SAS7BDAT, to summarize the data and generate plots.

ACCESSING THE DASHBOARD

To build the dashboard, the user needs to install R, RStudio® software and R libraries. Those libraries are listed in the code provided in the appendix. After the software is installed, the user needs to execute the code that opens a window which consists of the dashboard. Dashboard hosted publicly on shinyapps.io can be accessed by anyone and the data will not be stored or accessible to the authors. Dashboard and R code to build it can be accessed below.

Dashboard

R Shiny Code

COMPONENTS OF THE DASHBOARD

The dashboard mainly consists of two components (tabs)

1. Data tab
2. Plots tab

DATA TAB

The data tab contains one side panel and a main panel. Using the side panel, when the user becomes aware that data is ready to access, they can hit the upload file tab, which allows them to browse the data in a folder of their choice and select the desired data set or file. The utility can read CSV, Excel, and SAS (sas7bdat) files. Once the file is loaded, the data and its attributes will be displayed as rows and columns on the right side of the panel as shown in Figure 1 and Figure 2. This enables the user to browse through the data, review and explore the data points.

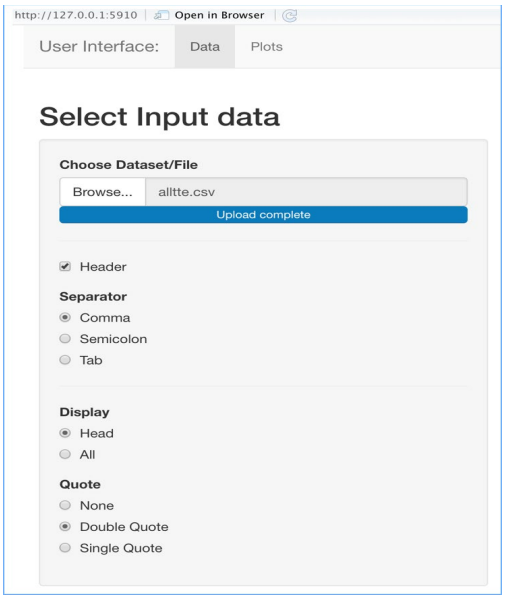
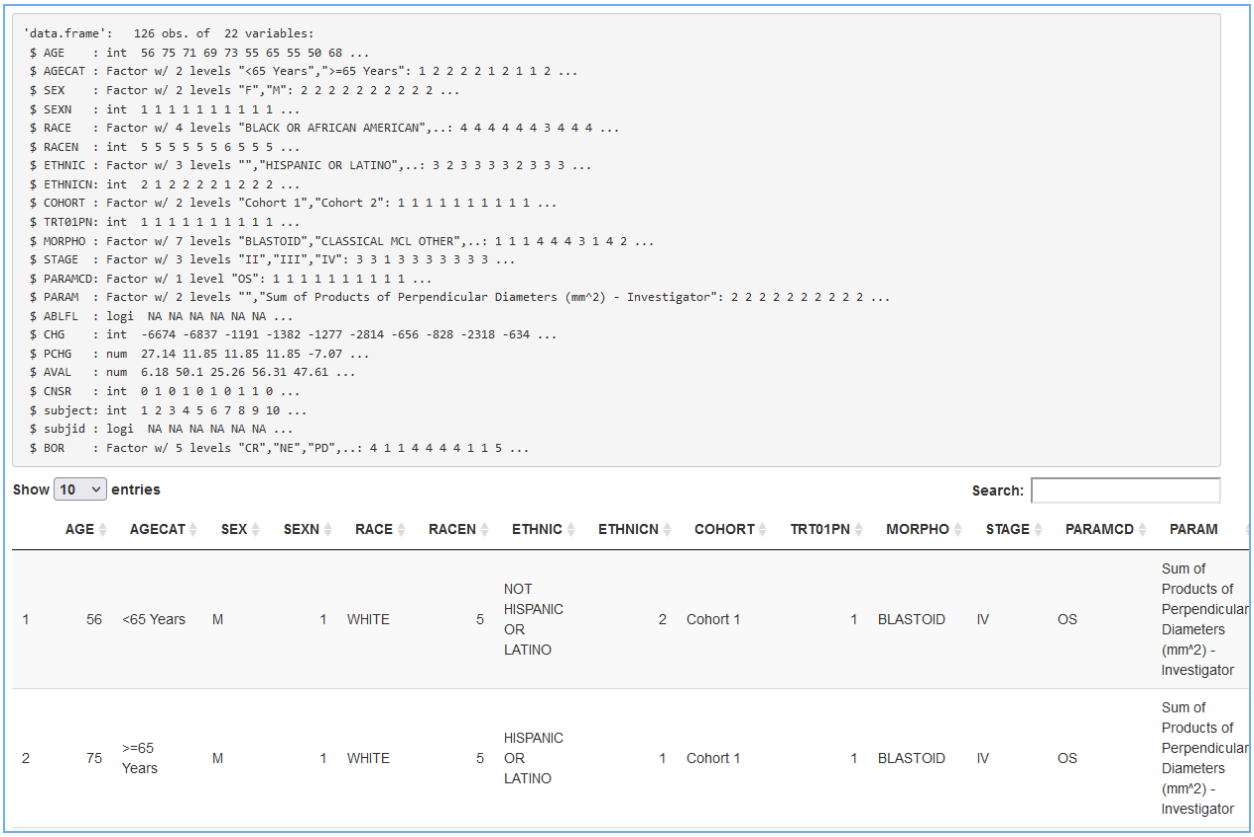


Figure 1. Uploading the file or data set



```
'data.frame':   126 obs. of  22 variables:
 $ AGE    : int  56 75 71 69 73 55 65 55 50 68 ...
 $ AGECAT : Factor w/ 2 levels "<65 Years",">=65 Years": 1 2 2 2 2 1 2 1 1 2 ...
 $ SEX    : Factor w/ 2 levels "F","M": 2 2 2 2 2 2 2 2 2 2 ...
 $ SEXN   : int  1 1 1 1 1 1 1 1 1 1 ...
 $ RACE   : Factor w/ 4 levels "BLACK OR AFRICAN AMERICAN",..: 4 4 4 4 4 4 3 4 4 4 ...
 $ RACEN  : int  5 5 5 5 5 5 6 5 5 5 ...
 $ ETHNIC : Factor w/ 3 levels "","HISPANIC OR LATINO",..: 3 2 3 3 3 3 2 3 3 3 ...
 $ ETHNICN: int  2 1 2 2 2 2 1 2 2 2 ...
 $ COHORT : Factor w/ 2 levels "Cohort 1","Cohort 2": 1 1 1 1 1 1 1 1 1 1 ...
 $ TRT01PN: int  1 1 1 1 1 1 1 1 1 1 ...
 $ MORPHO : Factor w/ 7 levels "BLASTOID","CLASSICAL MCL OTHER",..: 1 1 1 4 4 4 3 1 4 2 ...
 $ STAGE  : Factor w/ 3 levels "II","III","IV": 3 3 1 3 3 3 3 3 3 3 ...
 $ PARAMCD: Factor w/ 1 level "OS": 1 1 1 1 1 1 1 1 1 1 ...
 $ PARAM  : Factor w/ 2 levels "","Sum of Products of Perpendicular Diameters (mm^2) - Investigator": 2 2 2 2 2 2 2 2 2 2 ...
 $ ABLFL  : logi  NA NA NA NA NA NA ...
 $ CHG    : int  -6674 -6837 -1191 -1382 -1277 -2814 -656 -828 -2318 -634 ...
 $ PCHG   : num  27.14 11.85 11.85 11.85 -7.07 ...
 $ AVAL   : num  6.18 50.1 25.26 56.31 47.61 ...
 $ CNSR   : int  0 1 0 1 0 1 0 1 1 0 ...
 $ subject: int  1 2 3 4 5 6 7 8 9 10 ...
 $ subjid : logi  NA NA NA NA NA NA ...
 $ BOR    : Factor w/ 5 levels "CR","NE","PD",..: 4 1 1 4 4 4 4 1 1 5 ...
```

Show 10 entries Search:

	AGE	AGECAT	SEX	SEXN	RACE	RACEN	ETHNIC	ETHNICN	COHORT	TRT01PN	MORPHO	STAGE	PARAMCD	PARAM
1	56	<65 Years	M	1	WHITE	5	NOT HISPANIC OR LATINO	2	Cohort 1	1	BLASTOID	IV	OS	Sum of Products of Perpendicular Diameters (mm^2) - Investigator
2	75	>=65 Years	M	1	WHITE	5	HISPANIC OR LATINO	1	Cohort 1	1	BLASTOID	IV	OS	Sum of Products of Perpendicular Diameters (mm^2) - Investigator

Figure 2. Displaying data and attributes

PLOTS TAB

Once the data is read into the utility, the user can visualize the data using the plots tab. Within the plots tab there are multiple sub-tabs for various plots like scatter plot, box plot, KM plot and Waterfall plot.

Under each plot, there are various dropdown buttons to pick the respective variables to generate a specific plot. As an example, shown in Figure 3, there are three dropdown buttons available for the user to pick the “Analysis variable”, “Censor variable” and the “By Group variable” or the stratification variable.

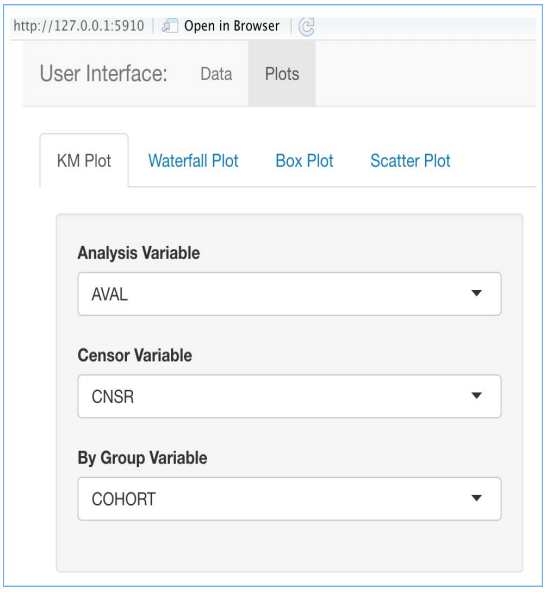


Figure 3 Plot type and Variable selection tabs under the plots tab

After all the required variables are selected using the dropdown menu for a specific plot, the respective figure is generated in a separate window. This helps the user to visualize the data using the generated figures.

Various plots generated using the plots tab

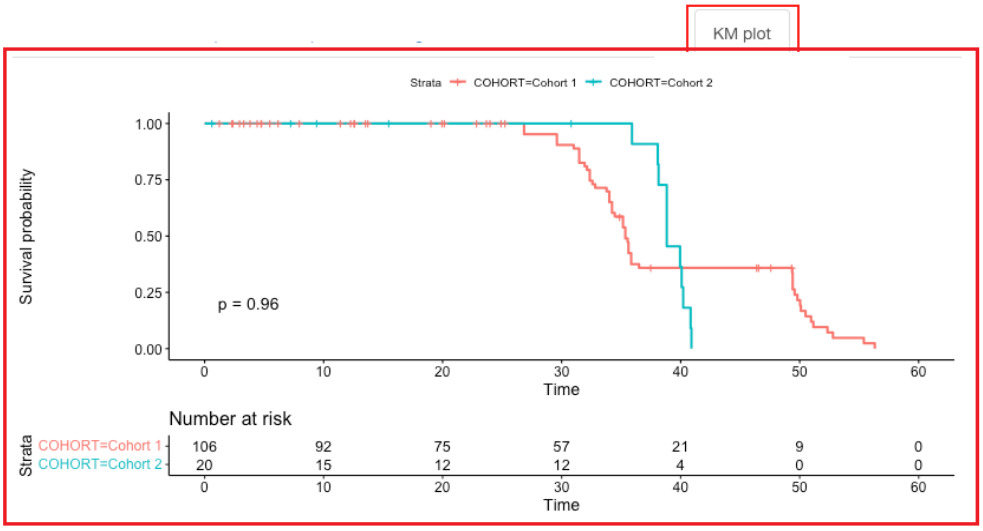


Figure 4. Kaplan Meier Plot of OS by COHORT

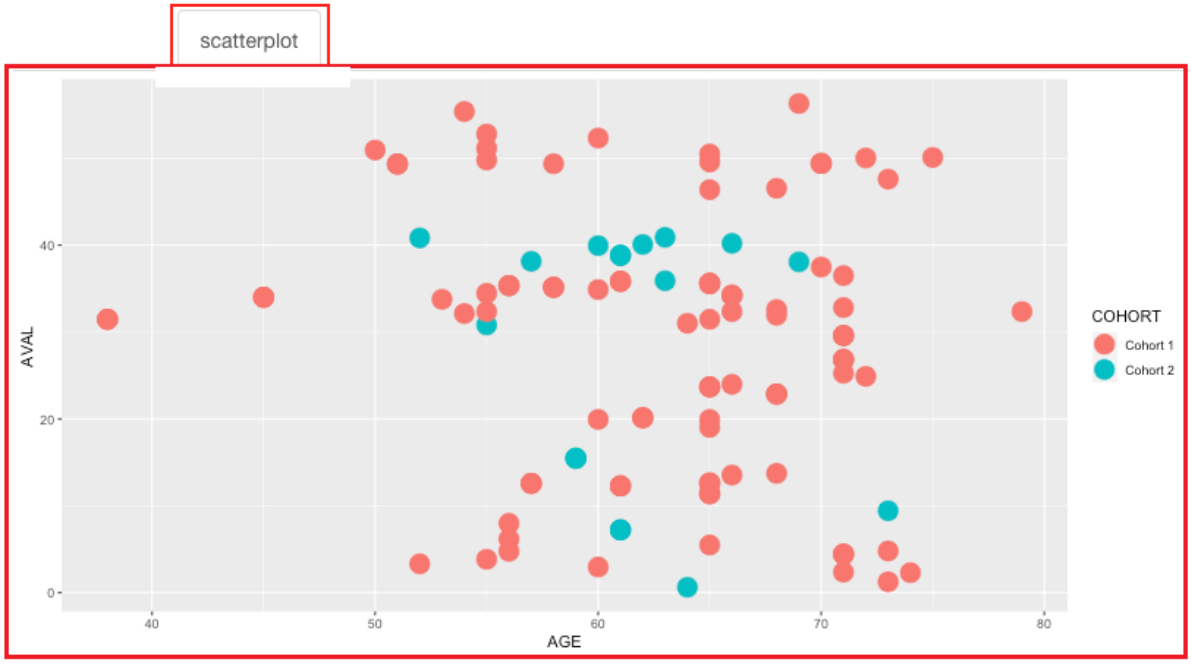


Figure 5. Scatter Plot of AVAL vs AGE

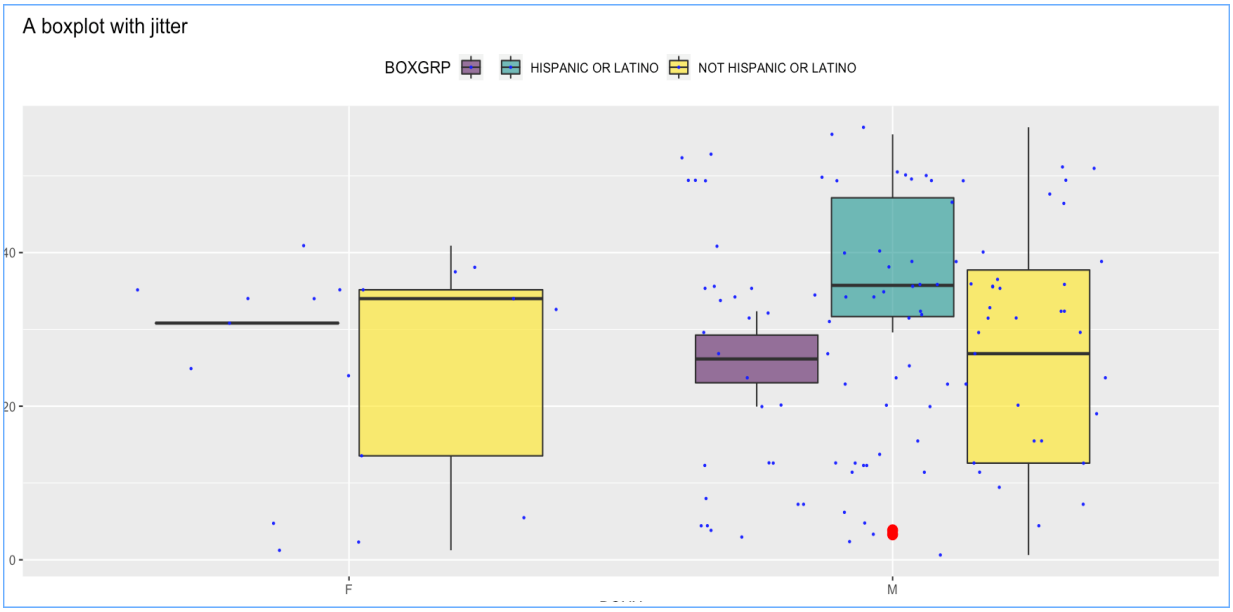


Figure 6. Box Plot of AVAL vs Sex

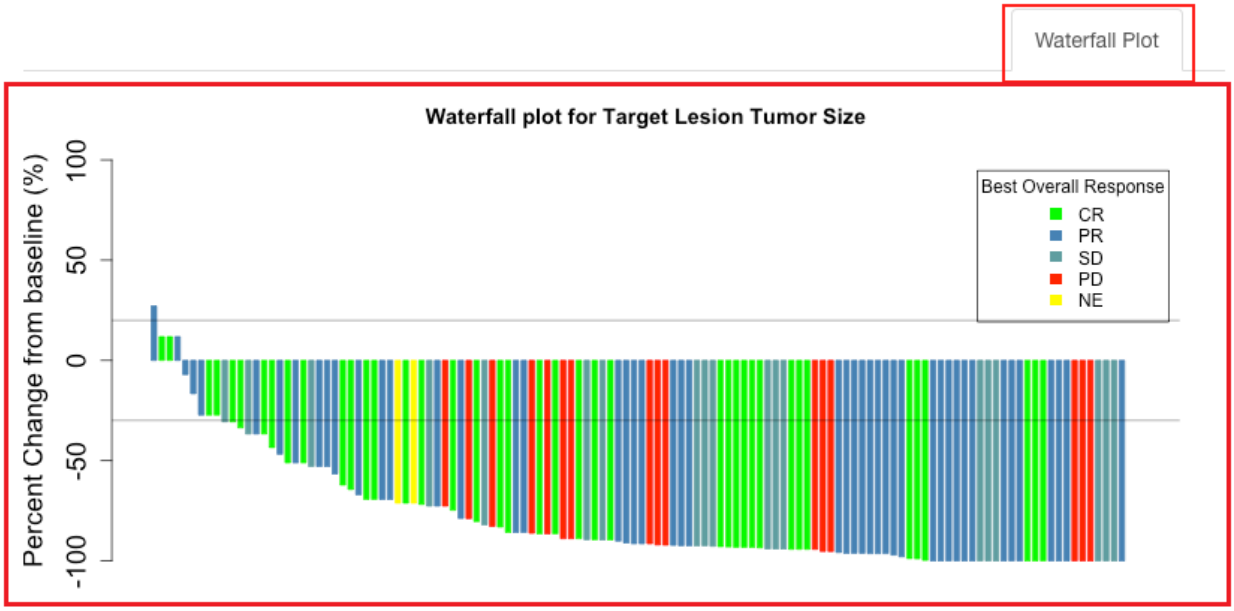


Figure 7. Waterfall plot of the percent change in the Sum of Diameters

LIMITATIONS

Although there are no limitations to functionalities that can be built into a dashboard using R Shiny and generate fancy and colorful plots for the data visualization purpose just with a click of few buttons, there are a few limitations to the dashboard we built and described in this paper:

1. The data needs to follow the ADaM standards
2. Users need to be familiar with the ADaM standards and should be familiar in picking the right variables for the right plot
3. Users do not yet could add custom titles and footnotes
4. Currently the utility doesn't support multiple sub-group analyses, i.e., any additional by group variables

CONCLUSION

Based on our experience working on data analyses and building dashboards employing R and R shiny to visualize the data gave us the firm belief that harnessing their power to assist the end users could be very beneficial in understanding the data, visualizing the data effortlessly and in making quick clinical decisions.

REFERENCES

Agresti, A. 2013. *Categorical Data Analysis*. 3rd ed. Hoboken, NJ: John Wiley & Sons.

Utilizing Visualizations Developed in R Shiny for Exploratory Safety Analysis by Li et al

https://cran.r-project.org/web/packages/shiny/index.html

https://shiny.rstudio.com/images/shiny-cheatsheet.pdf

https://jshaik.shinyapps.io/Pharmasug2022-app/

https://github.com/johnsshaik/RShiny-EfficacyPlots.git

ACKNOWLEDGMENTS

Sincere thanks to Shefalica Chand, Director, Statistical Programming at Seagen Inc., Kai Chen, Associate Director, Statistical Programming at Kite Pharma Inc., for their continued support, and encouragement throughout, and for their valuable assistance in reviewing this paper.

CONTACT INFORMATION

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

John Shaik
Sr Manager Statistical Programming, Seagen Inc.,
jshaik@seagen.com

Sreeram Kundoor
Manager Statistical Programming, Kite Pharma Inc.,
skundoor@kitepharma.com

APPENDIX I: R CODE TO BUILD THE DASHBOARD

```
library(shiny)
library(ggplot2)
library(plotly)
library(survival)
library(survminer)
library(viridis)
library(readxl)
library(haven)
library(vroom)
library(DT)
library(psych)
library(plotly)

ui <- fluidPage(navbarPage(
  "User Interface:",
  tabPanel(
    "Data",
    titlePanel("Select Input data"),
    sidebarLayout(
      sidebarPanel(
        fileInput(
          "file1",
          "Choose Dataset/File",
          multiple = TRUE,
          accept = c("text/csv",
                     "text/comma-separated-values,text/plain",
                     ".csv")
        )

      ),
      mainPanel(verbatimTextOutput("summary"),
```

```
            #tableOutput("contents")
            dataTableOutput("contents"))
  )
),




tabPanel(
  "Plots",
  tabsetPanel(
    tabPanel(
      "KM Plot",


      pageWithSidebar(
        headerPanel(''),

        sidebarPanel(
          uiOutput("variable_kanl"),
          uiOutput("variable_cnsr"),
          uiOutput("variable_bygrp")
        ),

        mainPanel(plotlyOutput('plot4'))
      )


    ),
    tabPanel(
      "Waterfall Plot",
      pageWithSidebar(
        headerPanel(''),


        sidebarPanel(
          uiOutput("variable_wfanl"),
          uiOutput("variable_wfgrp"),
          uiOutput("variable_wfxvar")
        ),
        mainPanel(plotlyOutput('plot5'))
      )

    ),

    tabPanel(
      "Box Plot",
      pageWithSidebar(
        headerPanel(''),

        sidebarPanel(
          uiOutput("variable_boxx"),
          uiOutput("variable_boxy"),
          uiOutput("variable_boxgrp")
        ),
```

```
      mainPanel(plotlyOutput('plot1'))
    )

  ),
  tabPanel(
    "Scatter Plot",
    pageWithSidebar(
      headerPanel(''),

      sidebarPanel(
        uiOutput("variable_sctx"),
        uiOutput("variable_scty"),
        uiOutput("variable_sctgrp")
      ),
      mainPanel(plotlyOutput('plot2'))
    )

  )


 )
),



))

server <- function(input, output, session) {
  onSessionEnded(stopApp)
  data <- reactive({
    req(input$file1)




    if (stringr::str_ends(input$file1$datapath, "csv")) {
      df <- read.csv(input$file1$datapath)
    } else if (stringr::str_ends(input$file1$datapath, "(xlsx|xls)")) {
      df <- read_excel(input$file1$datapath)
    } else if (stringr::str_ends(input$file1$datapath, "sas7bdat")) {
      df <- read_sas(input$file1$datapath)
    }


  })

  output$contents <- DT::renderDataTable(return(data()))
  output$summary <- renderPrint({
    str(as.data.frame(data()))


  })


  output$variable_kanl <- renderUI({
    selectInput("variableNames_kanl",
```

```
            label = "Analysis Variable",
            choices = names(data()))
})

output$variable_cnsr <- renderUI({
  selectInput("variableNames_cnsr",
            label = "Censor Variable",
            choices = names(data()))
})

output$variable_bygrp <- renderUI({
  selectInput("variableNames_bygrp",
            label = "By Group Variable",
            choices = names(data()))
})

output$variable_wfanl <- renderUI({
  selectInput("variableNames_wfanl",
            label = "Analysis Variable",
            choices = names(data()))
})

output$variable_wfgrp <- renderUI({
  selectInput("variableNames_wfgrp",
            label = "Response Variable",
            choices = names(data()))
})

output$variable_wfxvar <- renderUI({
  selectInput("variableNames_wfxvar",
            label = "X-axis Variable",
            choices = names(data()))
})


output$variable_boxx <- renderUI({
  selectInput("variableNames_boxx",
            label = "X-Axis variable",
            choices = names(data()))
})
output$variable_boxy <- renderUI({
  selectInput("variableNames_boxy",
            label = "Y-axis variable",
            choices = names(data()))
})
output$variable_boxgrp <- renderUI({
  selectInput("variableNames_boxgrp",
            label = "Color fill - grouping variable",
            choices = names(data()))
})

output$variable_sctx <- renderUI({
  selectInput("variableNames_sctx",
            label = "X-Axis variable",
            choices = names(data()))
})
```

```
  output$variable_scty <- renderUI({
    selectInput("variableNames_scty",
                label = "Y-axis variable",
                choices = names(data()))
  })
  output$variable_sctgrp <- renderUI({
    selectInput("variableNames_sctgrp",
                label = "Color fill - grouping variable",
                choices = names(data()))
  })

  survdat <- reactive({
    test <-
      data.frame(data()[[input$variableNames_kanl]], data()[[input$variableNames_cnsr]],
data()[[input$variableNames_bygrp]])
    colnames(test) <- c("AVAL", "CNSR", "BYGRP")
    return(test)


  })

  wfdat <- reactive({
    wftest <-
      data.frame(data()[[input$variableNames_wfanl]], data()[[input$variableNames_wfgrp]],
data()[[input$variableNames_wfxvar]])
    colnames(wftest) <- c("PCHG",  "WFGRP", "SUBJECT")
    wftest <- subset(wftest, SUBJECT < 50 & PCHG > -100)
    wftest <- wftest[order(wftest$PCHG), ]
    return(wftest)
  })

  boxdat <- reactive({
    boxtest <-
      data.frame(data()[[input$variableNames_boxx]], data()[[input$variableNames_boxy]],
data()[[input$variableNames_boxgrp]])
    colnames(boxtest) <- c("BOXX",  "BOXY", "BOXGRP")
    boxtest$BOXGRP <- factor(boxtest$BOXGRP)

    return(boxtest)
  })

  sctdat <- reactive({
    scttest <-
      data.frame(data()[[input$variableNames_sctx]], data()[[input$variableNames_scty]],
data()[[input$variableNames_sctgrp]])
    colnames(scttest) <- c("SCTX",  "SCTY", "SCTGRP")

    return(scttest)
  })

  output$plot4 <- renderPlotly({
    if (is.null(data)) {
      return(NULL)
    } else {
      srvpplt <-
        ggsurvplot(
```

```
      survfit(Surv(AVAL , CNSR) ~ BYGRP , data = survdat()) ,
      risk.table = TRUE,
      pval = TRUE,
      data = survdat() ,
      main = "KM estimates of Overall Survival"
    )
  ggplotly(srvpplt[[1]])

 }
})

output$plot5 <- renderPlotly({
 if (is.null(data)) {
  return(NULL)
 } else {
  ggplotly(
   ggplot(data = wfdat(), aes(SUBJECT, PCHG)) +
    geom_bar(aes(fill = WFGRP), stat = "identity") +

    theme_minimal() +
    theme(axis.text.x = element_text(angle = 90))
  )

 }
})


output$plot1 <- renderPlotly({
 if (is.null(data)) {
  return(NULL)
 } else {
  ggplotly(
   ggplot(boxdat(),
          aes(
           x = BOXX,
           y = BOXY,
           fill = BOXGRP
          )) + geom_boxplot(
           # custom boxes


           # custom outliers
           outlier.colour = "red",
           outlier.fill = "red",
           outlier.size = 3
          ) +
    scale_fill_viridis(discrete = TRUE, alpha = 0.6) +
    geom_jitter(
     color = "blue",
     size = 0.4,
     alpha = 0.9
    ) +
    theme(
     legend.position = "top",
     plot.title = element_text(size = 14)
    ) +
```

```
            ggtitle("A boxplot with jitter")
        )
      }
    })

    output$plot2 <- renderPlotly({
      if (is.null(data)) {
        return(NULL)
      } else {
        ggplotly(ggplot(sctdat(), aes(
          x = SCTX, y = SCTY, fill = SCTGRP
        )) + geom_point(
          shape = 21,
          color = "black",
          # fill = SCTGRP,
          size = 3
        ))
      }
    })




  }

shinyApp(ui, server)
```