PharmaSUG 2020 - Paper 298

RStats: A R-Shiny application for statistical analysis

Sean Yang, Hrideep Antony, Aman Bahl, Syneos Health

ABSTRACT

This paper will introduce RStats application which is an interactive and dynamic R-Shiny based application that can perform popular statistical analysis models that are frequently used in clinical trials. The application can perform even the advanced analytics such as Logistic regression, Survival analysis, Anova test, Correlation etc., along with the summary plots for users with no prior R programming or even with limited statistical knowledge. This application also eliminates the need for numerous lines of programming effort to create a similar statistical analysis.

INTRODUCTION

Most statistical programmers work frequently with SAS datasets to do statistical analyses using SAS. R can also perform these tasks because it has packages designed to work with SAS datasets i.e. Shiny package, a tool that provides a framework to develop GUI applications. Shiny's straightforward implementation of UI/Server interface makes it an ideal tool to create applications that can produce statistical tasks that a statistical programmer may often work with. As with any software program, there usually is more than one way to do things through R. The RStats application in this paper is one of the ways to perform these statistical analyses.

The process flow diagram in Figure 1 below explains the steps to perform the statistical analysis using this application:

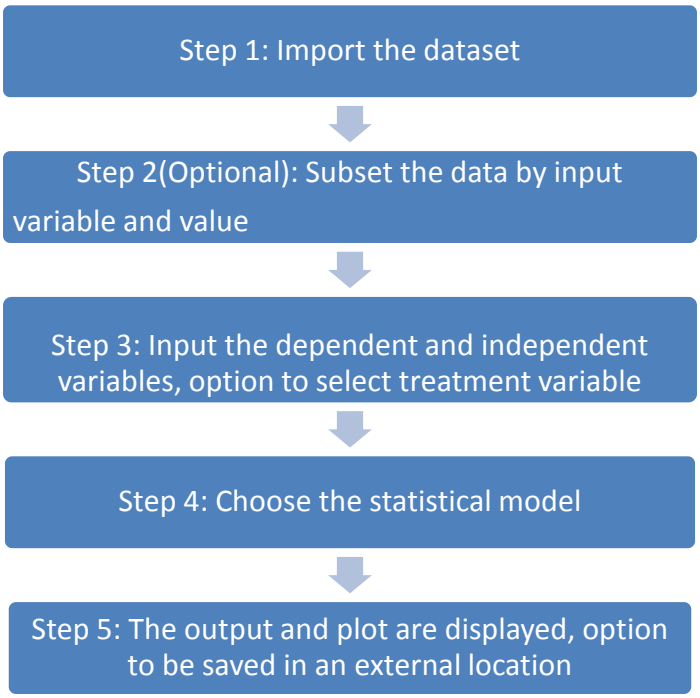


Figure 1: RStats flow diagram

STATISTICAL ANALYSIS APPLICATION

The Figure 2 below shows the live statistical analysis application which dynamically creates the plot and analysis output using any datasets presented by the user.

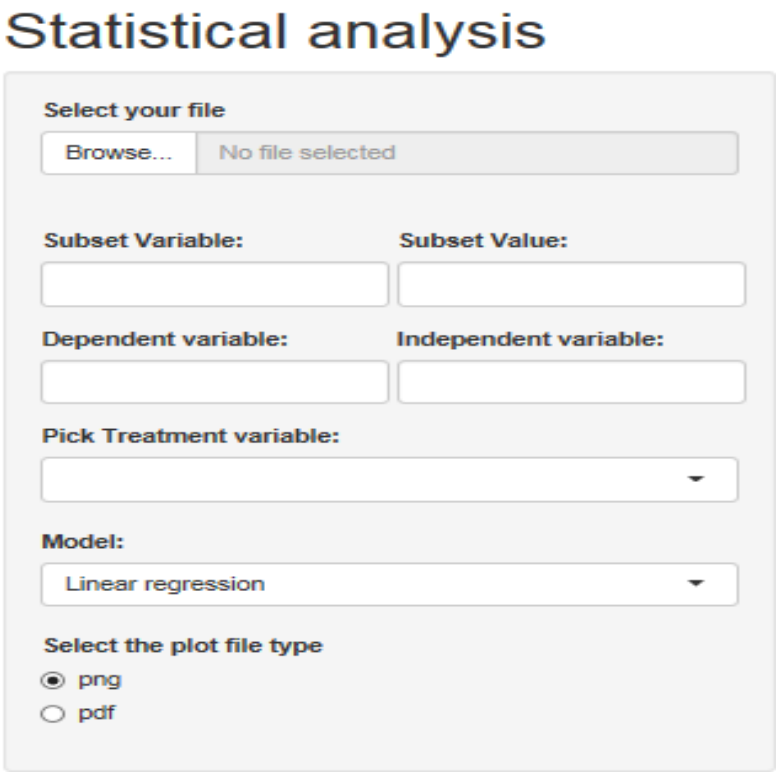


Figure 2: RStats user interface

Table 1 below has a list of analyses that can be conducted using this R-Shiny application.

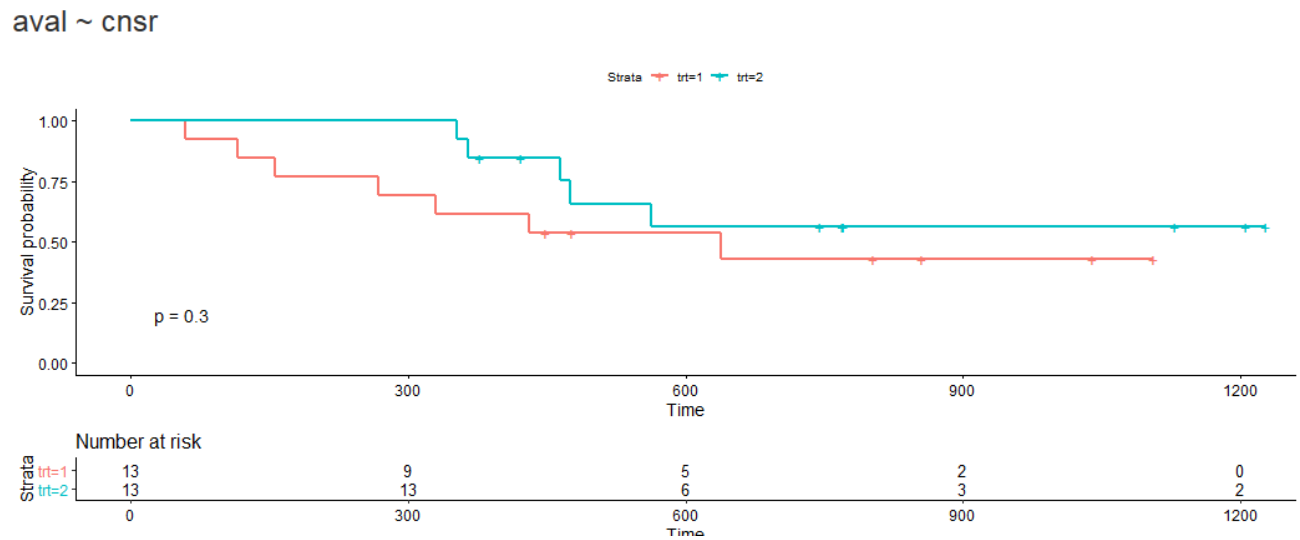
Model	Output	Plot
Linear regression	Estimate of parameters, p-values, R square	Scatterplot
Ttest	Mean, SD, p-value	Boxplot
Anova test	Mean, SD, p-value	Boxplot
Chi-square test	p-value	Bar plot
Logistic regression	Estimate of parameters,odds ratio, p-values	Bar plot
Survival analysis	p-value, risk table	KM curve
Correlation	p-value	Scatterplot

Table 1: Listing of models and the corresponding output and plot

EXAMPLE 1 – SURVIVAL ANALYSIS

For example, we want to study the time to first pain-free after dose, stratified by treatment. First select dataset ADTTE, subset as PARAMCD=TTPF, the dependent variable is AVAL and independent variable is CNSR, pick TRTAN as the treatment variable, then choose Survival analysis model.

Figure 3 below is the output of the survival analysis that the RStats application has generated.



Save the plot

```
Call: survfit(formula = Surv(Time, cen) ~ trt, data = new_data())

                trt=1
 time n.risk n.event survival std.err lower 95% CI upper 95% CI
   59     13       1    0.923  0.0739        0.789        1.000
  115     12       1    0.846  0.1001        0.671        1.000
  156     11       1    0.769  0.1169        0.571        1.000
  268     10       1    0.692  0.1280        0.482        0.995
  329      9       1    0.615  0.1349        0.400        0.946
  431      8       1    0.538  0.1383        0.326        0.891
  638      5       1    0.431  0.1467        0.221        0.840

                trt=2
 time n.risk n.event survival std.err lower 95% CI upper 95% CI
  353     13       1    0.923  0.0739        0.789        1.000
  365     12       1    0.846  0.1001        0.671        1.000
  464      9       1    0.752  0.1256        0.542        1.000
  475      8       1    0.658  0.1407        0.433        1.000
  563      7       1    0.564  0.1488        0.336        0.946
```

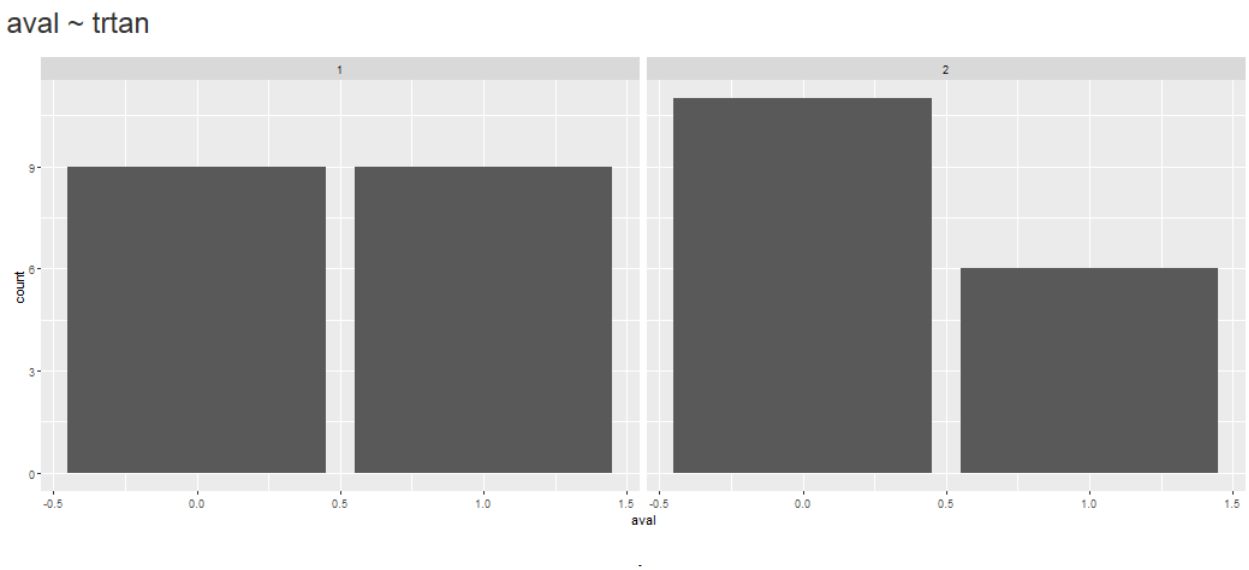
Figure 3: RStats summary output and plot for survival analysis

The KM curves are overlapped and p-value=0.46, so there's no significant difference in survival time between the treatments.

EXAMPLE 2 – LOGISTIC REGRESSION

We're interested to check the frequency and odds ratio of pain-free by treatment. After importing dataset ADEF, subset as PARAMCD=PAINFREE, the dependent variable is AVAL and the independent variable is TRTAN, finally choose the Logistic regression model.

Figure 4 below is the output of the logistic regression that the RStats application has generated.



Save the plot

```
[1] "Odds ratio: 1"                "Odds ratio: 0.545454545454571"

Call:
glm(formula = y ~ factor(trt), family = "binomial", data = new_data())

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-1.1774  -1.0552  -0.9331   1.1774   1.4432

Coefficients:
              Estimate Std. Error z value Pr(>|z|)
(Intercept)   1.450e-16  4.714e-01   0.000    1.000
factor(trt)2 -6.061e-01  6.927e-01  -0.875    0.382

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 47.804  on 34  degrees of freedom
Residual deviance: 47.028  on 33  degrees of freedom
AIC: 51.028

Number of Fisher Scoring iterations: 4
```

Figure 4: RStats summary output and plot for logistic regression

The odds ratio of treatment 2 vs. treatment 1 is 0.54 and p-value=0.382, no significant difference between the treatments.

CODE FOR BUILDING UI

```
ui<-pageWithSidebar(
  headerPanel("Statistical analysis"),
  sidebarPanel(
    fileInput('file1', 'Select your file',
              accept = c(
                'text/csv',
                'text/comma-separated-values','.csv',
                '.sas7bdat'
              )
    ),
    splitLayout(
      textInput("subvar","Subset Variable:"),
```

[1]

```
      textInput("subval","Subset Value:")),
    splitLayout(
      textInput("dep","Dependent variable:"),
      textInput("indep","Independent variable:")),
   selectInput("varlist","Pick Treatment
variable:",choices=c(colnames(data()))),
    selectInput("Model","Model:",
                list("Linear regression" = "lm",
                     "Logistic regression" = "logistic",
                     "Survival analysis" = "surv",
                     "Anova test" = "anova",
                     "Chisquare test" = "chisq",
                     "Correlation" = "corr",
                     "Bootstrap CI" = "BCa")),
    radioButtons("type", "Select the plot file
type",choices=list("png","pdf"))
  ),
  # Show the output and plot
  mainPanel(
    h3(textOutput("caption")),
    plotOutput("myPlot"),
    downloadButton("save1", "Save the plot"),
    verbatimTextOutput("regSum")
  )
)
```

[2]

[3]

[4]

[5]

UI CODE EXPLANATION

[1] Create the UI file input box so the user is able to select the input dataset.
[2] After the dataset is loaded, it is passed to the server. User is required to input the subset variable and value, independent and dependent variables. The treatment variable could be selected within a list of variables from the dataset.
[3] User can pick statistical model from the listing.
[4] A radio button in the main panel enables user to select the file type of saved plot.
[5] Create the main panel with plot and its download button, statistical summary output.

CODE FOR SERVER

```
server<-function(input, output, session) {
  # Import data
   data1 <- reactive({
    inFile <- input$file1
    if(is.null(file)){return()} [1]
    read_sas(inFile$datapath)
  })
  observeEvent(input$file1,{
    updateSelectInput(session,"varlist",choices=c(colnames(data1())))
  })
   # Compute the forumla text
  formulaText <- reactive({
    paste(input$dep,"~", input$indep)
  })
  output$caption <- renderText({
    formulaText()
  })
 # Data
  new_data<-reactive({
```

[1]

[2]

```
    raw_data <- data1()
    if (input$Model == 'surv') {
      df<-data.frame(
        Time = raw_data[[input$dep]],
        trt  = raw_data[[input$varlist]],
        cen  = raw_data[[input$indep]],
        sub  = raw_data[[input$subvar]]
       )
      if (input$subval == "") { df2<-df }
      else { df2<-subset(df,sub==input$subval) }
      return(df2)
    }
    if (input$Model == 'logistic') {
      df<-data.frame(
        y = raw_data[[input$dep]],
        trt = raw_data[[input$indep]],
        sub  = raw_data[[input$subvar]]
      )
      if (input$subval == "") { df2<-df }
      else { df2<-subset(df,sub==input$subval) }
      return(df2)
    }
# Statistics
  new_fit<-reactive({
    if (input$Model == 'surv') {
     fit<-survfit(Surv(Time , cen) ~ trt , data = new_data())
     return(fit)
    }
    if (input$Model == 'logistic') {
      fit<-glm(y ~ factor(trt), data = new_data(), family = "binomial")
      print(paste('Odds ratio:',exp(coef(fit))))
      return(fit)
    }
# Graph
  new_plot<-reactive({
    if (input$Model == 'surv') {
      graph<-ggsurvplot(new_fit(), risk.table = TRUE, pval = TRUE, data =
new_data())
      return(graph)
    }
    if (input$Model == 'logistic') {
      graph<-ggplot(new_data(),aes(y))+geom_bar()+facet_wrap('trt')
+labs(x=input$dep)
      return(graph)
    }
# Generate the plot
  output$myPlot <- renderPlot({
    new_plot()
    })
# Generate the summary of analysis
  output$regSum <- renderPrint({
    summary(new_fit())
  })
# Save the plot
  output$save1 <- downloadHandler(
    filename = function() {
      paste("myplot",input$type,sep=".")
```

[3]

[4]

[5]

[6]

```
    },
    content = function(file){
      if(input$type=="png") png(file)
      else pdf(file)
      print(new_plot())
      dev.off()
    })
}
```

[7]

SERVER CODE EXPLANATION

Note: Only Survival analysis and Logistic regression are presented.
[1] Once the user selects a dataset, reactive functions are used to go to the datapath and read in the SAS dataset.
[2] Calculate and return the formula text for printing as a caption.
[3] Subset the loaded dataset by the subset variable and value, then combine dependent, independent and treatment variables that are defined in UI, into a new dataset.
[4] Reactive functions are used to do dynamic statistical analysis.
[5] Reactive functions are used to create dynamic plot.
[6] Create plot and statistical summary which are called by the UI main panel.
[7] Use the file type selected in UI to save the plot to an external location.

CONCLUSION

The dynamic nature of the Shiny Apps makes them very powerful. They can automate route tasks and provide great efficiency, also can be easily customized. Shiny allows R users to put data insights into the hands of the decision-makers while providing a user-friendly framework that does not require any additional toolsets.

REFERENCES

https://cran.r-project.org/web/packages/shiny/index.html
https://shiny.rstudio.com/images/shiny-cheatsheet.pdf

ACKNOWLEDGMENTS

Sincere thanks to Steve Benjamin, Director Statistical Programming, Biostatistics and Global Contracts for his vision, great leadership, persistent support, and encouragement throughout and for his valuable assistance in reviewing this paper.

CONTACT INFORMATION

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

Sean Yang
Principal Statistical Programmer, Clinical Division, Syneos Health
Work Phone: +1- 905 296-4851
Email: sean.yang@syneoshealth.com
Web: http://www.syneoshealth.com

Hrideep Antony
Principal Statistical Programmer, Clinical Division, Syneos Health
Work Phone: +1-984 459 4785
Email: hrideep.antony@syneoshealth.com

Web: http://www.syneoshealth.com

Aman Bahl
Associate Director, Statistical Programming, Clinical Division, Syneos Health
Phone: +1-905-399-6715
E-mail: Aman.Bahl@syneoshealth.com
Web: http://www.syneoshealth.com