PharmaSUG 2020 - Paper AD-046

Normal is Boring, Let's be Shiny: Managing Projects in Statistical Programming Using the RStudio® Shiny® App.

Girish Kankipati and Hao Meng, Seattle Genetics, Inc., Bothell WA

ABSTRACT

The ability to deliver projects on schedule, within budget, and aligned with business goals is key to gaining an edge in today's fast-paced pharmaceutical and biotechnology industry. Project management plays an important role in achieving key milestones with high quality and optimal efficiency. While successful project management is an application of processes, methods, skills, and expertise, tools like SAS®, RStudio®, and Python™ can help track project status and better position the lead programmer to allocate appropriate resources.

One suitable app that fits the specific needs for status tracking in Statistical Programming is R Shiny®, a flexible programmer-friendly tool that can present an instant study status overview in graphical format with minimal coding and maintenance effort. This paper introduces an innovative design to track study status through an RStudio® Shiny® app that is interactive and reusable and can present status on demand. Based on simple server metadata, we can display a graphical representation of not only the total number of SDTM, ADaM, and TLFs that have been programmed and validated, but also the trend in progress to date to help lead programmers and statisticians determine any resource adjustments based on timely and effective status reporting that refreshes on a monthly, weekly, or daily basis to monitor ongoing study progress. Sample project tracker metadata, the high-level inner workings of our Shiny® app, and Shiny® graphs will be discussed in depth in this paper.

INTRODUCTION

Pharmaceutical/Biotech companies have adopted the use of foundational CDISC standards and other industry standards to facilitate various data handling, data mapping, and analysis/reporting activities. Implementing a complete end-to-end approach requires time, effort, and collaboration with cross-functional groups like Clinical Data Management, Biostatistics, and Statistical Programming. The end-to-end activities, shown in Figure 1, include data collection, data standardization, analysis and reporting, and submission of data to the regulatory authorities. Statistical Programming plays a key role in the creation of SDTM and ADaM data sets; in trial results reporting in tables, listings, and figures (TLFs); and in final data submission to FDA and other health authorities around the globe. The time and effort to create these deliverables varies from project to project. Some trials run from start to finish in a matter of weeks or months, while others, like Oncology trials, may take years to finish and require managing a significant number of resources during that time.

Figure 1: Programming Flow from Data Collection to TLF Production

Some of the challenges in managing these projects are:

1) Tracking the project status efficiently
2) Time projection to finish the project
3) Identifying total number of TLFs, data sets, and the assignments to programmers involved
4) No centralized summary reports for ongoing projects and those already locked
5) Lead programmers need to spend lots of time in meetings and through written communication like emails to track project status

This paper discusses one solution that helps track such progress by using the R Shiny® app.

BENEFITS

This paper will address the current challenges by building an RStudio® Shiny® app. Below are the potential benefits of this app with less coding effort and maintenance, to help bring efficiencies and process improvement.

1) Interactivity: The interaction between programmer and app is very easy and requires less hands-on experience to maintain the app.
2) Connectivity: This app connects to different sources such as drop-box, google sheet, SQL database by R and hence it helps in automation of the process. This helps to maintain data in one central repository.
3) Easy Access: Programmer can easily access the data base by connecting to the web page such as google chrome or internet explorer (available on smart phone, too).
4) Bookmarking: Programmer can bookmark the reports of the web pages and view the reports on a weekly or monthly basis. Summary plots can be download at the click of a button.
5) Study Reports: Tracking activities on ongoing and completed studies can be obtained from RStudio® Shiny® graphs. This gives important information about total number of data sets programmed and QCed, total number of programs with issues, ad-hoc deliverables, and dates for first patient in and database lock.
6) Programmer Friendly: Minimal coding efforts required to build the app.

R SHINY® CODE DEVELOPMENT

Here is an overview of the development of the RStudio® Shiny® code and tracker that involves three simple steps:

Figure 2: Steps in the Process of Developing the R Shiny® Tracker

Step 1: Save the data:

This is the first step in the process of creating the tracker. In this step, programmer will enter the actual data (e.g., type of output, programmer name, completion date etc.) directly through the R Shiny® app and send these data as a CSV file to Dropbox.com server. The status of programming deliverables such as SDTM, ADaM, and TLF in terms of development and QC status will be saved in a summary table, which will be displayed in the app. It creates one record for each entry as per provided information in the app (as shown in Figure 1 and Figure 2 below), and the data will be stored in Dropbox for further use. Once the data is saved there, the entry date and time is tracked by column Modified in Figure 1to help build the historical features of the reports.

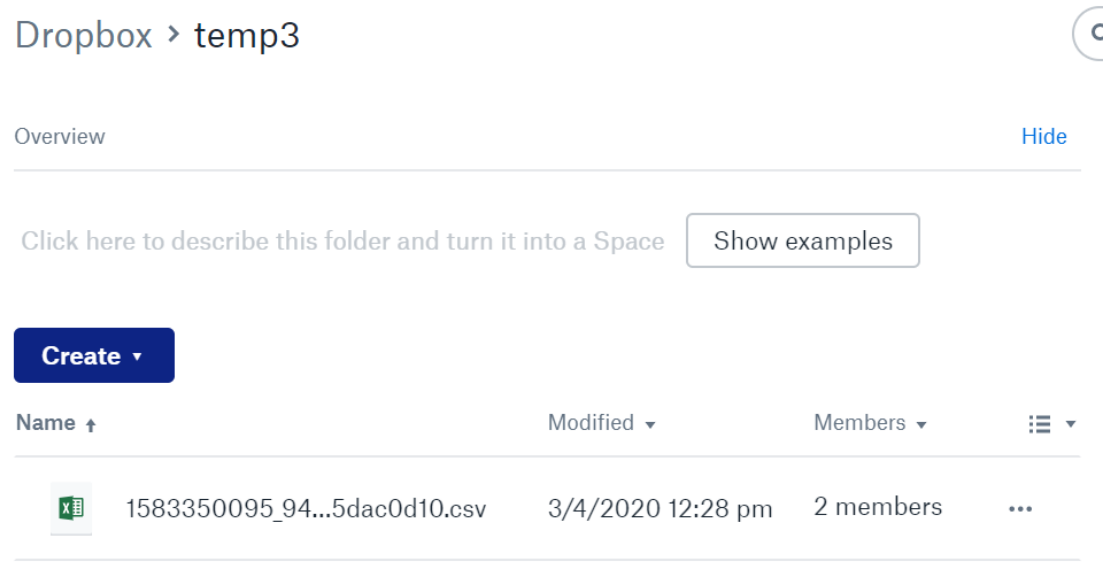


Figure 1 Record file in Dropbox

1583350095_9457e0cc6446d5e31378f8045dac0d10.csv

SDTM	TYPE	STATUS	DATE	USER
ADSL	ADaM	Complete	3/4/2020	XYZ

Figure 2 Actual data in CSV file

R Code:

```
  saveData <- function(data) {
  data <- t(data)
  # Create a unique file name
  fileName <- sprintf("%s_%s.csv", as.integer(Sys.time()),
digest::digest(data))
  # Write the data to a temporary file locally
  filePath <- file.path(tempdir(), fileName)
  # Write the file to the local system
  write.csv(data, filePath, row.names = FALSE, quote = TRUE)
  # Upload the file to Dropbox
  drop_upload(filePath, path = outputDir)}
```

Step 2: Load the data:

In this step, all the summary tables are combined into one data frame. The data which was stored in Dropbox is retrieved from the CSV files and printed in R Shiny® app as a summary table with column names Output, Category, Status, Date and Programmer. The same information is being displayed in *Figure 3*below. Some benefits of using Dropbox as repository store are shown below.

- Access controlled
- Traceability of data

- Real time updates
- One centralized location

R Code:

```
 loadData <- function() {
  # Read all the files into a list
  filesInfo <- drop_dir(outputDir)
  filePaths <- filesInfo$path_lower
  data <- lapply(filePaths, drop_read_csv, stringsAsFactors = FALSE)
  # Concatenate all data together into one data.frame
  data <- do.call(rbind, data)
  data <- datatable(data, colnames = c("Output", "Category", "Status",
"Date","Programmer"))
  data
}
```

Step 3: UI Framework:

In this step, data from Step 2 is used to create Shiny® graphs – the "Statistical Programming Tracker". In the graph, the X axis indicates dates and Y axis indicates the number of deliverables. The code for this UI is shared in the Appendix.

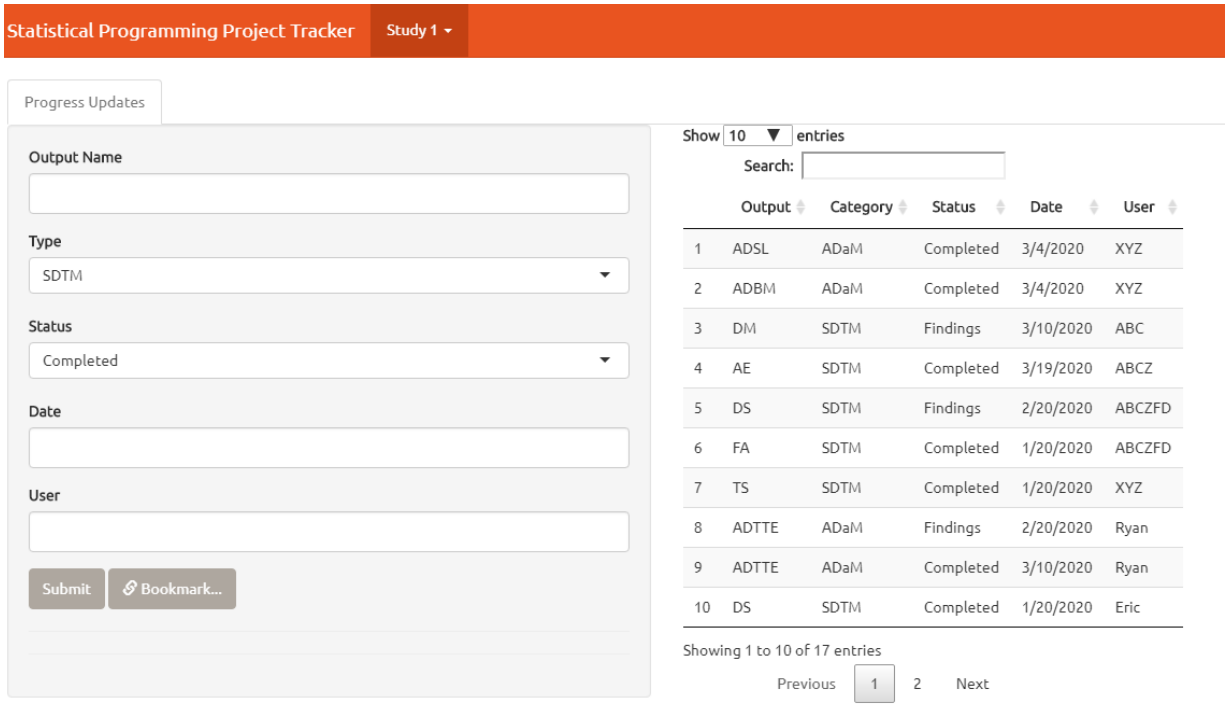


Figure *3* R Shiny® App for the Statistical Programming Status Tracker with Possible Parameters on the Left and Data Entries on the Right.

In the Progress Updates panel, programmers can enter related information to update progress. Once data is entered, hitting the submit button will send data to the Dropbox and correspondingly the summary table on the right side will be updated automatically.

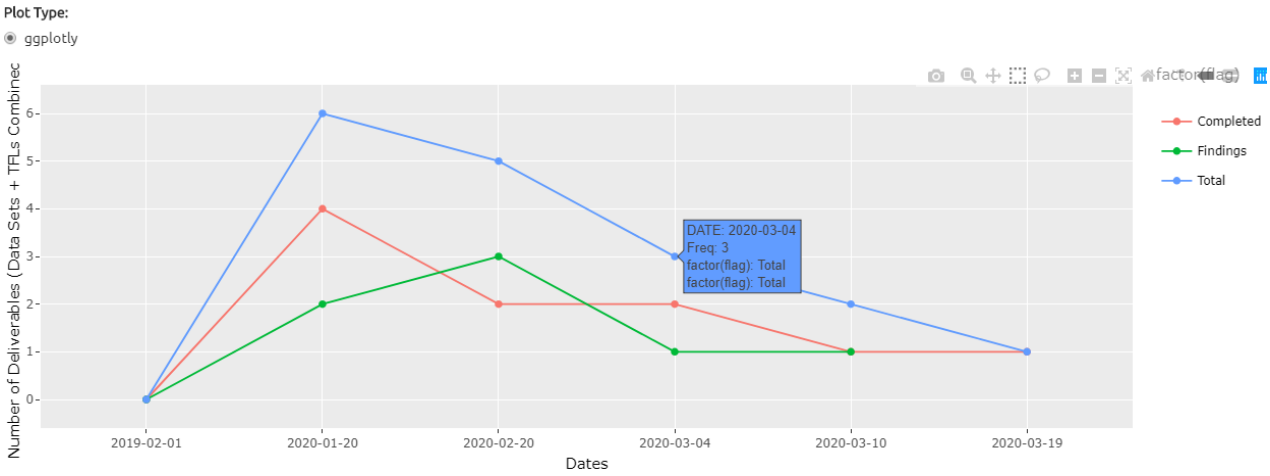
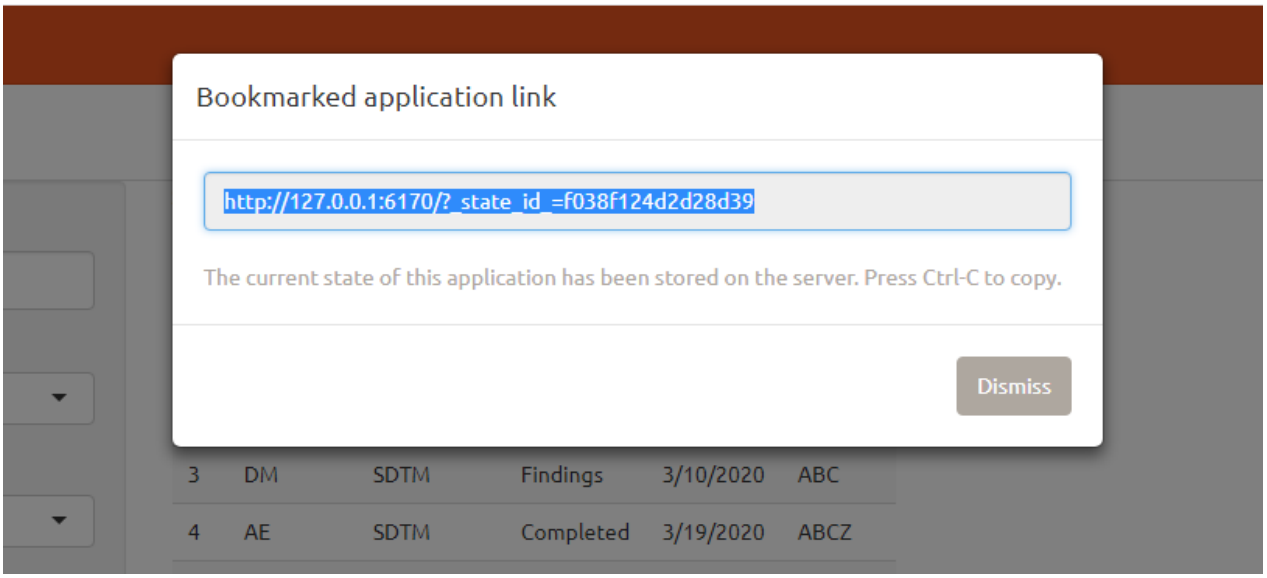


Figure 4: Plot in R Shiny® App With Total Number of Data Sets (SDTM + ADaM) and TLFs With Status of the Deliverables Each Week

Once the entered data is automatically retrieved from Dropbox, we use renderPlotly to create a reactive plot based on the retrieved data.

With the power of plotly package, we are able to use the default tool bar on the right upper corner to facilitate our review. For example, clicking on the camera button allows us download the summary plot as a png file, the zoom-in button helps us zoom in on a selected area which is especially helpful when a plot is busy, and additional metadata will be displayed when the mouse hovers over individual data points plotted in the graph.



To save the current working version, simply hit the Bookmark button, which saves the summary table and plots into an HTML link that can be retrieved by simply visiting the link in any browser.

CONCLUSION

Status tracking of a project is critical in the allotment of resources and budget and in planning the timelines, to ensure timely completion of the project. Building an R Shiny® app involving less maintenance, more automation, and helps lead programmers and biostatisticians efficiently track project status and data metrics on an ongoing basis. This paper shared study tracking elements often involved in a study analysis project. Apart from this, the R Shiny® package provides various functions which can be used to build different apps for different purposes (e.g., interactive data reviews, infrastructure improvements, etc.).

APPENDIX: Sample Code for the Status Tracking App

```
library(shiny)
library(shinythemes)
library(rdrop2)
library(DT)
library(plotly)
# Define the fields we want to save from the form
fields <- c("SDTM", "TYPE", "STATUS","DATE","PROGRAMMER")
outputDir <- "temp3"

saveData <- function(data) {
  data <- t(data)
  # Create a unique file name
  fileName <- sprintf("%s_%s.csv", as.integer(Sys.time()),
digest::digest(data))
  # Write the data to a temporary file locally
  filePath <- file.path(tempdir(), fileName)
  # Write the file to the local system
  write.csv(data, filePath, row.names = FALSE, quote = TRUE)
  # Upload the file to Dropbox
  drop_upload(filePath, path = outputDir)
}

loadData <- function() {
  # Read all the files into a list
  filesInfo <- drop_dir(outputDir)
  filePaths <- filesInfo$path_lower
  data <- lapply(filePaths, drop_read_csv, stringsAsFactors = FALSE)
  # Concatenate all data together into one data.frame
  data <- do.call(rbind, data)
  data <- datatable(data, colnames = c("Output", "Category", "Status",
"Date","Programmer"))
  data
}

# Shiny app with 3 fields for which the programmer can submit data

ui <- function(request){
  navbarPage("Statistical Programming Project Tracker", theme =
shinytheme("united"),
   navbarMenu("Study 1",

    tabPanel("Statistics",
     tabsetPanel(
       tabPanel("Progress Updates",
```

```
        sidebarLayout(
         sidebarPanel(
           textInput("SDTM", "Output Name", ""),
           selectInput("TYPE", "Type", choices =
c("SDTM","ADaM","TFL")),
           selectInput("STATUS", "Status", choices =
c("Completed","Findings")),
           textInput("DATE", "Date"),
           textInput("PROGRAMMER", "Programmer"),
           actionButton("submit", "Submit"),

           bookmarkButton(),
           hr(),
          # helpText("Note for Submit: After entering parameters, hit
Submit will back up the newly added record to outputDir, which is a
local filepath specified in the beginning of our program."),
           hr()
          # helpText("Note for Bookmark: Hit Bookmark to save current
Shiny state. Current state is saved either to the same folder where
App.R is saved (run locally, saved in folder shiny_bookmarks), or to
the server (run on server)")

         ),
         mainPanel(
           DT::dataTableOutput("responses", width = 300), tags$hr(),
           radioButtons("plotType", "Plot Type:", choices =
c("ggplotly")),
           plotlyOutput("plot"),
           verbatimTextOutput("hover"),
           verbatimTextOutput("click"),
           verbatimTextOutput("brush"),
           verbatimTextOutput("zoom")
         )
  )
  )
  )
  )
  )
  )
}
server <- function(input, output, session) {

  # Whenever a field is filled, aggregate all form data
  formData <- reactive({
    data <- sapply(fields, function(x) input[[x]])
    data
  })

  # Do not bookmark submit button
  setBookmarkExclude("submit")

  # When the Submit button is clicked, save the form data
  observeEvent(input$submit, {
    saveData(formData())

  })
```

```
  # Show the previous responses
  # (update with current response when Submit is clicked)
  output$responses <- DT::renderDataTable({
    input$submit
    loadData()

  })

  output$plot <- renderPlotly({
    # use the key aesthetic/argument to help uniquely identify selected
observations
    filesInfo <- drop_dir(outputDir)
    filePaths <- filesInfo$path_lower
    data <- lapply(filePaths, drop_read_csv, stringsAsFactors = FALSE)
    # Concatenate all data together into one data.frame
    dfa <- do.call(rbind, data)

    #df$DATE <- as.Date(df$DATE)
    dfa$DATE <- as.Date(dfa$DATE, format="%m/%d/%Y")
    df1  <- subset(dfa, STATUS == "Completed")
    df2  <- subset(dfa, STATUS == "Findings")

    res <- dfa %>% group_by(DATE) %>% summarise(Freq=n())
    res$flag = "Total"

    res1 <- df1 %>% group_by(DATE) %>% summarise(Freq=n())
    res1$flag = "Completed"

    res2 <- df2 %>% group_by(DATE) %>% summarise(Freq=n())
    res2$flag = "Findings"

    comb = rbind(res, res1, res2)

    o1 <- c('2019-02-01', 0, 'Total')
    o2 <- c('2019-02-01', 0, 'Completed')
    o3 <- c('2019-02-01', 0, 'Findings')
    comb <- rbind(comb, o1, o2,o3)
    comb$DATE <- format(as.Date(comb$DATE, "%m/%d/%Y") )
    comb <- comb[order(comb$DATE), ]
    key <- row.names(comb)

    p <- ggplot(comb, aes(x = DATE, y = Freq, group=factor(flag),
color=factor(flag))) +
      geom_point() +
      geom_line()+
      xlab(" Dates ") +
      ylab("Number of Deliverables (Data Sets + TFLs Combined)")
    ggplotly(p) %>% layout(dragmode = "select")


  })

}

shinyApp(ui, server, enableBookmarking = "server")
```

REFERENCES

SDTM to ADaM + Standard TLF Reporting Code = Efficient A&R: A Middle-to-End Approach by C Valerie Williams on CDISC.org:
https://www.cdisc.org/system/files/all/event/restricted/2017_International/INTX17%20Session%204%20Track%20A_Williams.pdf

Persistent data storage in Shiny apps (article by Dean Attali on Shiny.Rstudio.com):

https://shiny.rstudio.com/articles/persistent-data-storage.html

ACKNOWLEDGEMENTS

We would like to acknowledge our managers John Shaik and Shefalica Chand for reviewing our paper and providing valuable feedback.

CONTACT INFORMATION

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

Girish Kankipati
Seattle Genetics, Inc.
21823 - 30th Drive S.E.
Bothell, WA 98021
425-527-2140
gkankipati@seagen.com

Hao Meng
Seattle Genetics, Inc.
21823 - 30th Drive S.E.
Bothell, WA 98021
425-527-2065
hmeng@seagen.com