PhUSE 2016

Paper DV02

Communicating results with R&Shiny

Mika Mäkinen, Roche, Basel, Switzerland

ABSTRACT

Publishing results in web browser friendly form gives us an opportunity to move from producing outputs to communicating results. Instead of producing paper-like outputs, we can provide interactive html views with custom tables and colorful visualizations. Good visualization can convey a 100-page listing into one plot. Adding interactivity helps us to communicate the results in form that is both engaging and usable for stakeholders. Usable visualization enables the stakeholder to see patterns and anomalies in the data. In addition, they help programmers to understand the data better. Shiny offers out-of-box ready pipeline to publish R graphics in html. R packages have great tools for visualizations that can be enhanced with interactivity of Shiny. In this paper, we examine some visualizations that have be used to communicate results to stakeholders; reaching from basic interactive event plot that we have used in real study, to a concept of customized graphics that are able to tell the story of a study trough animation.

INTRODUCTION

Statistical programming is traditionally centered on providing printable TLGs to stakeholders. TLG package includes all the information that the study has been designed to investigate. However, it is hard to call TLG package printout, or PDF viewer, user friendly. TLG paper-like form is not modern usable solution to interpret the results. Using html for publishing outputs enables programmer to be free from restrictions of paper-like publishing. Therefore outputs can be designed to be, for example, larger than before and have elements, like animation, that are unavailable with paper publishing. With the new possibilities, outputs can be set to highlight certain things, to communicate important aspects of the data or the choice of importance can be given to user with interactive options. Either way, the process of producing and using the output differs from traditional output production and the programmer has to think how to communicate the results in an effective way.

This paper examines three cases of html publishing. Cases cover interactivity, changing underlying definitions and animation. As this article is in paper form, the outputs presented here are not in their right environment. Therefore two out of three outputs come with executable code, that allow reader to look into outputs as they are meant to be looked at, i.e. in web browser.

Publishing in html presents many challenges, of which some are out of the scope of this paper. For example software usability and use of right shapes and colors are challenges that might be new for statistical programmer, but are not covered here. This is not to say that these issues would be of little importance. On the contrary, for every production level output, little details should be carefully considered as user experience is in most cases dependent on these choices.

Presented outputs are simplified to be able to share the essential code in executable form. It is highly recommendable to look into usability and other similar issues before presenting outputs based on these code snippets. Code snippets are tested in Windows PC with Rstudio and in Unix server environment with Rstudio. As these setups are available for most, the author does not see need to support other combinations.

The paper consists of an introduction and three visualizations. First simulated dataset is introduced to be used with the executable code. The first and second visualizations are two relatively simple interactive plots. They are discussed using simulated data followed by a discussion of a case from a real study where they have been used. The third visualization is a concept of combining several plots and animating them over time as the events occurred during the study. At time of this article, this plot has not yet been applied to real life because the study is still ongoing.

SIMULATED DATA

The simulated data is from an imaginary study where patients were reporting events on their skin (Rash, Bruise, Swelling, Cut) and treatments that they used to treat these conditions (Gel, Cream, Tablet, Capsule). This setup does not reflect any real life study and it is just meant to be easily approachable and understandable setting. In this paper, simulated data is mainly used to visualize the relationship between event and treatment.

Data was simulated using R's sample function. For ease of simulation, every event was assigned with a treatment or NA for treatment. This link was, however, broken at the end of the simulation code to result with data, where events and treatments were close by each other, but there was no direct link between one event and treatment. This was done to simulate real life situation, where, especially oral, medicine can be considered to treat multiple conditions. Treatment also lack location on the body, so it is assumed that it was not gathered from patients. For this paper, data is simulated for 20 patients. Below data for an example patient and code needed to reproduce similar data.

patientNum	eventDay	event	eventCat	bodyLocation
100	55	Swelling	Event	8
100	65	Rash	Event	6
100	56	Gel	Treatment	NA
100	69	Capsule	Treatment	NA

```
##load libraries
library(grImport)
library(XML)
library(gridExtra)
library(RColorBrewer)
library(shiny)
library(ggplot2)

##1. create events from 1-maxevent per patient
##2. assign event with category, give weights
##3. apply body location for use in body 62 locations
##4. create treatment days in relation to event day, closest ones and NA:s are the most probable
##5. assign treatment with weights
##6. remove treatments where day is NA ie. no treatment applied
##7. assign treatments as event and eventDay, so that the connection between event and treatment is lost
##8. remove body location from Treatments
##9. Add patient number
##10. Create dataset
##11. Remove lines where eventDay is NA

createPatient <- function(pts,maxevent=80){
  eventDay <- sort(sample(1:365,size=sample(1:maxevent,size=1),replace=T))
  event <- sample(c("Rash","Bruise","Swelling","Cut"), size=length(eventDay),replace=T, prob=c(0.6,0.1,0.2,0.1))
  bodyLocation <- sample(1:62, size=length(eventDay),replace=T, prob=sample(c(0.1,0.2,0.3,0.4),size=62,replace=TRUE))
  treatmentDay <- eventDay + sample(c(0:9,NA), size=length(eventDay),replace=T,
prob=c(0.4,0.2,0.05,0.03,0.02,0.01,0.01,0.01,0.01,0.01,0.25))
  treatment <- sample(c("Gel","Cream","Tablet","Capsule"), size=length(eventDay),replace=T,prob=c(0.3,0.2,0.4,0.1))
  treatment <- ifelse(is.na(treatmentDay), NA, treatment)
  eventCat <- c(rep("Event",times=length(eventDay)),rep("Treatment",times=length(eventDay)))
  event <- c(event, treatment)
  eventDay <- c(eventDay, treatmentDay)
  bodyLocation <- ifelse(eventCat == "Treatment", NA, bodyLocation)
  patientNum <- rep(pts,times=(length(eventDay)))
  dataOut <- data.frame(patientNum, eventDay, event, eventCat,bodyLocation, stringsAsFactors = FALSE)
  dataOut <- dataOut[!is.na(dataOut$eventDay),]
return(dataOut)
}
##create data
ptsData <- do.call("rbind",lapply(100:120,createPatient,maxevent=80))
```

INTERACTIVITY IN MULTIPLE EVENT PLOT

Recurring events visualization can be problematic when there are many events. With paper one can always go from portrait to landscape, but the limitations are obvious. We used R and Shiny to create plot that is best viewed on HD-monitor. Plot has 8 different types of events and the user can display all or only certain type/s as needed. The visualization resulted in multiple meaningful combinations of events in real study application. With this visualization stakeholders were able to discuss about results, endpoint definitions and actually find some systematic anomalies in the data.

R/SHINY APPLICATION

Executable code for this plot can be found in the last subchapter of this chapter. We go here through the key choices of producing the code. Snapshot of visualization is presented in next subchapter.

Multiple event plot is a scatter plot with categorized y-axis. Ggplot2 is great library for developing scatterplots in an effective way. With ggplot function legend of a plot is really easy to define, and this was in this case decisive feature over base graphics. Most notable difference from default ggplot setup is that in this plot we wanted to remove the ggplot-typical cray background.

The Shiny app is used to select the events and treatments by the user. Per default ggplot will choose colors and shapes for individual plots, but as we are dropping and reselecting events, we want shape and color of event category to remain the same. That is why color and shape are explicitly fixed in the code. For Shiny, we use basic functionality. That is, CheckboxGroupInput function gives us the option to select multiple events at same time. User's choice is reflected in the server side with reacting to choice by sub selecting the data accordingly. The plot is rendered in the server and then plotted in UI by plotOutput function.

ASSESMENT OF VISUALIZATION WITH SIMULATED DATA

As we have previously simulated the data, deep analysis on what we see in plot is futile. However by looking at the plot we can get idea of what could be points of interest. For this screenshot, we have selected one event, Rash, and two treatments, Gel and Capsule. Let's assume that the study setup expected rash to be mainly treated with gel and capsule. From the visualization we can see, that patient 111 has treated all rash events exactly like we would have expected. If we look at patient 112, it seems that over half of rash events are not treated with expected treatments. This kind of discrepancy could be found, of course, also by using normal listing but from plot the visual comparison and assessment is more intuitive and instant. Also comparison between patients is easier in a plot than it would be in a listing.

COMMUNICATING RESULTS IN REAL STUDY

Multiple event plot was created for a non-interventional study. As the study was not blinded we were able to start to observe the results as soon as they were available. Visualization was enhanced with statistics that were reporting accumulation of data in numeric form. As visualization was available on server, stakeholders were able to access the results whenever they wanted. The data was updated in the background so the server had always the freshest data possible.

As we were working on a new disease area for the company, there were many things to learn. This is why the visualization was used within biometrics discussions to understand the data structure. Statisticians and statistical programmers were able to see the relations between events and treatments, and talk about differences to what was expected. At the start, mapping of data to sdtm was a challenge and this visualization was used to motivate the data management team and also to point out the needs that were not addressed in sdtm data.

For outside biometric stakeholders first implication of using Shiny was that they got the initial view of the data really fast. R and Shiny are agile way to publish data and we received comments complementing our speed from receiving initial data to having visualization available. Later the visualization was used within statistics, statistical programming and clinical science to refine the definitions of the primary and secondary endpoints in a new disease area where no standards are available. In fact, the team decided to add a new secondary endpoint to all studies based on the findings from the visualization in order to more accurately assess the true treatment effect. Furthermore, the clinical science team was immediately able to identify unexpected behaviors in the data which helped them to focus the medical review to these issues.

The visualization saved a lot of time and work for the whole team. It would have been very challenging for the statistics and clinical science team to come up with the right questions to be answered through traditional outputs because of the unexpected behaviors in the data. In addition, working together using the visualization in meetings created a nice team spirit and statistical programming became a valuable member of the team.

EXCUTABLE CODE TO PRODUCE SHINY PLOT

```
##add constant colors and shapes
myCol <- c(brewer.pal(9, "OrRd")[5:8], brewer.pal(9, "PuBu")[5:8])
eventLabels <- c("Rash","Bruise","Swelling","Cut",
                 "Gel","Cream","Tablet","Capsule")
names(myCol) <- eventLabels
mySha <- c(1,1,1,1,3,3,3,3)
names(mySha) <- eventLabels

##plot
eventPlot <-  function(dat, myCol, mySha){
p <-  ggplot(data = dat, aes(x = eventDay, y = as.factor(patientNum),  colour =event))
p +
  scale_y_discrete(drop=FALSE) +
  guides(fill=FALSE) + geom_point(size=3, alpha = 1, aes(shape=event)) +
  theme_bw() +
  theme(axis.text.x = element_text(angle = 90, hjust = 1),
        panel.background = element_rect(fill='white', colour='white')) +
  scale_colour_manual(name = "Event", values=myCol) +
  scale_shape_manual(name = "Event",values=mySha) +
  labs( x = "Days",  y = "Patient") +
  geom_vline(xintercept = 0, size=.3, color="white" ) +
  theme( # remove the vertical grid lines
    panel.grid.major.x = element_blank() ,
    panel.grid.minor.x = element_blank() ,
    panel.border=element_blank()
  )
}

##Shiny
ui <-fluidPage(
  checkboxGroupInput(inputId = 'phueve',
```

```
                label = 'Events:',
                choices =  eventLabels ,
                selected = eventLabels ,
                inline=TRUE),
  plotOutput("eveP")
)


server <- function(input, output){
  dataset <- reactive({
    ptsData[ptsData$event %in% input$phueve,]
  })
  output$eveP <- renderPlot({
    eventPlot(dataset(), myCol = myCol, mySha = mySha)
  })
}

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

VISUALIZATION OF CHANGE IN DEFINITIONS

Data from the previous study was used to assess definitions to be used in a next study. We used R to create distributions plots from the data and had sessions with stakeholders to assess the definitions. Shiny was used to interactively change the assumptions and immediately see how this would have changed the results of the previous study. With simulated data we are looking simply into finding the first treatment after an event using days from event to treatment to assess the validity of a treatment (i.e. if the treatment is too far from the event it is not considered to have been administered to treat that event). Target of this analysis would be to make a clear definition that would distinguish between events that are treated and events that are not treated.

R/SHINY APPLICATION

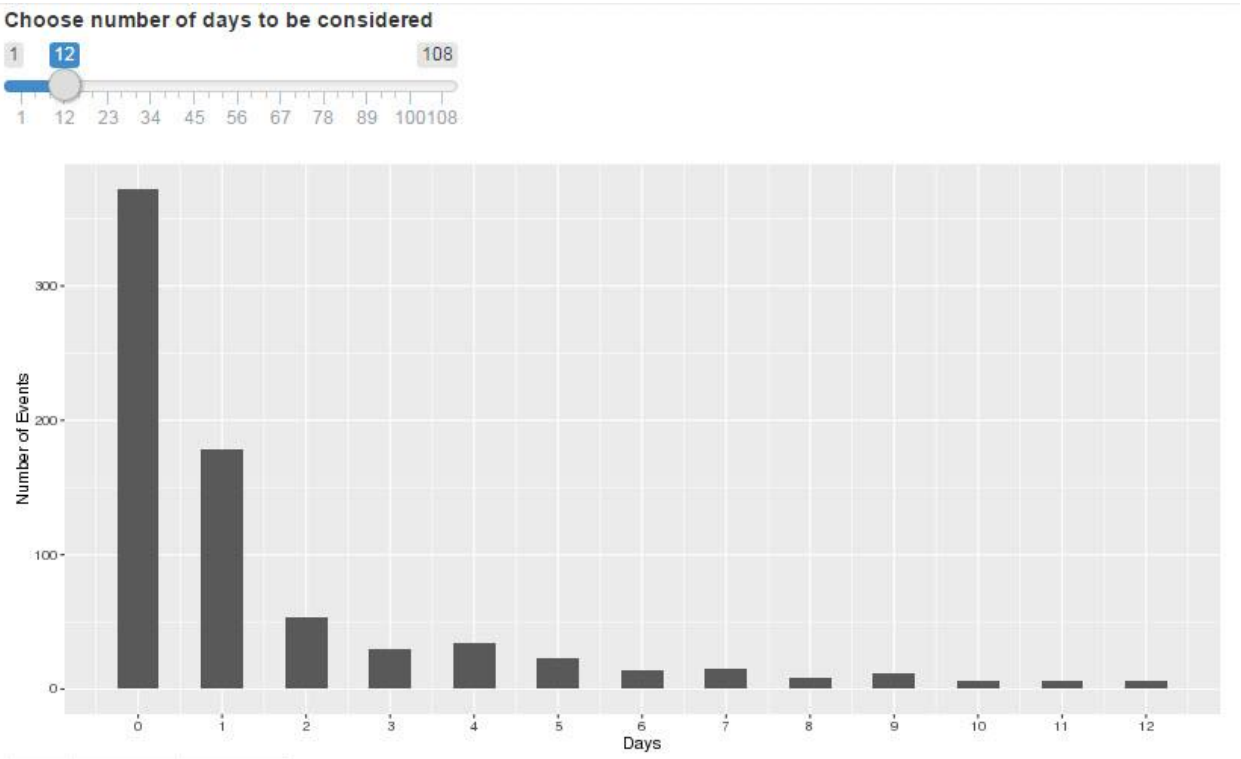
Executable code for this plot can be found in the last subchapter of this chapter. We go here through the key choices of producing the code. Snapshot of visualization is presented in next subchapter.

To assess the first treatment after an event, a dataset with the first treatment info is created. We use by function for calculations within patient. List created with by function is later combined to one dataset with do.call("rbind", data) call. Data is then plot to histogram using ggplot function.

Shiny uses sliderInput function to get user input of number of days after event day that should be included as treated. Input is reactive and affects "dataset" variable that then is used to feed into plot and accompanying table. These are then printed in UI.

ASSESMENT OF VISUALIZATION WITH SIMULATED DATA

The visualization is really simple: a histogram to present the distribution, and a table that gives information of the observations outside the histogram. User can input the number of days for event to be considered to be treated. Here the visualization could work as a starting point of discussion about good definitions. One could show it to stakeholders and ask questions like, should we consider all the treatments be valid if administrated within 7 days? Or maybe, does the gel work for bruises within the same timeframe than for cuts? These definitions can be then programmed in to shiny and used to redraw the histogram.



	included	excluded
nObs	757	46

COMMUNICATING RESULTS IN REAL STUDY

Distribution visualization was created for meeting series that addressed using non-interventional data as basis for exploratory definitions of event-treatment relationship. With non-interventional data we were able to show how change of definition of treated event would have changed the number of treated events in non-interventional study. Visualization was enhanced with table and listing of excluded and included events. Also options for defining how to count event as treated had more options than we demonstrated here with simulation.

One of the challenging parts of having large teams in study is to get people really committed to make decisions. For that visualization was helpful as it facilitated the discussion. During the meeting, we were able to show how change, for example, in time window would change the number of events to be considered as treated. During the meeting there was a comment from clinical science side that the definition should not be data driven. This was a valuable comment and probably helped clinical science to come up with clinically reasonable definition. Of course, our intention was not to make data driven definitions, but showing data driven visualizations facilitated decision making. Of note, in the end the visualization helped the team to understand that a time window based approach was not the right one for the question at hand and that the whole definition had to be re-defined.

EXCUTABLE CODE TO PRODUCE SHINY PLOT

```
##find the first treatments fr every bleed
firstTreatments <- by(ptsData, ptsData$patientNum, FUN=function(dat){
  trt <- dat[dat$eventCat=="Treatment",]$eventDay
  eve <- dat[dat$eventCat=="Event",]$eventDay
  out <-  lapply(eve,FUN=function(da,trt){
    diffToTrt <- trt-da
    diffToTrtNonneg <- diffToTrt[diffToTrt>=0]
    firstTre <- ifelse(length(diffToTrtNonneg) > 0, min(diffToTrtNonneg, na.rm=T), NA)
    data.frame(da,firstTre)
  },trt)
  dataout <- do.call("rbind", out)
```

```
  return(dataout)
})

firstData <- do.call("rbind", firstTreatments )
firstData <- firstData[!is.na(firstData$firstTre),]

##plot
histPlot <- function(dat){
p <- ggplot(data=dat, aes(dat$firstTre)) + geom_histogram(binwidth = 0.5)+
     scale_x_continuous(breaks = seq(min(dat$firstTre, na.rm=T),
                                max(dat$firstTre, na.rm=T),
                                by = ifelse(max(dat$firstTre, na.rm=T)<20,1,5))) +
     xlab("Days") +
     ylab("Number of Events")
p
}

histPlot(firstData)
##table

ui <-fluidPage(
  sliderInput(inputId= "ndays",
              label = "Choose number of days to be considered",
              value= 12,
              min = 1,
              max = max(firstData$firstTre),
              step = 1
              ),
  plotOutput("histP"),
  tableOutput("tabl")
              )

server <- function(input, output){
  dataset <- reactive({
    firstData[firstData$firstTre <= input$ndays,]
  })
  output$histP <- renderPlot({
    histPlot(dataset())
  })
  output$tabl <- renderTable(data.frame(included=nrow(dataset()),excluded=nrow(firstData)-nrow(dataset()), row.names =
"nObs"))
}

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

CUSTOM PLOT TO TELL THE STORY OF THE STUDY

CONCEPT

Statistical programming has traditionally concentrated on delivering outputs for regulatory reporting. This effort needs a rigorous approach where everything is explicitly explained in the output. A common convention is also that each data domain is reported separately and no combining of domains is done. With this concept, we wanted to examine what kind of visualizations would be possible, if we concentrated on telling a story of the study rather than focusing on rigorous number sharing.

This concept has never been used for a real study. Animation with simulated values has been shown to several study teams and feedback has been encouraging. It is good to remember that although the main customer of regulatory reporting are health authorities, there is lot of people working with the results, who are interested in them, but lack the statistical understanding and time to appreciate all the outputs. For this target group, more engaging solutions offer better way to communicate results.

The following plots are produced with R grid system. The main functions that were used to generate the visualizations are mentioned. The whole code is not shared because it is not as compact as the codes for the previous visualizations.

The visualization discussed here is multiple types of events for a patient during the study. The concept can be enhanced to include summary data of all the patients in the study, or by making side-by-side comparison between different groups. Snapshot of visualization is presented in last subchapter.

BODY HEATMAP

Body heatmap is a visualization that presents 62 body parts in one picture. Color intensity of a body part is dependent on amount of events that occurred in that location for a patient. This is a condensed way to show summary of the events for a given patient compared to a table. Correlations between body parts are better illustrated than, for example, using categorization, as no predetermined categorization has to be made.

To create body heatmap plot in R, a package called grImport was used. GrImport enables to read almost any vector graphic in to R and then to use the study data to manipulate the graph e,g, using different colors or size for different parts of the graph. Actual body outline picture was exported from the study protocol then vector drawing tool was used to draw individual body parts. Finally the ready picture was imported to R with grImport. To fill individual parts with different colors, all parts were mapped to data. Grid function pictureGrop was used to draw individual parts and they were combined with gropTree function. Finally grid.arrange was used to plot the whole picture.

EVENTS OVER TIME PLOT

Body heatmap shows events accumulating over time. To add more information to the same plot, events and treatments were plotted over time in the same way than in the first chapter of this paper. Here the aim was to create more a heatmap like visual look, where the question of interest is rather the number of events than details of each event. This approach gives better overall look of patient events during the study and works well in combination with the body heatmap.

This structure would allow different types of events to be added to the same plot. For example, in some studies some samples (e.g. bone marrow) are straining for patient to give, so they could be added to this plot as an event that affected patient during the study. As we are also plotting quality of life to the same visualization, having all the possible events affecting quality of life might create interesting correlations.

QUALITY OF LIFE PLOT

To illustrate the possibilities of animation, we add completely different dataset to be animated with skin event and treatment data. Skin conditions can be naturally combined with quality of life as problems with skin propably contribute to life quality. So we simulate data for quality of life between 0-100, measured once in 30 days. This data is then plotted with line on value-time axis.

COMBINING AND ANIMATING

Combining of the three plots could be done with grid system or in shiny. Here plots where combined with grid using grid.layout. When using grid to create complex visualizations, it is important to understand the use of functions like pushViewport and popViewport.

After most of the plot had been created with grid, the Shiny application was really simple to make, because the animation can be directly added as part of a slider. Animation (i.e. “play button”) is an out-of-the-box solution in Shiny and can be included as part of any slider. In order to control animations framerate, slider options allows programmer to choose steps of slider and interval of animation.

The resulting visualization is shown below. Here the animation is stopped at day 241 . User can whenever stop the animation and go back to previous frame with the slider. With real data, user could answer a question like; is there correlation between intensity of color in some body locations and quality of life, or maybe see if events clustering in time correlates with events clustering in body area.

CONCLUSION

Cases discussed in this paper demonstrate that visualizations which are viewable with a web browser are relatively easy to create. R and Shiny are good tools to create interactive visualizations. Independent of the tool, the most important aspect of visualization is the value that it brings to the user. Creating discussion, pointing out anomalies and engaging stakeholders are possible values that better visualizations can create. Being able to publish visualizations on a server makes them available to stakeholders all the time and makes the update process easy.

CONTACT INFORMATION

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

Mika Mäkinen
F. Hoffmann-La Roche AG
Grenzacherstrasse 124
Basel 4070, Switzerland
Building: 670, Room: 309.10
+41616878638
mika.maekinen@roche.com