Well, R is definitively here to stay and made its way into the data science tool zoo. For me as a statistician, I often feel alienated surrounded by these animals, but R is still also the statistician’s tool of choice (yes, it has come to age, but where are the predators ..?)

What was usually a big problem for us statistician, was to get our methods and models out to our customers, who (usually) don’t speak R. At this point Shiny comes in handy and offers a whole suite of bread and butter interface widgets, which can be deployed to web-pages and wired to R functions via all kinds of callback-routines.

Here is what I came up with (same data for better recognition ;-):

The whole magic is done with these lines of code:

library(MASS)

options(shiny.sanitize.errors = FALSE)

options(shiny.fullstacktrace = TRUE)

ui <- fluidPage(title="Shiny Linking Demo",

fluidRow(

column(5,

plotOutput("plot1",

click = "plot\_click",

brush = brushOpts("plot\_brush"),

width = 500,

height = 500

)),

column(5,

plotOutput("plot2",

click = "plot2\_click",

width = 500,

height = 500

))),

fluidRow(

column(5,

plotOutput("plot3",

click = "plot3\_click",

brush = brushOpts("plot3\_brush"),

width = 600,

height = 400

))

)

)

server <- function(input, output, session) {

keep <- rep(FALSE, 150)

shift <- FALSE

old\_brush <- -9999

var<- 1

keeprows <- reactive({

keepN <- keep

if (!is.null(input$plot\_click$x) | !is.null(input$plot3\_click$x))

keepN <- rep(FALSE, 150)

if (!is.null(input$plot\_brush$xmin) ) {

if( old\_brush != input$plot\_brush$xmin ) {

keepN <- brushedPoints(iris, input$plot\_brush,

xvar = "Sepal.Length",

yvar = "Sepal.Width",

allRows = TRUE)$selected\_

old\_brush <<- input$plot\_brush$xmin

}

}

if (!is.null(input$plot2\_click$x) ) {

keepN <- pmax(1,pmin(3,round(input$plot2\_click$x))) == as.numeric(iris$Species)

session$resetBrush("plot\_brush")

session$resetBrush("plot3\_brush")

}

if (!is.null(input$plot3\_brush$xmin) ) {

if( old\_brush != input$plot3\_brush$xmin ) {

var <<- round((input$plot3\_brush$xmin + input$plot3\_brush$xmax) / 2 )

coor\_min <- min(iris[,var]) + input$plot3\_brush$ymin \* diff(range(iris[,var]))

coor\_max <- min(iris[,var]) + input$plot3\_brush$ymax \* diff(range(iris[,var]))

keepN <- iris[, var] >= coor\_min & iris[, var] <= coor\_max

old\_brush <<- input$plot3\_brush$xmin

}

}

if( is.null(input$key) )

keep <<- keepN

else {

if( input$key )

keep <<- keepN | keep

else

keep <<- keepN

}

return(keep)

})

output$plot1 <- renderPlot({

plot(iris$Sepal.Length, iris$Sepal.Width, main="Drag to select points")

points(iris$Sepal.Length[keeprows()],

iris$Sepal.Width[keeprows()], col=2, pch=16)

})

output$plot2 <- renderPlot({

barplot(table(iris$Species), main="Click to select classes")

barplot(table(iris$Species[keeprows()]), add=T, col=2)

})

output$plot3 <- renderPlot({

parcoord(iris[,-5], col=keeprows() + 1, lwd=keeprows() + 1)

})

}

shinyApp(ui, server)

What makes this example somewhat special is:

* It does not need too much code
* It is relatively general, i.e. other plots may be added
* It uses traditional R graphics off the shelf
* It is not too slow

Of course it is a hack! But it proves that Shiny is capable to do interactive statistical graphics to some degree.