

Graphics in R - A Shiny Way to Program

Carolyn Nixon, Roche Products Ltd, Welwyn Garden City, UK

ABSTRACT

SAS® has long been the industry standard, but for the purpose of producing graphics the programming language and software environment R is fast becoming the new favorite. R has evolved vastly through the years, giving users three main ways to create graphs, base, lattice and ggplot graphics. In this paper you will learn about the evolution of these graphic packages, going into ggplot in more detail. In addition, a demonstration of how to incorporate R graphics into an interactive R Shiny application will be provided which is a dynamic way of visualizing data. The overall aim of this paper is to show users how R is leading the way in graphics and interactive visualization of our data.

Introduction

About R

R is a statistical programming language and software environment with particularly impressive graphics capabilities. R is not only free to use but is also open source, meaning R is constantly growing and will more than likely have the statistical methodology or graphical techniques you are looking to use¹.

Over the past couple of years, R has been a hot topic in the programming world and is being used more and more in the pharmaceutical industry by Data Analysts and Statisticians. The industry standard tends to be SAS® which is still widely used but R is catching up. R. Muenchen published an article on the popularity of R up to 2013² by the popularity of R in many different ways such as using job advertisements and surveys². This analysis shows how much attention R is now receiving and how it is being recognised worldwide.

Why would I recommend using R?

- R gives user's access to fast data exploration without having to learn a complex coding language.
- There are many packages available for users to choose from allowing easy automation and reproducibility.
- R creates quality graphics and figures.
- Customisation of almost every aspect of the plot is available and often easy to do.

Graphics in R

This paper will go through the three most commonly used graphical packages in R, base, lattice package and ggplot2 package. The positives and negatives of each will be addressed, with further information on why ggplot2 is now the best package to use. In addition, this paper will touch on R shiny, giving a simple example of an interactive graph.

Base

Base graphics cover the graphics that are already in R, so there is no need to download a package to have the base graphic functions available.

Base graphics are usually constructed through several separate functions. For example, one function to specify the x axis label and another to specify the data to be used. All documentation on base graphic parameters in the `par()` function.

You can think of base graphics as starting with a blank canvas⁵. Then the picture is drawn by adding the data to the plot, bit by bit (i.e function by function). One of the draw backs with base graphics is once you have started your plot you can't go back, you have to reconstruct your graph from the start again. In addition, with base graphics you have to specify and adjust your margins, this makes it hard to hold and transfer graph templates. Graphics produced using base plot tend to look very simple and clunky, performing more complex graphics is not easy to do.

Perhaps the most obvious base graphics function is `plot()`, however there are also `hist()`, `boxplot()` and many more. Examples 1 and 2 demonstrate how to use base graphics to create simple scatter plot.

Example 1

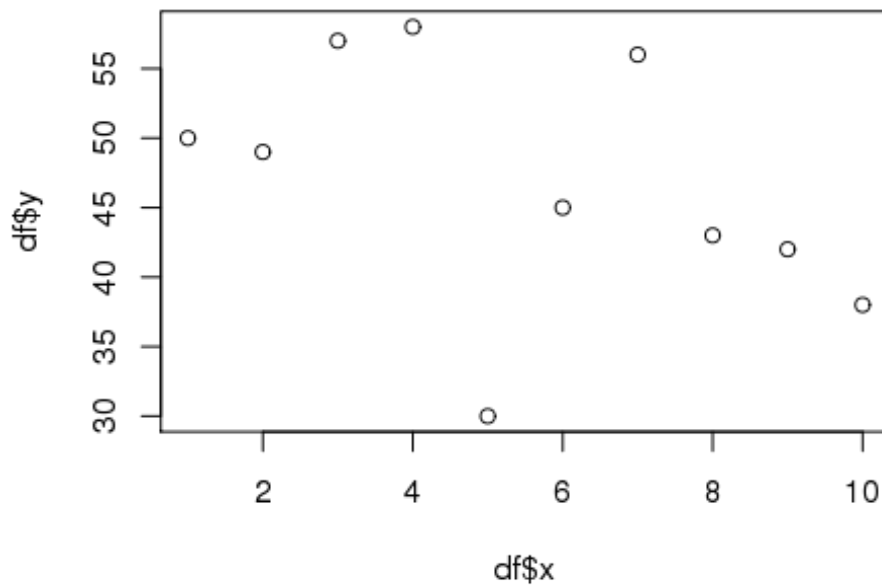
#Create data frame using vectors

```
x <- c(1,2,3,4,5,6,7,8,9,10)
y <- c(50,49,57,58,30,45,56,43,42,38)
z <- c("Blue", "Blue", "Brown", "Brown", "Brown", "Brown", "Blue", "Green", "Brown", "Brown")
zz <- c("Adult", "Adult", "Child", "Adult", "Adult", "Adult", "Adult", "Adult", "Child", "Adult")

df <- data.frame(x,y,z,zz)
```

	x	y	z	zz
1	1	50	Blue	Adult
2	2	49	Blue	Adult
3	3	57	Brown	Child
4	4	58	Brown	Adult
5	5	30	Brown	Adult
6	6	45	Brown	Adult
7	7	56	Blue	Adult
8	8	43	Green	Adult
9	9	42	Brown	Child
10	10	38	Brown	Adult

```
#Simple Scatterplot
plot(df$x,df$y)
```



Example 2

#Lets add more to this plot

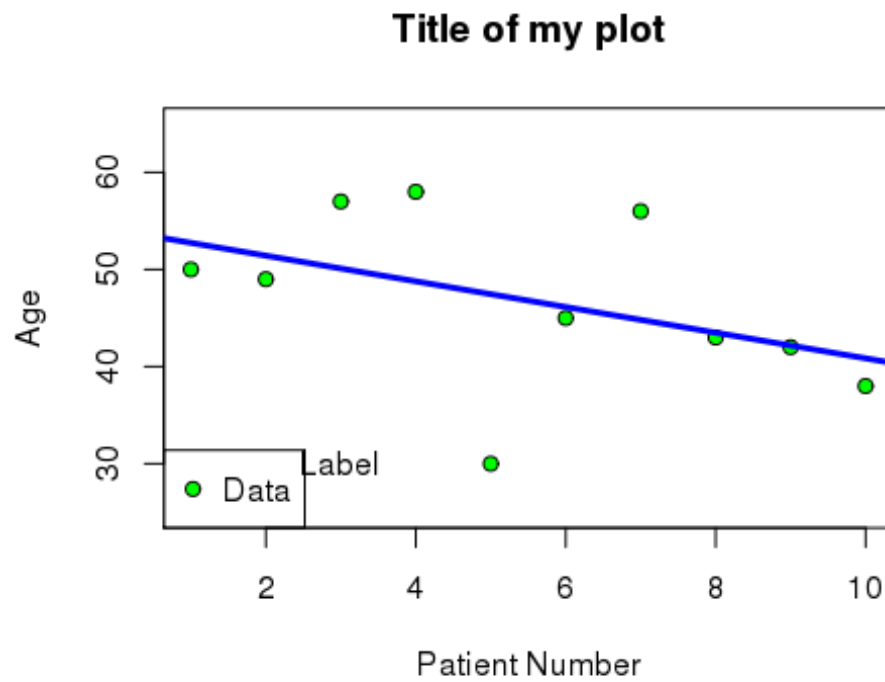
```
plot(x,                #x axis
     y,                #y axis
     xlab="Patient Number", #x axis label
     ylab="Age",         #y axis label
     pch = 21,          #plotting symbol
     mar = c(4, 4, 2, 2), #plot margin size
     col = "black",      #border colour for points
     bg = "green",       #background colour for points
     ylim = c(25,65))   #y axis limits

fit <- lm(y ~ x)        #fit a linear model between variables y and x
abline(fit,             #abline is a function that adds a line to the current plot
       lwd = 3,         #line width
       col = "blue")    #colour of line

title(main = "Title of my plot") #title of plot

text(3, 30, "Label") #label on graph

legend("bottomleft",    #add a legend to plot
       legend = "Data", #title of legend
       pch = 21,        #plotting symbol
       pt.bg = "green", #background colour for points
       col = "black")   #border colour for points
```



As you can see, base plot allows the user to create quick simple plots. However, adding further complex details can be tricky and having to run all the code from scratch to make a change to your plot makes base graphics cumbersome.

Lattice

The lattice package written by Deepayan Sarkar came slightly later than base graphics and contains code for trellis graphics. In contrast to base graphics, lattice needs to be downloaded as a package and is usually one single function. This package allows R to calculate spacing, margins and font sizes, although you can still specify these if needed.

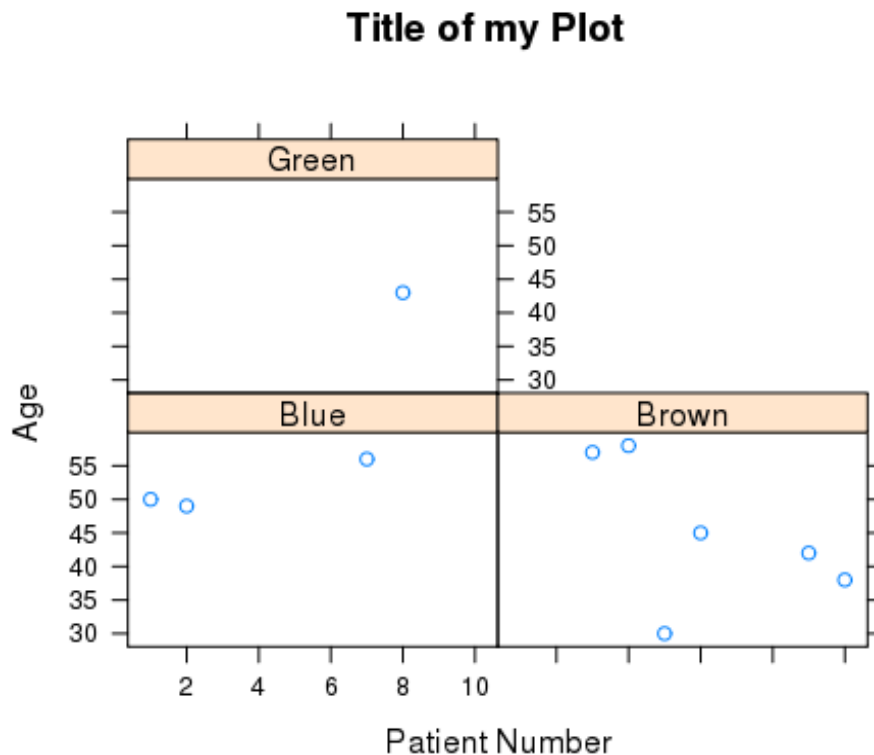
The downside of lattice is using one single function can be awkward, and the many options you have in lattice plot are not very intuitive. In addition as with base graphics, once you have created a plot you can't add to it.

Examples 3 and 4 demonstrate graphics using the lattice package (using the same dataset from examples 1 and 2):

Example 3

```
#Lattice Plot
library(package = "lattice")

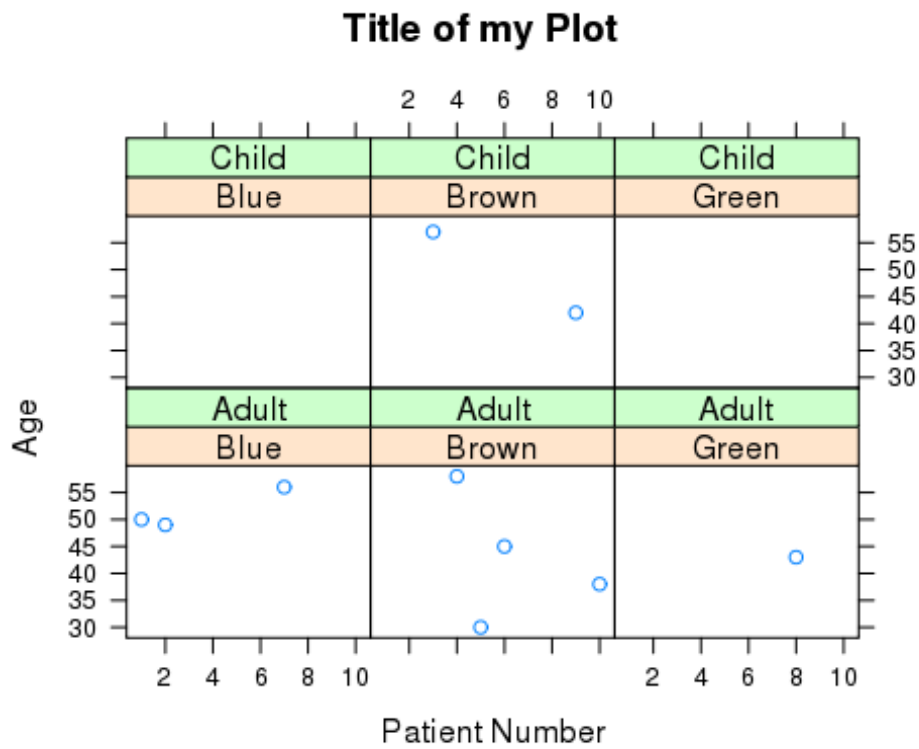
xyplot( y~x|z, #plot variables x by y given z
  data = df, #data
  main="Title of my Plot", #title
  ylab="Age", #y axis label
  xlab="Patient Number") #x axis label
```



Example 4

#Here is a further example adding two grouping/facet variables

```
xyplot( y~x|z*zz, #plot variables x by y given z
  data = df, #data
  main="Title of my Plot", #title
  ylab="Age", #y axis label
  xlab="Patient Number") #x axis label
```



Ggplot2

I would describe ggplot2, written by Hadley Wickham, as the best of both the base and lattice packages. Like lattice, ggplot2 automatically applies what it calculates to be the most appropriate spacing and margins. Just like base, ggplot2 starts like a blank canvas and each aspect of the graph is gradually added in separate functions. Ggplot2 is based on a book called 'Grammar of Graphics' by Leland Wilkerson³ which is designed to show e.g. statisticians and computer scientists how to visualize data in the best way. It looks at graphical components like verbs, adjectives and nouns in order to create new and useful graphics⁵.

The two basic functions in ggplot2 are `qplot` and `ggplot`.

`Qplot` should be used for simple and quick plots. `Ggplot` gives you the option for fine, granular control of all aspects of the plot and is therefore more flexible.

The basic components of using any functions in ggplot2 are:

- Data frame
- Aesthetic mapping (e.g. the colour and size of the data points)
- Geoms (e.g. points or lines to add)
- Facets
- Statistics (e.g. smoothing, quantiles)
- Scales (e.g. male = blue, female = red)
- Coordinate system

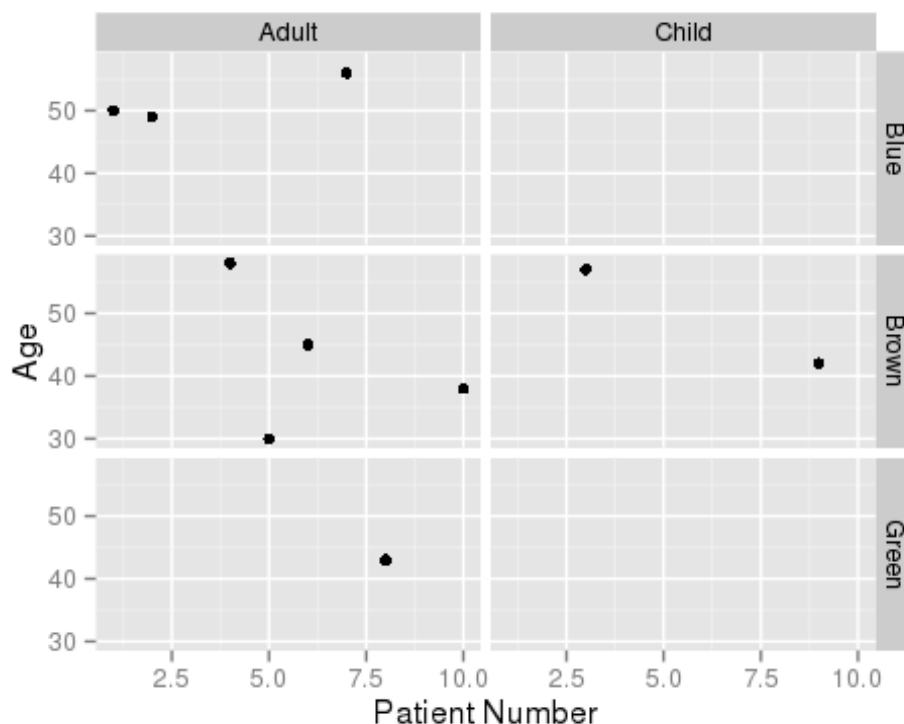
ggplot is both powerful and flexible. Ggplot is logical and therefore easy to pick up, as well as this there is a lot of online material and examples.

Examples 5 and 6 demonstrate the use of `qplot`, example 7 demonstrates `ggplot` (using the same dataset from examples 1 and 2):

Example 5

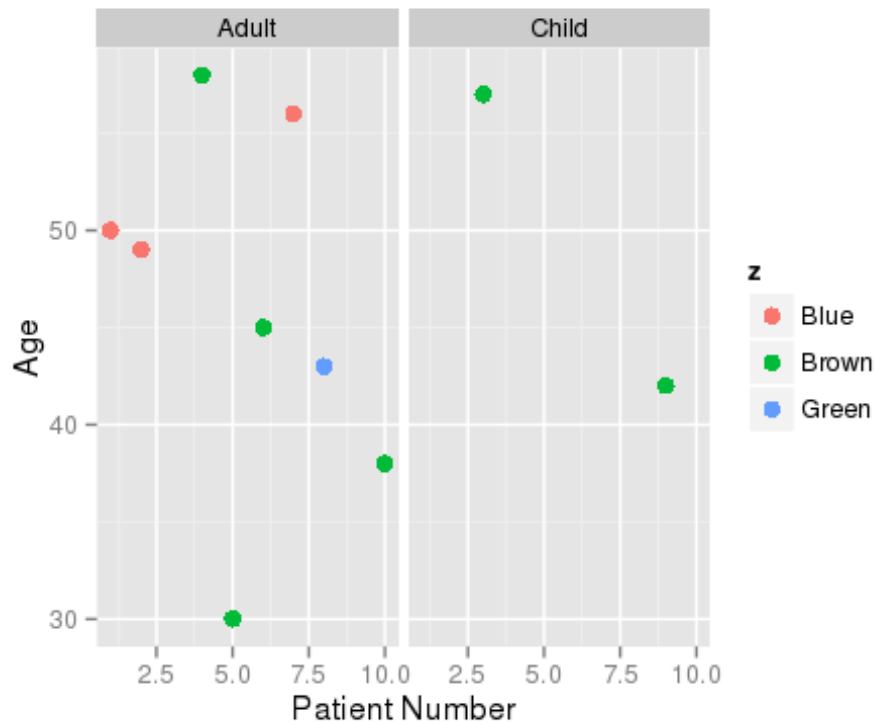
```
#Ggplot2
library("ggplot2")

#qplot (quick plot)
qplot(x, #xaxis
      y, #y axis
      data=df, #data
      facets=z~zz, #grouping/facets
      ylab="Age", #y axis label
      xlab="Patient Number") #x axis label
```



Example 6

```
qplot(x,
      y,
      data=df,
      color = z,
      facets=~zz,
      size=I(3),
      ylab="Age", #y axis Label
      xlab="Patient Number") #x axis Label
```



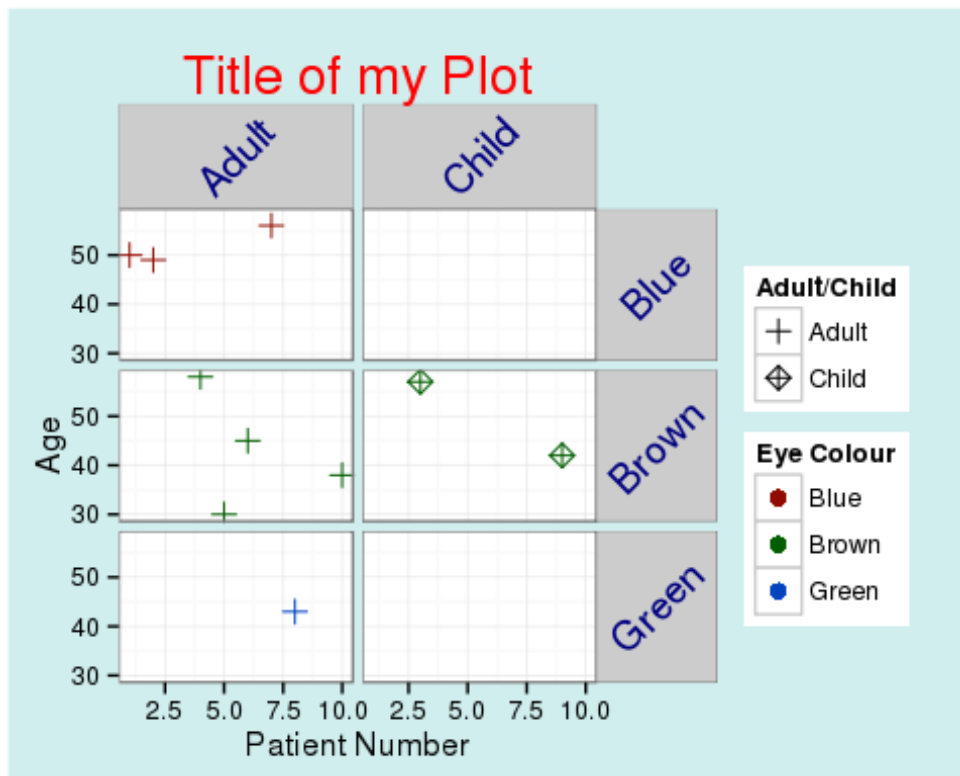
You have now seen an example of using qplot to create quick and simple graphs. Ggplot gives you access to the granular details in the graph as well as complex statistics and analysis. Here is an example of using ggplot:

Example 7

```
#ggplot

p <- ggplot(df, aes(x=x, y=y, group = z, color = z, shape = zz)) #very basic scatterplot
p <- p + geom_point(size = 3) #add points
p <- p + theme_bw() #remove background grid
p <- p + xlab("Patient Number") #x axis Label
p <- p + ylab("Age") #y axis Label
p <- p + ggtitle("Title of my Plot") #plot title
p <- p + facet_grid(z ~ zz) #groups/facets
p <- p + scale_colour_hue(name="Eye Colour", l=30) #Name of first legend and colours to use
p <- p + scale_shape_manual(name="Adult/Child", values= c(3,9)) #name of second legend and
symbols to use
p <- p + theme(strip.text.x = element_text(size = 15, colour = "navy", angle = 45)) #alter
text aesthetics for x axis facets
p <- p + theme(strip.text.y = element_text(size = 15, colour = "navy", angle = 45)) # alter
text aesthetics for u axis facets
p <- p + theme(plot.title = element_text(colour = "red", size = 20)) #alter title aesthetic
s
p <- p + theme(plot.background = element_rect(fill = "lightcyan2")) #alter plot background
colour

p
```



You can see from the last example that ggplot logic is easy to pick up, adding different layers or details to the plot with one line of code (no need to re-run from scratch as with lattice plot). Complex statistical analyses can be incorporated using the same logic.

R Shiny

R Shiny is a web application framework for R which allows users to create interactive web applications without the need to know any HTML, CSS or JavaScript⁴. The entire interface can be built using R, however it can also be written using HTML, CSS or JavaScript. It is also worth noting that R shiny can be run in any R environment, I tend to use Rstudio as it has a user friendly interface (and in my opinion it is superior to the UNIX SAS® or Windows SAS® interface).

R shiny allows you to customize your app in many different ways such as the layout, animations and users have access to a wide range of widgets. Another useful aspect of R shiny is it allows you to show reactive content, which gives the application a 'live' quality. R shiny is intuitive and the tutorial's provided on the official Shiny website provides beginners with everything they will need to start.

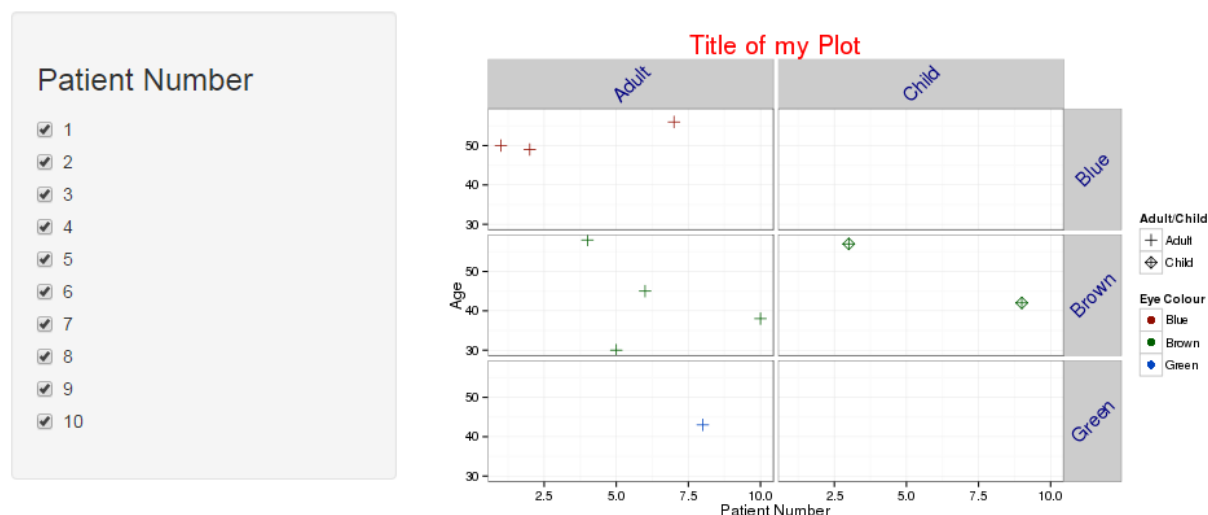
An R Shiny app has three different scripts:

1. ui.R: The code to define the how user interface should be set up.
2. Server.R: The server-side of the application, the code running behind the user interface.
3. global.R : the global environment, this code is visible and run before ui.R and server.R

Examples 8 and 9 are examples of a simple R shiny application using the ggplot output in example 7. This application allows the user to have a reactive plot where the user can choose which patients are included in the plot. The plot automatically updates every time the user chooses a tick box, whether it is one box or many. Below the examples you will see the code behind them

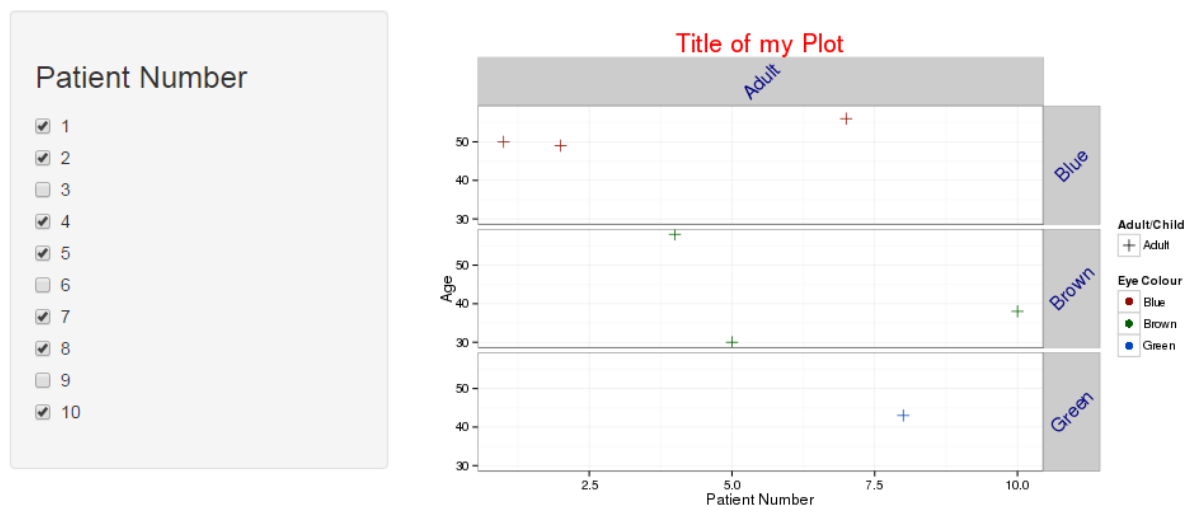
Example 8

Title of my Plot



Example 9 (Subset on patients 1, 2, 4, 5, 7, 8 and 10)

Title of my Plot



Conclusion

To summarize, there are three main graphics packages in R. Base, lattice package and ggplot2. Base graphics was the first of these as it is already built into R, and the lattice package was soon after giving users an easier way to create trellis plots. Ggplot2 is now the most recent of the three to be developed and best graphical package to use, whether you want to create a simple or complex plot. The package gives you the two functions qplot and ggplot, qplot should be used for simple plots, ggplot for more complex plots. The ggplot function allows the user to make granular, fine alterations to the plot and also gives access to complex statistical analyses. The syntax is easy to pick up as the code is logical and there are plenty of online examples.

Perhaps the star of the show however, is R Shiny. R Shiny allows users to create impressive visualizations of the data easily, by using R to manipulate HTML, CSS and JavaScript code. All R graphics can be incorporated in an R Shiny application and developed into a live, interactive application.

R is fantastic tool for graphics, providing a vast library of packages and therefore most statistical analysis methods. With R now being used globally, it is leading the way in graphics and interactive visualization of our data.

References

1. <http://www.r-project.org/about.html> - What is R? (*The R foundation*)
2. <http://r4stats.com/articles/popularity/> - The Popularity of Data Analysis Software (*Robert A. Muenchen*)
3. <http://www.springer.com/us/book/9780387245447> - The Grammar of Graphics (*Leland Wilkinson*)
4. <http://shiny.rstudio.com> – Shiny by R Shiny website
5. <https://www.coursera.org/course/rprog> - R programming Coursera course (*Roger Peng, Jeff Leek and Brian Caffo*)

Contact Information

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

Carolyn Nixon
Roche Products Ltd,
6 Falcon Way,
Shire Park,
Welwyn Garden City
AL7 1TW
United Kingdom

Work Phone: (44) 1707 36 6694
Email: Carolyn.nixon@roche.com