A visual tour of interactive graphics with R

Christophe Lalanne[*]

March, 2011

Abstract: We here describe simple use of interactive data analysis using the iPlots R package. The idea is to use brushing in linked graphics to foster exploratory analysis and model diagnostic. Other R packages are discussed.

Packages: iPlots • rgl • rggobi

1 Motivations

Far from being an exhaustive review of interactive and dynamic statistical graphics, the idea here is to review some of the available capabilities in R. A larger review is provided in Cook and Swayne (2007), using the GGobi software and its R interface.

We will focus on two aspects of interactive visualization, namely *brushing* (Becker and Cleveland, 1988) and *3D interactivty*.

2 The iPlots eXtreme package

The iPlots eXtreme package, aka Acinonyx (Urbanek, 2009), is available from <www.rforge.net>. It should supersede the traditional iPlots package. Although its functionnalities may appear rather limited at the moment, it already allows the user to explore data in an interactive manner, with linking and brushing enabled by default.

Let us assume a simple linear model of the form $y_i = 0.4 \times x_i + \varepsilon_i$, where $\varepsilon_i \sim \mathcal{N}(0, 1^2)$, that can be readily simulated in R as follows:

```
set.seed(101)
n <- 1000
x <- rnorm(n)
y <- 0.4*x + rnorm(n)
ip <- iplot(x, y)
```

Here is what R prints when we simply type ip at the command prompt:

```
Scatterplot y vs x (<AScatterPlot 0x100687e00>)
```

Well, it merely summarizes the type of object that is being plotted, and its address in memory. More information can be gathered by looking at its class:

[*] E-mail: ch.lalanne|at|gmail.com. Text available on www.aliquote.org, in /articles/tech/rvisuals.

```
[1] "iScatterplot" "iPlot"        "iVisual"      "iObject"
```

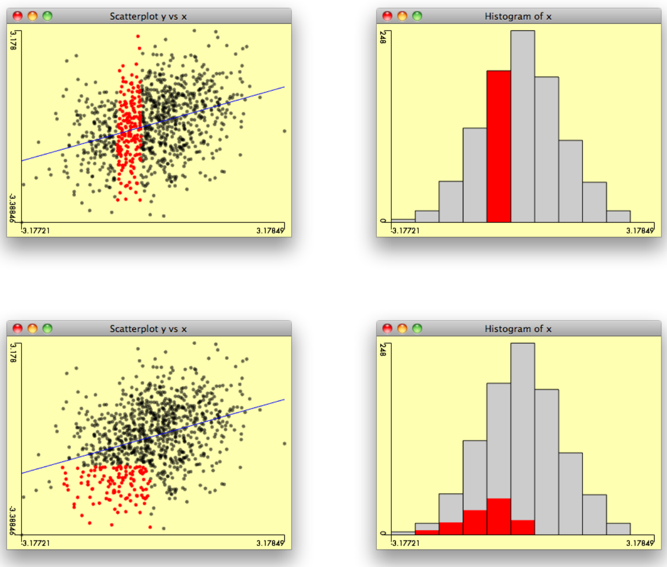
In fact, our scatterplot is a subclass of `iPlot` It does not support the '`formula`' interface, so data must be entered separately as `x` and `y`. However, overplotting is done by using transparency which results in nice-looking plots, while allowing to get a feel of the 2D density.

Now, adding a regression line is as simple as

```
ip + lm(y ~ x)
```

If we ask for an histogram of the x_i, the new plot will be automatically linked to the previous one. Note that it brings out a new graphic device, but we will learn shortly how to put them in a common frame.

```
ihist(x)
```



The top panel shows a scatterplot and an histogram for the same data, after we selected a certain range of `x` values. On the bottom panel, we do the reverse and select statistical units in the scatterplot.

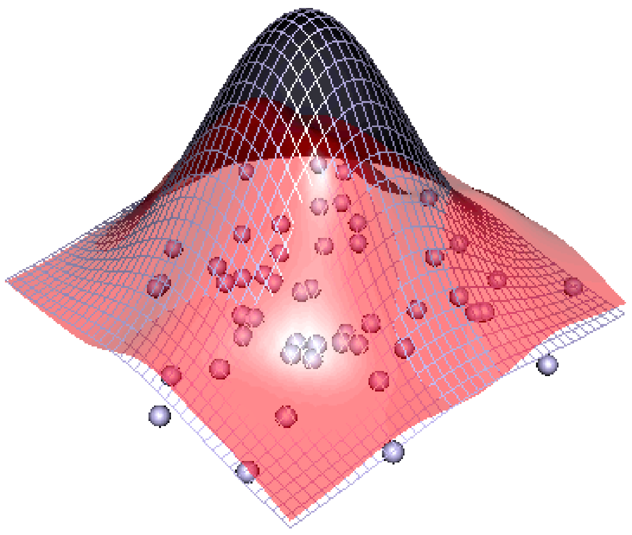
3 The `rgl` package

The `rgl` package, <http://rgl.neoscientists.org/>, uses OpenGL as a rendering engine, and provides interesting 3D viewing option, otherwise lacking in R.

To get a feel of `rgl` capabilities, just try

```
demo(bivar)
```

to show up a parametric density surface of a bivariate normal distribution.



The code to generate this figure is rather simple; here is a snipped version:

```
n <- 50; ngrid <- 40
x <- rnorm(n); y <- rnorm(n)
denobj <- kde2d(x, y,
n=ngrid)
den.z <-denobj$z
xgrid <- denobj$x
ygrid <- denobj$y
bi.z <- dnorm(xgrid)%*%t(dnorm(ygrid))
zscale<-20
# Draws simulated data
spheres3d(x,y,rep(0,n),radius=0.1)
# Draws non-parametric
density
surface3d(xgrid,ygrid,den.z*zscale,alpha=0.5)
# Draws parametric density
surface3d(xgrid,ygrid,bi.z*zscale,front="lines")
```

As an example, the following piece of code intends to show how PCA basically works. We first generate a matrix of random data, with a specific covariance structure, and then show the first three principal axes. Part of the code shown below comes from the excellent tutorials on **Information Visualisation** by Ross Ihaka.

```
sim.cor.data <- function(n=30, p=2, rho=0.6, sigma=1) {
  require(mvtnorm)
  H <- abs(outer(1:p, 1:p, "-"))
  V <- sigma * rho^H
  X <- rmvnorm(n, rep(0,p), V)
  return(X)
}
X <- sim.cor.data(n=100, p=5)
X.pca <- prcomp(X, scale=TRUE)
```

Now, constructing the 3D plots is done as follows.

```
rgl.open()
rgl.bg(color="white")

# display the 3D cloud
rgl.points(X.pca$x[,1:3], col="black", size=5, point_antialias=TRUE)

# set up a reference plane
xyz.lims <- apply(X.pca$x[,1:3], 2, range)
```

```
bot.plane <- min(xyz.lims[1,3]) - diff(xyz.lims[,3])/10
bot.plane <- mean(X.pca$x[,3])
rgl.surface(seq(xyz.lims[1,1],xyz.lims[2,1], length=10),
            seq(xyz.lims[1,2],xyz.lims[2,2], length=10),
            rep(bot.plane, 10*10),
            color="#CCCCFF", front="lines")
```

To capture the output, we can use `rgl.snapshot(filename)`, where `filename` is the name of the PNG file to be saved.

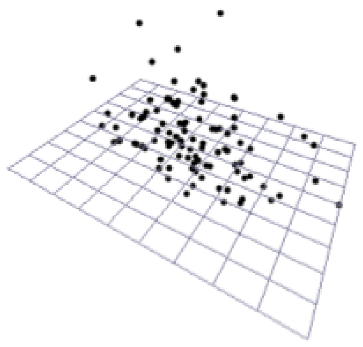
Instead of a reference plane, we could directly draw unit vectors

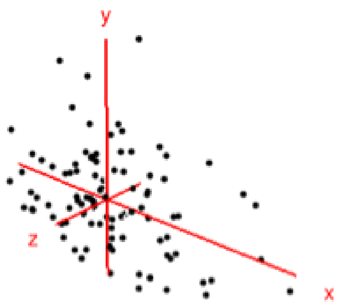
```
rgl.lines(c(0,1), c(0,0), c(0,0), col="red", lwd=2)
rgl.lines(c(0,0), c(0,1), c(0,0), col="red", lwd=2)
rgl.lines(c(0,0), c(0,0), c(0,1), col="red", lwd=2)
```

or axes (ranging from `min` to `max` observed values)

```
rgl.lines(xyz.lims[,1], c(0,0), c(0,0), col="red", lwd=2)
rgl.lines(c(0,0), xyz.lims[,2], c(0,0), col="red", lwd=2)
rgl.lines(c(0,0), c(0,0), xyz.lims[,3], col="red", lwd=2)
rgl.texts(c(xyz.lims[2,1]+.5,-.15,-.15),
          c(-.15,xyz.lims[2,2]+.5,-.15),
          c(-.15,-.15,xyz.lims[2,3]+.5), letters[24:26], col="red")
```

Both results are shown below.





Finally, there is no possibility of brushing an `rgl` device, but we can use *spinning* (here, 360°) with:

```
for(i in seq(0, 360, by = 1)) {
  rgl.viewpoint(theta = i, phi = 0)
  Sys.sleep(1/60)
}
```

There are alternative and more practical ways to the above, as found in e.g., `ordirgl` in the `vegan` package, or the `BiplotGUI` package that provides a complete environment for manipulating biplots (Gower and Hand, 1996), in 2D or 3D. For those who are seeking a more direct application of the commands discussed here, you can try to adapt the `sphpca` function in the `psy` package (Falissard, 1996).

4 Back to the basics

So far, we only talked about dedictaed environments for interactive visualization. However, the base R functionalities might still prove to be useful in some cases. In fact, the `tcltk` package offers a simple way to attach interactive buttons to the current device.

Let's say we want to intercatively display the most extremes individuals on a given matrix of scores. 'Extreme' could mean many things, but for now assume this is a percentile-based measure, for example the 5e and 95e percentile are used to flag individuals having extreme low or high scores.

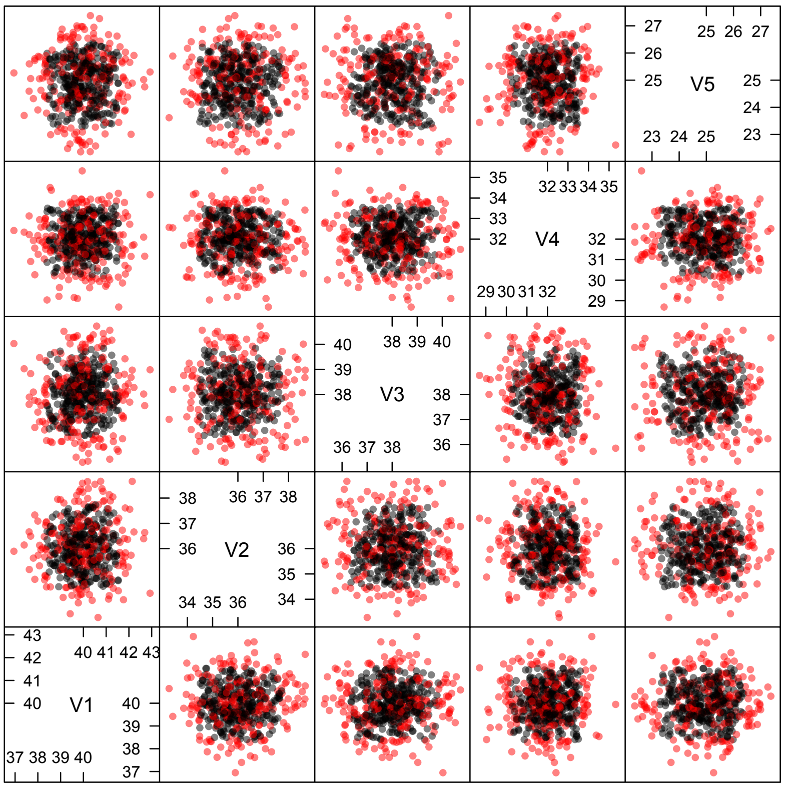
```
filter.perc <- function(x, cutoff=c(.05, .95), id=NULL, collate=FALSE) {
  lh <- quantile(x, cutoff, na.rm=TRUE)
  out <- list(x.low=which(x < lh[1]), x.high=which(x > lh[2]))
  if (!is.null(id)) {
    out[["x.low"]] <- id[out[["x.low"]]]
    out[["x.high"]] <- id[out[["x.high"]]]
  }
  if (collate)
    out <- unique(c(out[["x.low"]], out[["x.high"]]))
  return(out)
}

n <- 500
scores <- replicate(5, rnorm(n, mean=sample(20:40, 1)))
idx <- apply(scores, 2, filter.perc, id=NULL, collate=TRUE)
my.col <- as.numeric(1:n %in% unique(unlist(idx))) + 1
splom(~ scores, pch=19, col=my.col, alpha=.5, cex=.6)
```

A simple display for the distribution of these five series of scores is shown below, with individuals in red corresponding to those being in the lowest or highest fifth percentile. (Also, keep in mind that is done in a purely univariate manner.)

Now, what about varying the thresholds for highlighting individuals? Instead of repeating the same steps, we could simply add a dynamic selector to this display.

Using `aplpack::slider`, this can be implemented as follows:

```
do.it <- function() {
  require(aplpack)
  update.display <- function(...) {
    value <- slider(no=1)
    idx <- apply(scores, 2, filter.perc, cutoff=c(value, 1-value),
                 id=NULL, collate=TRUE)
```



Matrice de nuages de points

```
    my.col <- as.numeric(1:n %in% unique(unlist(idx))) + 1
    lp <- splom(~ scores, pch=19, col=my.col, alpha=.5, cex=.6)
    print(lp)
  }
  slider(update.display, sl.names="Percentile", sl.mins=0, sl.maxs=1,
         sl.deltas=.05, sl.defaults=.05)
}
splom(~ scores, pch=19, col=my.col, alpha=.5, cex=.6)
do.it()
```

There are a lot of other illustrations in the vignette *Some Slider Functions*, available on <http://cran.r-project.org/web/packages/aplpack/vignettes/sliderfns.pdf>.

5 Miscalleneous

TODO.

- discuss 3D PCA in psy
- mention BiplotGUI
- discuss ordirgl in vegan

```
library(Rcmdr)
attach(mtcars)
scatter3d(wt, disp, mpg)
```

References

Cook, D. and Swayne, D. (2007). *Interactive and Dynamic Graphics for Data Analysis With R and GGobi*. Springer. `http://www.ggobi.org/book/`.

Becker, R. and Cleveland, W. (1988). Brushing scatterplots. In Cleveland, W. and McGill, M., editors, *Dynamic Graphics for Statistics*, pages 201-224. Wadsworth & Brooks/Cole, Belmont, CA.

Urbanek, S. (2009). `iPlots eXtreme`. Next-generation interactive graphics for analysis of large data. In *UseR! 2009 Conference* . `http://www.r-project.org/conferences/useR-2009/slides/Urbanek.pdf`.

Gower, J. and Hand, D. (1996). *Biplots*. Chapman & Hall, London, UK.

Falissard, B. (1996). A spherical representation of a correlation matrix. *Journal of Classification, 13*(2), 167-280.