

Dynamic Graphics in R

Deepayan Sarkar

Department of Statistics
University of Wisconsin-Madison

October 19, 2005

R support

- ▶ Direct Manipulation
 - ▶ `locator()` identifies mouse click locations
 - ▶ Elements can be added to a graph, but not removed
 - ▶ Grid fakes removal by redrawing graph
- ▶ Traditional solution
 - ▶ Interface to external programs (`xgobi`, `ggobi`, `iPlot`, etc)

Dynamic Graphics

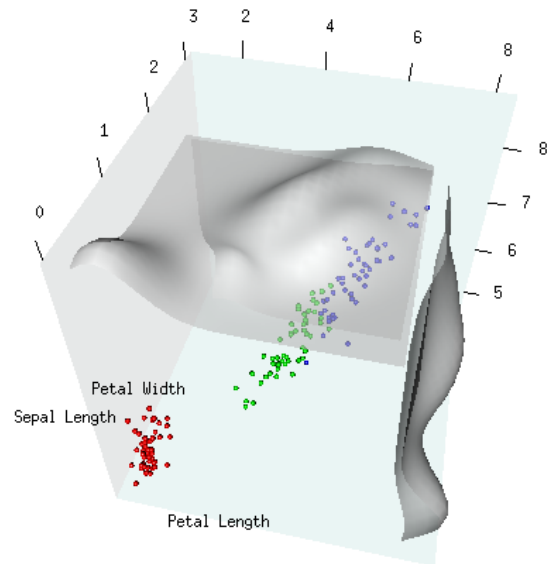
- ▶ **Direct manipulation**
 - ▶ Use mouse/keyboard to modify display
 - ▶ Linked displays
- ▶ **Dynamic display**
 - ▶ No change in data
 - ▶ Interactively modify rendering
 - ▶ Useful for 3-D displays
- ▶ **Animation**
 - ▶ Simple in principle
 - ▶ Combine multiple graphs into movie
- ▶ Poor native support in R

R support

- ▶ Dynamic 3D displays
 - ▶ Best current solution: OpenGL
 - ▶ Established 3-D system with support for
 - ▶ transparency
 - ▶ lighting
 - ▶ textures
 - ▶ Hardware acceleration with good graphics cards
 - ▶ `rgl` package provides low-level interface

Plan

- ▶ RGL basics and a detailed example
- ▶ Animation example



R support

- ▶ Animation
 - ▶ No direct support in R
 - ▶ Can produce multiple page PDF/Postscript
 - ▶ Easily combined by external software, e.g.
 - ▶ ImageMagick's `convert` program

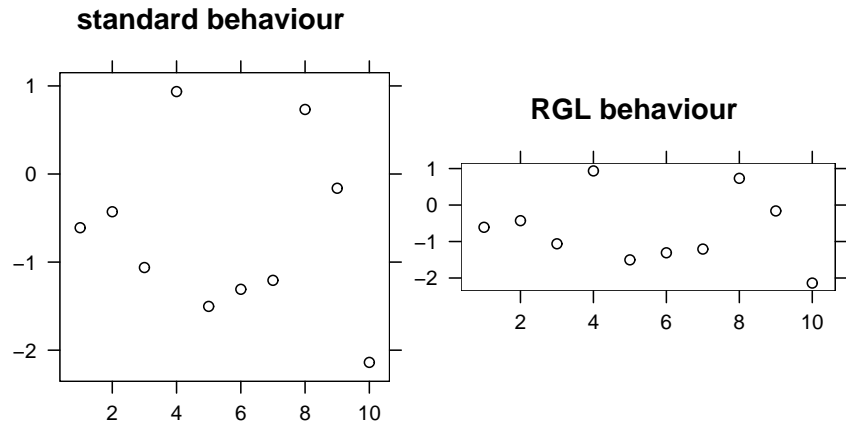
Goal

- ▶ Extend our “enhanced scatter plot” to three dimensions
 - ▶ 3-D scatter plot
 - ▶ Axes, labels
 - ▶ Marginal 2-D Density estimates

Coordinate system

- ▶ 3-D Cartesian coordinates (x, y, z)
- ▶ Absolute units (no in-built control of aspect ratio)

```
> plot(1:10, rnorm(10))
```



Scene management

- ▶ `bg3d`
 - ▶ Sets up background appearance
 - ▶ color, 'fog'
- ▶ `light3d`
 - ▶ Adds light source
 - ▶ Up to 8 allowed
- ▶ `clear3d, pop3d`
 - ▶ Clear or remove elements from a scene
- ▶ `view3d`
 - ▶ Change viewpoint (camera position)
- ▶ `bbox3d`
 - ▶ Add bounding box
 - ▶ Closest (built-in) thing to axes
 - ▶ Bounds *everything* in the scene

Graphical primitives

Type	Function
Points	<code>points3d</code>
Spheres	<code>spheres3d</code>
Lines	<code>lines3d</code>
Segments	<code>segments3d</code>
Triangles	<code>triangles3d</code>
Quadrilaterals	<code>quads3d</code>
Text	<code>text3d</code>
Surface	<code>rgl.surface</code>

Material Appearance

Name	Purpose
<code>color</code>	Color
<code>alpha</code>	Transparency
<code>lit</code>	whether lighting is to be used
<code>ambient, specular, ...</code>	lighting details
<code>texture</code>	Texture image
<code>etc</code>	

See `?rgl.material` for more

Function arguments

```
ecloud <-  
  function(x, y, z,  
           aspect = c(1, 1),  
           xlab = deparse(substitute(x)), [...]  
           xlim = extend(range(x)), [...]  
           bg = "white",  
           col = "pink",  
           col.box = "turquoise",  
           col.density = "grey",  
           col.axis = "black",  
           alpha.box = 0.1,  
           alpha.density = 0.8,  
           radius = 0.01,  
           dh = 0.2,  
           new = FALSE)
```

Function body

Checks and scene setup:

```
stopifnot(require(rgl))  
aspect <- rep(aspect, length = 2)  
if (new) open3d()  
clear3d()  
if (!is.null(bg)) bg3d(color = bg)
```

Rescale data

```
extend <- function(r) r + 0.05 * c(-1, 1) * diff(r)  
tx <- (x - xlim[1]) / diff(xlim)  
ty <- aspect[1] * (y - ylim[1]) / diff(ylim)  
tz <- aspect[2] * (z - zlim[1]) / diff(zlim)
```

Function body

Scatter, labels and bounding box

```
spheres3d(x = tx, y = ty, z = tz,  
          radius = radius,  
          color = col)  
bbox3d(color = c(col.box, col.axis),  
       xat = (pretty(xlim, 3) - xlim[1]) / diff(xlim),  
       yat = aspect[1] * (pretty(ylim, 3) - ylim[1]) / diff(ylim),  
       zat = aspect[2] * (pretty(zlim, 3) - zlim[1]) / diff(zlim),  
       xlab = format(pretty(xlim, 3)),  
       ylab = format(pretty(ylim, 3)),  
       zlab = format(pretty(zlim, 3)),  
       alpha = c(alpha.box, 1))  
text3d(x = c(0.5, 0, 0),  
       y = c(0, 0.5, 0),  
       z = c(0, 0, 0.5),  
       text = c(xlab, ylab, zlab),  
       color = "black")
```

Function body

Density (repeated for each plane)

```
require(MASS)  
dxy <- kde2d(tx, ty)  
dxy$z[] <- aspect[2] + dh * dxy$z / max(dxy$z)  
rgl.surface(x = dxy$x,  
           y = dxy$z,  
           z = dxy$y,  
           coords = c(1, 3, 2),  
           alpha = alpha.density)
```

Example call

```
with(iris,  
     ecloud(x = jitter(Petal.Length),  
            y = jitter(Petal.Width),  
            z = jitter(Sepal.Length),  
            xlab = "Petal Length",  
            ylab = "Petal Width",  
            zlab = "Sepal Length",  
            aspect = c(0.5, 2),  
            alpha.density = 0.8,  
            col = rainbow(3)[Species]))
```

Animation

- ▶ Produce multi-page PDF file in R
- ▶ Use the `convert` utility to make animated GIF
`convert -delay 10 in.pdf out.gif`
- ▶ Example: NASA data
- ▶ One plot for each timepoint
- ▶ Bubble plot
 - ▶ X, Y: location
 - ▶ radius of point: temperature
- ▶ Use R function `symbols()`

Other features

- ▶ RGL has other useful features
 - ▶ selection of region for choosing subsets
 - ▶ more general 3-D structures (mesh)
- ▶ Details available in package documentation

