R Graphics

Feng Li
`feng.li@cufe.edu.cn`

School of Statistics and Mathematics
Central University of Finance and Economics

Revised on August 15, 2018

Today we are going to learn...

A picture is worth a thousand words!

Basic R Graphical System

- The R graphics system can be broken into four distinct levels: graphics packages; graphics systems; a graphics engine, including standard graphics devices; and graphics device packages. Things can be quite complicated because there are two sets of graphical systems: the classical ones and those, more complex but much more powerful, from the lattice and grid packages.
- The basic R graphics system (e.g functions like "plot", "hist"...) are called **high-level graphics system** but it has a few problems. First, if you choose to divide the screen to display several plots, you cannot go back to an already produced plot to add to it – nor even to alter it. Second, you cannot nest those screen divisions.
- The grid system is a low-level graphics library that comes form the "Trellis" graphics in S-Plus. It addresses those problems in the classical graphics method. It is used, for instance, by the "lattice" library. *R Graphics*, a very popular book mainly about grid system was written by Paul Murrell.

The R graphics system

```
---------------------------------------------------------------------
Graphics Packages              maps...                    lattice ..
-----------------
Graphics Systems               graphics(classical)        grid
----------------
Graphics Engine & Devices                   grDevives
-------------------------
Graphics Device Packages                    gtkDevice ...
---------------------------------------------------------------------
```

Note: The full description of R graphics packages can be found in “CRAN Task View”
http://cran.r-project.org/web/views/Graphics.html

R graphics output can be produced in a wide variety of graphical formats (see Table). In R's terminology, output is directed to a particular output device and that dictates the output format that will be produced. A device must be created or opened" in order to receive graphical output and, for devices that create a File on disk, the device must also be closed in order to complete the output.
A complete R graphic work should consider the following steps

```
windows()  ## Start your favorite device
pdf(...)   ## File format and path if you want to save
plot(...)  ## Some plot here
...
dev.off()  ## Close the device
```

Graphical output formats

```
-------------------------------------------------------------------
Screen/GUI Devices
    X11()                           X Window window
    windows()                       Microsoft Windows window
    quartz()                        Mac OS X Quartz window
File Devices
    postscript()                    Adobe PostScript File (.ps,.eps
    pdf()                           Adobe PDF File (.pdf)
    pictex()                        LATEX PicTEX File
    xfig()                          XFIG
    bitmap()                        GhostScript conversion to File
    png()                           PNG bitmap File
    jpeg()                          JPEG bitmap File
Devices provided by add-on packages
    devGTK()                        GTK window (gtkDevice)
    devJava()                       Java Swing window (RJavaDevice)
    devSVG()                        SVG File (RSvgDevice)
-------------------------------------------------------------------
```

- Here are some R graphics examples (mainly about the classical graphical system).
- The aim for now is simply to provide an overall impression of the range of graphical images that can be produced using R.

The `par()` function

“par” can be used to set or query graphical parameters. Here are some very common usages. For more details check par function. 'las' numeric in 0,1,2,3; the style of axis labels. 0: always parallel to the axis (default), 1: always horizontal, 2: always perpendicular to the axis, 3: always vertical. Here is an example

```
par(mfrow=c(2,2)) # we plot them in a picture
par(las=0)
plot(0,main="par(las=0) default")
par(las=1)
plot(0,main="par(las=1)")
par(las=2)
plot(0,main="par(las=2)")
par(las=3)
plot(0,main="par(las=3)")
```

The position of axis labels

```
par(mfrow=c(2,2))
par(adj=0)
plot(0,main="par(adj=0)",ylab="ylab",xlab="xlab")
par(adj=.5)
plot(0,main="par(adj=.5)default",ylab="ylab",xlab="xlab")
par(adj=.7)
plot(1,main="par(adj=.7)",ylab="ylab",xlab="xlab")
par(adj=1)
plot(1,main="par(adj=1)",ylab="ylab",xlab="xlab")
```

The type of a graphical box

```
par(mfrow=c(2,4))
par(bty="o")
plot(0,main="par(bty=\"o\")")
par(bty="l")
plot(0,main="par(bty=\"l\")")
par(bty="7")
plot(0,main="par(bty=\"7\")")
par(bty="n")
plot(0,main="par(bty=\"n\")")
par(bty="c")
plot(0,main="par(bty=\"c\")")
par(bty="u")
plot(0,main="par(bty=\"u\")")
par(bty="]")
plot(0,main="par(bty=\"]\")")
```

Figure in figure

It is very useful to add a small graph in the main graph, the “fig” argument does exactly what we need. Look at this example

```
# give 100 random t with df 12
x <- rt(100, df=12)
# make a histgram
hist( x, col = "light blue")

# prepare to add a new small graph
op <- par(fig = c(.02,.42,.53,.99),new = TRUE)

# the qqnorm in the small graph
qqnorm(x, xlab = "", ylab = "", main = "", axes = FALSE)

# add a qq line in the small graph
qqline(x, col = "red", lwd = 2)

# set the line width of the small frame box
box(lwd=1)
# apply
par(op)
```

Double axises graph

Some economists like double axis graphs. Here how it works in R

```
set.seed(1)      # set the seed
par(mar=c(5,5,5,5))  # set the margin
# asumption this is Interest rate
x1<-sort(rnorm(100,mean=20,sd=5))
x2<-x1^3 # this is GDP
# plot GDP
plot(x2,axes=FALSE,type="l",col="blue",xlab="",ylab="")
axis(1) # add the axis on bottom
axis(2,col="blue") # add the asis on left

par(new=TRUE)  # add new plot on the current graph
plot(x1,,axes=FALSE,type="l",col="red",xlab="",ylab="")
axis(4,col="red") # add the axis on right

# We will give the three axises labels
mtext(c("Time","GDP(mil. $)","Interest rate(%)"),
      side=c(1,2,4),line=3)
title("GDP with interest rate") # give the main title
```

Figures layout

Assume we have 7 (odds) individual figures but we put them into a 2×4 frame which left a blank cell. To make things pretty we can do it like this:

```
# split display into two screens(1,2)
split.screen(c(2,1))
# split bottom half in two(3,4)
split.screen(c(1,2),2)

# activate screen (1), the top one and plot something
screen(1)
plot((1:10)^2,type="l",main="Screen(1)")
# activate screen (3), the top one and plot something
screen(3)
plot((1:10)^.5,type="l",main="Screen(3)")
# activate screen (4), the top one and plot something
screen(4)
plot((1:10),type="l",main="Screen(4)")
# exit split-screen mode
close.screen(all = TRUE)
```

Or we can use the more powerful function `layout()` in R. Check "layout" example

Add grid into the graph

```
par(mfrow=c(2,2))
plot(1:3)
# grid only in y-direction
grid(nx=NA, ny=5,col="gray")
plot(1:3)
# only x-direction
grid(nx=5,ny=NA, col="red")
plot(1:3)
# both x and Y
grid(nx = 5, ny = 5,lty="dashed",col="blue")
plot(1:3)
# x and y but different numbers
grid(nx = 3, ny = 5)
```

Math symbols & formula in graphics

Sometimes it is necessary to add mathematical formula in the graphics body. See “demo(plotmath)”

```
# make a hist but use the desity
hist(rnorm(100),xlim=c(-3,3),freq=FALSE)

# add a normal curve here
# 'expr' must be a function or an expression containing 'x'
x<-seq(-5,5,.01)
curve(dnorm(x),add=TRUE)

# add a density expression
text(x=2, y=.35,
     expression(paste("f(x)=",frac(1,sqrt(2*pi)*sigma),
                exp(-frac(1,2*sigma^2)*((x-mu)^2)))))
```

Long title problem

We have the situation like this: we have “a lot” to write in the title line which make the graphics really ugly. But we can wrap it like this

```
plot(0,main=paste(strwrap(‘‘This is a really long title
       that I can not type it properly but we can use
       strwap function to wrap it’’, width=55)))
```

Legend in graphics

Adding a legend is a easy thing. You can define color, type, position etc yourself. Look at this example

```
x<-seq(0,1,.01)
par(lwd=2)
plot(x,dbeta(x,5,1),type="l", ylim=c(0,3),
     col="green", lty=1,xlab="",ylab="")
lines(x,dbeta(x,.5,.5), col="red",lty=2)
lines(x,dbeta(x,1,3),col="blue",lty=3)
lines(x,dbeta(x,2,2), col="pink",lty=4)
lines(x,dbeta(x,2,5), col="black",lty=5)
lines(x,dbeta(x,1,1),col="orange",lty=6)
legend("top",
       c("Beta(5,1)","Beta(.5,.5)","Beta(1,3)",
         "Beta(2,2)","Beta(2,5)","Beta(1,1)"),
       lty=c(1,2,3,4,5,6),
       col=c("green","red","blue","pink","black","orange"),
       ncol=2)
```

R graphics in Multivariate Analysis

The graphics in multivariate analysis is quite easy to use, most of which can be picked up easily.
The build-in data set “mtcars” is the data extracted from the 1974 “Motor Trend” US magazine, and comprises fuel consumption and 10 aspects of automobile design and performance for 32 automobiles (1973-74 models). We can make the Star graph as below.

```
palette(rainbow(12)) # 12 color rainbow
stars(mtcars[, 1:7], len = 0.8, key.loc = c(12, 1.5),
      main = "Motor Trend Cars", draw.segments = TRUE)
```

To make a faces graph, a package “aplpack” is needed.

```
library("aplpack")
faces(matrix(sample(1:1000,128,),16,8),main="random faces")
```

Time series plot

```
# prepare the monthly time series data
# (8 series), from the Jan, 1961
z <- ts(matrix(rt(200 * 8, df = 3), 200, 8),
        start = c(1961, 1), frequency = 12)
# we can plot some of them for a
# period (1961.1-1969.12) in multiple frames
z1 <- window(z[,1:3], end = c(1969,12))
plot(z1, type = "b")
# or we can plot them in a single frame
plot(z, plot.type="single", lty=1:3, col=4:2)
# the lag opeator plot
# we use the monthly mean relative sunspot
# numbers from 1749 to 1983 in "sunspots"
# x_{t-1} aginst x_t
plot(lag(sunspots, 1), sunspots, pch = ".")
# the acf and pacf
# use the data "Luteinizing Hormone in Blood Samples"

acf(lh)
pacf(lh)
```

Grey band for standard deviation

Some time we need to show the confidence interval for the model. "polygon" can handle this easily. Look at this example

```
# set the seed
set.seed(125)
x<-1:200
# simulate an AR(1) model
y<-cumsum(rnorm(200))

# calculate the 95% level confident interval
int1<-y-1.96
int2<-y+1.96

# plot the data
plot(y,xlab="t",ylab=expression(y[t]))

# show the band
polygon(c(x, rev(x)), c(int1, rev(int2)),
        col="grey",border=NA)
# plot the line again
lines(y,type="l",col="blue",lwd=2)
```

3D plot (non-interactive)

We can use “persp”to plot non-interactive 3 dimensional graph. Let’s take the unnormalized *sinc function* as example.

$$\text{sinc}\left(z\right) = \text{sinc}\left(\sqrt{x^2+y^2}\right) = \frac{\sin\left(\sqrt{x^2+y^2}\right)}{\sqrt{x^2+y^2}}$$

```
x <- seq(-10, 10, .3)
y <- x
rotsinc <- function(x,y)
{
    sinc <- function(x) { y <- sin(x)/x ; y[is.na(y)] <- 1; y }
    sinc( sqrt(xˆ2+yˆ2) )
}
z <- outer(x, y, rotsinc)
persp(x, y, z, theta = 30, phi = 30, expand = 0.5,
      col = "lightgreen", ltheta = 120, shade = 0.75,
      ticktype = "detailed",
      xlab = "X", ylab = "Y", zlab = "Z")
title(main = expression(z == Sinc(sqrt(xˆ2 + yˆ2))))
```

Project 3D data onto 2D with different colors

Recall the *sinc function* we have tried. We can plot them in a 2D picture with color by using the “image” function.

```
x<-seq(-10,10,.3)
y<-x
z<-outer(x, y, rotsinc) # the rotsinc is defined in section 2.6
image(x, y, z, col=rainbow(50,start=.7,end=.1))
filled.contour(x,y,z) # show the color legend
```

Contour plot

Here is another way to disply 3D data in a 2D graph

```
N <- 50
x <-y<- seq(-1, 1, length=N)
xx <- matrix(x, nrow=N, ncol=N)
yy<-t(xx)
z <- 1 / (1 + xx^2 + (yy + .5 * sin(5*yy))^2)
# here is the contour plot
contour(x, y, z, main = "Contour plot")
```

3D interactive plot

“rgl” package is a 3D visualization device system (OpenGL). Since it is a little complicate, I just show some demos. You can use “rgl.snapshot” to export screen shot. There are several other packages/drivers are under development.

Suggested Reading

- For a quick guide for graphics in R, please visit `http://cran.r-project.org/web/views/Graphics.html`
- For a comprehensive guide to R graphics, see
  - Murrell, Paul. *R graphics*. CRC Press, 2005.