Advanced Graphics

Lecture 6

Nicholas Christian

BIOST 2094 Spring 2011

Outline

`lattice` **package**

- The `lattice` package is used to produce trellis graphics
- Trellis graphics display multivariate data by plotting subsets of the data on adjacent panels. Often each panel is a plot of the data for a given level of a categorical variable.
- Lattice graphics are completely unrelated to base graphics. For example, changing `par()` settings usually has no effect on Lattice plots.
- High-level functions create the graph and panel functions are responsible for producing the actual display.
- Low-level functions used with base graphics, `title()`, `lines()`, etc, are not applicable with trellis graphics.

Example - `xyplot()`

```
library(lattice)

x <- rnorm(50)
y <- rnorm(50)
sex <- gl(2,25, labels=c("Male", "Female"))
trt <- rbinom(50, size=1, prob=.5)

# Basic trellis scatterplot
# Plot y versus x, given sex
xyplot(y ~ x | sex)

# Custom trellis scatterplot
xyplot(y ~ x | sex, xlab="X-Axis", ylab="Y-Axis", pch=19, col="blue", cex=1.5,
        main="Y vs X by Sex",
        strip=strip.custom(bg="gray75", par.strip.text=list(font=2)))

# Including a grouping variable within each panel (group) and add a legend (key)
xyplot(y ~ x | sex, group=trt, xlab="X-Axis", ylab="Y-Axis",
        pch=c(19,0), col=c("blue", "red"), cex=1.5,
        main="Y vs X by Sex and Treatment",
        key=list(text=list(c("Treatment 1", "Treatment 2")),
                 points=list(pch=c(19,0), col=c("blue", "red")), columns=2),
        strip=strip.custom(bg="gray75", par.strip.text=list(font=2)))
```

Example - `xyplot()`

```
# Plot data using a custom panel function
# panel function arguments:
#            x = x-coordinates
#            y = y-coordinates
#   subscripts = indices of x and y from the original data
#       groups = grouping variable
#
# Need to use panel.points, the base graphics function points() is not
# applicable for lattice graphics
# The layout argument is used to change the arrangment of the panels
xyplot(y ~ x | sex, group=trt,
        xlab="X-Axis", ylab="Y-Axis",
        main="Y vs X by Sex and Treatment",
        layout=c(1,2),
        panel=function(x, y, subscripts, groups) {
                 panel.points(x, y,
                   col=ifelse(sex[subscripts]=="Male", "blue", "palevioletred1"),
                   pch=ifelse(groups==1, 19, 0), cex=1.5)
               },
        key=list(text=list(c("Male", "Female", "Treatment 1", "Treatment 2"),
                          col=c("blue", "palevioletred1", "black", "black")),
                 points=list(pch=c(NA, NA, 19,0)), columns=2),
        strip=strip.custom(bg="gray75", par.strip.text=list(font=2)))
```

Example - `xyplot()`

```
# Similiar approach for line graphs, include a value for type
xyplot(y ~ x | sex, group=trt,
        type="b", pch=c(19,0), col=c("blue", "red"), cex=1.5,
        xlab="X-Axis", ylab = "Y-Axis",
        main="Y vs X by Sex and Treatment",
        key=list(text=list(c("Treatment 1", "Treatment 2")),
                 lines=list(pch=c(19,0), type="b", col=c("blue", "red")),
                 columns=2, divide=1),
        strip=strip.custom(bg="gray75", par.strip.text=list(font=2)))
```

Example - `bwplot()`

```
y <- rnorm(100)
age <- cut(sample(16:60, 100, replace=TRUE), breaks=c(0,20,30,40,50,60))
trt <- gl(4, 25, labels=paste("Treatment", 1:4))

# Basic trellis boxplot, boxplot of y for each age group by treatment
bwplot(y ~ trt, groups=age)

# Custom trellis boxplot
# pch="|", prints a horizontal bar at the median
# as.table=TRUE, prints the panels left to right, top to bottom
# par.settings is described below
bwplot(y ~ age | trt, xlab="Age", ylab="Response", pch="|",
        main="Response for each Age Group by Treatment",
        strip=strip.custom(bg="gray75", par.strip.text=list(font=2)),
        as.table=TRUE,
        par.settings=list(box.rectangle=list(col="blue"),
                          box.umbrella=list(lty=1, col="blue"),
                          plot.symbol=list(col="blue", pch=19)))
```

Example - `bwplot()`

```
# Show some of the current graphical parameter settings
show.settings()

# trellis.par.get() get the current value of a graphic parameter
# trellis.par.set() set graphical parameters for *ALL* graphs

# After looking at the documentation for panel.bwplot(), I found
# that these settings affect the appearance of the boxplot
# (alpha is between 0 and 1 and controls the transparency)
trellis.par.get("box.rectangle")  # Box
trellis.par.get("box.umbrella")   # Whisker
trellis.par.get("plot.symbol")    # Outlier

# Rather than change these settings globally, I changed them
# only for the current graph using the par.settings argument
```

`diagram` package

- The `diagram` package is used for creating flow charts as well as transition diagrams and web diagram.
- The vignette included with this package has several good examples, `vignette("diagram")`

Example - CONSORT Diagram

- Create a CONSORT diagram, flow chart that depicts the movement of subjects through a clinical trial, for a breast cancer study.

```
library(diagram)
par(mar=c(1,1,1,1))

# Create an empty plot, bounded by [0,1] without axes, labels, titles
openplotmat()

# Get coordinates for shapes, argument specifies the number of shapes in each 'row'
xypos <- coordinates(c(1,2,2,1,2), relsize=.8)

# Draw arrows first, so arrows don't overlap shapes
treearrow(from=xypos[1,], to=xypos[2:3,], line.pos=.5, arr.pos=1)
straightarrow(from=xypos[2,], to=xypos[7,], arr.pos=1)
straightarrow(from=xypos[3,], to=xypos[8,], arr.pos=1)
straightarrow(from=xypos[4,], to=xypos[4,]-c(.15,0), arr.pos=1)
straightarrow(from=xypos[5,], to=xypos[5,]+c(.15,0), arr.pos=1)
straightarrow(from=xypos[4,]-c(0,xypos[4,2]-xypos[6,2]), to=xypos[6,]-c(.36,0), arr.pos=1)

RADX = .05      # Horizontal radius of the box
RADY = .05      # Vertical radius of the box
HEIGHT = .065   # Height of text
```

Example - CONSORT Diagram

```
# Draw shapes
textrect(xypos[1,], RADX, RADY, shadow.size=0, lcol="blue", lwd=2)
textplain(xypos[1,], lab=c("Subjects", "n=1,067"), height=HEIGHT)

textrect(xypos[2,], RADX, RADY, shadow.size=0, lcol="blue", lwd=2)
textplain(xypos[2,], lab=c("SNAD", "n=528"), height=HEIGHT)

textrect(xypos[3,], RADX, RADY, shadow.size=0, lcol="blue", lwd=2)
textplain(xypos[3,], lab=c("SNR", "n=539"), height=HEIGHT)

textrect(xypos[4,]-c(.15,0), RADX+.05, RADY, shadow.size=0, lcol="blue", lwd=2)
textplain(xypos[4,]-c(.15,0), lab=c("SN Positive or not determined", "n=170"), height=HEIGHT)

textrect(xypos[5,]+c(.15,0), RADX+.05, RADY, shadow.size=0, lcol="blue", lwd=2)
textplain(xypos[5,]+c(.15,0), lab=c("SN Positive or not determined", "n=148"), height=HEIGHT)

textrect(xypos[6,]-c(.36,0), RADX+.06, RADY+.01, shadow.size=0, lcol="blue", lwd=2)
textplain(xypos[6,]-c(.36,0), lab=c("SN Negative", "with early consent withdrawl", "n=2"),
           height=HEIGHT+.01)

textrect(xypos[7,], RADX+.03, RADY+.03, shadow.size=0, lcol="blue", lwd=2)
textplain(xypos[7,], lab=c("Analyzed n=356", "Lumpectomy n=307", "Mastectomy n=49"))

textrect(xypos[8,], RADX+.03, RADY+.03, shadow.size=0, lcol="blue", lwd=2)
textplain(xypos[8,], lab=c("Analyzed n=391", "Lumpectomy n=320", "Mastectomy n=71"))

title("CONSORT Diagram", line=-1)
```

tcltk package

- The tcltk package provides tools for adding graphical user interface (GUI) elements to R
- Tcl/Tk stands for, Tool Control Language + (GUI) Took Kit and is its own language that has been embedded in R by this package.
- Can create widgets (GUI window) with check boxes, ratio buttons, pull-down menus etc, that can be used to interact with R .
- For example the function rotate.persp() creates a widget and the settings in the widget are passed to the appropriate arguments in persp().

```
library(TeachingDemos)

x <- y <- seq(-1, 1, len=25)
z <- outer(x, y, FUN=function(x,y) -x*y*exp(-x^2-y^2))
rotate.persp(x,y,z)
```

Example - Tcl/Tk

```
library(tcltk)

# Create a new toplevel widget
w <- tktoplevel()

# Add text to widget
label.widget <- tklabel(w, text="CLICK OK!")

# Setup an "Ok" button that prints "Hello World!" to the console
button.Ok <- tkbutton(w,text="Ok",command=function()cat("Hello World!\n"))

# Setup a "Cancel button that closes the widget
button.Cancel <- tkbutton(w,text="Cancel",command=function()tkdestroy(w))

# Add the text and buttons to the widget
tkpack(label.widget, button.Ok, button.Cancel)
```

animation package

- The animation package creates an animated figure by connecting a series of R graphs, just like a cartoon.
- These animations can be displayed in the R console. The package will also generate code for including animations in web pages and LaTeX documents.
- The package contains several functions that use animation to demonstrate several different math and statistic topics.
- The developer, Yihui Xie, has a website with good examples, http://animation.yihui.name/animation:start
- Use the function Sys.sleep() and loops to create animations within R, without using the animation package.

Example - Weak Law of Large Numbers

```
wlln <- function(max.n=150) {
   par(mfrow=c(2,2), oma=c(0,0,2,0), mar=c(5.1,4.1,1.75,1))
   sample.norm <-  sample.exp <- array(dim=max.n)
   for(i in 1:max.n) {
       sample.norm[i] <- mean(rnorm(i, mean=0))
       sample.exp[i] <- mean(rexp(i, rate=1/3))

       par(mfrow=c(2,2), oma=c(0,0,3,0), mar=c(5.1,4.1,1,1))

       x <- seq(-3, 3, len=100)
       plot(x, dnorm(x), type="l", axes=F, ann=F, main="Normal Population with mean 0")
       polygon(x, dnorm(x), col="blue")
       axis(1)

       plot(sample.norm[1:i], type="o", xlab="Sample Size", ylab="Sample Mean", pch=19)
       abline(h=0, col="green3")

       x <- seq(0, 20, len=100)
       plot(x, dexp(x, rate=1/3),type="l",axes=F,ann=F,main="Skewed Population with mean 3")
       polygon(c(x, 0), c(dexp(x, rate=1/3), 0), col="blue")
       axis(1)

       plot(sample.exp[1:i], type="o", xlab="Sample Size", ylab="Sample Mean", pch=19)
       abline(h=3, col="green3")

       title(paste("Sample Size =", i), outer=TRUE)

       # Brief pause between plots
       #Sys.sleep(0.08)
   }
}
```

Example - `animation` package

```
library(animation)

# Create a latex file that includes the WLLN animation
saveLatex(wlln(max.n=100), img.name = "WLLN",
    latex.filename="weaklawlargenum.tex", outdir=getwd(),
    interval = 0.1, nmax = 100, ani.dev = "pdf", ani.type = "pdf",
    ani.width = 7, ani.height = 7,
    ani.opts = "controls,width=0.95\textwidth",
    documentclass = paste("\documentclassarticle",
        "\usepackage[papersize=7in,7in,margin=0.3in]geometry", sep = "\n"))

# Create an HTML file that includes the WLLN animation
saveHTML(wlln(max.n=100), img.name = "WLLN", htmlfile = "WLLN.html",
    outdir=getwd(), interval = 0.1, nmax = 100, ani.dev = "png",
    ani.type = "png", ani.width = 700, loop=FALSE, verbose=FALSE)

# Run some illustrations included with the package
help(package=animation)   # Get information on the packages contents
newton.method()
conf.int()
```