# Advanced Graphics

Leo Schalkwyk

SGDP, IoP

SGDP Summer School

# Outline

1. R Graphics Concepts
2. Modifying High-level Graphics
3. Images and drawings

# Coordinates

- high level functions set up a plotting area
  - slightly bigger than the range of the data by default
- around that is a margin where scale values and axis labels go
- around that is an outer margin (normally the actual margin)
- this is designed for convenience
  - coordinates of plot area are in data units
  - 'outward' coordinates or margin are in 'lines of text'
- high-level plot functions call `plot.new()`
  - this (optionally) wipes the plot device
- next they call `plot.window()`
  - this sets up the plot area coordinates

# Margins and Axes

- defaults for ranges margins and axes usually produce something reasonable
- for journals you often have to make labels really big
- for custom plots you may want something different
  - margins small or nonexistent
  - axes modified or nonexistent

# Parameters

- many plot parameters are used by plotting functions
- most plot functions use global default parameters
- these are passed to low-level plotting functions via `...`
- allows a consistent way to modify plotting behavior
- `lattice` and `ggplot` offer variant flavours
- `par()` gets and sets plotting parameters

# Devices

- plotting 'devices' are an additional abstraction
  - saves you having to think about pixels or dots
- think of it as a 'save as' mechanism
- similar to the way ghostscript, gnuplot and tex work
- plotting instructions are device independent
  - sent to a device to produce visible plot
- `x11()`, `windows()`, `quartz()`
  - show plot in a window on display (with options to save)
- `pdf`, `postscript`, `xfig`, `svg`
  - vector graphic file devices
- `png`, `jpeg`, `bmp`, `tiff`
  - bitmap file devices

# Colours

- colours for lines or points can be specified by number
  - default palette: black, red, green3, blue, cyan, magenta, yellow, gray
  - useful trick: `col=factor`
  - large list of named colours available (like x11)
- colours are specified in RGB space using three two digit hex numbers
  - *#FF0000* is red *#00FF00* green
  - optionally there's a fourth value for transparency
- several functions generate colour series
  - `heat.colors()`
  - `topo.colors()`
- `RColorBrewer` package provides nice colour sets

# Points and Labels

- points can also be specified by number, 18 different symbols
- letters can also be used but are usually ugly
- the function `text()` can put text anywhere you like
- `mtext()` puts text in the margins
- `legend(locator(1),...)` puts a legend wherever you click
- the plotting function `symbols()` for plotting circles or boxes
- `identify()` labels points you click on

# Overlaying Plots

- some plotting functions add to an existing plot
  - `lines()`, `points()`, `symbols()`, `text()`
  - `lines()` can be used to plot histograms or density plots
- some functions have an argument like `add=TRUE`
- you can force almost any plot to overlay with `par(new=TRUE)`
  - default is `par(new=FALSE)` (wiping the device first)
- plots often need modification to work as overlays
  - titles, axis labels, axes need to be suppressed on at least one
  - `axis()` can ten be used to add axis on opposite side
  - `par(fig=c(x,x,y,y))` gives you a new coordinate system

# Multiplot Layouts

- there are multiple ways to get multiple plots on one device
- simple option: `par(mfrow=c(5,5))` grid of 25 plots
  - often requires setting small margins
  - useful for surveying data
  - default plot for dataframes: eg `plot(DNase)`
- `layout()` allows arbitrary sizes and plotting order
- the `lattice` package is an alternative plotting universe
  - powerful multivariate plotting functions
  - pre-tuned to a distinctive graphical style
  - many of its own plotting functions

# Images

- an image is just one or more matrices of intensities
- `image()` plots matrices with colours of choice
  - looking for patterns (eg of missing data) in large tables
  - heatmaps
- the usual imaging convention is transposed and flipped compared to a matrix
  - `image(t(matrix(1:100,10)[10:1,]),col=rainbow(9))`
  - `image(t(volcano)[ncol(volcano):1,])`
  - the intensity convention is usually 0(darkest) to 1 (white)

# Drawing from scratch

- coordinates recall Oliver's sour faces exercise
  - set up a coordinate system
  - build up a picture from any plotting elements
- this is very useful for diagrams such as maps
- convenient coordinates in relevant units
- most plotting functions are vectorized
- `locator()` can be used to choose locations