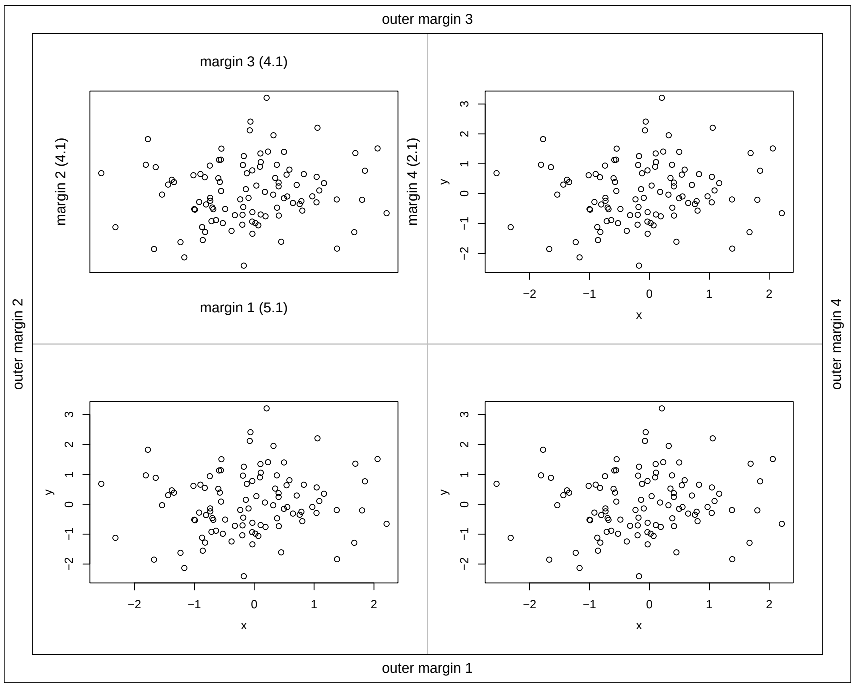
Tips for laying out plots in R

Steve Juggins
School of Geography, Politics and Sociology
University of Newcastle, UK
Stephen.Juggins@ncl.ac.uk
August 2005

1 Multiple plots on a page

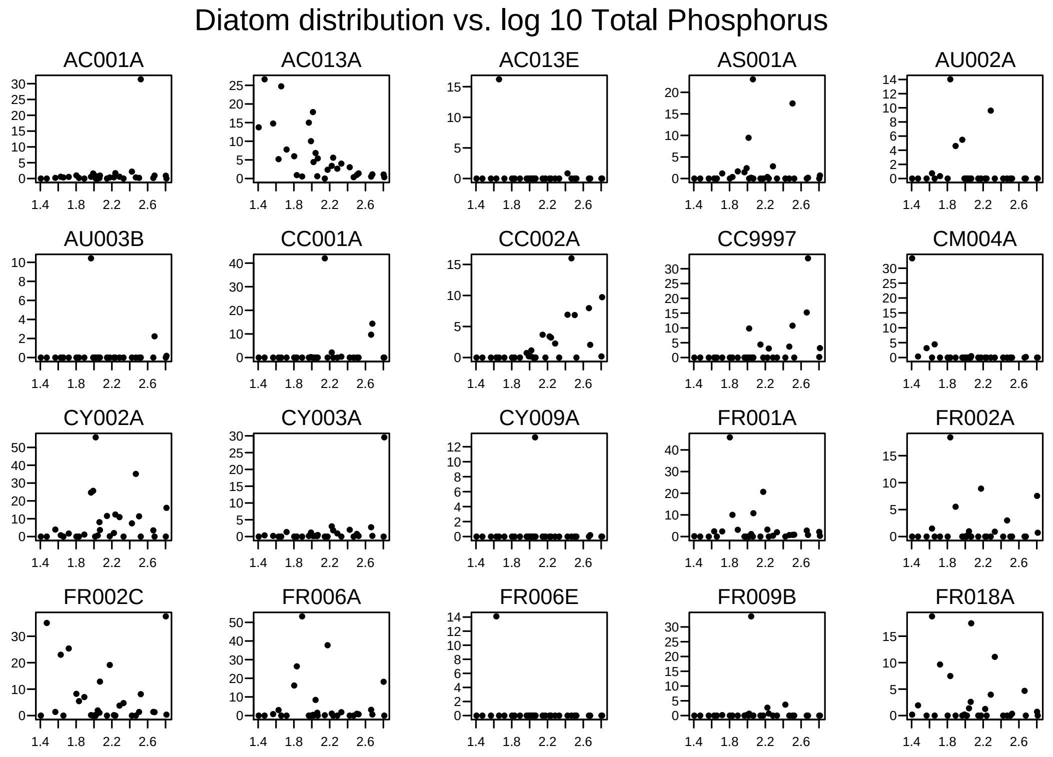
R contains a rich set of graphical parameters that can be used to customize the style of individual figures (here I follow R terminology and use *figure* to refer to an individual plot and *graph* to refer to the complete diagram, which may contain multiple figures). These graphical parameters are set to default values that will provide a well-proportioned plot when creating a single figure diagram. When working with multiple figure layouts some parameters, such as the axis fonts and plot margins, are not always approppariately rescaled, leading to plots iwth oversized margins and disproportiately small plotting areas. This section discusses some simple solutions.

`par(mfrow=)` provides a convenient way to group multiple figures or plots into a single graph. The graph below shows a 2 * 2 matrix of figures created using `par(mfrow=c(2,2))`, with the margin areas labelled. The figure and outer margins can be set using the `mar` and `oma` arguments to `par` respectively. Each argument expects a vector of four elements giving the bottom, left, top and right margins in text lines (corresponding to side=1, 2, 3 or 4). The default values for `mar` are shown in brackets. The outer margins are set to zero by default. Another useful argument to `par` is `mpg`. This accepts a vector of 3 elements specifying the distance in text lines between the figure (data area) margin and the axis title, axis labels and axis line respectively, with a default of `c(3, 1, 0)`.



Getting the right values for the figure margins is usually trial and error. In the graph below I create an outer margin at the top of the graph using `par(oma=c(0,0,2,0))` and write a title there using mtext with the argument outer=TRUE. Note that we save the current par settings in op and restore them after the plot.

```
> op <- par(no.readonly = TRUE)
> par(mfrow = c(4, 5))
> par(mar = c(2.5, 3.5, 1, 0.5))
> par(mgp = c(1.5, 0.5, 0))
> par(oma = c(0, 0, 3, 0))
> for (i in 1:20) {
+     plot(diat.max[, i] ~ env[, "TP"], ann = F, cex = 0.6, cex.axis = 0.8,
+         tcl = -0.4, las = 1, pch = 19)
+     mtext(colnames(diat.max)[i], side = 3, line = 0.2, cex = 0.8)
+ }
> mtext("Diatom distribution vs. log 10 Total Phosphorus", outer = TRUE,
+     side = 3, cex = 1.2, line = 1)
> par(op)
```



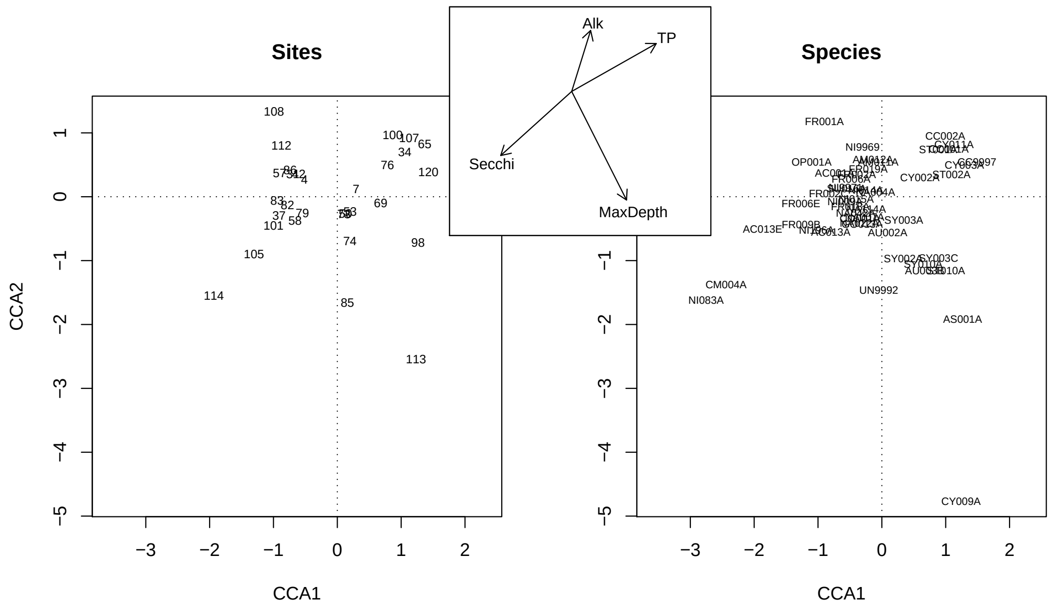
Comments: we first save the current graphics parameters in op, then use par(mar=c(2.5, 3.5, 1, .5)) to reduce the margins around the figure, par(mgp=c(1.5, .5, 0)) to reduce the spacing between the figure plotting region and the axis labels, and par(oma=c(0, 0, 3, 0)) to add an outer margin to the top of the graph. Within the plot command we use ann=F to turn off the axis labels, cex.axis=.8 to reduce the size of the axis values, tcl=-0.4 to reduce the size of the tick marks and las=1 to turn the y-axis labels perpendicular to the axis. We then use mtext rather than title to add a title to the plot because it is easier to place it closer to the top axis using this function. Finally we restore the original graphics settings.

2 More complex layouts

There are at least three ways of laying out more complex arrangements of plots. split.screen and layout both allow you to split the screen into arbitary sized sections that can each contain a figure. A more flexible approach that also allows you to overlay plots is to use par(fig=) to specify the exact position a figure will occupy. fig takes a vector of four elements to specify the left, right, bottom and top coordinates of the figure on a plotting region that spans from 0 to 1.

For example, `fig(0, .5, 0, 0.5)` will place a figure in the bottom left quarter of the plotting region. The use of `fig` is illustrated below by plotting a canononical correspondence analysis (CCA) ordination diagram. The CCA was produced using Jari Oksanen's `vegan` package. CCA diagrams often overlay plots sites, taxa and environmental variable in the same figure but this can lead to a cluttered plot. Here I plot sites and species in identically scaled but separate figures and overlay the environmental variables in a plot that covers the empty space between them.

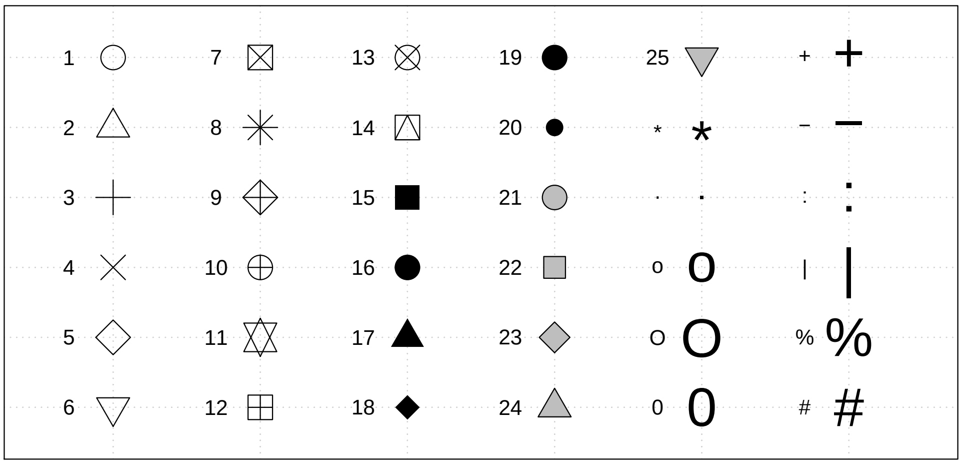
```
> op <- par(no.readonly = TRUE)
> par(fig = c(0, 0.5, 0, 0.95))
> plot(diat.cca, type = "none", scaling = 3)
> title("Sites")
> text(diat.cca, display = "wa", scaling = 3, cex = 0.7)
> par(fig = c(0.5, 1, 0, 0.95), new = TRUE)
> plot(diat.cca, type = "none", scaling = 3, ylab = "")
> title("Species")
> text(diat.cca, display = "species", scaling = 3, cex = 0.6)
> par(fig = c(0.41, 0.65, 0.6, 0.95), new = TRUE)
> par(mar = c(0, 0, 0, 0))
> bp <- scores(diat.cca, display = "bp", scaling = 3)
> plot(bp, type = "n", axes = FALSE, asp = 1)
> u <- par("usr")
> rect(u[1], u[3], u[2], u[4], col = "white")
> arrows(0, 0, bp[, 1] * 0.8, bp[, 2] * 0.8, length = 0.1)
> text(bp[, 1] * 0.9, bp[, 2] * 0.9, cex = 0.8, labels = rownames(bp),
+     xpd = NA)
> box()
> par(op)
```



Comments: we first save the current graphics parameters in op, then use `fig=` to set the position of the plotting area to the left half of the device and plot the first figure. We then do the same to plot the second figure on the right hand side of the device. We then adjust the positions passed to the third `fig` to specify a region in the top centre of the device, and since we don't want axis labels or values on this figure we reduce the margins to 0. We then use `plot` with `type="n"` to set up the plotting area but not draw anything. The call to `par("usr")` extracts the x-y data limits of the figure and these are used to plot a white-filled rectange over the region to obscure the underlying area. The arrows, text and frame are then added. Finally we restore the original graphics settings.

3 Plotting symbols

I can never remember the plotting symbols so here they are (courtesy of `example(points)` in the R help). Note that symbols 21-25 have their border colour specified by `col=` and fill colour specified by `bg=`.


1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
*
.
o
O
0
+
-
:
|
%
#