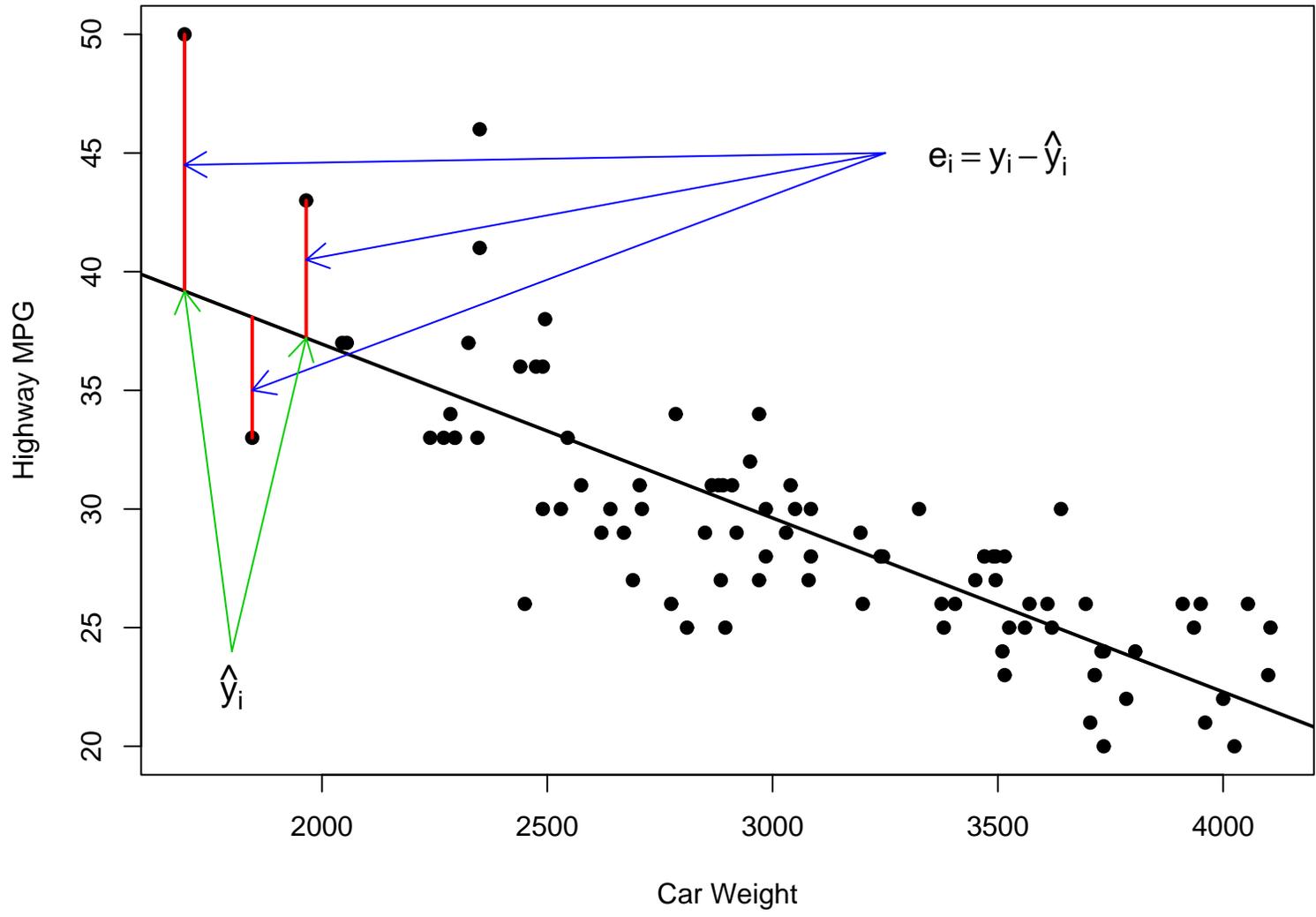


Graphics - Part III: Basic Graphics Continued

Statistics 135

Autumn 2005





```
carfit <- lsfit(Weight, HighMPG)
weights <- c(1695, 1845, 1965)
xmat <- matrix(c(1,1,1,weights), byrow=F, ncol=2)
carfitted <- xmat%*%matrix(carfit$coef, ncol=1)
plot(HighMPG ~ Weight, cars93, pch=16, xlab="Car Weight",
     ylab="Highway MPG")
abline(carfit, lwd=2)
segments(weights,obs,weights,carfitted,col=2, lwd=2)
arrows(3250,45,1695,44.5,col=4,length=0.15)
arrows(3250,45,1965,40.5,col=4,length=0.15)
arrows(3250,45,1845,35,col=4,length=0.15)
arrows(1800,24,1695,carfitted[1],col=3,length=0.15)
arrows(1800,24,1965,carfitted[3],col=3,length=0.15)
text(3500,45,expression(e[i] == y[i] - hat(y)[i]),cex=1.25)
text(1800,22.5,expression(hat(y)[i]),cex=1.25)
```

Plot Options

Many plotting options can be set within the plotting functions. Examples seen so far have been

- `xlab`, `ylab`: Axis labels
- `main`: Plot title
- `xlim`, `ylim`: Axis limits
- `pch`: Plotting symbols (see `help(points)` for more details)

Now these options are from high-level plotting functions. Many are available for all of these functions (like `main`). Others may be more command specific (`pch` which only is appropriate for `plot` - though used in some low-level functions as well).

There are a wide range of other plotting options, while available to the functions discussed so far, can be used in other ways and are not specific to certain functions. These additions options are part of the `par` function.

There are two types of options that can be set via `par`.

- Global plot features

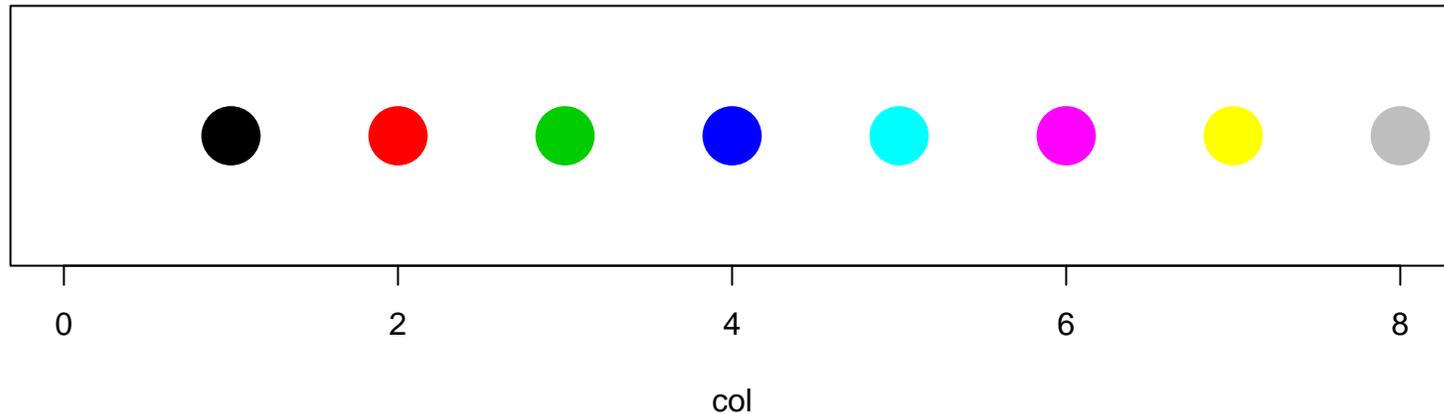
- Plot layout - `mfrow` or `mfcol` (sets number of rows and columns)
- Margins - `mar` or `mai`

- Item specific features

- Colour - `col`, `fg`, or `bg`

There are three way to specify colour in **R**, by a code (from 1:8), by name (see the functions `color` or `colours` for possibility), or by RGB code, given as a hex code string of the form "RRGGBB"

Colour Codes



```
par(mfrow=c(1,1), pty="m", mar=c(4,1,4,1)+0.1)
plot(0:8, rep(0,9), col=0:8, pch=16, cex=4, yaxt="n",
     xlab="col", ylab="", main="Colour Codes")
```

With these codes, 0 is the background colour, in the above case, white.

Colour by RGB

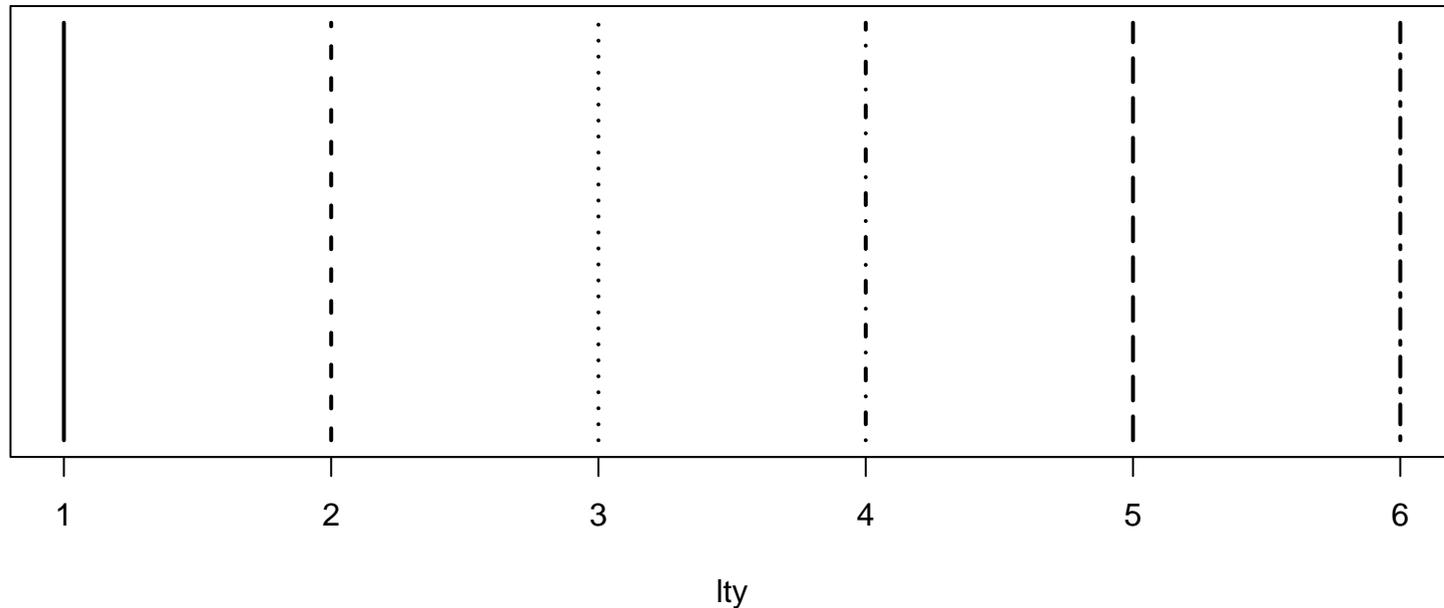
RR	CE	00	DD	88
GG	00	88	CC	88
BB	8A	44	00	88
				
	#CE008A	#008844	#DDCC00	#888888

```
plot(1:4, rep(0,4), pch=16, cex=4, xaxt="n", yaxt="n",
     xlim=c(0.5,4.25), ylim=c(-0.5,1), xlab="", ylab="",
     main="Colour by RGB",
     col=c("#CE008A", "#008844", "#DDCC00", "#888888"))
```

(The text commands are omitted in the above code.)

– Line types - lty

Line Codes



```
par(mfrow=c(1,1), pty="m", mar=c(4,1,4,1)+0.1)
plot(c(1,6), 0:1, type="n", yaxt="n", ylab="", xlab="lty",
     main="Line Codes")
segments(1:6,rep(0,6), 1:6, rep(1,6), lty=1:6, lwd=2)
```

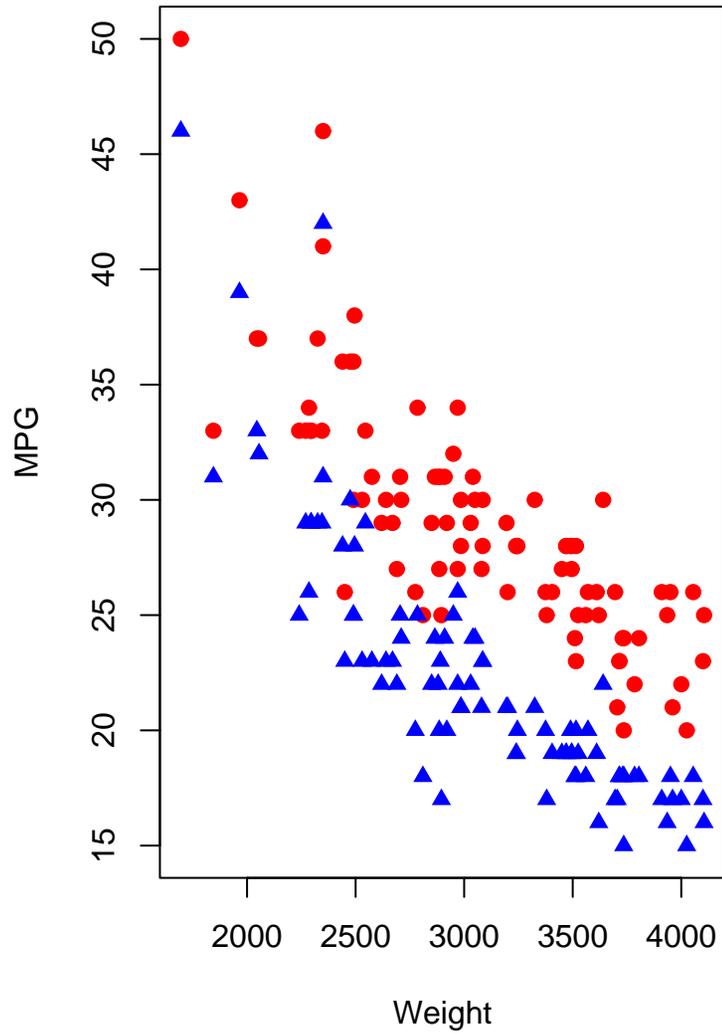
- Line widths - `lwd` (A multiplier of the default line width)
- Fonts - `cex` (sets font size), `font` (sets type face), or `family` (font family). Note that `cex` (and other `par` parameters) has methods for `axis`, `lab`, `main`, and `sub` for more specific adjustment of features. It can be used to adjust the plot symbol size, as done in earlier examples.
- Axes - `xaxp` (tick marks), `xaxs` (axis style), `xaxt` ("n": no axis printed, "s": standard axis), `xlog` (plot axis on log scale). Similar for y axis.

The global plot features must be set before plotting by the `par` command.

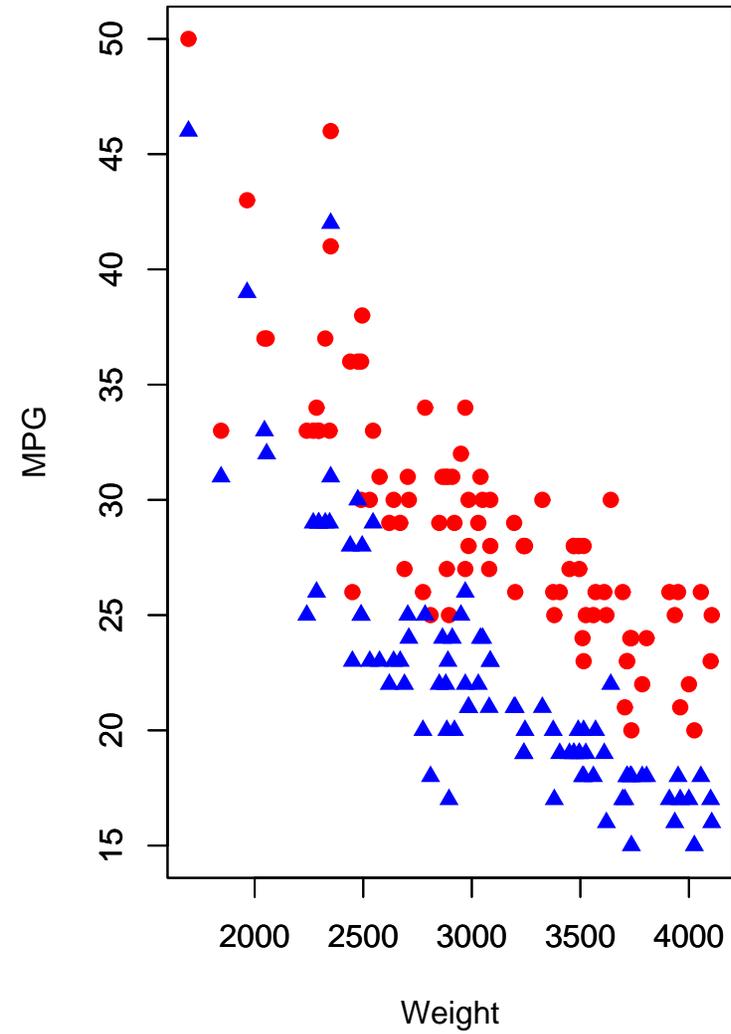
The item specific features may be set within a plotting function, or can be set by the `par` function to reset a default.

For example,

One plot command



Two plot commands

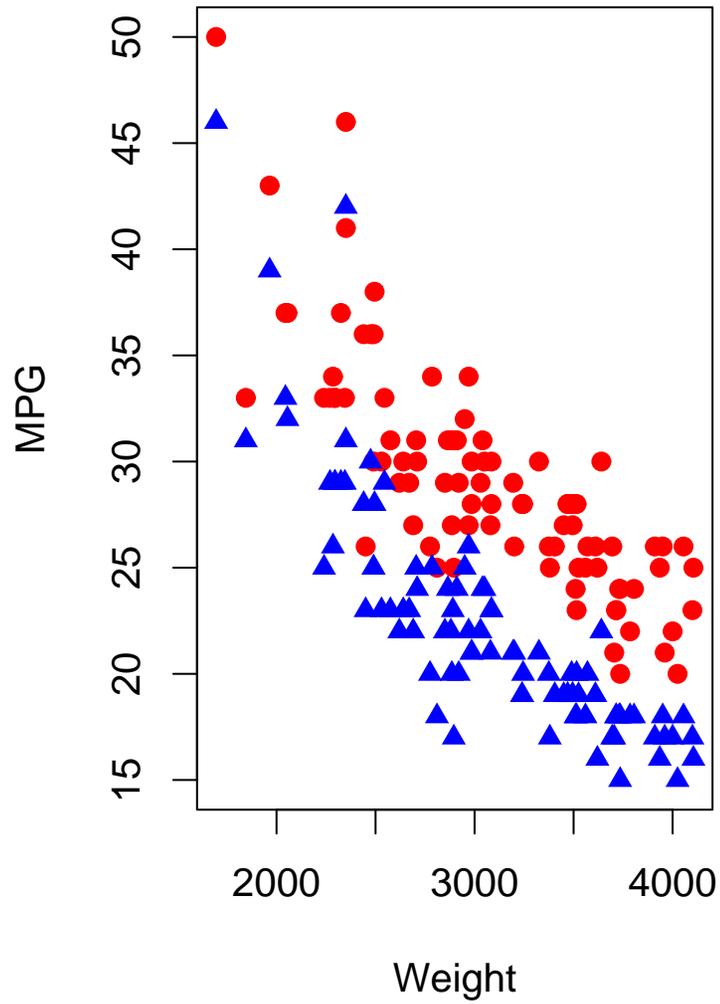


```
# Set global properties of plot
par(mfrow=c(1,2), mar=c(4,4,3,1)+0.1)

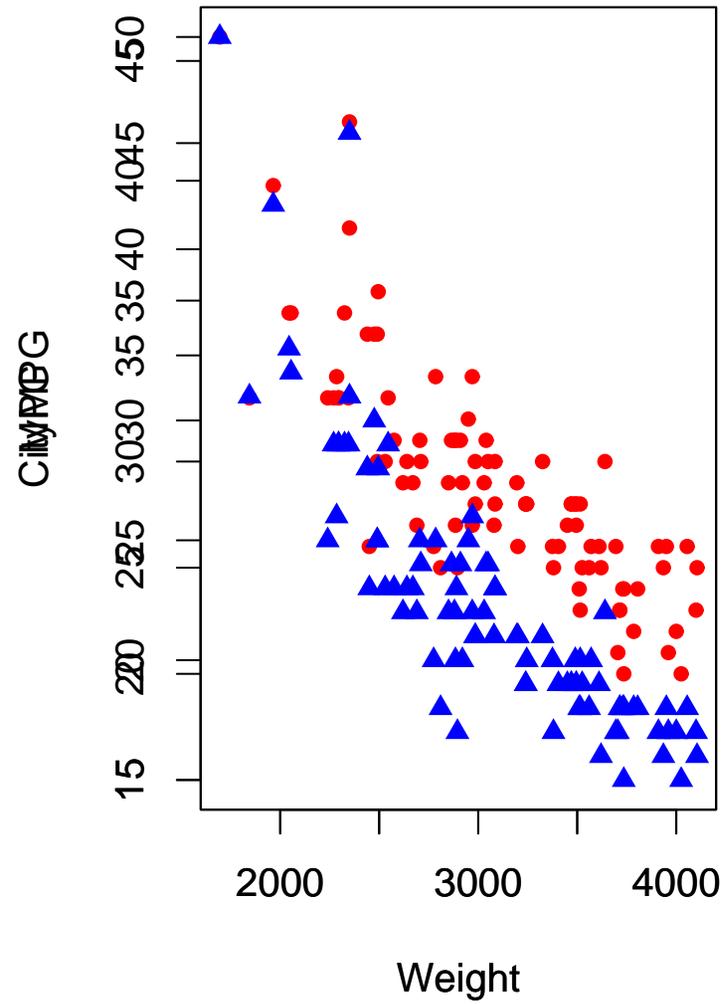
# Sets default plot region type - "m": maximum, "s": square
par(pty="m")

# Create plot
plot(Weight,HighMPG, pch=16, col=2, xlab="Weight", ylab="MPG",
     main="One plot command", ylim=c(15,50))
points(Weight,CityMPG, pch=17, col=4) # col="blue"
plot(Weight,HighMPG, pch=16, col=2, xlab="Weight", ylab="MPG",
     main="Two plot commands", ylim=c(15,50))
par(new=T) plot(Weight,CityMPG, pch=17, col=4, xlab="", ylab="",
               main="", ylim=c(15,50))
```

One plot command



Two plot commands – messed



```
par(mfrow=c(1,2), pty="m", mar=c(4,4,3,1)+0.1)
par(cex=1.25) # increase text and symbols by 25%

plot(Weight,HighMPG, pch=16, col=2, xlab="Weight", ylab="MPG",
     main="One plot command", ylim=c(15,50))
points(Weight,CityMPG, pch=17, col=4)
plot(Weight,HighMPG, pch=16, col=2, xlab="Weight", ylab="MPG",
     main="Two plot commands - messed up", ylim=c(15,50), cex=0.75)
par(new=T)
plot(Weight,CityMPG, pch=17, col=4)
```

To see what all the par settings are give the par(). Specific setting can also be observed by passing the desired option to the function.

```
> par("cex", "mfrow")
```

```
$cex [1] 1
```

```
$mfrow [1] 1 2
```

```
> par()
```

```
$xlog [1] FALSE
```

```
$ylog [1] FALSE
```

```
$adj [1] 0.5
```

```
$ann [1] TRUE
```

```
$ask [1] FALSE
```

blah, blah, blah finally getting to the end

```
$xaxt [1] "s"
```

```
$xpd [1] FALSE
```

```
$yaxp [1] 15 50 7
```

```
$yaxs [1] "r"
```

```
$yaxt [1] "s"
```

Graphic Devices

There are a number of different formats available for graphics. If you are just wanting to see graphs on your screen, you usually don't need to worry about them (unless you are in unix/linux setup under Xwindows). Also some of these are platform specific, such as `windows` and `win.metafile`. To see what is available for your platform, see `help(Devices)`.

- `windows`: The graphics driver for Windows (on screen, to printer and to Windows metafile).
- `win.metafile`: Save in a file in Windows metafile format.
- `win.printer`: Sends to Windows printer.
- `x11`: The graphics device for Xwindows (on screen)
- `postscript`: Writes Encapsulated PostScript graphics commands to a file

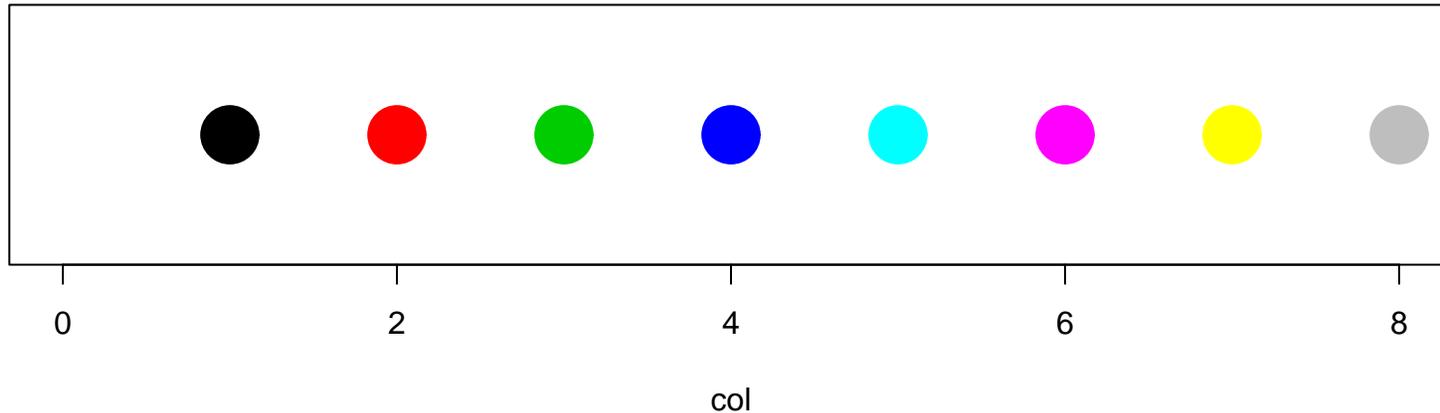
- `jpeg`: JPEG bitmap device
- `pdf`: Write PDF graphics commands to a file
- `pictex`: Writes LaTeX/PicTeX graphics commands to a file
- `png`: PNG bitmap device
- `bmp`: BMP bitmap device
- `xfig`: Device for XFIG graphics file format
- `bitmap`: bitmap pseudo-device via 'GhostScript' (if available).

In some cases, you can create the graphics file directly from the on screen image. Formats available can vary by system. For example, in the Windows' version, you can create metafiles and postscript files from the graphics window. It is also possible to cut and paste a graphics file in Windows. In the Mac version, I can't get the graphics Window to be saved in any format.

Even though you can sometimes save what you want from the graphics window, it is usually better to write out the graphic into the desired format directly. For example, all the graphics included in these overheads are postscript files with specific characteristics.

For example, an earlier plot was created with the commands:

Colour Codes



```
postscript('../colourcode.eps', horiz=F, width=9, height=3)
par(mfrow=c(1,1), pty="m", mar=c(4,1,4,1)+0.1) plot(0:8, rep(0,9),
col=0:8, pch=16, cex=4, yaxt="n",
  xlab="col", ylab="", main="Colour Codes")
dev.off()
```

The `postscript` command describes how the file is to be saved and the `dev.off` command says that the graphic is completed and to save the file.

The complete set of options for the `postscript` function are

```
postscript(file = ifelse(onefile, "Rplots.ps", "Rplot%03d.ps"),
           onefile = TRUE,
           paper, family, encoding, bg, fg,
           width, height, horizontal, pointsize,
           pagecentre, print.it, command,
           title = "R Graphics Output", fonts = NULL)
```

Arguments:

`file`: a character string giving the name of the file. If it is `''`, the output is piped to the command given by the argument `'command'`.

For use with `'onefile=FALSE'` give a `'printf'` format such as `'Rplot%03d.ps'` (the default in that case).

`paper`: the size of paper in the printer. The choices are `'a4'`, `'letter'`, `'legal'` and `'executive'`

(and these can be capitalized).

`horizontal`: the orientation of the printed image, a logical. Defaults to true, that is landscape orientation on paper sizes with width less than height.

`width`, `height`: the width and height of the graphics region in inches. The default is to use the entire page less a 0.25 inch border on each side.

`pointsize`: the default point size to be used.

If you want to want to include graphics into a Word document (on Windows), use the `win.metafile`.

```
win.metafile(filename = "", width = 7, height = 7, pointsize = 12)
```

The options work similarly to `postscript`.

If you want to check out exactly what a graphic would look like on the screen under a specified format, use the `win.graph` function (under Windows).

```
win.graph(width = 7, height = 7, pointsize = 12)
```

The other graphics commands work similarly. See their help pages for more info.