Graphical parameters

(The R code used for this website is here)

The basic graphics in R is very flexible in what all can be modified. The graphical parameters can be modified on several levels:

1. by calling the function `par` before the figure is actually plotted;
2. inside the high- or low-level graphical function;
3. by wrapping parameters into the list together with the argument of the plotting function.

If you call the function `par ()` (with emtpy parenthesis), it will return the list of all graphical parameters with their default setting:

```
par ()
```

```
$xlog
[1] FALSE

$ylog
[1] FALSE

$adj
[1] 0.5

$ann
[1] TRUE

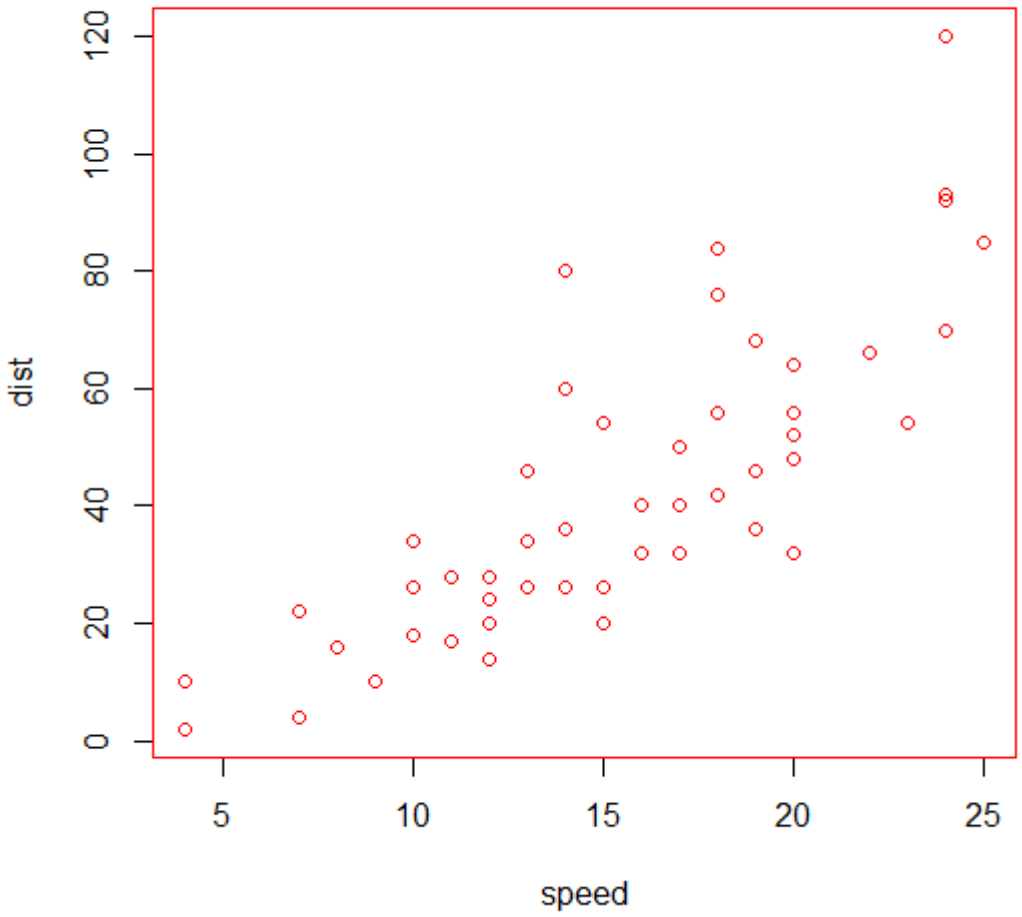
$ask
[1] FALSE
...
```

The list of all parameters is here, details about what the parameters mean can be found in `?par` (link here)

1. Using par () function

Most of the parameters can be changed inside the `par` function. The difference from changing these parameters e.g. in the `plot` function is that the change is persistent - it will apply to all figures plotted in that graphical device, unless you change the value again or close the device. The function `par` needs to be called before the figure is actually plotted. For example, if you want to change the colour in which items in the figure are plotted, you can do it by calling `par` function with given argument, and then by plotting the figure:

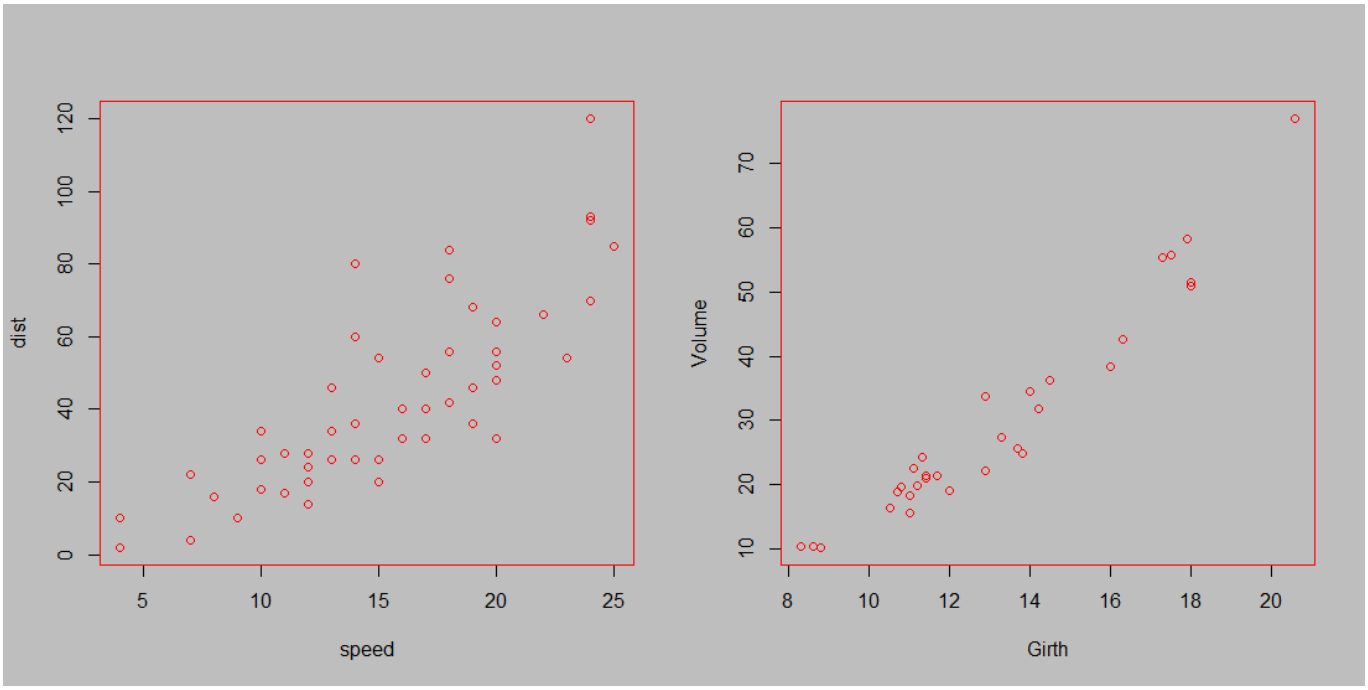
```
par (col = 'red')
plot (dist ~ speed, data = cars)
```



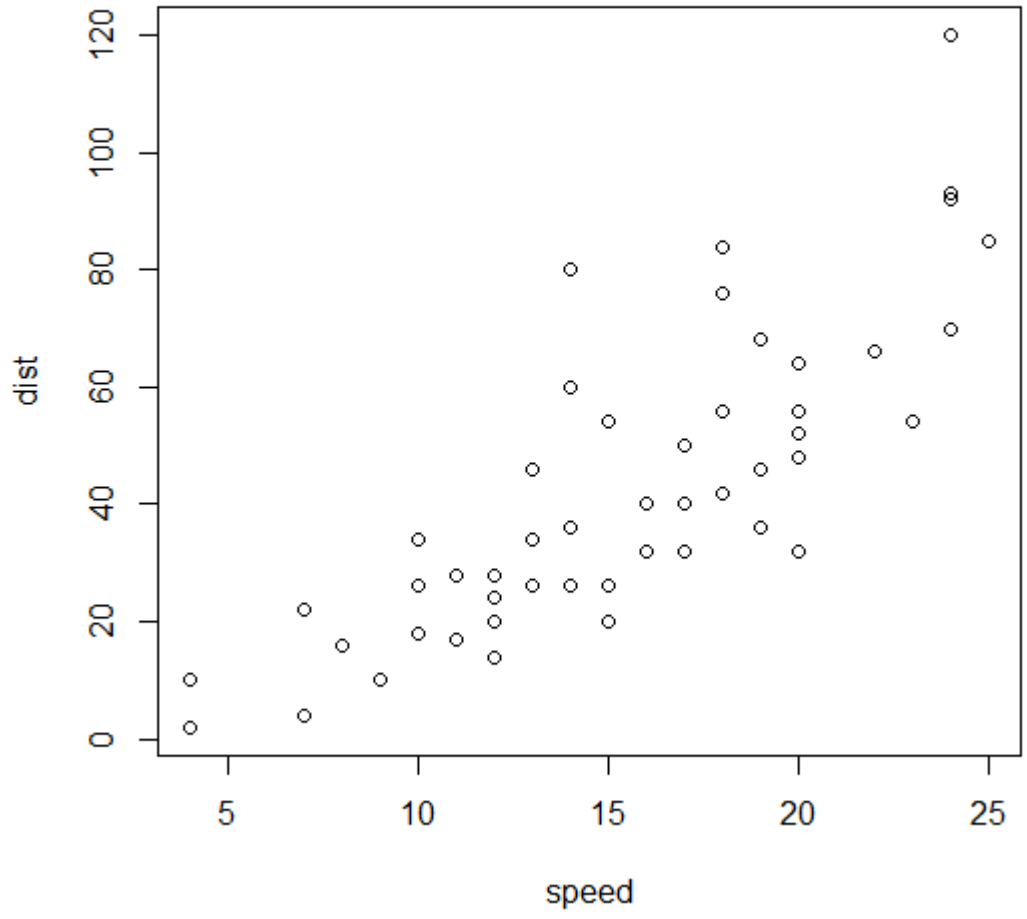
When the plotting function draws the figure, it will check the current setting of the graphical parameters to know which values to use; if we change it (as in this case for the parameters `col`), it will use this value to plot the figure.

One way to make sure that you can set the parameters to the state before change is to store the initial value when you change them:

```
old_par <- par (mfrow = c(1,2), col = 'red', bg = 'grey')
plot (dist ~ speed, cars)
plot (Volume ~ Girth, trees)
```




```
par (old_par)
plot (dist ~ speed, cars)
```



Parameters which can be modified **only** by calling the `par` function are listed in the following table (from *R*

Graphics by P. Murrel):

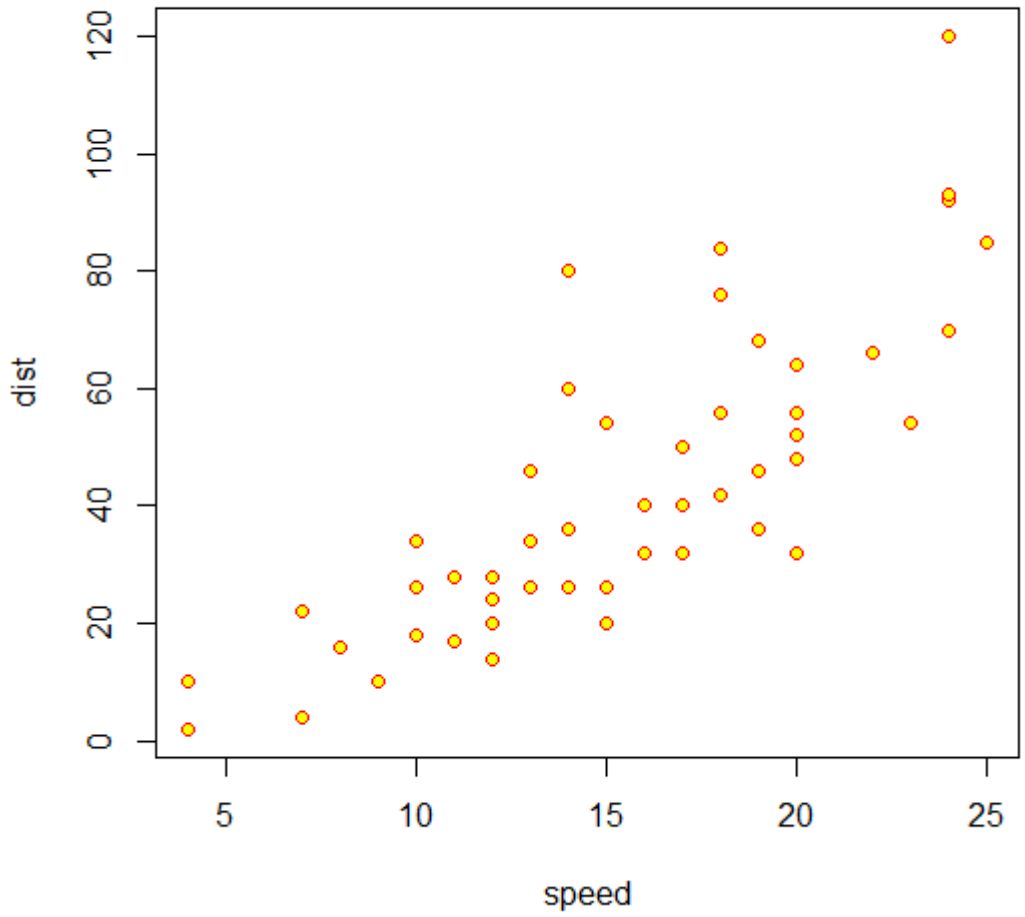
Table 3.2
Low-level traditional graphics state settings. This set of graphics state settings can be queried and set via the `par()` function. Each setting is described in more detail in the relevant **Section**.

Setting	Description	Section
`ask`	prompt user before new page?	3.2.8
`family`	font family for text	3.2.3
`fig`	location of figure region (normalized)	3.2.6
`fin`	size of figure region (inches)	3.2.6
`lend`	line end style	3.2.2
`lheight`	line spacing (multiplier)	3.2.3
`ljoin`	line join style	3.2.2
`lmitre`	line mitre limit	3.2.2
`mai`	size of figure margins (inches)	3.2.6
`mar`	size of figure margins (lines of text)	3.2.6
`mex`	line spacing in margins	3.2.6
`mfcol`	number of figures on a page	3.3.1
`mfg`	which figure is used next	3.3.1
`mfrow`	number of figures on a page	3.3.1
`new`	has a new plot been started?	3.2.8
`oma`	size of outer margins (lines of text)	3.2.6
`omd`	location of inner region (normalized)	3.2.6
`omi`	size of outer margins (inches)	3.2.6
`pin`	size of plot region (inches)	3.2.6
`plt`	location of plot region (normalized)	3.2.6
`ps`	size of text (points)	3.2.3
`pty`	aspect ratio of plot region	3.2.6
`usr`	range of scales on axes	3.4.7
`xlog`	logarithmic scale on x-axis?	3.2.5
`ylog`	logarithmic scale on y-axis?	3.2.5

2. Modify the argument inside the high- or low-level graphical function

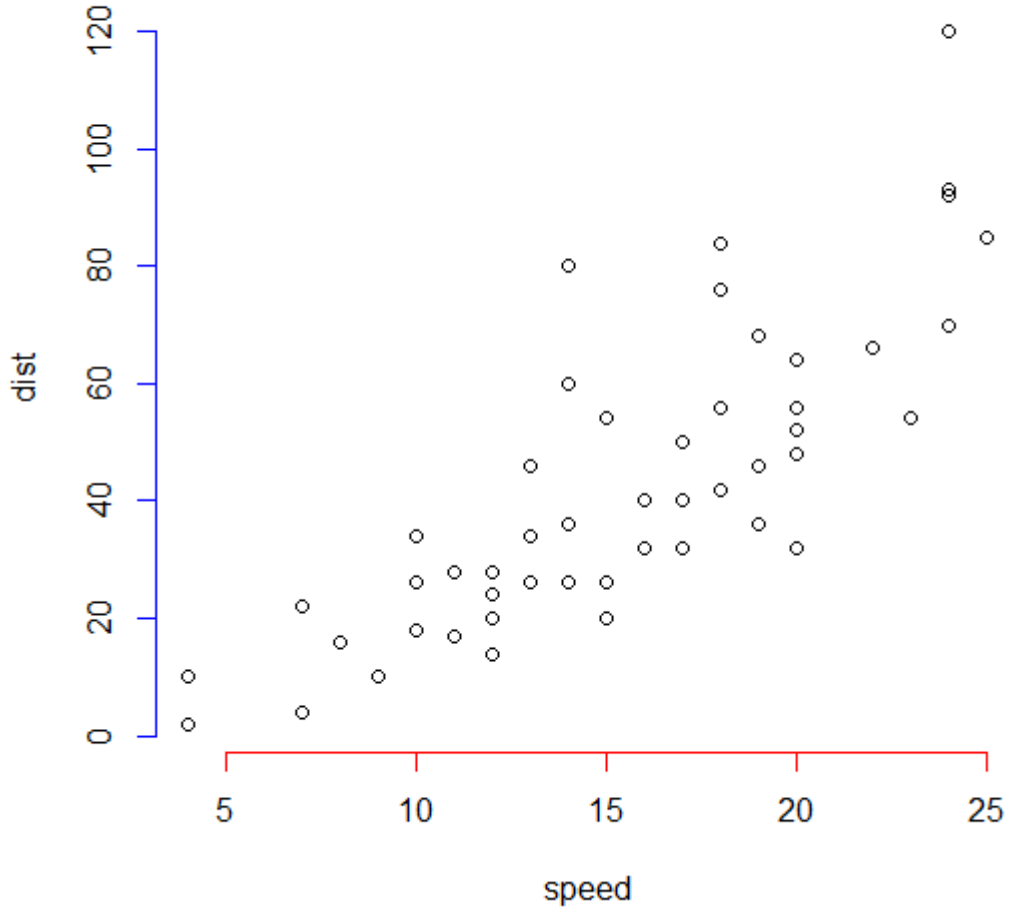
You can modify the values of these parameters when you plot the figure itself, e.g. using the `plot` function:

```
plot (dist ~ speed, data = cars, pch = 21, col = 'red', bg = 'yellow')
```



or using the low level function:

```
plot (dist ~ speed, data = cars, axes = FALSE)
axis (1, col = 'red')
axis (2, col = 'blue')
```



The list of parameters that can be used in this way is here:

Table 3.1
High-level traditional graphics state settings. This set of graphics state settings can be queried and set via the `par()` function *and* can be used as arguments to other graphics functions (e.g., `plot()` or `lines()`). Each setting is described in more detail in the relevant **Section**.

Setting	Description	Section
`adj`	justification of text	3.2.3
`ann`	draw plot labels and titles?	3.2.3
`bg`	"background" color	3.2.1
`bty`	type of box drawn by `box()`	3.2.5
`cex`	size of text (multiplier)	3.2.3
`cex.axis`	size of axis tick labels	3.2.3
`cex.lab`	size of axis labels	3.2.3
`cex.main`	size of plot title	3.2.3
`cex.sub`	size of plot sub-title	3.2.3
`col`	color of lines and data symbols	3.2.1
`col.axis`	color of axis tick labels	3.2.1
`col.lab`	color of axis labels	3.2.1
`col.main`	color of plot title	3.2.1
`col.sub`	color of plot sub-title	3.2.1
`fg`	"foreground" color	3.2.1
`font`	font face (bold, italic) for text	3.2.3
`font.axis`	font face for axis tick labels	3.2.3
`font.lab`	font face for axis labels	3.2.3
`font.main`	font face for plot title	3.2.3
`font.sub`	font face for plot sub-title	3.2.3
`gamma`	gamma correction for colors	3.2.1
`lab`	number of ticks on axes	3.2.5
`las`	rotation of text in margins	3.2.3
`lty`	line type (solid, dashed)	3.2.2
`lwd`	line width	3.2.2
`mgp`	placement of axis ticks and tick labels	3.2.5
`pch`	data symbol type	3.2.4
`srt`	rotation of text in plot region	3.2.3
`tck`	length of axis ticks (relative to plot size)	3.2.5
`tcl`	length of axis ticks (relative to text size)	3.2.5
`tmag`	size of plot title (relative to other labels)	3.2.3
`type`	type of plot (points, lines, both)	3.2.4
`xaxp`	number of ticks on x-axis	3.2.5
`xaxs`	calculation of scale range on x-axis	3.2.5
`xaxt`	x-axis style (standard, none)	3.2.5
`xpd`	clipping region	3.2.7
`yaxp`	number of ticks on y-axis	3.2.5
`yaxs`	calculation of scale range on y-axis	3.2.5
`yaxt`	y-axis style (standard, none)	3.2.5

3. Wrapping parameters into the list together with the argument of the plotting function

This is the finest control of the graphical parameters. You can wrap them together with the value of the argument when you call the argument inside the high- or low-level graphical function.

For example, if you want to change the appearance of the main text plotted above the scatterplot (making it larger, red and italics), you can wrap the relevant arguments (`cex`, `col` and `font`) into the list and assign to the argument `title`:

```
plot (dist ~ speed, data = cars, main = list ('Scatterplot', cex = 4, col = 'red',
font = 3))
```

Scatterplot

