

Chapter 9

More on plotting

9.1 Introduction

To learn more about plotting, we will use the data set `mtcars`, concerning fuel consumption and ten aspects of automobile design and performance for 32 US automobiles (1973–74 models). This data set comes with R. Load this data and have a brief look at the data as follows:

```
> data(mtcars)
> attach(mtcars)
```

```
The following object(s) are masked from mtcars ( position 3 ) :
```

```
am carb cyl disp drat gear hp mpg qsec vs wt
```

```
The following object(s) are masked from mtcars ( position 6 ) :
```

```
am carb cyl disp drat gear hp mpg qsec vs wt
```

```
> names(mtcars)
```

```
[1] "mpg" "cyl" "disp" "hp" "drat" "wt" "qsec" "vs" "am"
[10] "gear" "carb"
```

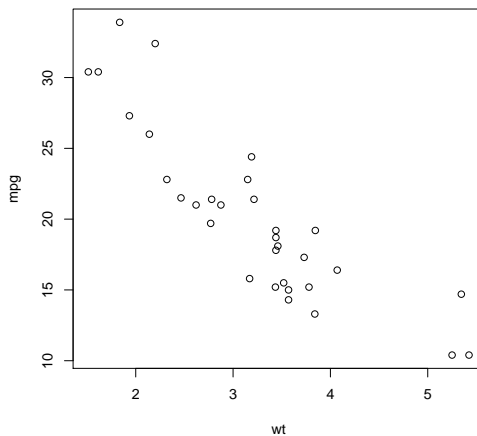
```
> head(mtcars)
```

	mpg	cyl	disp	hp	drat	wt	qsec	vs	am	gear	carb
Mazda RX4	21.0	6	160	110	3.90	2.620	16.46	0	1	4	4
Mazda RX4 Wag	21.0	6	160	110	3.90	2.875	17.02	0	1	4	4
Datsun 710	22.8	4	108	93	3.85	2.320	18.61	1	1	4	1
Hornet 4 Drive	21.4	6	258	110	3.08	3.215	19.44	1	0	3	1
Hornet Sportabout	18.7	8	360	175	3.15	3.440	17.02	0	0	3	2
Valiant	18.1	6	225	105	2.76	3.460	20.22	1	0	3	1

As with any data analysis, we like to start with plots. In this section, we discuss plotting options. You might like to make changes to the plotting commands in a `.R` file, as explained above, and then `source` this file. This is much easier than retyping similar commands over and over.

Let's initially examine the relationship between fuel consumption (`mpg`, or miles per gallon) and vehicle weight (`wt`):

```
> plot(mpg ~ wt)
```



Note that the *formula notation* has been used, where the tilde `~` means ‘is modelled by’ or ‘is described by’. This plot is pretty basic, but still quite decent for a default plot.

9.2 Adding labels

We can alter the axis labels as follows:

```
> plot(mpg ~ wt, xlab = "Weight (in pounds)", ylab = "Consumption (miles per gallon)")
```

We can also add a major title:

```
> plot(mpg ~ wt, xlab = "Weight (in pounds)", ylab = "Consumption (miles per gallon)",
+      main = "Fuel consumption against Weight")
```

If the main title is too long, you can split it across lines:

```
> plot(mpg ~ wt, xlab = "Weight (in pounds)", ylab = "Consumption (miles per gallon)",
+      main = "Fuel consumption against Weight")
```

There is also a sub-title that can be set, but I usually find it makes the graphics too cluttered and looks too much like an extension of the horizontal-axis label.

After you have drawn a figure, titles can be added using the `title()` command.

9.3 Changing the plotting characters

Points can be plotted using many different types of characters, defined by using `pch`. We can specify the character directly:

```
> plot(mpg ~ wt, pch = "@", xlab = "Weight (in pounds)",
+      ylab = "Consumption (miles per gallon)", main = "Fuel consumption against Weight")
```

Or we can use a predefined list (`pch=19` is one of my favourites):

```
> plot(mpg ~ wt, pch = 19, xlab = "Weight (in pounds)",
+      ylab = "Consumption (miles per gallon)", main = "Fuel consumption against Weight")
```

To see the list of predefined plotting characters, see `example(points)`.

9.4 Changing colours

Different colours can be used also:

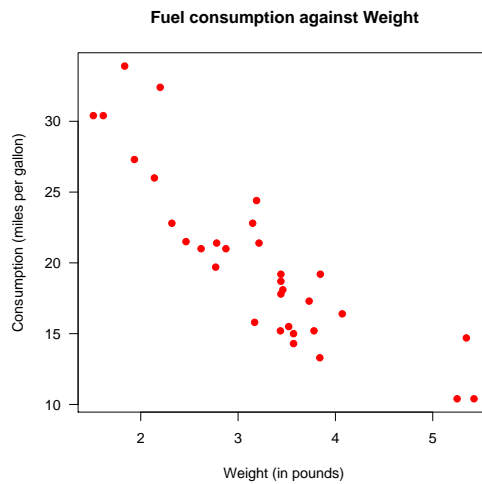
```
> plot(mpg ~ wt, pch = 19, col = "red", xlab = "Weight (in pounds)",
+      ylab = "Consumption (miles per gallon)", main = "Fuel consumption against Weight")
```

Colours may be defined explicitly as above, or numerically using a colour code. More on this later. (The standard `x11` colour names are used.)

9.5 Changing the format of the axes

I prefer my axis labels to always be horizontal. In R, sometimes this is almost essential if you make small graphics. Try this:

```
> plot(mpg ~ wt, pch = 19, col = "red", las = 1, xlab = "Weight (in pounds)",
+      ylab = "Consumption (miles per gallon)", main = "Fuel consumption against Weight")
```



9.6 Selecting parts of a data frame

Often, a simple scatterplot hides information. For example, in the relationship between fuel consumption and weight, the number of cylinders is probably quite important. In the data frame, the number of cylinders, `cyl`, is numeric. For some purposes, it is useful to think of the number of cylinders as a *factor* or *categorical variable*.

Consider this:

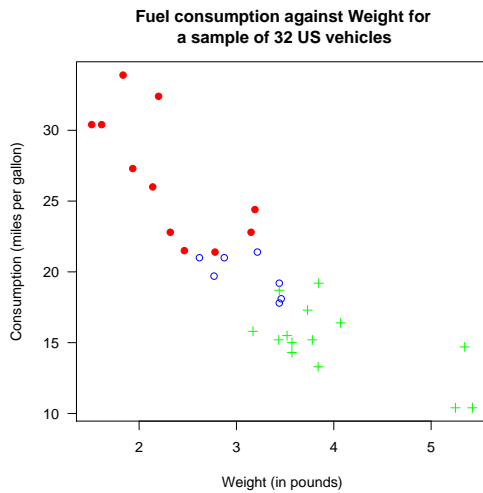
```
> coplot(mpg ~ wt | factor(cyl), columns = 3)
```

But we can do something similar with the standard scatterplot also. First, we plot the data invisibly (using `type="n"`, where `n` means ‘no plot at all (please)’).

```
> plot(mpg ~ wt, type = "n", las = 1, xlab = "Weight (in pounds)",
+      ylab = "Consumption (miles per gallon)", main = "Fuel consumption against Weight for
```

This sets up the axes, axis labels, and so forth, but doesn’t plot any data. Now we want to *add* data points, but separately for the number of cylinders. To do this, we select *subsets* of the data frame to plot (and note the use of `==` rather than `=`).

```
> plot(mpg ~ wt, type = "n", las = 1, xlab = "Weight (in pounds)",
+      ylab = "Consumption (miles per gallon)", main = "Fuel consumption against Weight for
> points(mpg[cyl == 4] ~ wt[cyl == 4], pch = 19, col = "red")
> points(mpg[cyl == 6] ~ wt[cyl == 6], pch = 1, col = "blue")
> points(mpg[cyl == 8] ~ wt[cyl == 8], pch = 3, col = "green")
```



Note that this way of selecting part of a data frame applies in general:

```
> mean(mpg[cyl == 4])
```

```
[1] 26.66364
```

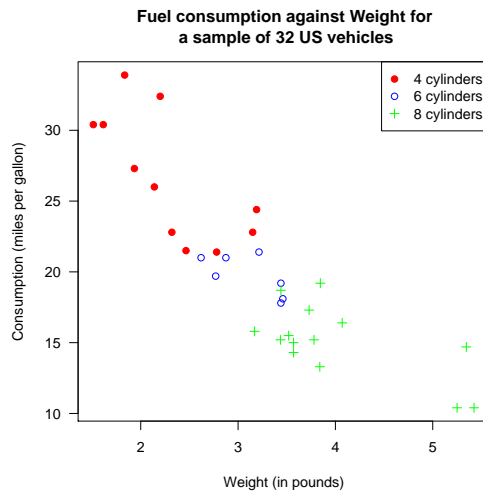
```
> median(mpg[cyl == 8])
```

```
[1] 15.2
```

9.7 Adding a legend

Finally, we might add a legend.

```
> plot(mpg ~ wt, type = "n", las = 1, xlab = "Weight (in pounds)",
+      ylab = "Consumption (miles per gallon)", main = "Fuel consumption against Weight for\n a samp
> points(mpg[cyl == 4] ~ wt[cyl == 4], pch = 19, col = "red")
> points(mpg[cyl == 6] ~ wt[cyl == 6], pch = 1, col = "blue")
> points(mpg[cyl == 8] ~ wt[cyl == 8], pch = 3, col = "green")
> legend("topright", col = c("red", "blue", "green"), pch = c(19,
+      1, 3), legend = c("4 cylinders", "6 cylinders", "8 cylinders"))
```



Now it is clear that cars with more cylinders are, in general, heavier and use more fuel.

9.8 Saving figures

After a picture is produced, it can be saved in a variety of formats. In Windows, you can use the menu to save a figure in some formats¹ Easiest is to save in (encapsulated) postscript:

```
dev.print(postscript, file="PracticePlot.eps")
```

(Notice this is called ‘print-ing’ in R; in reality, the picture is saved in a file.) This is so common, another function is usually used with sensible defaults set:

```
hist( wt )
dev.copy2eps(file="PracticePlot.eps")
```

To save as a png or jpeg file, use

```
hist(wt)
dev.print(png, file="PracticePlot.png")
dev.print(jpeg, file="PracticePlot.jpg")
```

(Again notice this is called ‘print-ing’; in reality, the picture is saved in a file.)

There other method is to open a png, jpg, eps, pdf or another R ‘device’ and plot *directly* to this device:

¹I don’t use Windows, so I know little about it. Perhaps you can right-click on the Windows and save it in some formats also. Noenttheless, everything in this section applies to all operating systems.

```
pdf(file="PracticePlot.pdf")
hist(wt)
dev.off()
```

Note the device must be turned off using `dev.off()` before the file is created. Nothing useful appears on the screen.

Once again, you can specify a different folder/directory by explicitly giving it.

9.9 Other options

There are many options that can be set. In fact, for a beginner the list is intimidating; above we have explored some of the more common options so we didn't need to examine the plethora of options! But it's now time to look at the available options. A list is found by typing:

```
?par
```

And don't forget:

```
> detach(mtcars)
```


Bibliography

- [1] D J Hand, F Daly, A D Lunn, K Y McConway, and E Ostrowski. *A Handbook of Small Data Sets*. Chapman and Hall, London, 1996.