

# Chapter 7

## Simple analyses

From the previous Chapter, a linear relationship is not unreasonable between the length and breadth of jellyfish. To fit a linear model, use the `lm` function, which stands for linear model. Perhaps it will be useful to look at `?lm` now.

### 7.1 Formula notation

In this section, we use the jellyfish data again:

```
> jf <- read.table("http://www.sci.usq.edu.au/staff/dunn/Datasets/Books/Hand/Hand-R/jelly-R.dat",
+   header = TRUE)
> attach(jf)
```

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

```
    Breadth Length Site
```

```
    The following object(s) are masked from jf ( position 4 ) :
```

```
    Breadth Length Site
```

```
> names(jf)
```

```
[1] "Breadth" "Length"  "Site"
```

To use this function, we need to understand R's *formula notation*. Formula notation is used frequently in R. It is best seen in practice:

```
> jf.lm <- lm(Breadth ~ Length)
```

The formula `Breadth ~ Length` works as follows:

- The tilde  $\sim$  means ‘described by’ or ‘modelled by’.
- On the left of the equation is the *response variable*, or the *dependent variable*, or the ‘Y’ variable.
- On the right of the tilde are the *predictors* or the *covariates*, or the *independent variables*, or the ‘X’. This right-hand side can be quite complex; we will see this later.

So the above model fits a linear regression model of the form

$$\text{Breadth} = \hat{\beta}_0 + \hat{\beta}_1 \text{Length}.$$

Notice the constant term is fitted by default as this is nearly always what is wanted. To explicitly omit the constant term, use

```
> lm(Breadth ~ Length - 1)
```

Call:

```
lm(formula = Breadth ~ Length - 1)
```

Coefficients:

```
Length
0.8497
```

or

```
> lm(Breadth ~ Length + 0)
```

Call:

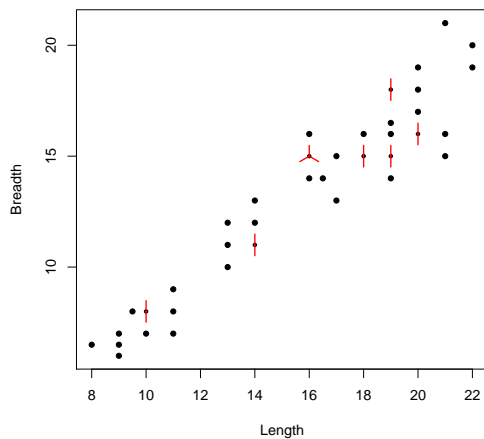
```
lm(formula = Breadth ~ Length + 0)
```

Coefficients:

```
Length
0.8497
```

Note that plotting can also use formula notation, and is often more natural:

```
> sunflowerplot(Breadth ~ Length)
```



## 7.2 Function output

In the above call to `lm`, the output was directed to a variable called `jf.lm`:

```
> jf.lm <- lm(Breadth ~ Length)
```

What is `jf.lm`? It is an R *object*. In fact, almost everything in R is called an *object*.

But what does this object contain? Let's have a look:

```
> jf.lm
```

Call:

```
lm(formula = Breadth ~ Length)
```

Coefficients:

(Intercept)	Length
-1.4423	0.9351

In fact, this is only a brief overview of the object `jf.lm`. We learn more by typing

```
> summary(jf.lm)
```

Call:

```
lm(formula = Breadth ~ Length)
```

Residuals:

Min	1Q	Median	3Q	Max
-3.19560	-0.81180	0.05847	0.70711	2.80440

```

Coefficients:
              Estimate Std. Error t value Pr(>|t|)
(Intercept) -1.44226    0.75199  -1.918  0.0616 .
Length      0.93514    0.04604  20.311 <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 1.287 on 44 degrees of freedom
Multiple R-Squared:  0.9036,    Adjusted R-squared:  0.9014
F-statistic: 412.5 on 1 and 44 DF,  p-value: < 2.2e-16

```

(Recall we used `summary` before to summarize data objects. Since `jf.lm` is a different type of object, `summary` does something different.)

This `summary` provides quite a lot of information: a brief summary of the residuals, the parameter estimates, their standard errors, the corresponding  $t$ -ratios and corresponding  $P$ -values, plus some more.

`jf.lm` is actually an object that contains a *lot* of information:

```

> names(jf.lm)

[1] "coefficients" "residuals"    "effects"      "rank"
[5] "fitted.values" "assign"       "qr"           "df.residual"
[9] "xlevels"      "call"        "terms"       "model"

```

So, for example, we can see the coefficients of the model by typing

```

> jf.lm$coefficients

(Intercept)      Length
-1.4422622    0.9351363

```

(Note this is similar to how we accessed variables within a data frame if it is not `attach`-ed.) The same format can be used to see all the other components also: try

```

> jf.lm$residuals

```

Typing `jf.lm$residuals` becomes a bit cumbersome after a while; for the commonly requested components, there are usually shortcuts:

```

> coef(jf.lm)

(Intercept)      Length
-1.4422622    0.9351363

> resid(jf.lm)[1:5]

```

```

      1          2          3          4          5
0.46117214 -0.97396413 -0.47396413  0.02603587  0.55846774

```

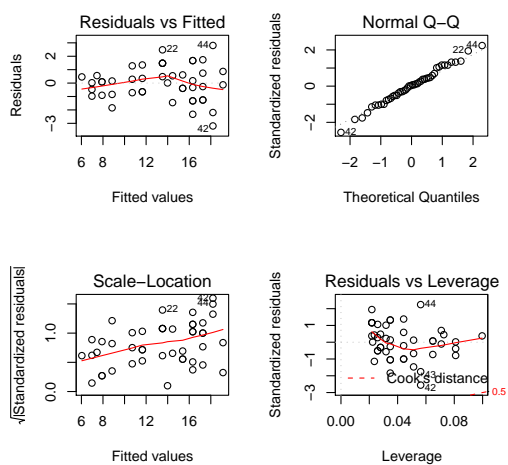
(Note the [1:5 ] means to extract just the first five residuals.)

You can also plot objects:

```

> par(mfrow = c(2, 2))
> plot(jf.lm)
> par(mfrow = c(1, 1))

```



(Again, this demonstrates R's object oriented approach: when you call `plot`, R determines what type of object you wish to plot, and plots sensible things accordingly.)

And don't forget:

```

> detach(jf)

```

