

Applied Statistics

Contents

1	Preliminaries	3
2	Introduction: A simple linear model	3
2.1	A simple linear model	4
2.1.1	Simple least squares estimation	5
2.2	Sampling properties of $\hat{\beta}$	5
2.3	So how old is the universe?	6
2.4	Adding a distributional assumption	8
2.4.1	Testing hypotheses about β	8
2.4.2	Confidence intervals	9
3	Linear models in general	10
3.1	Notation summary	12
3.2	R and linear models in general	12
3.2.1	A quadratic model	12
3.2.2	A model with factors	14
4	Linear model theory	15
4.1	Unbiasedness and variance	16
4.2	Checking	16
4.3	Further inference results: intervals and testing	17
4.3.1	$(\hat{\beta}_i - \beta_i)/\hat{\sigma}_{\hat{\beta}_i} \sim t_{n-p}$	17
4.3.2	Testing $H_0 : \mathbf{C}\beta = \mathbf{d}$	18
4.4	The geometry of linear models	18
4.5	Results in terms of \mathbf{X}	19
5	Linear models in R	19
5.1	The <code>cars</code> data again	19
5.1.1	Confidence intervals and hypothesis tests for single parameters	19
5.1.2	F-ratio tests about several parameters	22
5.1.3	Checking the model assumptions: residuals	23
5.1.4	Residual plots: common problems	26
5.2	Tyre wear data and backward selection	28
5.2.1	<code>drop1</code>	30
5.2.2	AIC	31
5.2.3	<code>step</code>	31
5.3	Forward and forward-backward selection	32
5.4	r^2 : How close is the fit?	33
5.5	Prediction	33
6	Causality, confounding and randomization	35
6.1	Confounding: some simulated examples	36
6.2	Controlled experiments and randomization	37
6.3	Instrumental variables	38

7	Practical modelling with factors and interactions	39
7.1	Identifiability	40
7.2	Multiple factors	41
7.3	'Interactions' of factors	42
7.4	Factor continuous interactions	43
7.5	Using factor variables in R	43
7.5.1	Factor interactions in model formulae	44
7.5.2	A simple example	44
7.6	The warpbreak data	46
7.6.1	Follow up	48
7.7	ANOVA tables	50
7.7.1	Example of erroneous dropping of a main effect	51
7.7.2	drop1 again	52
7.8	An agricultural field trial	52
8	How to approach an analysis	55
9	Beyond linear models	57
9.1	Maximum likelihood estimation	57
9.1.1	A simple example	58
9.1.2	Practical likelihood maximization	59
9.1.3	Illustrating $\hat{\theta}$ uncertainty	60
10	Introducing GLMs	61
10.1	Simple single covariate examples of GLMs	61
11	Inference with GLMs	63
11.1	The exponential family of distributions	63
11.2	GLM fitting	64
11.3	GLM checking and inference	65
12	glm in R	66
12.1	Heart attack example	69
12.2	Melanoma case-control study	73
12.3	Insurance claims testing example	74
12.4	Semiconductor wafer model selection example	74
12.4.1	AIC based selection	76
12.5	Remarks on model selection	79
12.6	Interpreting model coefficients	79
13	Mixed models	80
13.1	Mixed model theory	81
13.2	Computer intensive inference via resampling	82
14	Mixed models in R	84
14.1	Simple example	84
14.2	Loblolly pines: a more complicated example	85
14.3	Several levels of nesting	90
15	Generalized Additive Models	91
16	GAMs in R	93
16.1	A simple smoothing example	94
16.2	A GAM for fish egg survey data	98
16.3	Smooth interactions and smooth ANOVA decompositions	101

1 Preliminaries

This course builds on previous statistics units that you have taken. In particular it assumes that you are familiar with the key ideas behind parametric statistical inference.

- *Statistics* is about using data to extract meaningful information about the system generating the data, when the data generating process is such that the data will vary randomly from one replication of the data generating process to the next, even if the underlying system stays the same.
- A *statistical model* is a simplified mathematical representation of the data generating process (a sort of mathematical cartoon of the system that we want to learn about). It will usually depend on some known things, such as other variables we have measured alongside the data, and some unknown parameters: β , say.
- A key point about a statistical model is that *if* we knew the values of β , then the model should be able to simulate data that ‘looked like’ the real data. In principle, given values for the parameters, a statistical model also allows us to assign relative probabilities to observing one data set, as opposed to another.
- Once we have a model, the major goal of statistics is then to make *inferences* about β from the data. There are 4 questions:
 1. What value of β is most consistent with the data?
 2. What range of values of β is consistent with the data
 3. Is some pre-specified value of, or restriction on, β consistent with the data?
 4. Are there any values of β at all for which the model is consistent with the data?
- The answers to these questions are provided by
 1. Parameter estimation, especially maximum likelihood estimation, which seeks to find the $\hat{\beta}$ making the observed data as probable as possible, according to the model.
 2. Confidence interval estimation, which uses the data to compute intervals with a specified probability of including the true parameter values, over replication of the data generating process .
 3. Hypothesis testing, which seeks to assess the plausibility of some hypothesis, by computing a measure of how improbable the data are under that hypothesis: the *p-value* (see later). A low p-values suggests that the data would be improbable if the hypothesis were true, so it’s probably false.
 4. Model checking: could the model have produced the data at all? If it could not then none of the theory in 1-3 is of any use at all. Useful checking is often done graphically.

In terms of technicalities, the course particularly assumes that you are familiar with random variables, p.d.f.s, c.d.f.s and inverse c.d.f.s (quantile functions). It also assumes that if you are told $\hat{\beta} \sim N(\beta, \sigma_{\hat{\beta}}^2)$ or $(\hat{\beta} - \beta)/\hat{\sigma}_{\hat{\beta}} \sim t_d$ (plus any values needed), then you could test hypotheses about β and find confidence intervals for β (in your sleep, after a heavy night out, with one hand tied behind your back etc).

2 Introduction: A simple linear model

How old is the universe? The standard big-bang model of the origin of the universe says that it expands uniformly, and locally, according to Hubble’s law,

$$y = \beta x,$$

where y is the relative velocity of any two galaxies separated by distance x , and β is “Hubble’s constant” (in standard astrophysical notation $y \equiv v$, $x \equiv d$ and $\beta \equiv H_0$). β^{-1} gives the approximate age of the universe, but β is unknown and must somehow be estimated from observations of y and x , made for a variety of galaxies at different distances from us.

Figure 1 plots velocity against distance for 24 galaxies, according to measurements made using the Hubble Space Telescope. Velocities are assessed by measuring the Doppler effect red shift in the spectrum of light observed from the galaxies concerned, although some correction for ‘local’ velocity components is required. Distance

measurement is much less direct, and is based on the 1912 discovery, by Henrietta Leavitt, of a relationship between the period of a certain class of variable stars, known as the Cepheids, and their luminosity. The intensity of Cepheids varies regularly with a period of between 1.5 and something over 50 days, and the mean intensity increases predictably with period. This means that, if you can find a Cepheid, you can tell how far away it is, by comparing its apparent brightness to its period predicted intensity.

It is clear, from the figure, that the observed data do not follow Hubble's law exactly, but given the measurement process, it would be surprising if they did. Given the apparent variability, what can be inferred from these data? In particular: (i) what value of β is most consistent with the data? (ii) what range of β values is consistent with the data and (iii) are some particular, theoretically derived, values of β consistent with the data? Statistics is about trying to answer these three sorts of questions.

One way to proceed is to formulate a linear statistical model of the way that the data were generated, and to use this as the basis for inference. Specifically, suppose that, rather than being governed directly by Hubble's law, the observed velocity is given by Hubble's constant multiplied by the observed distance plus a 'random variability' term. That is

$$y_i = \beta x_i + \epsilon_i, \quad i = 1 \dots 24, \quad (1)$$

where the ϵ_i terms are independent random variables such that $\mathbb{E}(\epsilon_i) = 0$ and $\mathbb{E}(\epsilon_i^2) = \sigma^2$. The random component of the model is intended to capture the fact that if we gathered a replicate set of data, for a new set of galaxies, Hubble's law would not change, but the apparent random variation from it would be different, as a result of different measurement errors. Notice that it is not implied that these errors are completely unpredictable: their mean and variance are assumed to be fixed, it is only their particular values, for any particular galaxy, that are not known.

2.1 A simple linear model

This section develops statistical methods for a simple linear model of the form (1). This allows the key concepts of linear modelling to be reviewed without the distraction of any mathematical difficulty.

Formally, consider n observations, x_i, y_i , where y_i is an observation on random variable, Y_i , with expectation, $\mu_i \equiv \mathbb{E}(Y_i)$. Suppose that an appropriate model for the relationship between x and y is:

$$Y_i = \mu_i + \epsilon_i \text{ where } \mu_i = x_i \beta. \quad (2)$$

Here β is an unknown parameter and the ϵ_i are mutually independent zero mean random variables, each with the same variance σ^2 . So the model says that Y is given by x multiplied by a constant plus a random term. Y is an example of a *response variable*, while x is an example of a *predictor variable*. Figure 2 illustrates this model for a case where $n = 8$.

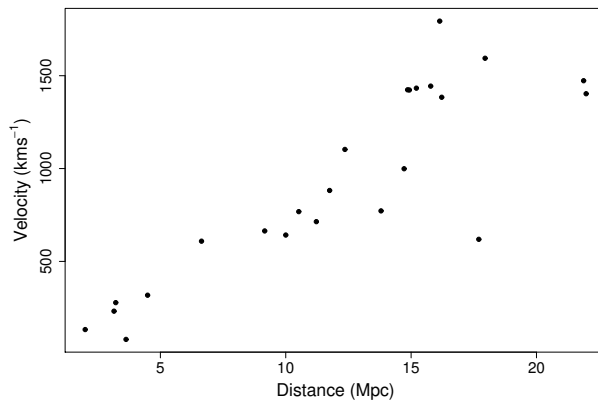


Figure 1: A Hubble diagram showing the relationship between distance, x , and velocity, y , for 24 galaxies containing Cepheid stars. The data are from the Hubble Space Telescope key project to measure the Hubble constant.

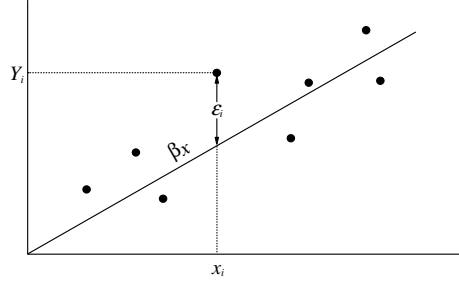


Figure 2: Schematic illustration of a simple linear model with one explanatory variable.

2.1.1 Simple least squares estimation

How can β , in model (2), be estimated from the x_i, y_i data? A sensible approach is to choose a value of β that makes the model fit closely to the data. To do this we need to define a measure of how well, or how badly, a model with a particular β fits the data. One possible measure is the **residual sum of squares** (RSS) of the model:

$$S = \sum_{i=1}^n (y_i - \mu_i)^2.$$

For the current model the RSS is

$$S = \sum_{i=1}^n (y_i - x_i\beta)^2.$$

If we have chosen a good value of β , close to the true value, then the model predicted μ_i should be relatively close to the y_i , so that S should be small, whereas poor choices will lead to μ_i far from their corresponding y_i , and high values of S . Hence β can be estimated by minimizing S with respect to (w.r.t.) β and this is known as the method of *least squares*.

To minimize S , for the current simple model, differentiate w.r.t. β :

$$\frac{\partial S}{\partial \beta} = - \sum_{i=1}^n 2x_i(y_i - x_i\beta)$$

and set the result to zero to find $\hat{\beta}$, the least squares estimate of β :

$$- \sum_{i=1}^n 2x_i(y_i - x_i\hat{\beta}) = 0 \Rightarrow \sum_{i=1}^n x_i y_i - \hat{\beta} \sum_{i=1}^n x_i^2 = 0 \Rightarrow \hat{\beta} = \sum_{i=1}^n x_i y_i / \sum_{i=1}^n x_i^2.*$$

2.2 Sampling properties of $\hat{\beta}$

To evaluate the reliability of the least squares estimate, $\hat{\beta}$, it is useful to consider the sampling properties of $\hat{\beta}$. That is, we should consider some properties of the distribution of $\hat{\beta}$ values, which would be obtained from repeated independent replication of the x_i, y_i data used for estimation. To do this, it is helpful to introduce the concept of an *estimator*, which is obtained by replacing the observations, y_i , in the estimate of $\hat{\beta}$ by the random variables, Y_i , to obtain

$$\hat{\beta} = \sum_{i=1}^n x_i Y_i / \sum_{i=1}^n x_i^2.$$

Clearly the *estimator*, $\hat{\beta}$, is a random variable and we can therefore discuss its distribution. For now, consider only the first two moments of that distribution.

* $\partial^2 S / \partial \beta^2 = 2 \sum x_i^2$ which is clearly positive, so a minimum of S has been found.

The expected value of $\hat{\beta}$ is obtained as follows:

$$\mathbb{E}(\hat{\beta}) = \mathbb{E}\left(\sum_{i=1}^n x_i Y_i / \sum_{i=1}^n x_i^2\right) = \sum_{i=1}^n x_i \mathbb{E}(Y_i) / \sum_{i=1}^n x_i^2 = \sum_{i=1}^n x_i^2 \beta / \sum_{i=1}^n x_i^2 = \beta.$$

So $\hat{\beta}$ is an unbiased estimator — its expected value is equal to the true value of the parameter that it is supposed to estimate.

Unbiasedness is a reassuring property, but knowing that an estimator gets it right on average, does not tell us much about how good any one particular estimate is likely to be: for this we also need to know how much estimates would vary from one replicate data set to the next — we need to know the estimator variance.

From general probability theory we know that if Y_1, Y_2, \dots, Y_n are *independent* random variables and a_1, a_2, \dots, a_n are real constants then

$$\text{var}\left(\sum_i a_i Y_i\right) = \sum_i a_i^2 \text{var}(Y_i).$$

But we can write

$$\hat{\beta} = \sum_i a_i Y_i \text{ where } a_i = x_i / \sum_i x_i^2,$$

and from the original model specification we have that $\text{var}(Y_i) = \sigma^2$ for all i . Hence,

$$\text{var}(\hat{\beta}) = \sum_i x_i^2 / \left(\sum_i x_i^2\right)^2 \sigma^2 = \left(\sum_i x_i^2\right)^{-1} \sigma^2. \quad (3)$$

In most circumstances σ^2 itself is an unknown parameter and must also be estimated. Since σ^2 is the variance of the ϵ_i , it makes sense to estimate it using the variance of the ‘estimated’ ϵ_i , the model **residuals**. In general these are defined as $\hat{\epsilon}_i = y_i - \hat{\mu}_i$, where $\hat{\mu}_i$ are the **fitted values**, the model estimates of $\mu_i = \mathbb{E}(y_i)$. For the current simple model $\hat{\mu}_i = x_i \hat{\beta}$, so $\hat{\epsilon}_i = y_i - x_i \hat{\beta}$. An unbiased estimator of σ^2 for this model is then:

$$\hat{\sigma}^2 = \frac{1}{n-1} \sum_i (y_i - x_i \hat{\beta})^2$$

(proof of unbiasedness is given later for the general case). Plugging this into (3) obviously gives an unbiased estimate of the variance of $\hat{\beta}$.

2.3 So how old is the universe?

The least squares calculations derived above are available as part of the statistical package and environment R. The function `lm` fits linear models to data, including the simple example currently under consideration. The Cepheid distance — velocity data shown in figure 1 are stored in a data frame[†] `hubble`. The following R code fits the model and produces the (edited) output shown.

```
> library(gamair); data(hubble)
> hub.mod <- lm(y~x-1,data=hubble)
> summary(hub.mod)
```

Call:

```
lm(formula = y ~ x - 1, data = hubble)
```

Coefficients:

	Estimate	Std. Error
x	76.581	3.965

[†]A data frame is just a two dimensional array of data in which the values of different variables are stored in different named columns.

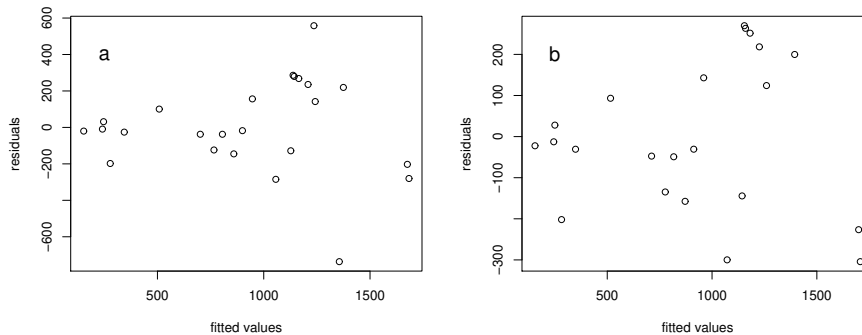


Figure 3: Residuals against fitted values for (a) the model (1) fitted to all the data in figure 1 and (b) the same model fitted to data with two substantial outliers omitted.

The call to `lm` passed two arguments to the function. The first is a *model formula*, $y \sim x - 1$, specifying the model to be fitted: the name of the response variable is to the left of ‘ \sim ’ while the predictor variable is specified on the right; the ‘ -1 ’ term indicates that the model has no ‘intercept’ term, i.e. that the model is a straight line through the origin. The second (optional) argument gives the name of the data frame in which the variables are to be found. `lm` takes this information and uses it to fit the model by least squares: the results are returned in a ‘fitted model object’, which in this case has been assigned to an object called `hub.mod` for later examination. ‘ $<-$ ’ is the assignment operator, and `hub.mod` is created by this assignment (overwriting any previously existing object of this name).

The `summary` function is then used to examine the fitted model object. Only part of its output is shown here: $\hat{\beta}$ and the estimate of the standard error of $\hat{\beta}$ (the square root of the estimated variance of $\hat{\beta}$, derived above). Before using these quantities it is important to check the model assumptions. In particular we should check the plausibility of the assumptions that the ϵ_i are independent and all have the same variance. The way to do this is to examine residual plots.

Recall that the ‘fitted values’ for this model are defined as $\hat{\mu}_i = \hat{\beta}x_i$, while the residuals are simply $\hat{\epsilon}_i = y_i - \hat{\mu}_i$. A plot of residuals against fitted values is often useful and the following produces the plot in figure 3(a).

```
plot(fitted(hub.mod), residuals(hub.mod), xlab="fitted values",
     ylab="residuals")
```

What we would like to see, in such a plot, is an apparently random scatter of residuals around zero, with no trend in either the mean of the residuals, or their variability, as the fitted values increase. A trend in the mean violates the independence assumption, and is usually indicative of something missing in the model structure, while a trend in the variability violates the constant variance assumption. The main problematic feature of figure 3(a) is the presence of two points with very large magnitude residuals, suggesting a problem with the constant variance assumption. It is probably prudent to repeat the model fit, with and without these points, to check that they are not having undue influence on our conclusions. The following code omits the offending points and produces a new residual plot shown in figure 3(b).

```
> hub.mod1 <- lm(y~x-1, data=hubble[-c(3,15),])
> summary(hub.mod1)
```

Call:

```
lm(formula = y ~ x - 1, data = hubble[-c(3, 15), ])
```

Coefficients:

	Estimate	Std. Error
x	77.67	2.97

```
> plot(fitted(hub.mod1), residuals(hub.mod1),
```

```
+      xlab="fitted values",ylab="residuals")
```

The omission of the two large outliers has improved the residuals and changed $\hat{\beta}$ somewhat, but not drastically.

The Hubble constant estimates have units of $(\text{km})\text{s}^{-1} (\text{Mpc})^{-1}$. A Mega-parsec is $3.09 \times 10^{19}\text{km}$, so we need to divide $\hat{\beta}$ by this amount, in order to obtain Hubble's constant with units of s^{-1} . The approximate age of the universe, in seconds, is then given by the reciprocal of $\hat{\beta}$. Here are the two possible estimates expressed in years:

```
> hubble.const <- c(coef(hub.mod),coef(hub.mod1))/3.09e19
> age <- 1/hubble.const
> age/(60^2*24*365)
12794692825 12614854757
```

Both fits give an age of around 13 billion years. So we now have an idea of the best estimate of the age of the universe, but what range of ages would be consistent with the data?

2.4 Adding a distributional assumption

So far everything done with the simple model has been based only on the model equations and the two assumptions of independence and equal variance, for the response variable. If we wish to go further, and find confidence intervals for β , or test hypotheses related to the model, then a further distributional assumption will be necessary.

Specifically, assume that $\epsilon_i \sim N(0, \sigma^2)$ for all i , which is equivalent to assuming $Y_i \sim N(x_i\beta, \sigma^2)$. Now we have already seen that $\hat{\beta}$ is just a weighted sum of Y_i , but the Y_i are now assumed to be normal random variables, and a weighted sum of normal random variables is itself a normal random variable. Hence the estimator, $\hat{\beta}$, must be a normal random variable. Since we have already established the mean and variance of $\hat{\beta}$, we have that

$$\hat{\beta} \sim N\left(\beta, \left(\sum x_i^2\right)^{-1} \sigma^2\right). \quad (4)$$

2.4.1 Testing hypotheses about β

One thing we might want to do is to try and evaluate the consistency of some hypothesized value of β with the data. For example some Creation Scientists estimate the age of the universe to be 6000 years, based on a reading of the Bible. This would imply that $\beta = 163 \times 10^{6\ddagger}$. The consistency with data of such a hypothesized value for β can be based on the probability that we would have observed the $\hat{\beta}$ actually obtained, if the true value of β was the hypothetical one.

Specifically, we can test the null hypothesis, $H_0 : \beta = \beta_0$, versus the alternative hypothesis, $H_1 : \beta \neq \beta_0$, for some specified value β_0 , by examining the probability of getting the observed $\hat{\beta}$, or one further from β_0 , assuming H_0 to be true. If σ^2 were known then we could work directly from (4), as follows.

The probability required is known as the **p-value** of the test. It is

the probability of getting a value of $\hat{\beta}$ at least as favourable to H_1 as the one actually observed, if H_0 is actually true.

Actually this definition holds for the p-value of any hypothesis test, if the specific ' $\hat{\beta}$ ' is replaced by the general '*a test statistic*'. In this case it helps to distinguish notationally between the estimate, $\hat{\beta}_{\text{obs}}$, and estimator $\hat{\beta}$. The p-value is then

$$\begin{aligned} p &= \Pr\left[|\hat{\beta} - \beta_0| \geq |\hat{\beta}_{\text{obs}} - \beta_0| \mid H_0\right] \\ &= \Pr\left[\left|\frac{\hat{\beta} - \beta_0}{\sigma_{\hat{\beta}}}\right| \geq \left|\frac{\hat{\beta}_{\text{obs}} - \beta_0}{\sigma_{\hat{\beta}}}\right| \mid H_0\right] \\ &= \Pr[|Z| > |z|] \end{aligned}$$

[‡]This isn't really valid, of course, since the Creation Scientists are not postulating a big bang theory.

where $Z \sim N(0, 1)$, $z = (\hat{\beta}_{\text{obs}} - \beta_0)/\sigma_{\hat{\beta}}$ and $\sigma_{\hat{\beta}}^2 = (\sum x_i^2)^{-1}\sigma^2$. Hence, having formed z , the p-value is easily worked out, using the cumulative distribution function for the standard normal, built into any statistics package. Small p-values suggest that the data are inconsistent with H_0 , while large values indicate consistency. 0.05 is often used as the boundary between ‘small’ and ‘large’ in this context.

In reality σ^2 is usually unknown. Broadly the same testing procedure can still be adopted, by replacing σ with $\hat{\sigma}$, but we need to somehow allow for the extra uncertainty that this introduces (unless the sample size is very large). It turns out that if $H_0 : \beta = \beta_0$ is true then

$$T \equiv \frac{\hat{\beta} - \beta_0}{\hat{\sigma}_{\hat{\beta}}} \sim t_{n-1}$$

where n is the sample size, $\hat{\sigma}_{\hat{\beta}}^2 = (\sum x_i^2)^{-1}\hat{\sigma}^2$, and t_{n-1} is the t-distribution with $n - 1$ degrees of freedom. The value $n - 1$ comes from the number of data minus the number of estimated model parameters, and the result is proven in section 4.3.1. It is clear that large magnitude values of T favour H_1 , so using T as the test statistic, in place of $\hat{\beta}$, we can calculate a p-value by evaluating

$$p = \Pr[|T| > |t|]$$

where $T \sim t_{n-1}$ and $t = (\hat{\beta}_{\text{obs}} - \beta_0)/\hat{\sigma}_{\hat{\beta}}$. This is easily evaluated using the c.d.f. of the t distributions, built into any decent statistics package. Here is some code to evaluate the p-value for $H_0 : \text{the Hubble constant is } 163000000$.

```
> cs.hubble <- 163000000
> t.stat <- (coef(hub.mod1) - cs.hubble) /
+ summary(hub.mod1)$coefficients[2]
> pt(t.stat, df=21) * 2 # 2 because of |T| in p-value defn.
3.906388e-150
```

i.e. as judged by the test statistic, t , the data would be hugely improbable if $\beta = 1.63 \times 10^8$. It would seem that the hypothesized value can be rejected rather firmly (in this case, using the data with the outliers increases the p-value by a factor of 1000 or so).

Hypothesis testing is particularly useful when there are good reasons to want to stick with some null hypothesis, until there is good reason to reject it. This is often the case when comparing models of differing complexity: it is often a good idea to retain the simpler model until there is quite firm evidence that it is inadequate. Note one interesting property of hypothesis testing. If we choose to reject a null hypothesis whenever the p-value is less than some fixed level, α (often termed the *significance level* of a test), then we will inevitably reject a proportion, α , of correct null hypotheses. We could try and reduce the probability of such mistakes by making α very small, but in that case we pay the price of reducing the probability of rejecting H_0 when it is false!

2.4.2 Confidence intervals

Having seen how to test whether a *particular* hypothesized value of β is consistent with the data, the question naturally arises of what *range* of values of β would be consistent with the data? To answer this, we need to select a definition of ‘consistent’: a common choice is to say that any parameter value is consistent with the data if it results in a p-value of ≥ 0.05 , when used as the null value in a hypothesis test.

Sticking with the Hubble constant example, and working at a significance level of 0.05, we would have rejected any hypothesized value for the constant, that resulted in a t value outside the range $(-2.08, 2.08)$, since these values would result in p-values of less than 0.05. The R function `qt` can be used to find such ranges: e.g. `qt(c(0.025, 0.975), df=21)` returns the range of the middle 95% of t_{21} random variables. So we would accept any β_0 fulfilling:

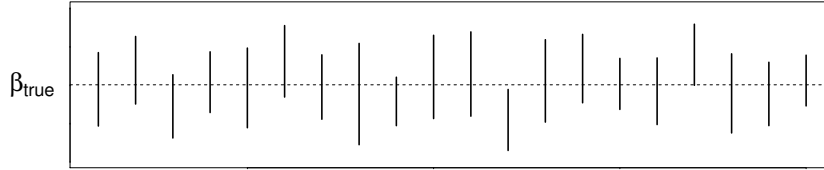
$$-2.08 \leq \frac{\hat{\beta} - \beta_0}{\hat{\sigma}_{\hat{\beta}}} \leq 2.08$$

which re-arranges to give the interval

$$\hat{\beta} - 2.08\hat{\sigma}_{\hat{\beta}} \leq \beta_0 \leq \hat{\beta} + 2.08\hat{\sigma}_{\hat{\beta}}.$$

Such an interval is known as a ‘95% Confidence interval’ for β .

The defining property of a 95% confidence interval is this: if we were to gather an infinite sequence of independent replicate data sets, and calculate 95% confidence intervals for β from each, then 95% of these intervals would include the true β , and 5% would not. It is easy to see how this comes about. By construction, a hypothesis test with a significance level of 5% rejects the correct null hypothesis for 5% of replicate data sets, and accepts it for the other 95% of replicates. Hence 5% of 95% confidence intervals must exclude the true parameter, while 95% include it. The following illustrates this, showing intervals for a parameter β computed for 20 replicate datasets...



For the Hubble example, a 95% CI for the constant (in the usual astro-physicists units) is given by:

```
> sigb <- summary(hub.mod1)$coefficients[2]
> h.ci <- coef(hub.mod1)+qt(c(0.025, 0.975), df=21)*sigb
> h.ci
[1] 71.49588 83.84995
```

This can be converted to a confidence interval for the age of the universe, in years, as follows:

```
> h.ci <- h.ci*60^2*24*365.25/3.09e19 # convert to 1/years
> sort(1/h.ci)
[1] 11677548698 13695361072
```

i.e. the 95% CI is (11.7,13.7) billion years. Actually this ‘Hubble age’ is the age of the universe if it has been expanding freely, neither being slowed down by gravitation, nor accelerating for reasons not fully understood. In fact this free expansion assumption does not appear to be quite right, so our answer will be out by a bit more than the CI might suggest.

3 Linear models in general

The simple linear model, introduced above, can be generalized by allowing the response variable to depend on multiple predictor variables (plus an additive constant). These extra predictor variables can themselves be transformations of the original predictors. Here are some examples, for each of which a response variable datum, y_i , is treated as an observation on a random variable, Y_i , where $\mathbb{E}(Y_i) \equiv \mu_i$, the ϵ_i are zero mean random variables, and the β_j are model parameters, the values of which are unknown and will need to be estimated using data.

1. $\mu_i = \beta_0 + x_i\beta_1$, $Y_i = \mu_i + \epsilon_i$, is a straight line relationship between y and predictor variable, x .
2. $\mu_i = \beta_0 + x_i\beta_1 + x_i^2\beta_2 + x_i^3\beta_3$, $Y_i = \mu_i + \epsilon_i$, is a cubic model of the relationship between y and x .
3. $\mu_i = \beta_0 + x_i\beta_1 + z_i\beta_2 + \log(x_i z_i)\beta_3$, $Y_i = \mu_i + \epsilon_i$, is a model in which y depends on predictor variables x and z and on the log of their product.

Each of these is a linear model because the ϵ_i terms and the model parameters, β_j , enter the model in a linear way. Notice that the predictor variables can enter the model non-linearly. Exactly as for the simple model, the parameters of these models can be estimated by finding the β_j values which make the models best fit the observed data, in the sense of minimizing $\sum_i (y_i - \mu_i)^2$. The theory for doing this will be developed in section 4, and that development is based entirely on re-writing the linear model using matrices and vectors.

To see how this re-writing is done, consider the straight line model given above. Writing out the μ_i equations for all n pairs, (x_i, y_i) , results in a large system of linear equations:

$$\begin{aligned}\mu_1 &= \beta_0 + x_1\beta_1 \\ \mu_2 &= \beta_0 + x_2\beta_1 \\ \mu_3 &= \beta_0 + x_3\beta_1 \\ &\vdots \\ \mu_n &= \beta_0 + x_n\beta_1\end{aligned}$$

which can be re-written in matrix-vector form as

$$\begin{pmatrix} \mu_1 \\ \mu_2 \\ \mu_3 \\ \vdots \\ \mu_n \end{pmatrix} = \begin{pmatrix} 1 & x_1 \\ 1 & x_2 \\ 1 & x_3 \\ \vdots & \vdots \\ 1 & x_n \end{pmatrix} \begin{pmatrix} \beta_0 \\ \beta_1 \end{pmatrix}.$$

So the model has the general form

$$\boldsymbol{\mu} = \mathbf{X}\boldsymbol{\beta}.$$

i.e. the expected value vector $\boldsymbol{\mu}$ is given by a **model matrix** (also known as a design matrix), \mathbf{X} , multiplied by a **parameter vector**, $\boldsymbol{\beta}$. All linear models can be written in this general form. Of course since $y_i = \mu_i + \epsilon_i$, we can also write the model as

$$\mathbf{y} = \mathbf{X}\boldsymbol{\beta} + \boldsymbol{\epsilon}.$$

As a second illustration, the cubic example, given above, can be written in matrix vector form as

$$\begin{pmatrix} \mu_1 \\ \mu_2 \\ \mu_3 \\ \vdots \\ \mu_n \end{pmatrix} = \begin{pmatrix} 1 & x_1 & x_1^2 & x_1^3 \\ 1 & x_2 & x_2^2 & x_2^3 \\ 1 & x_3 & x_3^2 & x_3^3 \\ \vdots & \vdots & \vdots & \vdots \\ 1 & x_n & x_n^2 & x_n^3 \end{pmatrix} \begin{pmatrix} \beta_0 \\ \beta_1 \\ \beta_2 \\ \beta_3 \end{pmatrix}.$$

Models in which data are divided into different groups, each of which are assumed to have a different mean, are less obviously of the form $\boldsymbol{\mu} = \mathbf{X}\boldsymbol{\beta}$, but in fact they can be written in this way, by use of dummy indicator variables. Again, this is most easily seen by example. Consider the model

$$\mu_i = \beta_j \text{ if observation } i \text{ is in group } j,$$

and suppose that there are three groups, each with 2 data. Then the model can be re-written

$$\begin{pmatrix} \mu_1 \\ \mu_2 \\ \mu_3 \\ \mu_4 \\ \mu_5 \\ \mu_6 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} \beta_0 \\ \beta_1 \\ \beta_2 \end{pmatrix}.$$

Variables indicating the group to which a response observation belongs, are known as *factor* variables. Somewhat confusingly, the groups themselves are known as *levels* of a factor. So the above model involves one factor, ‘group’, with three levels. Models of this type, involving factors, are commonly used for the analysis of designed experiments. In this case the model matrix depends on the design of the experiment (i.e. on which units belong to which groups), and for this reason the terms ‘design matrix’ and ‘model matrix’ are often used interchangeably. Whatever it is called, \mathbf{X} is absolutely central to understanding the theory of linear models, generalized linear models and generalized additive models.

3.1 Notation summary

- The *response variable* vector \mathbf{y} , has *expected value* vector $\boldsymbol{\mu} = \mathbb{E}(\mathbf{y})$.
- According to the linear model $\boldsymbol{\mu} = \mathbf{X}\boldsymbol{\beta}$, where \mathbf{X} is the *model matrix* and $\boldsymbol{\beta}$ the *parameter vector* (or *coefficient vector*).
- Hence the linear model can also be written as $\mathbf{y} = \mathbf{X}\boldsymbol{\beta} + \boldsymbol{\epsilon}$, where $\boldsymbol{\epsilon}$ is a *random error* vector.
- Once the model is estimated, the *parameter estimates* are denoted $\hat{\boldsymbol{\beta}}$. This notation will also be used for the *parameter estimator*, the context indicating which is meant.
- The *fitted value* vector is $\hat{\boldsymbol{\mu}} = \mathbf{X}\hat{\boldsymbol{\beta}}$.
- The *residual* vector is $\hat{\boldsymbol{\epsilon}} = \mathbf{y} - \hat{\boldsymbol{\mu}}$.

3.2 R and linear models in general

R has functions for setting up model matrices automatically, on the basis of a *model formula* specifying the structure for the columns of the model matrix. We'll look at a couple of examples here.

3.2.1 A quadratic model

The `cars` data frame contains data on stopping distances of cars against speed when the driver was first signalled to stop. It typically takes some fixed amount of 'reaction time', after the stop signal, for the driver to apply the brakes, hence the car will travel a distance proportional to initial speed before beginning to slow at all: so we need a linear dependence on `speed` in the model. The kinetic energy of a car is proportional to the square of its speed, and once the brakes are applied they can dissipate this energy (ultimately as heat) at a constant rate per unit distance travelled: hence the braking phase contributes a distance proportional to the square of `speed` to the total stopping distance. i.e. a suitable model for the data is:

$$\text{distance}_i = \beta_0 + \beta_1 \text{speed}_i + \beta_2 \text{speed}_i^2 + \epsilon_i$$

where the ϵ_i are i.i.d. $N(0, \sigma^2)$ random variables. Of course we would expect the parameter β_0 to be zero, given the justification for the model, but there is no harm in leaving it in for the moment. If there is statistical evidence that it is not zero, then this would suggest something wrong with our physical model: so it provides a useful check.

The R function `model.matrix` can be used to set up the model matrix for this model of the `cars` data, as follows.

```
> cars # check what data look like
  speed dist
1     4     2
2     4    10
3     7     4
.      .     .
.      .     .
50    25    85
> X <- model.matrix(~speed+I(speed^2), data=cars)
```

Like `lm`, `model.matrix` takes a model formula and a data frame as arguments. It returns a model matrix. In this case the formula can be *one sided* as the response variable is immaterial when setting up the model matrix. Everything to the right of `~` in the formula specifies the structure of the model matrix. In this case we specify that there should be one column containing the speeds, one column containing the squares of the speeds and, since we did not suppress it, there will also be a column of 1s, corresponding to the model intercept, β_0 . Because some arithmetic symbols have special meanings (see later in notes) within model formulae, we have to 'protect' the term `speed^2`, by writing it as `I(speed^2)`, which means that we want `^2` to have its usual arithmetic meaning here. `I()` is the *identity function*: it just returns its argument. Here is what `X` looks like...

```
> X
      (Intercept) speed I(speed^2)
1              1      4         16
2              1      4         16
3              1      7         49
.              .      .           .
.              .      .           .
50             1     25        625
```

Having got **R** to set up an appropriate model matrix, we *could* use it much as we used the single covariate x in the Hubble data example...

```
> stop.mod0 <- lm(dist~X-1,data=cars)
> summary(stop.mod0)
[edited]
Coefficients:
              Estimate Std. Error t value Pr(>|t|)
X(Intercept)   2.47014    14.81716   0.167   0.868
Xspeed          0.91329     2.03422   0.449   0.656
XI(speed^2)     0.09996     0.06597   1.515   0.136
```

```
Residual standard error: 15.18 on 47 degrees of freedom
Multiple R-Squared: 0.9133,    Adjusted R-squared: 0.9078
F-statistic: 165.1 on 3 and 47 DF,  p-value: < 2.2e-16
```

Notice the `Coefficients` section of the table, in particular. It now has a row for each model coefficient, with each row having much the same interpretation as we have already covered for the single parameter case.

Actually, we would not usually form a model matrix explicitly, and then use `lm` in the way just done. Instead we would supply the more elaborate model formula directly to `lm`, from which it can create the appropriate model matrix automatically. For example:

```
> stop.model <- lm(dist~speed+I(speed^2),data=cars)
> summary(stop.model)
[edited]
Coefficients:
              Estimate Std. Error t value Pr(>|t|)
(Intercept)   2.47014    14.81716   0.167   0.868
speed          0.91329     2.03422   0.449   0.656
I(speed^2)     0.09996     0.06597   1.515   0.136
```

```
Residual standard error: 15.18 on 47 degrees of freedom
Multiple R-Squared: 0.6673,    Adjusted R-squared: 0.6532
F-statistic: 47.14 on 2 and 47 DF,  p-value: 5.852e-12
```

The results are, of course, identical to those produced by the more long winded approach. Looking at the summary information in more detail, all the p-values for testing the single parameter hypotheses, $H_0 : \beta_j = 0$, are very high. Does this mean that we can accept all such hypotheses, and conclude that all the parameter values could really be 0? Absolutely not! Each test is only valid if the other parameters are allowed to take non-zero values: this is because the parameter estimators are not independent and so the p-values can not be either. It is possible to test that all parameters except the intercept are simultaneously zero, and this is the test that the final p-value on the last line of the summary relates to: clearly such a hypothesis has no support from the data.

The high p-values in the summary do suggest that *something* could be removed from our model, and a sensible approach is to try removing the term with the highest p-value. This is the intercept, and on the physical grounds, given in the model motivation, we might expect the intercept to be zero.

```

> stop.model2 <- lm(dist~speed+I(speed^2)-1,data=cars)
> summary(stop.model2)
[edited]
Coefficients:
            Estimate Std. Error t value Pr(>|t|)
speed      1.23903    0.55997   2.213  0.03171 *
I(speed^2)  0.09014    0.02939   3.067  0.00355 **
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 15.02 on 48 degrees of freedom
Multiple R-Squared:  0.9133,    Adjusted R-squared:  0.9097
F-statistic: 252.8 on 2 and 48 DF,  p-value: < 2.2e-16

```

Now both β_1 and β_2 have much lower associated p-values: there are no grounds for dropping any more model terms. We should, of course, check residual plots for this model, exactly as was done for the Hubble data, but for the moment we will move on.

3.2.2 A model with factors

The data frame `PlantGrowth` contains data on plant weights after growth under 3 alternative experimental treatments. The data frame contains columns `weight` and `group`, the latter indicating which treatment group the weight measurement corresponds to. The groups are `ctrl`, `trt1` and `trt2`.

```

> PlantGrowth
  weight group
1   4.17  ctrl
2   5.58  ctrl
.      .    .
.      .    .
11  4.81 trt1
.      .    .
.      .    .
30  5.26 trt2

```

An appropriate model for these data is

$$\mathbb{E}(\text{weight}_i) = \beta_j \text{ if observation } i \text{ is in group } j,$$

where groups 1, 2 and 3 are `ctrl`, `trt1` and `trt2`, respectively. We have already seen how such a model can be made into a linear model, using a design matrix of zeros and ones, but how can such models be set up in R?

Within R, it is possible to give a variable a special class, `factor`, which indicates that its values are to be taken as identifiers of groups, and that it should be handled specially when used to set up a model matrix. `group` is one of these `factor` variables, as the following shows.

```

> class(PlantGrowth$group)
[1] "factor"
> PlantGrowth$group
[1] ctrl ctrl ctrl ctrl ctrl ctrl ctrl ctrl ctrl trt1 trt1 trt1 trt1 trt1
[16] trt1 trt1 trt1 trt1 trt1 trt1 trt2 trt2 trt2 trt2 trt2 trt2 trt2 trt2 trt2
Levels: ctrl trt1 trt2

```

Here is how to produce the appropriate model matrix:

```

> X <- model.matrix(~group-1,data=PlantGrowth)

```

```
> X
      groupctrl grouptrt1 grouptrt2
1          1          0          0
2          1          0          0
.
10         1          0          0
11         0          1          0
.
20         0          1          0
21         0          0          1
.
30         0          0          1
```

Again, we could use `X` directly in a call to `lm`, but it is usually preferable to have `lm` set up the model matrix internally, and simply report the results of the subsequent fitting.

```
> plant.mod <- lm(weight~group-1, data=PlantGrowth)
> summary(plant.mod)
[edited]
Coefficients:
              Estimate Std. Error t value Pr(>|t|)
groupctrl      5.0320      0.1971  25.53  <2e-16 ***
grouptrt1      4.6610      0.1971  23.64  <2e-16 ***
grouptrt2      5.5260      0.1971  28.03  <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.6234 on 27 degrees of freedom
Multiple R-Squared:  0.9867,    Adjusted R-squared:  0.9852
F-statistic: 665.5 on 3 and 27 DF,  p-value: < 2.2e-16
```

As you can see, the output is much as before. However, one difference is that, for models involving factor variables, it is often difficult to interpret the p-values for individual coefficients in any meaningful way. Really we would want to test the hypothesis that the whole factor has no effect, versus the alternative that it does, rather than getting hung up on individual parameters associated with the factor. To understand how to do this we will need some more theory for linear models.

4 Linear model theory

As in the simple one parameter case, point estimates of the linear model parameters, β , can be obtained by the method of least squares; that is, by minimising the residual sum of squares

$$S = \sum_{i=1}^n \epsilon_i^2 = \sum_{i=1}^n (y_i - \mu_i)^2,$$

with respect to β , where $\mu = \mathbf{X}\beta$. Fitting a model so that we minimise the squared error terms, or equivalently try to make the μ_i as close as possible to the y_i , has intuitive appeal. From now on the random vector, \mathbf{y} , and its observed value will not be distinguished notationally, the context making clear which is meant.

To use least squares with a linear model, written in general matrix-vector form, first recall the link between the Euclidean length of a vector and the sum of squares of its elements. If \mathbf{v} is any vector of dimension, n , then $\|\mathbf{v}\|^2 \equiv \mathbf{v}^T \mathbf{v} \equiv \sum_{i=1}^n v_i^2$. Hence

$$S = \|\mathbf{y} - \mu\|^2 = \|\mathbf{y} - \mathbf{X}\beta\|^2.$$

Since \mathcal{S} is simply the squared (Euclidian) length of the vector $\mathbf{y} - \mathbf{X}\beta$, its value will be unchanged if $\mathbf{y} - \mathbf{X}\beta$ is rotated or reflected. This observation is the basis for a practical method for finding $\hat{\beta}$ and for developing the distributional results required to use linear models.

Specifically, as with any real matrix, \mathbf{X} can always be decomposed

$$\mathbf{X} = \mathbf{Q} \begin{pmatrix} \mathbf{R} \\ \mathbf{0} \end{pmatrix} = \mathbf{Q}\mathbf{R}, \quad (5)$$

where \mathbf{R} is a $p \times p$ upper triangular matrix,[†] and \mathbf{Q} is an $n \times n$ orthogonal matrix, the first p columns of which form \mathbf{Q} . Recall that orthogonal matrices rotate/reflect vectors, but do not change their length. Orthogonality also means that $\mathbf{Q}\mathbf{Q}^\top = \mathbf{Q}^\top\mathbf{Q} = \mathbf{I}_n$. Multiplying $\mathbf{y} - \mathbf{X}\beta$ by \mathbf{Q}^\top implies that

$$\|\mathbf{y} - \mathbf{X}\beta\|^2 = \|\mathbf{Q}^\top\mathbf{y} - \mathbf{Q}^\top\mathbf{X}\beta\|^2 = \left\| \mathbf{Q}^\top\mathbf{y} - \begin{pmatrix} \mathbf{R} \\ \mathbf{0} \end{pmatrix} \beta \right\|^2.$$

Defining p vector \mathbf{f} and $n - p$ vector \mathbf{r} so that $\begin{pmatrix} \mathbf{f} \\ \mathbf{r} \end{pmatrix} \equiv \mathbf{Q}^\top\mathbf{y}$, yields[‡]

$$\|\mathbf{y} - \mathbf{X}\beta\|^2 = \left\| \begin{pmatrix} \mathbf{f} \\ \mathbf{r} \end{pmatrix} - \begin{pmatrix} \mathbf{R} \\ \mathbf{0} \end{pmatrix} \beta \right\|^2 = \|\mathbf{f} - \mathbf{R}\beta\|^2 + \|\mathbf{r}\|^2.$$

The length of \mathbf{r} does not depend on β and $\|\mathbf{f} - \mathbf{R}\beta\|^2$ can be reduced to zero by choosing β so that $\mathbf{R}\beta$ equals \mathbf{f} . Hence, provided that \mathbf{X} and therefore \mathbf{R} have full column rank,

$$\hat{\beta} = \mathbf{R}^{-1}\mathbf{f} \quad (6)$$

is the least squares estimator of β . Notice that $\|\mathbf{r}\|^2 = \|\mathbf{y} - \mathbf{X}\hat{\beta}\|^2$, the *residual sum of squares* for the model fit.

4.1 Unbiasedness and variance

It is easy to show that $\hat{\beta}$ is unbiased (has expectation equal to the true β).

$$\mathbb{E}(\hat{\beta}) = \mathbb{E}(\mathbf{R}^{-1}\mathbf{Q}^\top\mathbf{y}) = \mathbf{R}^{-1}\mathbf{Q}^\top\mathbb{E}(\mathbf{y}) = \mathbf{R}^{-1}\mathbf{Q}^\top\mathbf{X}\beta = \mathbf{R}^{-1}\mathbf{Q}^\top\mathbf{Q}\mathbf{R}\beta = \beta.$$

The covariance matrix for $\hat{\beta}$ is also straightforward. The covariance matrix of \mathbf{y} is $\mathbf{I}\sigma^2$ and so the covariance matrix of $\mathbf{f} = \mathbf{Q}^\top\mathbf{y}$ is $\mathbf{Q}^\top\mathbf{Q}\sigma^2 = \mathbf{I}\sigma^2$. Hence the covariance matrix of $\hat{\beta} = \mathbf{R}^{-1}\mathbf{f}$ is

$$\mathbf{V}_{\hat{\beta}} = \mathbf{R}^{-1}\mathbf{R}^{-\top}\sigma^2. \quad (7)$$

4.2 Checking

Further inference, beyond simply estimating β , will require that the assumptions of independence and constant variance of the ϵ_i hold. This means that we need some means for checking these assumptions, if our inference is to be sound. Formally we will also assume normality of the ϵ_i , but in reality there is a certain robustness to this assumption, as a result of the central limit theorem.

The most useful checks of model assumptions are often graphical, as these tend to indicate not just that an assumption is violated, but also *how* it is violated. This in turn may allow us to fix the problem. Once we have estimated a linear model, all the information about model fit is contained in the model *residuals*

$$\hat{\epsilon}_i = y_i - \hat{\mu}_i$$

(where $\hat{\mu} = \mathbf{X}\hat{\beta}$, of course). To check the model assumptions the $\hat{\epsilon}_i$ are usually plotted against the model predictor variables and the *fitted values*, $\hat{\mu}_i$. Systematic pattern in the mean of the residuals will violate the independence assumption, and often results from something wrong with how the dependence on a predictor variable has been specified. Systematic pattern in the variability of the residuals indicates violation of the constant variance assumption. This is most often manifested in the plot of $\hat{\epsilon}_i$ against $\hat{\mu}_i$.

[†]That is, $R_{i,j} = 0$ if $i > j$

[‡]If the final equality is not obvious recall that $\|\mathbf{x}\|^2 = \sum_i x_i^2$, so if $\mathbf{x} = \begin{pmatrix} \mathbf{v} \\ \mathbf{w} \end{pmatrix}$, $\|\mathbf{x}\|^2 = \sum_i v_i^2 + \sum_i w_i^2 = \|\mathbf{v}\|^2 + \|\mathbf{w}\|^2$.

4.3 Further inference results: intervals and testing

Having covered estimation and checking, we can now consider how to answer the inferential questions:

1. What range of parameter values are consistent with the data?
2. Are some pre-specified values (or restrictions) for the parameters consistent with the data?

In the specific case of linear models, we want to find confidence intervals for the β_i and to *test hypotheses* about the β_i . For example, we often want to test the null hypothesis that one or several elements of β are zero, corresponding to terms being omitted from the linear model.

Confidence intervals and hypothesis tests are probabilistic concepts, so at this stage we need to turn our linear model into a fully probabilistic model, by specifying a full distribution for the ϵ_i . We will simply assume that independently $\epsilon_i \sim N(0, \sigma^2)$, which implies that $\mathbf{y} \sim N(\mathbf{X}\beta, \mathbf{I}\sigma^2)$, and hence, upon recycling results from above

$$\hat{\beta} \sim N(\beta, \mathbf{V}_{\hat{\beta}}) \text{ where } \mathbf{V}_{\hat{\beta}} = \mathbf{R}^{-1}\mathbf{R}^{-\top}\sigma^2.$$

This is not yet in the most directly usable form: it gives the distribution of $\hat{\beta}$ in terms of the unknowns β and σ^2 . As we saw in section 2.4.1, if we want to make inferences about β_i it is helpful if we can start from a *pivotal* statistic, which has a distribution with no unknowns.

4.3.1 $(\hat{\beta}_i - \beta_i)/\hat{\sigma}_{\hat{\beta}_i} \sim t_{n-p}$

First, consider the distribution of $\mathbf{Q}^\top \mathbf{y}$. It is a linear transform of a normal random vector, so must also be a normal random vector, and from standard theory on transformation matrices its covariance matrix is

$$\mathbf{V}_{\mathbf{Q}^\top \mathbf{y}} = \mathbf{Q}^\top \mathbf{I} \mathbf{Q} \sigma^2 = \mathbf{I} \sigma^2.$$

Since, for the normal distribution only, zero covariance implies independence, the elements of $\mathbf{Q}^\top \mathbf{y}$ are mutually independent. Furthermore,

$$\begin{aligned} \mathbb{E} \begin{pmatrix} \mathbf{f} \\ \mathbf{r} \end{pmatrix} &= \mathbb{E}(\mathbf{Q}^\top \mathbf{y}) = \mathbf{Q}^\top \mathbf{X} \beta = \begin{pmatrix} \mathbf{R} \\ \mathbf{0} \end{pmatrix} \beta \\ &\Rightarrow \mathbb{E}(\mathbf{f}) = \mathbf{R}\beta \text{ and } \mathbb{E}(\mathbf{r}) = \mathbf{0}. \end{aligned}$$

So we have that

$$\mathbf{f} \sim N(\mathbf{R}\beta, \mathbf{I}_p \sigma^2) \text{ and } \mathbf{r} \sim N(\mathbf{0}, \mathbf{I}_{n-p} \sigma^2)$$

with both vectors independent of each other.

Now, because the $n - p$ elements of \mathbf{r} are i.i.d. $N(0, \sigma^2)$ random variables,

$$\frac{1}{\sigma^2} \|\mathbf{r}\|^2 = \frac{1}{\sigma^2} \sum_{i=1}^{n-p} r_i^2 \sim \chi_{n-p}^2.$$

The mean of a χ_{n-p}^2 r.v. is $n - p$, so this result is sufficient (but not necessary) to imply that

$$\hat{\sigma}^2 = \|\mathbf{r}\|^2 / (n - p) \tag{8}$$

is an unbiased estimator of σ^2 . The independence of the elements of \mathbf{r} and \mathbf{f} also implies that $\hat{\beta}$ and $\hat{\sigma}^2$ are independent.[§]

Now consider a single-parameter estimator, $\hat{\beta}_i$, with standard deviation, $\sigma_{\hat{\beta}_i}$, given by the square root of element i, i of $\mathbf{V}_{\hat{\beta}}$. An unbiased estimator of $\mathbf{V}_{\hat{\beta}}$ is $\hat{\mathbf{V}}_{\hat{\beta}} = \mathbf{V}_{\hat{\beta}} \hat{\sigma}^2 / \sigma^2 = \mathbf{R}^{-1} \mathbf{R}^{-\top} \hat{\sigma}^2$, so an estimator, $\hat{\sigma}_{\hat{\beta}_i}$, is given by the square root of element i, i of $\hat{\mathbf{V}}_{\hat{\beta}}$, and it is clear that $\hat{\sigma}_{\hat{\beta}_i} = \sigma_{\hat{\beta}_i} \hat{\sigma} / \sigma$. Hence,

$$T = \frac{\hat{\beta}_i - \beta_i}{\hat{\sigma}_{\hat{\beta}_i}} = \frac{\hat{\beta}_i - \beta_i}{\sigma_{\hat{\beta}_i} \hat{\sigma} / \sigma} = \frac{(\hat{\beta}_i - \beta_i) / \sigma_{\hat{\beta}_i}}{\sqrt{\frac{1}{\sigma^2} \|\mathbf{r}\|^2 / (n - p)}} \sim \frac{N(0, 1)}{\sqrt{\chi_{n-p}^2 / (n - p)}} \sim t_{n-p} \tag{9}$$

[§]Recall that $\|\mathbf{r}\|^2 = \|\mathbf{y} - \mathbf{X}\hat{\beta}\|^2$.

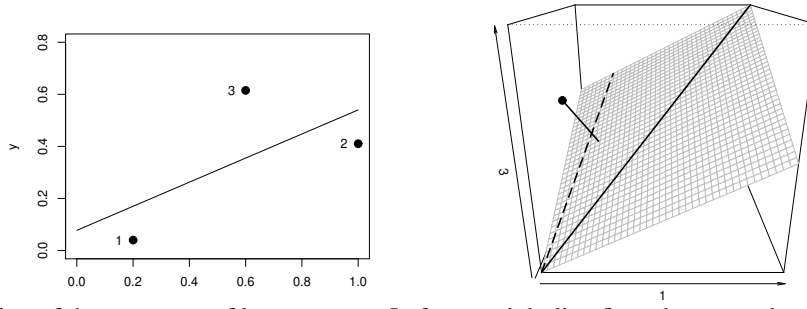


Figure 4: Illustration of the geometry of least squares. Left: a straight line fit to three x, y data. Right: the space in which the y coordinates of the data define a single point, while the columns of the model matrix (solid and dashed line) span the subspace shown in grey. The least squares estimate of $\mathbb{E}(\mathbf{y})$ is the orthogonal projection of the data point onto the model subspace.

(where the independence of $\hat{\beta}_i$ and $\hat{\sigma}^2$ has been used, along with the definition of a t random variable). This result can be used for any β_i exactly as it was used for the single β in sections 2.4.1 and 2.4.2 (with the implied adjustment of degrees of freedom, of course).

4.3.2 Testing $H_0 : \mathbf{C}\beta = \mathbf{d}$

Now consider a more general testing problem. Recall that factor variables serve to classify observations into different groups. In a linear model each factor variable will usually have several associated parameters - in principle, one for each group. If we want to test whether the factor variable should be in the model or not, we therefore need to test whether all the factor's parameters could simultaneously be zero. This is a special case of the general null hypothesis $H_0 : \mathbf{C}\beta = \mathbf{d}$. Note that approaching such a testing problem with a separate test for each parameter of the factor is a bad idea: how would you go about combining the different p-values, for example?

Suppose that \mathbf{C} is a $q \times p$ matrix and \mathbf{d} a q vector, where $q < p$. Under H_0 we have $\mathbf{C}\hat{\beta} - \mathbf{d} \sim N(\mathbf{0}, \mathbf{C}\mathbf{V}_{\hat{\beta}}\mathbf{C}^T)$, from basic properties of the transformation of normal random vectors. Any positive definite matrix can be factored into the crossproduct of a triangular matrix with itself: a *Cholesky decomposition*. Forming the Cholesky decomposition $\mathbf{L}^T\mathbf{L} = \mathbf{C}\mathbf{V}_{\hat{\beta}}\mathbf{C}^T$, it is then easy to show that $\mathbf{L}^{-T}(\mathbf{C}\hat{\beta} - \mathbf{d}) \sim N(\mathbf{0}, \mathbf{I})$, so,

$$(\mathbf{C}\hat{\beta} - \mathbf{d})^T(\mathbf{C}\mathbf{V}_{\hat{\beta}}\mathbf{C}^T)^{-1}(\mathbf{C}\hat{\beta} - \mathbf{d}) = (\mathbf{C}\hat{\beta} - \mathbf{d})^T\mathbf{L}^{-1}\mathbf{L}^{-T}(\mathbf{C}\hat{\beta} - \mathbf{d}) \sim \sum_{i=1}^q N(0, 1)^2 \sim \chi_q^2.$$

As in the previous section, plugging in $\hat{\mathbf{V}}_{\hat{\beta}} = \mathbf{V}_{\hat{\beta}}\hat{\sigma}^2/\sigma^2$ gives the computable test statistic and its distribution under H_0 :

$$\begin{aligned} F &= \frac{1}{q}(\mathbf{C}\hat{\beta} - \mathbf{d})^T(\mathbf{C}\hat{\mathbf{V}}_{\hat{\beta}}\mathbf{C}^T)^{-1}(\mathbf{C}\hat{\beta} - \mathbf{d}) = \frac{\sigma^2}{q\hat{\sigma}^2}(\mathbf{C}\hat{\beta} - \mathbf{d})^T(\mathbf{C}\mathbf{V}_{\hat{\beta}}\mathbf{C}^T)^{-1}(\mathbf{C}\hat{\beta} - \mathbf{d}) \\ &= \frac{(\mathbf{C}\hat{\beta} - \mathbf{d})^T(\mathbf{C}\mathbf{V}_{\hat{\beta}}\mathbf{C}^T)^{-1}(\mathbf{C}\hat{\beta} - \mathbf{d})/q}{\frac{1}{\hat{\sigma}^2}\|\mathbf{r}\|^2/(n-p)} \sim \frac{\chi_q^2/q}{\chi_{n-p}^2/(n-p)} \sim F_{q,n-p}. \end{aligned} \quad (10)$$

This result can be used to compute a p-value for the test.

When our interest is in testing whether several elements of β could simultaneously be zero, then $\mathbf{d} = \mathbf{0}$ while \mathbf{C} is made up of a selection of rows of the $p \times p$ identity matrix. In that case let RSS_1 denote the residual sum of squares for the full model, and RSS_0 denote the residual sum of squares for the reduced model in which the H_0 is true. It can be shown that the above test statistic then reduces to

$$F = \frac{(\text{RSS}_0 - \text{RSS}_1)/q}{\text{RSS}_1/(n-p)}.$$

4.4 The geometry of linear models

Least squares estimation of linear models amounts to finding the orthogonal projection of the n dimensional response data \mathbf{y} onto the p dimensional linear subspace spanned by the columns of \mathbf{X} . The linear model states that

$\mathbb{E}(\mathbf{y})$ lies in the space spanned by all possible linear combinations of the columns of the $n \times p$ model matrix \mathbf{X} , and least squares seeks the point in that space that is closest to \mathbf{y} in Euclidean distance. Figure 4 illustrates this geometry for the model,

$$\begin{pmatrix} .05 \\ .40 \\ .65 \end{pmatrix} = \begin{pmatrix} 1 & .1 \\ 1 & 1 \\ 1 & .6 \end{pmatrix} \begin{pmatrix} \beta_0 \\ \beta_1 \end{pmatrix} + \begin{pmatrix} \epsilon_1 \\ \epsilon_2 \\ \epsilon_3 \end{pmatrix}.$$

Given this view of least squares fitting as orthogonal projection it is interesting to look at the projection matrix that maps the data \mathbf{y} to the fitted values $\hat{\boldsymbol{\mu}}$. We have

$$\hat{\boldsymbol{\mu}} = \mathbf{X}\hat{\boldsymbol{\beta}} = \mathbf{Q}\mathbf{R}\mathbf{R}^{-1}\mathbf{Q}^T\mathbf{y} = \mathbf{Q}\mathbf{Q}^T\mathbf{y}.$$

So the required projection matrix is $\mathbf{A} = \mathbf{Q}\mathbf{Q}^T$, which is often known as the *influence matrix*, or *hat matrix* of the linear model. (Don't forget that \mathbf{Q} is an $n \times p$ matrix with orthogonal columns, but non-orthogonal rows, so that $\mathbf{Q}^T\mathbf{Q} = \mathbf{I}$ but $\mathbf{Q}\mathbf{Q}^T \neq \mathbf{I}$.) One interesting property of \mathbf{A} is that it is idempotent: $\mathbf{A}\mathbf{A} = \mathbf{A}$. In a way this is obvious, as the orthogonal projection of $\hat{\boldsymbol{\mu}}$ onto the column space of \mathbf{X} must be $\hat{\boldsymbol{\mu}}$.

4.5 Results in terms of \mathbf{X}

The presentation so far has been in terms of the QR based method actually used to fit linear models in practice. By taking this approach, results (9) and (10) can be derived relatively easily. However, for historical reasons, these results are more usually presented in terms of the model matrix, \mathbf{X} , rather than the components of its QR decomposition.

For example the covariance matrix of $\hat{\boldsymbol{\beta}}$ becomes $(\mathbf{X}^T\mathbf{X})^{-1}\sigma^2$, which is easily seen to be equivalent to (7):

$$\begin{aligned} \mathbf{V}_{\hat{\boldsymbol{\beta}}} &= (\mathbf{X}^T\mathbf{X})^{-1}\sigma^2 = (\mathbf{R}^T\mathbf{Q}^T\mathbf{Q}\mathbf{R})^{-1}\sigma^2 = (\mathbf{R}^T\mathbf{R})^{-1}\sigma^2 \\ &= \mathbf{R}^{-1}\mathbf{R}^{-T}\sigma^2. \end{aligned}$$

The expression for the least squares estimates is $\hat{\boldsymbol{\beta}} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y}$, which is equivalent to (6):

$$\hat{\boldsymbol{\beta}} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y} = \mathbf{R}^{-1}\mathbf{R}^{-T}\mathbf{R}^T\mathbf{Q}^T\mathbf{y} = \mathbf{R}^{-1}\mathbf{Q}^T\mathbf{y} = \mathbf{R}^{-1}\mathbf{f}.$$

It follows that the influence/hat matrix can be written as $\mathbf{A} = \mathbf{X}(\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T$. These results are of theoretical interest, but should not usually be used for computational purposes.

5 Linear models in R

Having covered most of the theory needed to understand linear modelling, it's time to focus on application of that theory, so this section will cover the use of linear models in more detail.

5.1 The `cars` data again

We'll use the `cars` data again in order to relate the theory covered in section 4 to practical analysis with R in the context of a now familiar example.

5.1.1 Confidence intervals and hypothesis tests for single parameters

This section will use the result of section 4.3.1, to find confidence intervals for, and test hypotheses about, model parameters. First refit the full quadratic model:

$$\text{dist}_i = \beta_1 + \beta_2\text{speed}_i + \beta_3\text{speed}_i^2 + \epsilon_i, \quad \epsilon_i \text{ i.i.d. } N(0, \sigma^2)$$

to the data in the `cars` data frame in R.

```
cm <- lm(dist~speed+I(speed^2), data=cars)
```

From the resulting fitted model object, `cm`, it is possible to extract all the quantities needed to make the T statistics, covered in section 4.3.1. The following extracts, $\hat{\beta}$, $\hat{V}_{\hat{\beta}}$ and the $\hat{\sigma}_{\hat{\beta}_i}$'s respectively:

```
beta.hat <- coef(cm)
V.beta <- vcov(cm)
sigma.beta <- sqrt(diag(V.beta))
```

Confidence intervals

Now, using the result from section 4.3.1, exactly as it was used in sections 2.4.1 and 2.4.2, we can find confidence intervals for the β_j 's. Let's start by finding a 90% CI for β_2 . This will involve using the `qt` function to evaluate the value above which 5% of t_{47} distributed random variables lie (a further 5% lying below the negative of that value, by symmetry of the distribution). For example

```
> qt(.95, df=47)
[1] 1.677927
```

implies that a t_{47} r.v. has a probability 0.05 of being greater than 1.678. 47 is actually the correct number of degrees of freedom to use in the current case, since there are 50 observed distances (you can use `nrow(cars)` to double check this) and the model has 3 parameters. So the following evaluates the required interval

```
> beta.hat[2] + qt(.95, df=47)*sigma.beta[2]
      speed
4.326560
> beta.hat[2] - qt(.95, df=47)*sigma.beta[2]
      speed
-2.499985
```

i.e. a 90% CI for β_2 is $(-2.5, 4.33)$.

Actually, it is possible to calculate intervals for all 3 parameters at the same time. The following finds 95% CI's for all the parameters

```
> beta.hat + qt(.975, df=47)*sigma.beta
(Intercept)      speed      I(speed^2)
 32.2784284    5.0056113    0.2326702
> beta.hat - qt(.975, df=47)*sigma.beta
(Intercept)      speed      I(speed^2)
-27.33815279   -3.17903606   -0.03275162
```

So, for example, a 95% CI for β_3 is $(-0.033, 0.233)$.

Hypothesis tests

The result from section 4.3.1 is also useful for testing hypotheses about the values of the β_j . For example, consider testing $H_0 : \beta_2 = 0$ vs. $H_1 : \beta_2 \neq 0$. To test this we first form the t-statistic, $T = \hat{\beta}_2 / \hat{\sigma}_{\hat{\beta}_2}$,

```
> T <- beta.hat[2]/sigma.beta[2]
> T
      speed
0.448962
```

If H_0 is true then this should be an observation of a t_{47} r.v., but if H_1 is true then it should have too large a magnitude for this (basically because $\hat{\beta}_2$ won't then be 'close' to the hypothesized value, zero). As usual we quantify the evidence for or against H_0 using a p-value. This is the probability of a t_{47} r.v. having a value at least as favourable to H_1 as the T actually observed. i.e. $\Pr[t_{47} > .449 \text{ or } t_{47} < -.449] = \Pr[|t_{47}| > .449]$. This is easily evaluated in R.

```
> p.val <- 2*pt(abs(T), df=47, lower.tail=FALSE)
> p.val
      speed
0.6555224
```

(The `lower.tail=FALSE` argument causes `pt` to return $\Pr[t_{47} > .449]$ rather than $\Pr[t_{47} < .449]$.) A p-value of 0.66 suggests that the data are quite compatible with H_0 , i.e. provided that the model contains β_1 and β_3 terms, β_2 seems to be superfluous. However it might be that one of β_1 and β_3 is even more superfluous, so we should test whether either of those might be zero, before dropping any terms.

In fact we can work out equivalent p-values for all 3 parameters simultaneously:

```
> T <- beta.hat/sigma.beta
> p.val <- 2*pt(abs(T), df=47, lower.tail=F)
> beta.hat; sigma.beta; T; p.val
(Intercept)      speed      I(speed^2)
 2.4701378    0.9132876    0.0999593 ## parameter estimates
14.81716473    2.03422044    0.06596821 ## estimator standard dev.
 0.1667079    0.4489620    1.5152647 ## T statistics
 0.8683151    0.6555224    0.1364024 ## p-values
```

Actually, p-values for testing whether parameters might be zero are very often useful, so R produces them by default in the summary of a fitted linear model.

```
> summary(cm)
...
Coefficients:
              Estimate Std. Error t value Pr(>|t|)
(Intercept)   2.47014    14.81716   0.167    0.868
speed          0.91329     2.03422   0.449    0.656
I(speed^2)     0.09996     0.06597   1.515    0.136
...
```

Notice how the columns of the `Coefficients` table contain exactly the values that we have just calculated. These single parameter tests are often used to decide whether to drop a term from the model: usually the term with the highest p-value greater than some threshold, such as 0.05, would be dropped (provided that there are no prior grounds for insisting on its retention in the model). The model is refitted without the term, and the new p-values re-examined to see if any further simplification is possible. This is repeated until all terms have p-values lower than, say, 0.05.

Different hypotheses and model selection

Sometimes other types of hypotheses may be tested. For example, it is hard to think of a decent physical interpretation for a negative β_1 . Hence it might be more appropriate to test $H_0 : \beta_1 = 0$ vs. $H_1 : \beta_1 > 0$. In this case *the probability of a t_{47} being at least as favourable to H_1 as the T actually observed* is $\Pr[t_{47} > 14.82]$: i.e. large magnitude negative values of T now count for H_0 , so only one tail of the t_{47} distribution features in the p-value calculation. Here is the p-value calculation in R

```
> T <- beta.hat[1]/sigma.beta[1]
> p.val <- pt(T, df=47, lower.tail=F)
> p.val
(Intercept)
 0.4341575
```

Again, no reason to reject the null (provided the other terms are in the model), but notice that the p-value is halved by the change in alternative.

Another use of the section 4.3.1 result is to test whether β_j might have some hypothesized value other than zero. For example we might be able to calculate what β_3 ought to be, based on the frictional properties of the brakes

and tyres: suppose this calculation says that $\beta_3 = 0.2$. We can test this theoretical value, by testing $H_0 : \beta_3 = 0.2$ vs. $H_1 : \beta_3 \neq 0.2$, as follows.

```
> T <- (beta.hat[3]-.2)/sigma.beta[3]
> p.val <- 2*pt(abs(T),df=47,lower.tail=F)
> p.val
I(speed^2)
0.136091
```

So not much evidence with which to reject H_0 . However, this test provides a very good example of why it is often a good idea to base statistical conclusions on the simplest model consistent with the data. If we repeat the test, based on a model without an intercept, then things are rather different.

```
> cm1 <- lm(dist~speed+I(speed^2)-1,data=cars)
> beta.hat <- coef(cm1)
> V.beta <- vcov(cm1)
> sigma.beta <- sqrt(diag(V.beta))
> T <- (beta.hat[2]-.2)/sigma.beta[2]
> p.val <- 2*pt(abs(T),df=48,lower.tail=F)
> p.val
I(speed^2)
0.0004934814
```

The p-value has dropped enormously, and we now have firm evidence to reject $H_0 : \beta_3 = 0.2$. What has happened here, is that the parameter estimators are so correlated in the full model, that all the estimators have very high variance. As a result we have a hard time rejecting a null hypothesis even if it quite a long way from the truth. The redundant parameters cause a loss of precision, and consequent loss of *power* when testing. The more parsimonious model gives much better precision, and higher power when testing.

Note however, that there is a price to pay for *selecting* a model statistically and then performing hypothesis tests as if there were no question about the truth of the selected model. By doing this we fail to account for our initial uncertainty about which model was right when performing the hypothesis tests. This tends to mean that our p-values are a bit lower than if we had properly accounted for all uncertainty: i.e. the p-values are really only approximate now.

5.1.2 F-ratio tests about several parameters

This section illustrates the use of the F-ratio result from section 4.3.2, with R. Writing the cars model as

$$\text{dist}_i = \mu_i + \epsilon_i \text{ where } \epsilon_i \text{ i.i.d } N(0, \sigma^2),$$

suppose that we had some prior reason to want to test:

$$H_0 : \mu_i = \text{speed}_i^2 \beta \text{ against } H_0 : \mu_i = \beta_1 + \text{speed}_i \beta_2 + \text{speed}_i^2 \beta_3$$

(where the null is equivalent to $H_0 : \beta_1 = \beta_2 = 0$). This is exactly the situation in which the results of section 4.3.2 are useful. Recall that the result is:

$$F = \frac{(\text{RSS}_0 - \text{RSS}_1)/(p_1 - p_0)}{\hat{\sigma}^2} \sim F_{q,n-p} \text{ if } H_0 \text{ is true.}$$

$p_1 - p_0$ is the difference in degrees of freedom between the models (difference in number of parameters); $\hat{\sigma}^2 = \text{RSS}_1/(n - p_1)$ is the residual variance estimator, from the alternative (larger) model. Fitting the null model, it is straightforward to calculate this F ratio statistic:

```
> cm0 <- lm(dist~I(speed^2)-1,data=cars)
> rss0 <- sum(residuals(cm0)^2)
> rss1 <- sum(residuals(cm)^2)
> F <- ((rss0-rss1)/2)/(rss1/(47))
> F
[1] 2.412267
```

From the result of section 4.3.2 this ought to be an observation on an $F_{2,47}$ random variable, if H_0 is true. Otherwise it should be improbably large for an $F_{2,47}$ r.v. As usual we quantify the compatibility of the test statistic (and hence the data) with the stated distribution by calculating a p-value. That is we work out the probability that an $F_{2,47}$ r.v. would be at least as favourable to H_1 as the observed $F = 2.41$, if H_0 is true. So we seek $\Pr[F_{2,47} \geq 2.41]$, which is easily evaluated using the `pf` function.

```
> p.val <- pf(F, 2, 47, lower.tail=FALSE)
> p.val
[1] 0.1006278 ## p-value
```

...the data provide little evidence to reject H_0 , in this case.

While the above calculations are useful for linking the theory in section 4.3.2 to what is actually done to use the result for a model and data, there is usually no need to ‘spell out’ the calculations like this. **R** has built in functions to automate the process. For example the test just conducted can be done using the `anova` function. . .

```
> anova(cm0, cm)
Analysis of Variance Table

Model 1: dist ~ I(speed^2) - 1
Model 2: dist ~ speed + I(speed^2)
  Res.Df    RSS Df Sum of Sq    F Pr(>F)
1      49 11935.9
2      47 10824.7  2    1111.2  2.4123 0.1006
```

The quantities in this *ANOVA table* exactly match the quantities used in the previous, long winded, calculation.

It is also possible to compare a longer sequence of models using `anova`. Each pair in the sequence is compared in exactly the same way as two models would be compared, except that the residual variance estimate in the denominator of each F-ratio statistic is always taken from the ‘largest’ model (i.e. the one with most parameters).

Finally, the following is possible:

```
> anova(cm)
Analysis of Variance Table

Response: dist
      Df  Sum Sq Mean Sq F value    Pr(>F)
speed    1  21185.5  21185.5   91.986 1.211e-12 ***
I(speed^2) 1    528.8    528.8    2.296   0.1364
Residuals 47  10824.7    230.3
```

This single argument call causes `anova` to successively strip terms from your model, at each stage testing the null hypothesis that the current version of the model is correct, against the alternative that the previous version was right. F-ratio tests are used each time, but again $\hat{\sigma}^2$ according to the original model is always used as the denominator of the F ratio statistic. The tables have to be read from bottom row up. So, the `I(speed^2)` row compares models with and without the dependence on speed^2 , and suggests that we could drop it. Turning to the `speed` row, this is about comparing a model with intercept and linear dependence on `speed` to a model with only an intercept: it suggests that we should hang on to the `speed` term.

Actually this single argument use of `anova` only really makes sense when examining models for balanced data from designed experiments. In other cases the order in which the model is automatically reduced usually makes little sense, and the p-values are only useful up to the first p-value that is small enough that we would reject the smaller model in the pair being compared. As we have already seen, if we reduce the `cars` model in a sensible order, we would drop the intercept term, and then nothing else.

5.1.3 Checking the model assumptions: residuals

In the previous few sections models for the `cars` data have been used in order to test hypotheses and find confidence intervals, but we have not actually checked that the model assumptions are reasonable. If the assumptions

are not reasonable, then we have been wasting our time! For this reason it is usually important to check model assumptions immediately after fitting models, and before doing anything that relies on the assumptions being reasonable. The only reason that this was not done above, was in order to ensure that illustration of the practicalities of using the results of sections 4.3.1 and 4.3.2 followed as soon as possible after those sections.

Model checking is always based on looking at *residuals*. Recall that the residuals, $\hat{\epsilon}_i$, are simply the response data, y_i , minus the fitted values, $\hat{\mu}_i$, i.e.

$$\hat{\epsilon} = \mathbf{y} - \hat{\boldsymbol{\mu}} \text{ where } \hat{\boldsymbol{\mu}} = \mathbf{X}\hat{\boldsymbol{\beta}}.$$

The residuals contain all the information left in the data after fitting the model. If the model is adequate then they should behave like observations on the ϵ_i r.v.s. In other words the ϵ_i should look like a collection of independent observations on an $N(0, \sigma^2)$ r.v. Actually this is only approximately true: using the influence matrix, \mathbf{A} , from section 4.4 we have $\hat{\epsilon} = (\mathbf{I} - \mathbf{A})\mathbf{y}$. So by standard transformation of covariance matrices and the idempotence of \mathbf{A} , the covariance matrix of the residuals is $(\mathbf{I} - \mathbf{A})\sigma^2$. Neither exactly independent nor with exactly constant variance. We can't do anything about the slight lack of independence, but the residuals can be standardized to have constant variance, by defining

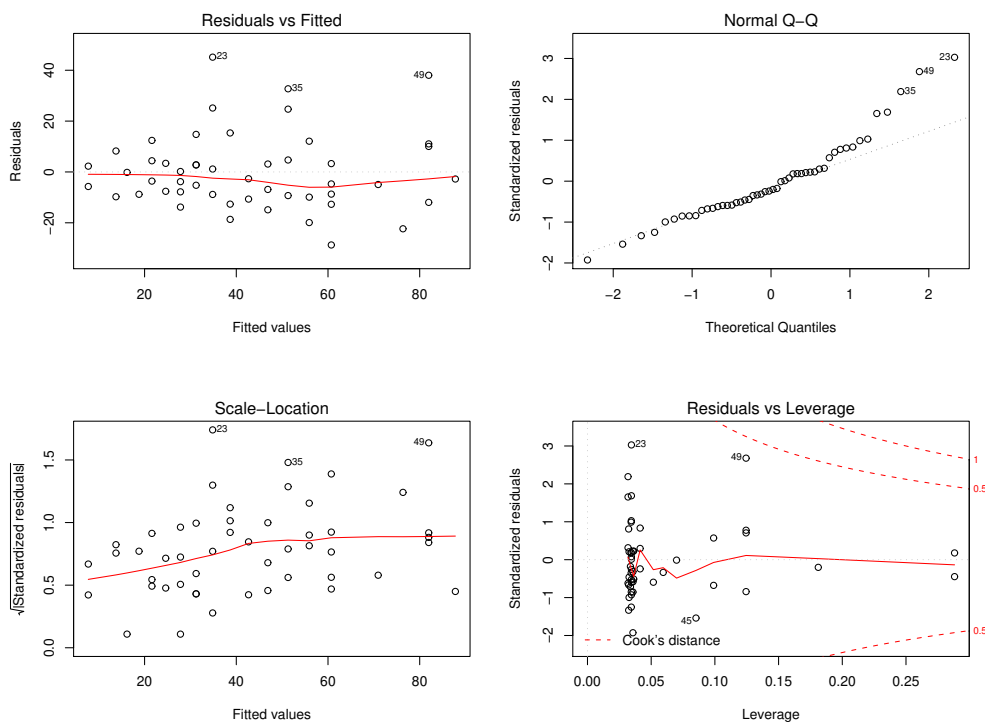
$$\tilde{\epsilon}_i = \hat{\epsilon}_i / \sqrt{1 - A_{ii}}$$

The way to check residuals is to examine a variety of diagnostic plots, always looking for evidence of any violation of the linear modelling assumptions:

1. The ϵ_i are independent.
2. The ϵ_i have constant variance.
3. The ϵ_i follow a normal distribution.

Because of the central limit theorem, the first two of these are more important than the third.

The `plot` function will provide a number of useful residual plots if passed a fitted linear model object. For example, `par(mfrow=c(2,2)); plot(cm)` results in...



- The upper left plot shows the model residuals, $\hat{\epsilon}_i$, against the model fitted values, $\hat{\mu}_i$, where $\hat{\mu} = \mathbf{X}\hat{\beta}$ and $\hat{\epsilon} = \mathbf{y} - \hat{\mu}$. The residuals should be evenly scattered above and below zero (the distribution of fitted values is not of interest). A trend in the mean of the residuals would violate the assumption of independent response variables, and usually results from an erroneous model structure: e.g. assuming a linear relationship with a predictor, when a quadratic is required, or omitting an important predictor variable. A trend in the variability of the residuals suggests that the variance of the response is related to its mean, violating the constant variance assumption: transformation of the response or use of a GLM may help, in such cases. The plot shown does not indicate any problem.
- The lower left plot is a scale-location plot. The raw residuals are standardized by dividing by their estimated standard deviation, $\hat{\sigma}\sqrt{1 - A_{ii}}$ (\mathbf{A} is the influence matrix). The square root of the absolute value of each standardized residual is then plotted against the equivalent fitted value. It can be easier to judge the constant variance assumption from such a plot, and the square root transformation reduces the skew in the distribution, which would otherwise be likely to occur. There is some suggestion of variance increasing with mean in this plot, but the effect is not very large, so is unlikely to cause a big problem.
- The upper right panel is a normal Q-Q (quantile-quantile) plot. The standardized residuals are sorted and then plotted against the quantiles of a standard normal distribution. If the residuals are normally distributed then the resulting plot should look like a straight line relationship, perturbed by some random scatter. In this case the few large residuals show signs of being a bit too big to have come from the same normal distribution as the others, but again, the effect is not large relative to what can occur by chance even when the model assumptions are fine.
- The lower right panel plots the standardized residuals against the *leverage* of each datum. The leverage is simply A_{ii} , which measures the *potential* for particular datum to influence the overall model fit. The combination of a large residual and high leverage implies that the corresponding datum has a substantial influence on the overall fit. Large leverage with a small residual generally implies that, although the datum would exert undue influence on the fit, if out of line with the other data, it is actually consistent with the other data, and does not have too great an influence. Similarly a large residual may not have much influence on the fit, if the corresponding datum has low leverage.

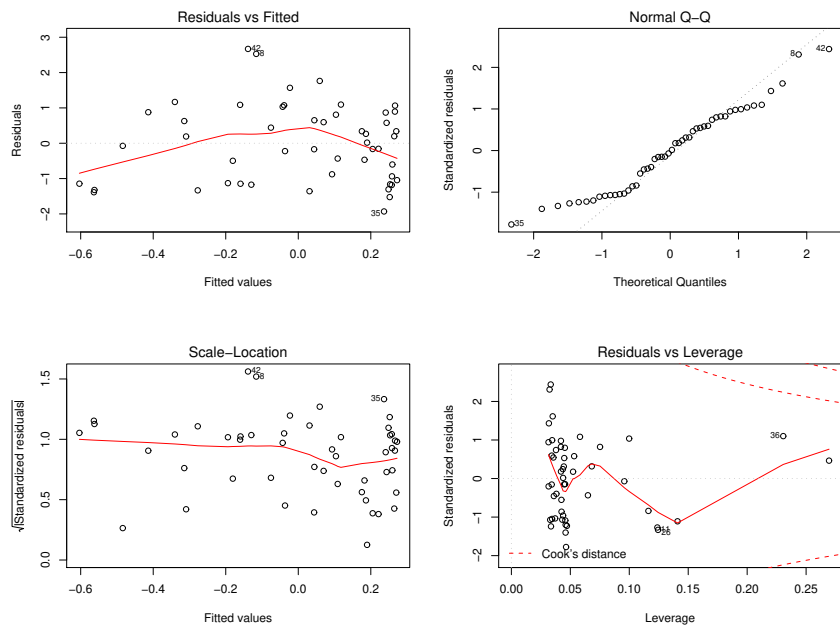
A good summary of how much influence each datum actually has is provided by its *Cook's distance*. Cook's distance is a measure of how much influence each observation has on the fitted model. If $\hat{\mu}_i^{[k]}$ is the i^{th} fitted value, when the k^{th} datum is omitted from the fit, then Cook's distance is

$$d_k = \frac{1}{(p+1)\hat{\sigma}^2} \sum_{i=1}^n (\hat{\mu}_i^{[k]} - \hat{\mu}_i)^2, \quad (11)$$

where p is the number of parameters and n the number of data. A very large value of d_k indicates a point that has a substantial influence on the model results. If the Cook's distance values indicate that model estimates may be very sensitive to just one or two data, then it is usually prudent to repeat any analysis without the offending points, in order to check the robustness of the modelling conclusions. Cook's distance can be shown to be a function of leverage and standardized residual, so contours of Cook's distance are shown on the plot, from which the Cook's distance for any datum can be read. In this case none of the points look wildly out of line.

It is important to realize that residual plots are quite variable, even when all the model assumptions are met. To get a feel for this it can be helpful to simulate some residuals for cases where the assumptions are fine...

```
set.seed(18)           ## just ensures exact repeatability
y <- rnorm(50)          ## simulate a normal response
x <- runif(50)           ## and a predictor
sm <- lm(y~x+I(x^2))    ## fit a quadratic to data
par(mfrow=c(2,2))
plot(sm)                ## plot residuals
```



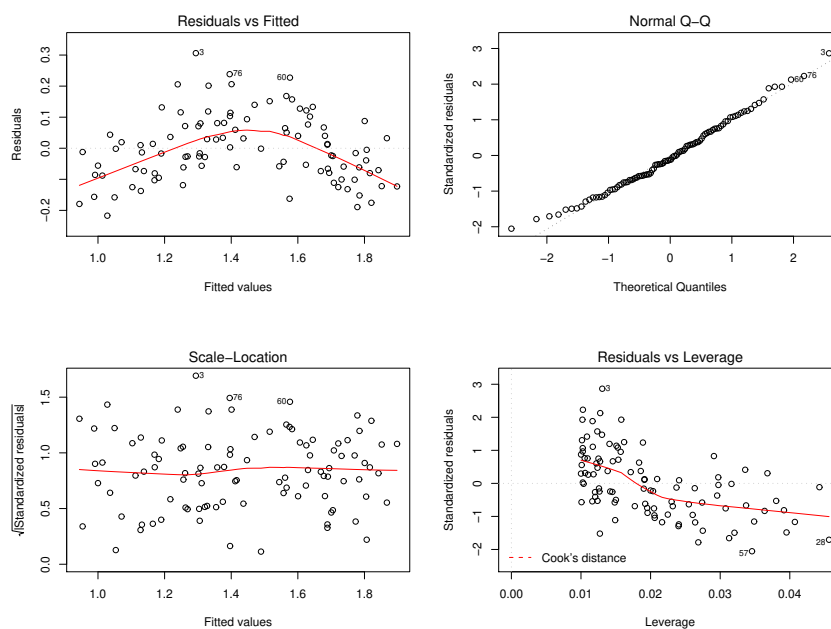
Repeating this simulation exercise a few times (without the `set.seed` line, of course), will give you a feel for the degree of variability to expect when the model assumptions are ok. Note that there is a sample size dependence in this variability!

5.1.4 Residual plots: common problems

When looking at residual plots, the most important things to check for are

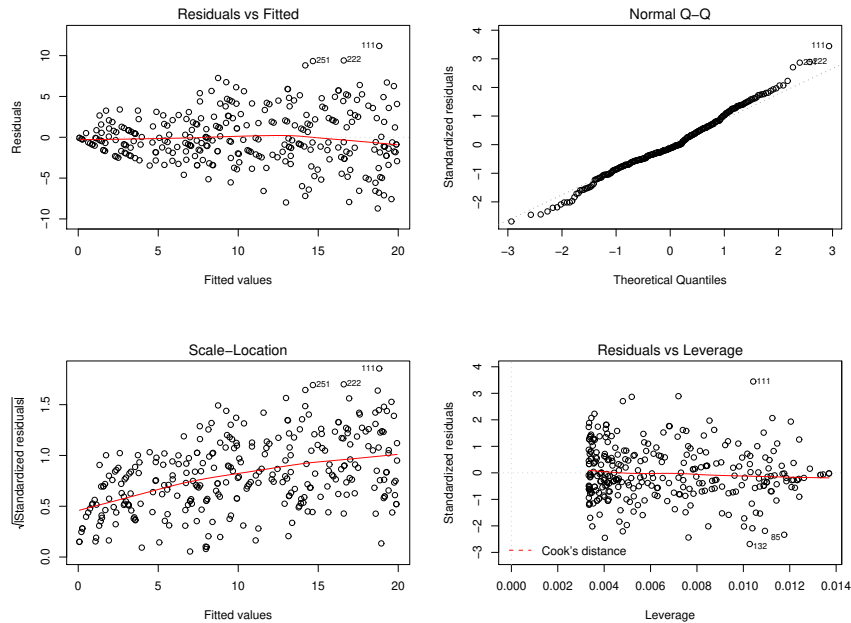
1. Anything that suggests that the residuals might not all have zero expected value.
2. Anything that suggests that the residuals might not all have the same variance.

Here is a problematic set of plots.

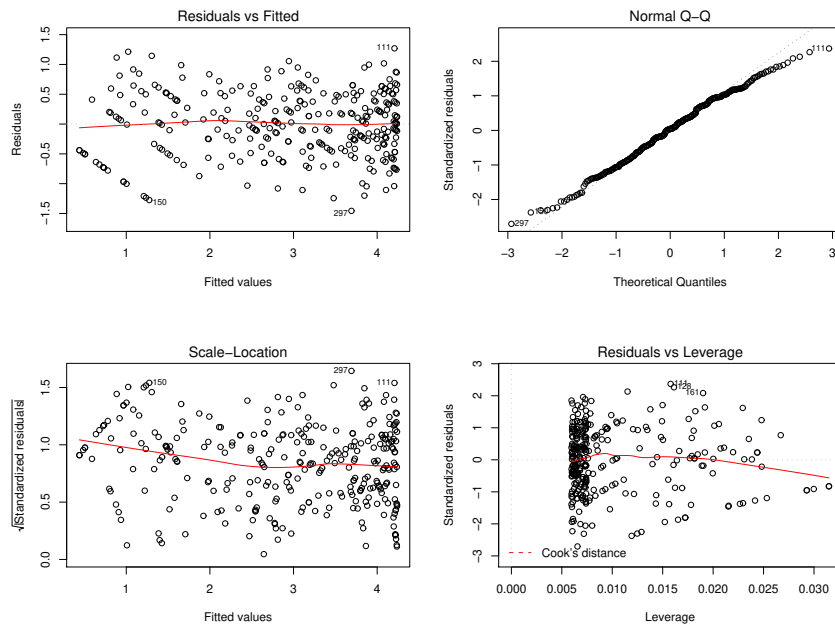


Notice the clear trend in the mean of the residuals plotted against fitted values. This sort of pattern means that the independence assumption is violated, and usually indicates that something is missing from the model. Perhaps a dependence on the square of some predictor variable?

Here is a set of plots with a different problem. . .



In this case the mean seems to be more or less zero for all the residuals, as it should be. However the variance of the residuals shows a clear increase with the mean. This pattern of increasing variance with increasing mean is quite a common pattern in practice, and obviously violates the constant variance assumption. In this case it can sometimes help to switch to modelling some transformation of the original response variable. For example the following plots are from a model of the *square root* of the response variable from the previous plot.



5.2 Tyre wear data and backward selection

The MASS library, which is an add on for R, contains a data set, `Rubber`, on the relationship between the rate of wear of tyres (`loss` in grammes per hour), and the hardness and tensile strength of the tyre rubber (`hard` and `tens` respectively, measured in kgm^{-2} and ‘Shore units’ respectively).

```
> library(MASS)
> Rubber
      loss hard tens
1    372   45  162
2    206   55  233
3    175   61  232
4    154   66  231
5    136   71  231
.      .    .    .
.      .    .    .
29   215   81  134
30   148   86  127
```

The data were gathered in order to develop a model with which to predict wear rate as a function of hardness and tensile strength. All that is really known at the outset is that the loss rate should be some smooth function of the two predictors, so we could try a polynomial model in which all combinations of powers of `tens` and `hard` were present, up to third order. i.e. we could try the model:

$$y_i = \beta_0 + \beta_1 h_i + \beta_2 t_i + \beta_3 t_i h_i + \beta_4 t_i^2 + \beta_5 h_i^2 + \beta_6 h_i^2 t_i + \beta_7 h_i t_i^2 + \beta_8 t_i^3 + \beta_9 h_i^3 + \epsilon_i$$

where y is `loss`, h is `hard`, t is `tens` and the ϵ_i are i.i.d. $N(0, \sigma^2)$ r.v.s. In the interests of precision it is worth trying to see whether all the terms in this model are needed, or whether something simpler would suffice. To this end we can successively remove the model parameter for which the hypothesis of zero value yields the highest p-value, above some threshold (e.g. 0.05), until all terms in the model have p-values below the threshold.

Here is how this *backward selection* can be achieved in R.

```
## fit full model ....
> m1 <- lm(loss~hard + tens + I(hard*tens) + I(hard^2) + I(tens^2)+
+         I(hard^2*tens)+I(tens^2*hard)+I(tens^3)+I(hard^3), Rubber)
> plot(m1) ## not shown
> summary(m1)
...
Coefficients:
              Estimate Std. Error t value Pr(>|t|)
(Intercept)   -6.040e+02  4.543e+03  -0.133    0.896
hard          -1.240e+01  7.213e+01  -0.172    0.865
tens           3.123e+01  5.295e+01   0.590    0.562
I(hard * tens) -2.924e-01  5.085e-01  -0.575    0.572
I(hard^2)       4.968e-01  7.613e-01   0.653    0.521
I(tens^2)      -1.572e-01  2.081e-01  -0.755    0.459
I(hard^2 * tens) 2.469e-03  1.675e-03   1.474    0.156
I(tens^2 * hard) -7.527e-05  9.350e-04  -0.081    0.937 ## highest!
I(tens^3)       3.384e-04  2.788e-04   1.214    0.239
I(hard^3)      -4.736e-03  3.897e-03  -1.215    0.238
...
### The 'I(tens^2 * hard)' term has the highest p-value: drop it ...

> m2 <- update(m1, .~.-I(tens^2*hard))
> summary(m2)
```

```
...
Coefficients:
              Estimate Std. Error t value Pr(>|t|)
(Intercept) -9.349e+02  1.890e+03  -0.495   0.6261
hard        -8.602e+00  5.324e+01  -0.162   0.8732 ## highest!
tens         3.520e+01  1.910e+01   1.843   0.0795 .
I(hard * tens) -3.299e-01  1.993e-01  -1.655   0.1127
I(hard^2)      4.927e-01  7.414e-01   0.665   0.5136
I(tens^2)     -1.715e-01  1.060e-01  -1.618   0.1207
I(hard^2 * tens) 2.536e-03  1.419e-03   1.787   0.0884 .
I(tens^3)      3.545e-04  1.906e-04   1.860   0.0770 .
I(hard^3)     -4.780e-03  3.766e-03  -1.269   0.2182
...
```

'hard' has highest p-value, so drop it ...

```
> m3 <- update(m2, ~.-hard)
...
## more iterations follow until ...
...
> m5 <- update(m4, ~.-I(hard^2))
> summary(m5)
```

```
Call:
lm(formula = loss ~ tens + I(hard * tens) + I(tens^2) + I(hard^2 *
  tens) + I(tens^3) + I(hard^3) - 1, data = Rubber)
```

```
Residuals:
    Min       1Q   Median       3Q      Max
-56.512 -12.656   4.415  12.057  59.316
```

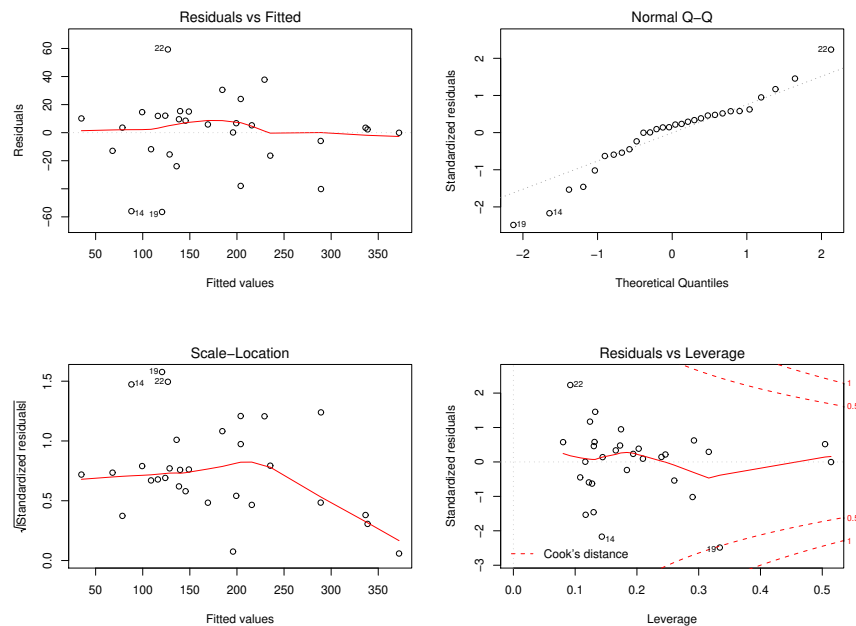
```
Coefficients:
              Estimate Std. Error t value Pr(>|t|)
tens         1.774e+01  2.201e+00   8.063 2.75e-08 ***
I(hard * tens) -8.237e-02  3.466e-02  -2.377  0.02580 *
I(tens^2)     -1.227e-01  1.706e-02  -7.196 1.95e-07 ***
I(hard^2 * tens) 8.920e-04  3.027e-04   2.947  0.00704 **
I(tens^3)      2.621e-04  4.129e-05   6.347 1.46e-06 ***
I(hard^3)     -9.594e-04  1.616e-04  -5.936 3.99e-06 ***
---
```

```
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
Residual standard error: 27.86 on 24 degrees of freedom
Multiple R-Squared:  0.9838,    Adjusted R-squared:  0.9797
F-statistic: 242.6 on 6 and 24 DF,  p-value: < 2.2e-16
```

```
> plot(m5)
```

Notice that at the last step every remaining term is significant at the 5% level. In the above the `update` function was used to *update* the fitted model: by specifying only the terms to be dropped from the previous fit there is less typing to do (compared to calling `lm` each time), and hence less chance of error. Here are the final residual plots.



For only 30 data, these plots look ok. Note, however, that on each plot the 3 most extreme points are labelled with their row number. In this case the most extreme 3 points on every plot are observations 14, 19 and 22: so it would be worth looking more carefully at these. In particular it might be worth repeating the analysis with and without observation 19, which has a high Cook's distance and may therefore be having undue influence on the results.

5.2.1 drop1

`drop1` is a useful R function which will work through the terms in your model, dropping each term and comparing the fit of the model without that term, to the full model fit. For a linear model the comparison can be done by F ratio testing or AIC (see below). For example, for the full tyre wear model we get

```
> drop1(m1, test="F")
```

Single term deletions

Model:

```
loss ~ hard + tens + I(hard * tens) + I(hard^2) + I(tens^2) +
      I(hard^2 * tens) + I(tens^2 * hard) + I(tens^3) + I(hard^3)
```

	Df	Sum of Sq	RSS	AIC	F value	Pr(F)
<none>			15931.5	208.2		
hard	1	23.5	15955.0	206.3	0.0296	0.8652
tens	1	277.2	16208.7	206.8	0.3480	0.5618
I(hard * tens)	1	263.5	16195.0	206.7	0.3308	0.5716
I(hard^2)	1	339.3	16270.7	206.9	0.4259	0.5214
I(tens^2)	1	454.4	16385.9	207.1	0.5705	0.4589
I(hard^2 * tens)	1	1730.8	17662.2	209.3	2.1728	0.1560
I(tens^2 * hard)	1	5.2	15936.6	206.3	0.0065	0.9366
I(tens^3)	1	1173.6	17105.1	208.4	1.4733	0.2390
I(hard^3)	1	1176.5	17108.0	208.4	1.4770	0.2384

The p-value for each row is that obtained by testing the null hypothesis that the model *without* that row's term is correct, against the alternative that the full model is required. So a high p-value implies that the data are quite consistent with the model in which the term concerned has been dropped.

Notice that because each model comparison here involves just one parameter, the p-values obtained by F ratio testing are the same as those obtained from the t-tests used in `summary`. So why bother with `drop1`? The answer is that it is much more convenient than `summary` when there are factor variables in the model. In such a case `summary` will report p-values for each parameter associated with the factor, which are not suitable for deciding on whether the whole factor variable should be included in the model or not. By contrast, `drop1` will compare the models with and without the factor variable (thereby dealing with all its parameters at once) which is what is needed for model selection.

5.2.2 AIC

The default model comparison method for `drop1` is AIC: Akaike's Information Criterion . When we compare models by hypothesis testing we are basically asking 'what is the simplest model that is defensible for these data', since hypothesis testing always sticks with the simplest model unless the data provide strong evidence that it is less adequate than the alternative. AIC is a different approach, it is an attempt to estimate a particular measure of the 'distance' between the fitted model, and the underlying 'true model' that generated the data. If we model select by AIC then we are trying to find the the model that will be as 'close' as possible to the truth, in the sense of having the smallest error when the model is used for prediction. Models with lower AIC are considered to be better than models with higher AIC. Formally if l is the maximized log likelihood of the model and p the number of parameters it has then

$$\text{AIC} = -2l + 2p.$$

The AIC function will compute the AIC of a fitted model object in R. e.g.

```
> AIC(m1)
[1] 295.3820
```

and here is what `drop1` does by default...

```
> drop1(m1)
Single term deletions

Model:
loss ~ hard + tens + I(hard * tens) + I(hard^2) + I(tens^2) +
      I(hard^2 * tens) + I(tens^2 * hard) + I(tens^3) + I(hard^3)
      Df Sum of Sq    RSS    AIC
<none>                 15931.5  208.2
hard                   1     23.5 15955.0  206.3
tens                   1    277.2 16208.7  206.8
I(hard * tens)         1    263.5 16195.0  206.7
I(hard^2)               1    339.3 16270.7  206.9
I(tens^2)               1    454.4 16385.9  207.1
I(hard^2 * tens)        1   1730.8 17662.2  209.3
I(tens^2 * hard)        1      5.2 15936.6  206.3
I(tens^3)               1   1173.6 17105.1  208.4
I(hard^3)               1   1176.5 17108.0  208.4
```

The row labelled `<none>` gives the AIC of the full model. Irritatingly, this differs from that given by `AIC` because of differences in which model independent constants have been included in the log likelihood of the model: of course differences in AIC are unaffected by such details. The remaining rows report the AIC of the model without the term concerned (but with all other model terms). The lowest AIC is for the model with `I(tens^2 * hard)` dropped (although you need to use AIC to confirm that it is actually lower than the model with `hard` dropped), so this would be the first candidate for dropping.

5.2.3 step

Backwards selection by iteratively calling `drop1`, and deleting the term it implies is rather automatic, and could clearly be automated completely. Function `step` does this, based on AIC model selection. `step` actually does

slightly more than backwards selection, since at each stage (after the second), it tries to see whether the model could be improved by re-including any of the currently dropped terms, before it considers further deletions. Here is the outcome of applying step to the tyre model:

```
> step(m1)
Start:  AIC=208.25
loss ~ hard + tens + I(hard * tens) + I(hard^2) + I(tens^2) +
      I(hard^2 * tens) + I(tens^2 * hard) + I(tens^3) + I(hard^3)
```

	Df	Sum of Sq	RSS	AIC
- I(tens^2 * hard)	1	5.2	15936.6	206.3
- hard	1	23.5	15955.0	206.3
- I(hard * tens)	1	263.5	16195.0	206.7
- tens	1	277.2	16208.7	206.8
- I(hard^2)	1	339.3	16270.7	206.9
- I(tens^2)	1	454.4	16385.9	207.1
<none>			15931.5	208.2
- I(tens^3)	1	1173.6	17105.1	208.4
- I(hard^3)	1	1176.5	17108.0	208.4
- I(hard^2 * tens)	1	1730.8	17662.2	209.3

[omitted steps]

Call:

```
lm(formula = loss ~ tens + I(hard * tens) + I(hard^2) + I(tens^2) +
    I(hard^2 * tens) + I(tens^3) + I(hard^3), data = Rubber)
```

Coefficients:

(Intercept)	tens	I(hard * tens)	I(hard^2)
-1.185e+03	3.630e+01	-3.436e-01	3.803e-01
I(tens^2)	I(hard^2 * tens)	I(tens^3)	I(hard^3)
-1.752e-01	2.637e-03	3.612e-04	-4.308e-03

Notice that this model is different to the one we arrived at by hypothesis testing, and has more terms. AIC generally tends to select more complex models than hypothesis testing, because of its different objective: the ‘best’ model, in a prediction error sense, is likely to be more complicated than ‘the simplest model that the data will allow’.

5.3 Forward and forward-backward selection

In the tyre wear example, it was fairly easy to decide to start with a cubic model and use backward selection. Given the aim of finding the optimum rubber we needed at least quadratic model, and quartic would have been excessively complex for the number of data. Backward selection is also nice theoretically. We start with a model that is complicated enough that we are fairly sure that it is not wrong, it’s just that it is overcomplicated. By starting with a more or less correct model, we ensure that the distributional assumptions underpinning the hypothesis tests (or AIC computation) are valid, so that the whole procedure is well founded.

However, it is often the case that there is no obvious candidate for the ‘most complex reasonable model’ or that such a model would be far too complicated to use in practice. In this case we might choose to start with a relatively simple model (e.g. some initial subset of variables enter as simple linear effects) hoping that it is not too far wrong. We then repeatedly use hypothesis testing or AIC comparison to consider adding in extra variables, or higher order terms, one at a time. If an added term improves the fit significantly we keep it in the model. Clearly in using this method we are starting from a model that we are accepting is wrong, and if the procedure proceeds for more than one step, then it must rely on comparing models when both of them are wrong (we have no rigorous theory for this

case). None the less we can often end up with an adequate model this way, in which case the theoretical problems with the method used to find it are often unimportant. This method is ‘forward selection’.

Forward and backward selection can be combined in various ways. For example, run forward selection until no further variables are added, then run backward selection until no further variables are dropped, then run forward selection again, and so on until successive forward and backward steps both produce no change in the model.

The `step` function can perform forward or forward-backward selection. For example, here is the tyre example again, but this time starting from a model with simple linear terms in `hard` and `tens`...

```
m0 <- lm(loss~hard+tens,Rubber)
step(m0,scope=loss~hard + tens + I(hard*tens) + I(hard^2) + I(tens^2)+
      I(hard^2*tens)+I(tens^2*hard)+I(tens^3)+I(hard^3),direction="both")
```

[much omitted output]

Call:

```
lm(formula = loss ~ hard + tens + I(tens^2 * hard) + I(hard^3) +
    I(tens^3) + I(hard^2 * tens), data = Rubber)
```

Coefficients:

(Intercept)	hard	tens	I(tens^2 * hard)
1.780e+03	-7.546e+00	-8.282e+00	-7.584e-04
I(hard^3)	I(tens^3)	I(hard^2 * tens)	
-2.036e-03	1.433e-04	2.248e-03	

...slightly different to the previous model.

5.4 r^2 : How close is the fit?

It is useful to have a general measure of how closely a model fits the response data, and the r^2 statistic provides this. r^2 measures the proportion (or percentage) of the variance in the original data that is ‘explained’ by the fitted model. The proportion of variance left unexplained, is the observed variance of the residuals, divided by the observed variance of the response data. The proportion variance explained is hence one minus the same ratio, so

$$r^2 = 1 - \frac{\sum_i \hat{\epsilon}_i^2 / n}{\sum_i (y_i - \bar{y})^2 / n}.$$

Notice that this conventional definition (from which the n ’s can be cancelled) uses biased variance estimators. As a result r^2 tends to overestimate how well a model is doing. The *adjusted* r^2 avoids this, to some extent, by using unbiased estimators:

$$r_{\text{adj}}^2 = 1 - \frac{\sum_i \hat{\epsilon}_i^2 / (n - p)}{\sum_i (y_i - \bar{y})^2 / (n - 1)},$$

where p is the number of model parameters. Note that r_{adj}^2 can be negative.

A high r^2 is always better than a low r^2 , but a low r^2 should not necessarily be taken as indicating that a model is poor: some response data simply contain a good deal of random variability about which nothing can be done.

Note that opinions differ about the appropriate definition of r^2 when a model contains no intercept term. In R the definition of r^2 uses 0 in place of \bar{y} in this circumstance, which has the somewhat undesirable effect of making the r^2 increase substantially if an intercept term is dropped from a model.

5.5 Prediction

Suppose that we have a fitted model, and would like to use the model to make predictions for a set of predictor variable values (which may have been used for fitting, or may be new). For example, suppose that a prediction is required for `tens` = 200 and `hard` = 50 using the model fitted to the `Rubber` data.

1. Use the new predictor variables to create a vector $\tilde{\mathbf{x}}$ in exactly the same way as the predictors would have been used to create a row of the model matrix if they had been used for model fitting.
2. The *prediction* is then $\tilde{\mu} = \tilde{\mathbf{x}}^T \hat{\boldsymbol{\beta}}$.
3. Obviously several predictions can be made simultaneously, for different predictor combinations, by replacing $\tilde{\mathbf{x}}$ by a prediction matrix, $\tilde{\mathbf{X}}$, say.

For example the finally selected tyre wear model was:

$$y_i = \beta_1 t_i + \beta_2 t_i h_i + \beta_3 t_i^2 + \beta_4 h_i^2 t_i + \beta_5 t_i^3 + \beta_6 h_i^3 + \epsilon_i$$

Hence to make a prediction for \tilde{h}, \tilde{t} , we form the vector $\tilde{\mathbf{x}}^T = [\tilde{t}, \tilde{t}\tilde{h}, \tilde{t}^2, \tilde{h}^2\tilde{t}, \tilde{t}^3, \tilde{h}^3]$, and the prediction is $\tilde{\mu} = \tilde{\mathbf{x}}^T \hat{\boldsymbol{\beta}}$. With the values $\tilde{h} = 50$ and $\tilde{t} = 200$ then $\tilde{\mathbf{x}}^T = [200, 10000, 40000, 500000, 8000000, 125000]$.

The variance of a prediction is easily obtained using standard results on the transformation of covariance matrices, and the covariance matrix of $\hat{\boldsymbol{\beta}}$. i.e.

$$\text{var}(\tilde{\mu}) = \tilde{\mathbf{x}}^T \mathbf{V}_{\hat{\boldsymbol{\beta}}} \tilde{\mathbf{x}} \quad (\equiv \sigma_{\tilde{\mu}}^2, \text{ say}).$$

It is then straightforward to show that

$$\frac{\tilde{\mu} - \mu}{\hat{\sigma}_{\tilde{\mu}}} \sim t_{n-p}$$

(where μ denotes the true expected value of the response, given the predictors). So an $100(1 - \alpha)\%$ CI for μ is

$$\tilde{\mu} \pm t_{1-\alpha/2, n-p} \hat{\sigma}_{\tilde{\mu}},$$

where $t_{1-\alpha/2, n-p}$ is the value below which a t_{n-p} r.v. will lie with probability $1 - \alpha/2$.

The R function `predict` facilitates the calculation of predictions, $\tilde{\mu}$, associated standard errors, $\hat{\sigma}_{\tilde{\mu}}$, and corresponding confidence intervals. Here is a simple prediction of expected loss rate at `tens = 200` and `hard = 50`. Notice how a *data frame*, `newdata`, is supplied containing the values at which predictions are required.

```
> predict(m5, newdata=data.frame(tens=200, hard=50))
[1] 237.9392
```

We can also ask for standard errors, $\hat{\sigma}_{\tilde{\mu}}$ to be returned. . .

```
> predict(m5, newdata=data.frame(tens=200, hard=50), se=TRUE)
$fit
[1] 237.9392
$se.fit
[1] 16.67017
$df
[1] 24
$residual.scale
[1] 27.85691
```

Confidence intervals for the expected response at the given predictor values can also be obtained:

```
> predict(m5, newdata=data.frame(tens=200, hard=50), interval="confidence")
           fit           lwr           upr
[1,] 237.9392 203.5337 272.3448
```

Note that the default confidence level is .95, but this can be altered using the `level` argument.

Of course it is also possible to obtain several predictions (and associated standard errors and intervals) at once. For example

```
> predict(m5, newdata=data.frame(tens=c(200, 190), hard=c(50, 60)))
           1           2
237.9392 201.6707
```

Prediction intervals

The *confidence intervals* for predictions, considered so far, are random intervals that have some specified probability of including the true *expected response* corresponding to some predictor variable values. Sometimes it is useful to obtain **prediction intervals** which are intervals within which we would expect a new independent observation of a response variable to lie, with some specified probability, given some particular values for the predictors.

Let Y be such a new observation. We could write Y as

$$Y = \tilde{\mu} + \tilde{\epsilon}$$

where $\tilde{\mu}$ is the predicted expectation of Y according to the fitted model. Now $\mathbb{E}(Y) = \mu$ and it is easy to see that $\mathbb{E}(\tilde{\mu}) = \mu$, so

$$\mathbb{E}(Y - \tilde{\mu}) = 0.$$

Furthermore

$$\text{var}(Y - \tilde{\mu}) = \sigma^2 + \sigma_{\tilde{\mu}}^2 (= \sigma_{Y-\tilde{\mu}}^2, \text{ say}),$$

by the independence of Y and $\tilde{\mu}$ (Y is a new observation, not used in the estimation of the model used to calculate $\tilde{\mu}$). A fairly routine argument then shows that

$$\frac{Y - \tilde{\mu}}{\hat{\sigma}_{Y-\tilde{\mu}}} \sim t_{n-p},$$

a result which can be used to create the required *prediction interval*. Specifically

$$\begin{aligned} \Pr\left(-t_{1-\alpha/2, n-p} < \frac{Y - \tilde{\mu}}{\hat{\sigma}_{Y-\tilde{\mu}}} < t_{1-\alpha/2, n-p}\right) &= 1 - \alpha \\ \Rightarrow \Pr\left(\tilde{\mu} - t_{1-\alpha/2, n-p}\hat{\sigma}_{Y-\tilde{\mu}} < Y < \tilde{\mu} + t_{1-\alpha/2, n-p}\hat{\sigma}_{Y-\tilde{\mu}}\right) &= 1 - \alpha \end{aligned}$$

i.e. a $100(1 - \alpha)\%$ *prediction interval* for a new response observation at a given set of predictor values is

$$\tilde{\mu} \pm t_{1-\alpha/2, n-p}\hat{\sigma}_{Y-\tilde{\mu}}.$$

Such intervals are easily obtained in R using the `predict` function, with the `interval="prediction"` option, as follows...

```
> predict(m5, newdata=data.frame(tens=200, hard=50), interval="prediction")
      fit      lwr      upr
[1,] 237.9392 170.9371 304.9413
```

Notice that this is much wider than the corresponding confidence interval for the predicted mean response: it has to be, as this is the interval within which 95% of new replicate observations should lie (while the confidence interval was concerned with bracketing the long term average of such replicate observations).

6 Causality, confounding and randomization

There are several different purposes for which we may wish to use the methods of statistical inference.

1. For prediction. For example we build a model of patients' probability of heart disease based on lifestyle factors such as diet and exercise, estimate the model from a representative group of subjects whose heart disease status is known, and use it to predict the status of new subjects.
2. To establish association between variables. For example we suspect that being educated in a single sex school may be negatively associated with length of marriage, and use statistical modelling to assess whether this putative association is real, or could just be due to chance variability in the data.
3. To establish causation. Often, especially in science, we want to know whether something is an actual cause of something else. Famous examples are: does smoking cause cancer? and does the MMR vaccine cause autism?

The methods we have covered so far are quite sufficient for addressing 1 and 2, but the *causal inference* required in 3 is more difficult, and the sort of methods covered so far are not sufficient to establish causality *on their own*.

To understand the issues, it is important to really have grasped the difference between correlation (association) and causality. An oft quoted example is the correlation between the population of storks and the birth rate, in Europe. The correlation is real, but there is obviously no causation. Rather, industrialization raised health care, living and educational standards in Europe leading to reduced birth rates, but that same industrialization led to a reduction in suitable habitat for storks. That is, the association is caused by other factors which may be causative to both variables of interest.

Confusing correlation with causation is a famously good way of selling newspapers. A study from the university of Bath was reported as revealing that men who preferred women with a smaller than usual hip to waist ratio were more likely to have autistic children. The association was real, but it was the interpretation in terms of the preference being somehow causative that made the story interesting. In fact, the likely explanation of the association is that higher than average testosterone in the womb seems to be a causative factor in autism, and women with higher than average testosterone tend to have smaller than usual hip to waist ratio. For obvious reasons they also tend to have children with men who find them attractive. This explanation makes for a dull press release, of course. Again the key is a hidden variable related to the variables of direct interest.

6.1 Confounding: some simulated examples

To appreciate the effect of correlated predictors on interpretation, and on attempts at causal inference, it is worth looking at some simulations, where we are in complete control of the correlation structure between variables. The following code simulates two correlated predictor variables, x and h :

```
require(mgcv) ## supplies rmvn
V <- matrix(c(1,.95,.95,1),2,2) ## cov matrix
n <- 1000 ## number of data
set.seed(7) ## just for reproducibility
X <- rmvn(n,rep(0,2),V)
x <- X[,1];h <- X[,2]
```

Now consider the case when $y_i = x_i + \epsilon_i$, but we fit the model $y_i = \beta_0 + \beta_1 x_i + \beta_2 h_i + \epsilon_i$.

```
> y <- x + rnorm(n)
> summary(lm(y ~ x + h))
...
              Estimate Std. Error t value Pr(>|t|)
(Intercept) -0.003268    0.031170  -0.105    0.9165
x             0.812568    0.098924   8.214 6.59e-16 ***
h             0.209603    0.099475   2.107  0.0354 *
...
> summary(lm(y ~ x ))
...
              Estimate Std. Error t value Pr(>|t|)
(Intercept) -0.002117    0.031219  -0.068    0.946
x             1.010417    0.031191  32.395 <2e-16 ***
...
```

Because of the high correlation between x and h it is difficult to distinguish their effects, with the result that when both are included in the model the ‘explanation’ of the response tends to get shared out between them, leading to rather variable estimates of the real effect of x . In contrast when the spurious h term is omitted from the model, the estimate of β_1 is much more accurate. Actually, for most replicates of the data simulation, β_2 would not have appeared to be significantly different from zero, so we would have been likely to drop the term, and arrive at the correct model using just the inferential tools already developed.

However, it is worth also looking at what would have happened if we had only observe h but not x :

```
> summary(lm(y ~ h))
...
              Estimate Std. Error t value Pr(>|t|)
(Intercept) -0.007193   0.032187  -0.223   0.823
h            0.985161   0.032336  30.466  <2e-16 ***
...
```

The high correlation between h and x means that we find a strongly significant association between h and y , despite the fact that h played no part at all in the simulation of y ! In this case x is an example of a hidden *confounding variable*: it is correlated with both our predictor, h , and our response y , and therefore messes up our inference about the true causal influence of h on y (which is zero in this case).

Here is a second example of confounding. Now both h and x have a causal effect on y , and if we only observe x , we will overestimate its true effect (overestimate, because the correlation is positive).

```
> y <- x + h + rnorm(n)
> summary(lm(y ~ x))
...
              Estimate Std. Error t value Pr(>|t|)
(Intercept)  0.01714    0.03249   0.528   0.598
x            1.92020    0.03246  59.155  <2e-16 ***
...
> summary(lm(y ~ x + h))
...
              Estimate Std. Error t value Pr(>|t|)
(Intercept)  0.01257    0.03145   0.400   0.689
x            1.13516    0.09980  11.374  < 2e-16 ***
h            0.83168    0.10036   8.287  3.71e-16 ***
...
```

i.e. when h is hidden from us, our estimate of β_1 is almost twice as large as it should be in this case.

Note that this tendency of the model fitting to try and compensate for missing confounding variables, is a profound difficulty for causal inference, but a blessing for prediction. The fact that the model has adjusted for the missing confounders in this way actually improves prediction.

6.2 Controlled experiments and randomization

The gold standard for establishing causality is to use data from a properly designed experiment, in which we use the process of *randomization* to break any possible association between unmeasured confounders and the predictor whose effect we want to measure. The idea is to turn the systematic variation in the response caused by confounders into something that we can model as independent random variability between experimental units.

An example conveys the idea. Suppose that we want to examine the relationship between exercise and fat mass in people aged 40 to 50. There are a huge number of factors contributing to people's fat mass, and if we simply look at a sample of people who already exercise by different amounts it will be very difficult to isolate the effect of exercise separate from all the other confounders (diet, working hours, profession, whether they have children, drinking habits, smoking, wealth etc.). Suppose instead that we set up a study in which participants enrol and are then assigned to one of several 'treatments', consisting of exercise programs of differing intensity (or a control of none). Now if we assign subjects to the different treatments randomly then there can be no possible association between all the other variables contributing to fat mass, and the one that we are interested in: amount of exercise. Indeed all the variability in fat mass attributable to the confounders can now be treated as random variability within each treatment. Any effect of exercise that we then find is causal: the thing we controlled having an effect on the variable we are interested in. Of course this approach does not stop us from including measured covariates in the analysis model, in order to reduce the amount of variability being modelled as random, and thereby increase precision, but it does mean that we no longer have to worry about the effect of hidden confounders.

To appreciate the issue mathematically, consider a linear model for which the full model matrix is (\mathbf{X}, \mathbf{H}) (w.l.o.g. assume that the columns of \mathbf{H} are centred). If we had not observed the variables in \mathbf{H} , then our estimates

of the effects of \mathbf{X} would be $\tilde{\beta}_x = (\mathbf{X}^\top \mathbf{X})^{-1} \mathbf{X}^\top \mathbf{y}$, whereas if we observed all the data then it would be

$$\begin{pmatrix} \hat{\beta}_x \\ \hat{\beta}_h \end{pmatrix} = \begin{pmatrix} \mathbf{X}^\top \mathbf{X} & \mathbf{X}^\top \mathbf{H} \\ \mathbf{H}^\top \mathbf{X} & \mathbf{H}^\top \mathbf{H} \end{pmatrix}^{-1} \begin{pmatrix} \mathbf{X}^\top \\ \mathbf{H}^\top \end{pmatrix} \mathbf{y}.$$

Clearly if $\mathbf{X}^\top \mathbf{H} \neq \mathbf{0}$ then $\tilde{\beta}_x \neq \hat{\beta}_x$, that is the missing confounders in \mathbf{H} interfere with the estimation of β_x . However if the confounders are independent of \mathbf{X} , then at least in the large sample limit, $\mathbf{X}^\top \mathbf{H} = \mathbf{0}$, so that

$$\begin{pmatrix} \hat{\beta}_x \\ \hat{\beta}_h \end{pmatrix} = \begin{pmatrix} \mathbf{X}^\top \mathbf{X} & \mathbf{0} \\ \mathbf{0} & \mathbf{H}^\top \mathbf{H} \end{pmatrix}^{-1} \begin{pmatrix} \mathbf{X}^\top \\ \mathbf{H}^\top \end{pmatrix} \mathbf{y} = \begin{pmatrix} (\mathbf{X}^\top \mathbf{X})^{-1} & \mathbf{0} \\ \mathbf{0} & (\mathbf{H}^\top \mathbf{H})^{-1} \end{pmatrix} \begin{pmatrix} \mathbf{X}^\top \\ \mathbf{H}^\top \end{pmatrix} \mathbf{y}.$$

i.e. when the confounders are independent of the response we get the same estimate of β_x whether or not they are included in the model (and its expected value is correct). Randomization ensures that this occurs.

6.3 Instrumental variables

So properly designed controlled experiments with random allocation of experimental units to treatments are the best way of establishing causal effects. However there are many cases in which experimental manipulation is impractical, unethical and/or illegal. For example, if we are interested in establishing the causative effect of alcohol consumption on heart disease, it is simply unethical to conduct a study in which we randomly allocate subjects to a heavy drinking treatment (the same goes for any treatment where we have prior cause to suspect it may be harmful). Economics is another field beset with such problems: for example it is impractical to conduct a controlled experiment to establish the causative factors controlling a company's share price, and anything that came close would likely count as illegal market manipulation. This sort of problem is so ubiquitous in economics that it is from the field of *econometrics* that much of the work on causal inference from observational data comes.

In this section we will look at one method: instrumental variables. The approach can be applied more widely than just to linear models, but the ideas are easiest to grasp in the linear model context. Let's again use the setup in which \mathbf{X} is the model matrix for the observed effects, \mathbf{H} is the model matrix for the hidden confounders (column centred - i.e. its mean has been subtracted from each column) and the response data are generated by,

$$\mathbf{y} = \mathbf{X}\beta_x + \mathbf{H}\beta_h + \epsilon.$$

Since we don't have access to \mathbf{H} we fit $\mathbf{y} = \mathbf{X}\beta_x + \mathbf{e}$, and therein lies the problem, since in reality $\mathbf{e} = \mathbf{H}\beta_h + \epsilon$, which is unlikely to meet the linear model assumptions about the residual vector. Indeed,

$$\mathbb{E}(\hat{\beta}_x) = (\mathbf{X}^\top \mathbf{X})^{-1} \mathbf{X}^\top (\mathbf{X}, \mathbf{H})\beta = \beta_x + (\mathbf{X}^\top \mathbf{X})^{-1} \mathbf{X}^\top \mathbf{H}\beta_h \neq \beta_x.$$

Now the problem here is that the space spanned by \mathbf{X} is not orthogonal to \mathbf{e} , because of the interdependence of \mathbf{X} and \mathbf{H} . This would not be a problem if we could somehow orthogonalize \mathbf{X} and \mathbf{e} , for example by projecting \mathbf{X} onto a space that is orthogonal to \mathbf{e} .

Suppose then, that we have available some *instrumental variables* giving rise to a (column centred) model matrix \mathbf{Z} , where \mathbf{Z} is not part of the true model for \mathbf{y} , but is correlated with \mathbf{X} and is independent of \mathbf{H} (and hence \mathbf{e}). We'll assume that \mathbf{Z} has at least as high a rank as \mathbf{X} , and w.l.o.g. its columns are centred to have zero mean. We could then project \mathbf{X} onto the column space of \mathbf{Z} . That is replace \mathbf{X} with $\mathbf{A}_z \mathbf{X}$ where $\mathbf{A}_z = \mathbf{Z}(\mathbf{Z}^\top \mathbf{Z})^{-1} \mathbf{Z}^\top$, and regress \mathbf{y} on $\mathbf{A}_z \mathbf{X}$. In that case

$$\hat{\beta}_x = (\mathbf{X}^\top \mathbf{A}_z \mathbf{X})^{-1} \mathbf{X}^\top \mathbf{A}_z \mathbf{y}$$

and we now have

$$E(\hat{\beta}_x) = (\mathbf{X}^\top \mathbf{A}_z \mathbf{X})^{-1} \mathbf{X}^\top \mathbf{A}_z \mathbf{X} \beta_x + (\mathbf{X}^\top \mathbf{A}_z \mathbf{X})^{-1} \mathbf{X}^\top \mathbf{A}_z \mathbf{H} \beta_h \simeq \beta_x,$$

since $\mathbf{A}_z \mathbf{H} \simeq \mathbf{0}$ as a result of $\mathbf{Z}^\top \mathbf{H} \simeq \mathbf{0}$ by the independence of \mathbf{Z} and \mathbf{H} (alternatively we could also take expectations over the distribution of the predictors to obtain equality).

Here is a simulated illustration. First simulate observed x , confounder h and instrument z , by simulating multivariate random variables with appropriate correlation structure, using x and h to simulate response y .

```
V <- matrix(c(1,.7,0.9,.7,1,0,.9,0,1),3,3)
library(mgcv); n <- 1000; set.seed(0)
X <- rmvn(n,rep(0,3),V)
x <- X[,1]; h <- X[,2]; z <- X[,3]
y <- x + h + rnorm(n)*.3
```

Now fit the model naively, and then again using the instrumental variable.

```
> summary(lm(y ~ x)) ## naive fit
...
              Estimate Std. Error t value Pr(>|t|)
(Intercept)  0.01760    0.02426   0.726   0.468
x            1.70146    0.02425  70.173 <2e-16 ***
...
> zx <- fitted(lm(x~z-1)) ## project x onto z
> summary(lm(y ~ zx)) ## fit iv model
...
              Estimate Std. Error t value Pr(>|t|)
(Intercept)  0.08042    0.05434   1.48   0.139
zx          1.01547    0.07557  13.44 <2e-16 ***
...
```

Clearly the IV model gives an estimate that is much closer to correct. The price paid is precision: the standard error is substantially larger than we would get if h had been available for inclusion in the model (and that is without correcting for the uncertainty in the regression of x on z). This approach is not the only way of using instrumental variables, and their use is not restricted to linear models, but this method is sufficient for introducing the idea.

The success of such methods rests on being able to find good instrumental variables, which are independent of the potential confounders, but strongly correlated with the observed variables (weak correlation will inflate the standard errors of the estimates). Another way of stating this is that the instrumental variables must be correlated with the response variable *only via the instrument's correlation with the observed predictor variables*.

Finding such variables is obviously not easy, and one might question how often such a magical combination of correlation with X but independence of H can really apply, and be knowable when the confounders are not. However the technique of 'Mendelian randomization', provides an application of instrumental variable use that does seem to be practical. Take the example of alcohol and heart disease. There are genetic polymorphisms that predict the propensity to consume alcohol, but are completely unrelated to any other plausible predictors of heart disease. Hence a subject's genotype (with respect to this polymorphism) can be used as an instrumental variable. The method is called 'Mendelian randomization' after the genetic pioneer Gregor Mendel, and the fact that the genotype is assigned randomly by meiosis when passed from parents to offspring, thereby avoiding correlation with things that it does not actually cause.

'Mendelian randomization' is important in epidemiology when trying to uncover health effects of environmental exposures that can not (ethically) be manipulated experimentally, however finding appropriate polymorphisms and establishing that they are uncorrelated with any unobserved confounders is obviously challenging.

7 Practical modelling with factors and interactions

Most of the models covered so far have been for situations in which the predictor variables are continuous variables, but there are many situations in which the predictor variables are more qualitative in nature, and serve to divide the responses into groups. Examples might be eye-colour of subjects in a psychology experiment, which of three alternative hospitals were attended by patients in a drug trial, manufacturers of cars used in crash tests etc. Variables like these, which serve to classify the units on which the response variable has been measured into distinct categories, are known as *factor variables*, or simply *factors*. The different categories of the factor are known as *levels* of the factor. For example levels of the factor 'eye colour' might be 'blue', 'brown', 'grey' and 'green', so that we would refer to eye colour as a factor with four levels. Note that 'levels' is quite confusing

terminology: there is not necessarily any natural ordering of the levels of a factor. Hence ‘levels’ of a factor might best be thought of as ‘categories’ of a factor, or ‘groups’ of a factor.

Factor variables are handled using dummy variables. Each factor variable can be replaced by as many dummy variables as there are levels of the factor — one for each level of the factor. For each response datum, only one of these dummy variables will be non-zero: the dummy variable for the single level that applies to that response. Consider an example to see how this works in practice: 9 laboratory rats are fed too much, so that they divide into 3 groups of 3: ‘fat’, ‘very fat’ and ‘enormous’. Their blood insulin levels are then measured 10 minutes after being fed a standard amount of sugar. The investigators are interested in the relationship between insulin levels and the factor ‘rat size’. Hence a model could be set up in which the predictor variable is the factor ‘rat size’, with the three levels ‘fat’, ‘very fat’ and ‘enormous’. Writing y_i for the i^{th} insulin level measurement, a suitable model might be:

$$\mathbb{E}(Y_i) \equiv \mu_i = \begin{cases} \beta_0 & \text{if rat is fat} \\ \beta_1 & \text{if rat is very fat} \\ \beta_2 & \text{if rat is enormous} \end{cases}$$

and this is easily written in linear model form, using a dummy predictor variable for each level of the factor:

$$\begin{pmatrix} \mu_1 \\ \mu_2 \\ \mu_3 \\ \mu_4 \\ \mu_5 \\ \mu_6 \\ \mu_7 \\ \mu_8 \\ \mu_9 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 \\ 1 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 1 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 1 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} \beta_0 \\ \beta_1 \\ \beta_2 \end{pmatrix}.$$

A key difference between dummy variables and directly measured predictor variables, is that the dummy variables, and parameters associated with a factor, are almost always treated as a group during model selection — it does not usually make sense for a subset of the dummy variables associated with a factor to be dropped or included on their own: either all are included or none. The F ratio tests derived in section ?? are designed for hypothesis testing in this situation.

7.1 Identifiability

When modelling with factor variables, model ‘identifiability’ becomes an important issue. It is quite easy to set up models, involving factors, in which it is impossible to estimate the parameters uniquely, because an infinite set of alternative parameter vectors would give rise to exactly the same expected value vector. A simple example illustrates the problem. Consider again the fat rat example, but suppose that we wanted to formulate the model in terms of an overall mean insulin level, α , and deviations from that level, β_j , associated with each level of the factor. The model would be something like:

$$\mu_i = \alpha + \beta_j \text{ if rat } i \text{ is rat size level } j$$

(where j is 0, 1 or 2, corresponding to ‘fat’, ‘very fat’ or ‘enormous’). The problem with this model is that there is not a one-to-one correspondence between the parameters and the fitted values, so that the parameters can not be uniquely estimated from the data. This is easy to see. Consider any particular set of α and β values, giving rise to a particular μ value: any constant c could be added to α and simultaneously subtracted from each element of β without changing the value of μ . Hence there is an infinite set of parameters giving rise to each μ value, and therefore the parameters can not be estimated uniquely. The model is not ‘identifiable’.

This situation can be diagnosed directly from the model matrix. Written out in full, the example model is

$$\begin{pmatrix} \mu_1 \\ \mu_2 \\ \mu_3 \\ \mu_4 \\ \mu_5 \\ \mu_6 \\ \mu_7 \\ \mu_8 \\ \mu_9 \end{pmatrix} = \begin{pmatrix} 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} \alpha \\ \beta_0 \\ \beta_1 \\ \beta_2 \end{pmatrix}.$$

But the columns of the model matrix are not independent, and this lack of full column rank means that the formulae for finding the least squares parameter estimates break down[¶]. Identifiable models have model matrices of full column rank; unidentifiable ones are column rank deficient.

The solution to the identifiability problem is to impose just enough linear constraints on the model parameters, that the model becomes identifiable. For example, in the fat rat model, we could impose the constraint that

$$\sum_{j=0}^2 \beta_j = 0.$$

This would immediately remove the identifiability problem, but does require use of a linearly constrained least squares method. A simpler constraint is to set one of the unidentifiable parameters to zero, which requires only that the model is re-written, without the zeroed parameter, rather than a modified fitting method. For example, in the fat rat case, we could set α to zero, and recover the original identifiable model. This is perfectly legitimate, since the reduced model is capable of reproducing any expected values that the original model could produce.

In the one factor case, this discussion of identifiability may seem to be un-necessarily complicated, since it is so easy to write down the model directly, in an identifiable form. However, when models involve more than one factor variable, the issue can not be avoided.

7.2 Multiple factors

It is frequently the case that more than one factor variable should be included in a model, and this is straightforward to do. Continuing the fat rat example, it might be that the sex of the rats is also a factor in insulin production, and that the appropriate model is:

$$\mu_i = \alpha + \beta_j + \gamma_k \text{ if rat } i \text{ is rat size level } j \text{ and sex } k$$

where k is 0 or 1 for male or female. Written out in full (assuming the rats are MMFFFMFMM) the model is

$$\begin{pmatrix} \mu_1 \\ \mu_2 \\ \mu_3 \\ \mu_4 \\ \mu_5 \\ \mu_6 \\ \mu_7 \\ \mu_8 \\ \mu_9 \end{pmatrix} = \begin{pmatrix} 1 & 1 & 0 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 & 1 & 0 \end{pmatrix} \begin{pmatrix} \alpha \\ \beta_0 \\ \beta_1 \\ \beta_2 \\ \gamma_0 \\ \gamma_1 \end{pmatrix}.$$

It is immediately obvious that the model matrix is of column rank 4, implying that two constraints are required to make the model identifiable. You can see the lack of column independence by noting that column 5 is column 1

[¶]In terms of section 4, \mathbf{R} will not be full rank, and will hence not be invertible; in terms of section 4.5, $\mathbf{X}^T \mathbf{X}$ will not be invertible.

minus column 6, while column 2 is column 1 minus columns 3 and 4. An obvious pair of constraints would be to set $\beta_0 = \gamma_0 = 0$, so that the full model is

$$\begin{pmatrix} \mu_1 \\ \mu_2 \\ \mu_3 \\ \mu_4 \\ \mu_5 \\ \mu_6 \\ \mu_7 \\ \mu_8 \\ \mu_9 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 \\ 1 & 1 & 0 & 1 \\ 1 & 1 & 0 & 1 \\ 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 \end{pmatrix} \begin{pmatrix} \alpha \\ \beta_1 \\ \beta_2 \\ \gamma_1 \end{pmatrix}.$$

When you specify models involving factors in **R**, it will automatically impose identifiability constraints for you, and by default these constraints will be that the parameter for the ‘first’ level of each factor is zero. Note that ‘first’ is essentially arbitrary here — the order of levels of a factor is not important. However, if you need to change which level is ‘first’, in order to make parameters more interpretable, see the `relevel` function.

7.3 ‘Interactions’ of factors

In the examples considered so far, the effect of factor variables has been purely additive, but it is possible that a response variable may react differently to the combination of two factors, than would be predicted by simply adding the effect of the two factors separately. For example, if examining patient blood cholesterol levels, we might consider the factors ‘hypertensive’ (yes/no) and ‘diabetic’ (yes/no). Being hypertensive or diabetic would be expected to raise cholesterol levels, but being both is likely to raise cholesterol levels much more than would be predicted from just adding up the apparent effects when only one condition is present. When the effect of two factor variables together differs from the sum of their separate effects, then they are said to *interact*, and an adequate model in such situations requires *interaction terms*. Put another way, if the effects of one factor change in response to another factor, then the factors are said to interact.

Let us continue with the fat rat example, but now suppose that how insulin level depends on size varies with sex. An appropriate model is then

$$\mu_i = \alpha + \beta_j + \gamma_k + \delta_{jk} \text{ if rat } i \text{ is rat size level } j \text{ and sex } k,$$

where the δ_{jk} terms are the parameters for the interaction of rat size and sex. Writing this model out in full it is clear that it is spectacularly unidentifiable:

$$\begin{pmatrix} \mu_1 \\ \mu_2 \\ \mu_3 \\ \mu_4 \\ \mu_5 \\ \mu_6 \\ \mu_7 \\ \mu_8 \\ \mu_9 \end{pmatrix} = \begin{pmatrix} 1 & 1 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \end{pmatrix} \begin{pmatrix} \alpha \\ \beta_0 \\ \beta_1 \\ \beta_2 \\ \gamma_0 \\ \gamma_1 \\ \delta_{00} \\ \delta_{01} \\ \delta_{10} \\ \delta_{11} \\ \delta_{20} \\ \delta_{21} \end{pmatrix}.$$

In fact, for this simple example, with rather few rats, we now have more parameters than data. There are of course many ways of constraining this model to achieve identifiability. One possibility (the default in **R**) is to set $\beta_0 = \gamma_0 = \delta_{00} = \delta_{01} = \delta_{10} = \delta_{20} = 0$. The resulting model can still produce any fitted value vector that the full model can produce, but all the columns of its model matrix are independent, so that the model is identifiable:

$$\begin{pmatrix} \mu_1 \\ \mu_2 \\ \mu_3 \\ \mu_4 \\ \mu_5 \\ \mu_6 \\ \mu_7 \\ \mu_8 \\ \mu_9 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 \\ 1 & 1 & 0 & 1 & 1 & 0 \\ 1 & 1 & 0 & 1 & 1 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 \end{pmatrix} \begin{pmatrix} \alpha \\ \beta_1 \\ \beta_2 \\ \gamma_1 \\ \delta_{11} \\ \delta_{21} \end{pmatrix}.$$

Of course, the more factor variables are present, the more interactions are possible, and the higher the order of the possible interactions: for example if three factors are present then each factor could interact with each factor, giving three possible ‘two-way’ interactions, while all the factors could also interact together, giving a three-way interaction (e.g. the way in which insulin levels dependence on rat size is influenced by sex is itself influenced by blood group — perhaps with interactions beyond two-way, equations are clearer than words).

7.4 Factor continuous interactions

Suppose we had simply measured the weight, w_i , of rats, rather than classified them into three groups. In this case

$$\mu_i = \alpha + \gamma_j + \beta w_i + \delta_j w_i \text{ if rat } i \text{ is sex } j$$

might be appropriate. That is to say insulin levels vary linearly with weight, but in a different way for male and female rats. Here γ_j would be the ‘main effect’ of sex βw_i the ‘main effect’ of weight and $\delta_j w_i$ is the weight-sex ‘interaction’. Similar confounding issues exist between the main effects of weight and the weight-sex interaction as exist between factor interactions and main effects, and similar constraints are needed. Recalling the assumed ordering of rats, MMFFMFMM, then an identifiable version of the model is

$$\begin{pmatrix} \mu_1 \\ \mu_2 \\ \mu_3 \\ \mu_4 \\ \mu_5 \\ \mu_6 \\ \mu_7 \\ \mu_8 \\ \mu_9 \end{pmatrix} = \begin{pmatrix} 1 & 0 & w_1 & 0 \\ 1 & 0 & w_2 & 0 \\ 1 & 1 & w_3 & w_3 \\ 1 & 1 & w_4 & w_4 \\ 1 & 1 & w_5 & w_5 \\ 1 & 0 & w_6 & 0 \\ 1 & 1 & w_7 & w_7 \\ 1 & 0 & w_8 & 0 \\ 1 & 0 & w_9 & 0 \end{pmatrix} \begin{pmatrix} \alpha \\ \gamma_1 \\ \beta \\ \delta_1 \end{pmatrix}.$$

7.5 Using factor variables in R

It is very easy to work with factor variables in R. All that is required is that you let R know that a particular variable is a factor variable. For example, suppose z is a variable declared as follows:

```
> z <- c(1,1,1,2,2,1,3,3,3,3,4)
> z
[1] 1 1 1 2 2 1 3 3 3 3 4
```

and it is to be treated as a 4 level factor. The function `as.factor()` will ensure that z is treated as a factor:

```
> z <- as.factor(z)
> z
[1] 1 1 1 2 2 1 3 3 3 3 4
Levels: 1 2 3 4
```

When a factor variable is printed, a list of its levels is also printed — this provides an easy way to tell if a variable is a factor variable. Note also that the digits of z are treated purely as labels: the numbers 1 to 4 could have been any labels. For example, x could be a factor with 3 levels, declared as follows:

```

> x <- c("A", "A", "C", "C", "C", "er", "er")
> x
[1] "A"  "A"  "C"  "C"  "C"  "er" "er"
> x <- as.factor(x)
> x
[1] A  A  C  C  C  er er
Levels: A C er

```

Once a variable is declared as a factor variable, then **R** can process it automatically within model formulae, by replacing it with the appropriate number of binary dummy variables (and imposing any necessary identifiability constraints on the specified model).

7.5.1 Factor interactions in model formulae

Within model formulae the notation $a:b$ indicates that the interaction of variables a and b should be included in the model, while $a*b$ is exactly equivalent to $a + b + a:b$, meaning that main effects and interactions should be included. Note that if a and b are both factors then $y \sim a:b$ and $y \sim a*b$ result in exactly the same model fits, but with different meanings for the parameters and different identifiability constraints.

With reference to the fat rat examples, if `insulin` is the response, `size` and `sex` are the factor variables, and `weight` the continuous variable, then

- `insulin ~ size - 1` gives the section 7 model.
- `insulin ~ size` gives the section 7.1 model (same fit as above, but different parameter interpretation).
- `insulin ~ size + sex` gives the section 7.2 model.
- `insulin ~ size*sex` gives the section 7.3 model.
- `insulin ~ sex*weight` gives the section 7.4 model.

Another useful notation is provided by terms like $(a+b+c)^2$, which specifies main effects and interactions up to the order of the given power - 2 in this case. i.e. $(a+b+c)^2 = a + b + c + a:b + a:c + b:c$.

7.5.2 A simple example

As an example of the use of factor variables, again consider the `PlantGrowth` data frame supplied with **R**. These are data on the growth of plants under control conditions and two different treatment conditions. The factor `group` has three levels `ctrl1`, `trt1` and `trt2`, and it is believed that the growth of the plants depends on this factor. First check that `group` is already a factor variable:

```

> PlantGrowth$group
[1] ctrl1 ctrl1 ctrl1 ctrl1 ctrl1 ctrl1 ctrl1 ctrl1 ctrl1 ctrl1 trt1
[12] trt1 trt1 trt1 trt1 trt1 trt1 trt1 trt1 trt1 trt1 trt2 trt2
[23] trt2 trt2 trt2 trt2 trt2 trt2 trt2 trt2 trt2 trt2
Levels: ctrl1 trt1 trt2

```

...since a list of levels is reported, it must be. The response variable for these data is `weight` of the plants at some set time after planting, and the aim is to investigate whether the `group` factor controls this, and if so, to what extent.

```

> pgm.1 <- lm(weight ~ group, data=PlantGrowth)
> plot(pgm.1)

```

As usual, the first thing to do, after fitting a model, is to check the residual plots shown in figure 5. In this case there is some suggestion of decreasing variance with increasing mean, but the effect does not look very pronounced, so it is probably safe to proceed.

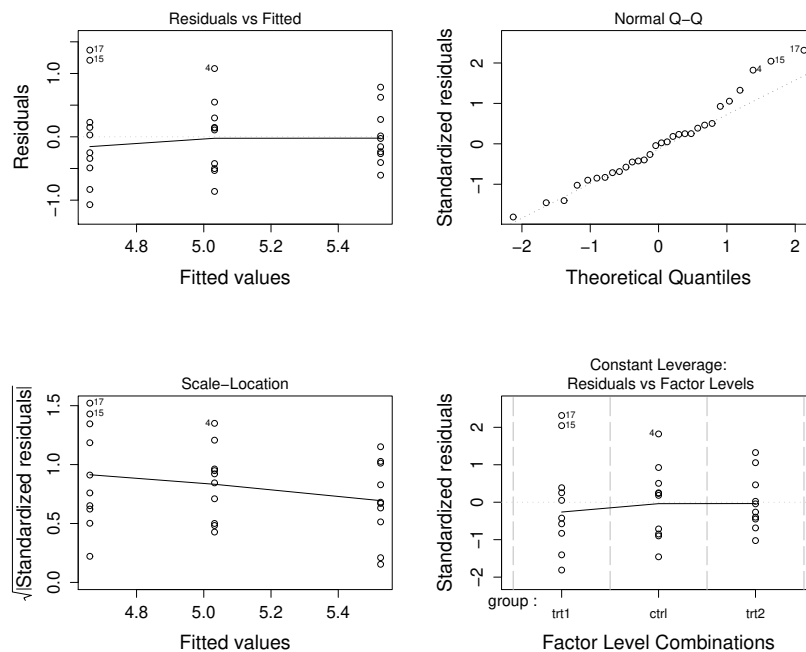


Figure 5: Model checking plots for the plant growth example. Note that, since the leverages are all the same, in this case, the lower right plot is now simplified.

```
> summary(pgm.1)
[edited]
Coefficients:
              Estimate Std. Error t value Pr(>|t|)
(Intercept)    5.0320     0.1971  25.527  <2e-16 ***
grouptrt1     -0.3710     0.2788  -1.331   0.1944
grouptrt2      0.4940     0.2788   1.772   0.0877 .
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.6234 on 27 degrees of freedom
Multiple R-Squared: 0.2641,    Adjusted R-squared: 0.2096
F-statistic: 4.846 on 2 and 27 DF,  p-value: 0.01591
```

Notice how R reports an intercept parameter and parameters for the two treatment levels, but, in order to obtain an identifiable model, it has not included a parameter for the control level of the group factor. So the estimated overall mean weight (in the population that these plants represent, given control conditions) is 5.032, while treatment 1 is estimated to lower this weight by 0.37, and treatment 2 to increase it by 0.49. However, neither parameter individually appears to be significantly different from zero. (Don't forget that `model.matrix(pgm.1)` can be used to check up on the form of the model matrix used in the fit.)

Model selection based on the summary output is very difficult for models containing factors. It makes little sense to drop the dummy variable for just one level of a factor from a model, and if we did, what would we then do about the model identifiability constraints? Usually, it is only of interest to test whether the whole factor variable should be in the model or not, and this amounts to testing whether all its associated parameters are simultaneously zero or not. The F-ratio tests derived in section 4.3.2 are designed for just this purpose, and in R, such tests can be invoked using the `anova` function. For example, we would compare `pgm.1` to a model in which the expected response is given by a single parameter that does not depend on `group`:

```

> pgm.0<-lm(weight~1,data=PlantGrowth)
> anova(pgm.0,pgm.1)
Analysis of Variance Table

Model 1: weight ~ 1
Model 2: weight ~ group
  Res.Df    RSS Df Sum of Sq    F Pr(>F)
1      29 14.2584
2      27 10.4921  2      3.7663 4.8461 0.01591 *
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

```

The output here gives the F-ratio statistic used to test the null hypothesis that the simpler model is correct, against the alternative that `pgm.1` is correct. If the null is true then the probability of obtaining so large an F value is only 0.016, suggesting that the null hypothesis is not very plausible.

So we see that the data provide evidence for an effect of the `group` factor variable on `weight`, which appeared marginal or absent when we examined p-values for the individual model parameters. This comes about because we have, in effect, considered all the parameters associated with the factor simultaneously, thereby obtaining a more powerful test than any of the single parameter tests could be. In the light of this analysis, the most promising treatment to look at is clearly treatment 2, since this gives the largest and ‘most significant’ effect and it is a positive effect.

7.6 The warpbreak data

Now consider an example with two factors as predictors. The `warpbreak` data in R come from an industrial experiment, attempting to find optimal conditions for weaving with the minimum number of breaks in the wool being woven. Two types of wool were tested at each of 3 weaving tensions (9 replicates of each combination), and the number of breaks in a standard length of cloth was recorded.

First have a quick look at the data

```

> names(warpbreaks)
[1] "breaks"      "wool"        "tension"
> ## Produce a pretty display of the data (see ?xtab for details)
> warpbreaks$replicate <- rep(1:9, len = 54)
> ftable(xtabs(breaks ~ wool + tension + replicate, data = warpbreaks))
      replicate  1  2  3  4  5  6  7  8  9
wool tension
A      L      26 30 54 25 70 52 51 26 67
      M      18 21 29 17 12 18 35 30 36
      H      36 21 24 18 10 43 28 15 26
B      L      27 14 29 19 29 31 41 20 44
      M      42 26 19 16 39 28 21 39 29
      H      20 21 24 17 13 15 15 16 28

```

Let y_{ijk} denote the number of breaks for the k^{th} replicate at the i^{th} wool type and j^{th} tension. A possible model for these data would be

$$y_{ijk} = \mu + \alpha_i + \beta_j + \gamma_{ij} + \epsilon_{ijk}$$

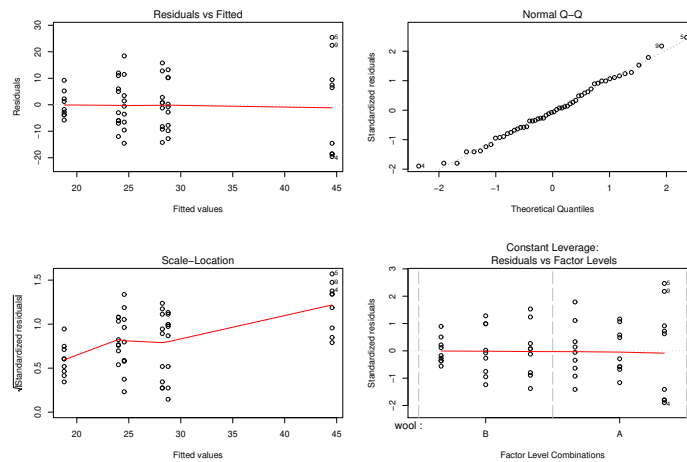
where the 2 α_i parameters are for the *main effect* of wool, the 3 β_j parameters are for the main effect of tension and the 6 γ_{ij} parameters represent the interaction of wool type and tension (μ is the overall mean number of breaks). Identifiability constraints will be needed for this model, of course, but these can be generated automatically in R.

First fit and check the model (the interaction term is specified by `tension:wool` in the model formula).

```

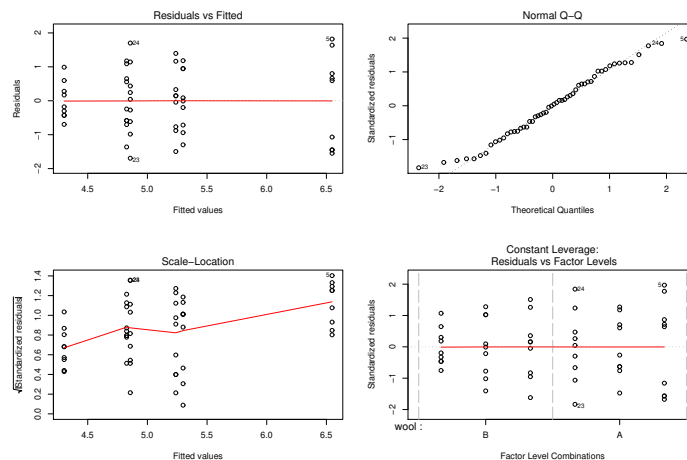
> wm0 <- lm(breaks~wool + tension + tension:wool,data=warpbreaks)
> par(mfrow=c(2,2))
> plot(wm0)

```



The variance seems to be increasing with the mean here (left 2 panels of the residual plots). This is perhaps not surprising as the data are really counts of fairly rare events: something that might be best modelled by a Poisson distribution, rather than a normal. Transformation of the response data (the y 's — `break`) can help in this circumstance. Power transformations or log transformations are often worth trying. For Poisson data it can be shown that a square root power transform is sensible, so it is worth trying that.

```
> wm <- lm(breaks^0.5~wool + tension + tension:wool,data=warpbreaks)
> par(mfrow=c(2,2))
> plot(wm)
```



The revised plots show modest improvement, and experimenting with other transformations does not do much better, so the square root transform is probably sensible.

Next, have a quick look at the model summary.

```
> summary(wm)
```

Call:

```
lm(formula = breaks^0.5 ~ wool + tension + tension:wool, data = warpbreaks)
```

Residuals:

Min	1Q	Median	3Q	Max
-1.69410	-0.70129	0.01772	0.66046	1.81902

Coefficients:

	Estimate	Std. Error	t value	Pr(> t)	
(Intercept)	6.5476	0.3266	20.046	< 2e-16	***
woolB	-1.3094	0.4619	-2.835	0.006692	**
tensionM	-1.7216	0.4619	-3.727	0.000511	***
tensionH	-1.6912	0.4619	-3.661	0.000625	***
woolB:tensionM	1.7821	0.6533	2.728	0.008874	**
woolB:tensionH	0.7553	0.6533	1.156	0.253350	

Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.9799 on 48 degrees of freedom

Multiple R-Squared: 0.3606, Adjusted R-squared: 0.294

F-statistic: 5.415 on 5 and 48 DF, p-value: 0.0004998

Notice that the model actually has only 6 identifiable parameters (rather than the 12 we originally specified). Notice also that the model actually only explains some 30% of the variability in the data (Adjusted R-squared). As always with factor variables, it is not sensible to try and base model selection on the p-values in the summary.

To perform model selection we can test the null hypothesis that the data are actually described by the model

$$y_{ijk} = \mu + \alpha_i + \beta_j + \epsilon_{ijk}$$

against the alternative that the model with interactions is required. An F-ratio test (see section 4.3.2) is the way to do this, as follows.

```
> wml <- lm(breaks^0.5~wool + tension,data=warpbreaks)
```

```
> anova(wml,wm)
```

Analysis of Variance Table

Model 1: breaks^0.5 ~ wool + tension

Model 2: breaks^0.5 ~ wool + tension + tension:wool

	Res.Df	RSS	Df	Sum of Sq	F	Pr(>F)
1	50	53.291				
2	48	46.089	2	7.201	3.75	0.03067 *

The p-value is quite low here, suggesting that there is evidence to reject the null hypothesis, and accept the alternative that the model with interaction is required (recall the reasoning: the p-value indicates that the observed F ratio is rather improbable under the null, suggesting that the null is not true).

Note that it is meaningless to try and remove main effects while leaving in a corresponding interaction, a point which will be elaborated shortly. For the moment this means that we are done, and have selected a model (the original one). So, the way that tension influences number of breaks is dependent on type of wool.

7.6.1 Follow up

Having selected the model, which says that the number of breaks depends on an interaction of wool type and tension, we would now want to examine the parameter estimates, with a view to comparing the effects of the different factor levels, or combinations of factor levels. Exactly what follow up analysis is appropriate depends on the questions being addressed: it isn't possible to give a universal recipe. Instead let's look at what is appropriate for the warpbreaks data.

For warpbreaks we would like to know the combination(s) of wool type and tension that minimize break-ages. From the summary of the `wm` fitted model object this is not all that easy to work out, because of the way that the identifiability constraints have changed the meanings of the remaining parameters. However, we can modify the identifiability constraints in order to get more interpretable parameters. In the current case we could actually choose to set $\alpha_i = \beta_j = \mu = 0$ and only leave the γ_{ij} terms in the model. This will mean that γ_{ij} becomes the expected number of breaks for the i^{th} wool type and the j^{th} tension: just what's needed.


```
> wma <- lm(breaks^0.5~tension:wool-1,data=warpbreaks)
> range(fitted(wm)-fitted(wma)) ### proof: it's the same model as before!
[1] -7.993606e-15  7.993606e-15
> summary(wma)
```

Call:

```
lm(formula = breaks^0.5 ~ tension:wool - 1, data = warpbreaks)
```

Residuals:

	Min	1Q	Median	3Q	Max
	-1.69410	-0.70129	0.01772	0.66046	1.81902

Coefficients:

	Estimate	Std. Error	t value	Pr(> t)
tensionL:woolA	6.5476	0.3266	20.05	<2e-16 ***
tensionM:woolA	4.8259	0.3266	14.78	<2e-16 ***
tensionH:woolA	4.8564	0.3266	14.87	<2e-16 ***
tensionL:woolB	5.2381	0.3266	16.04	<2e-16 ***
tensionM:woolB	5.2987	0.3266	16.22	<2e-16 ***
tensionH:woolB	4.3022	0.3266	13.17	<2e-16 ***

Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.9799 on 48 degrees of freedom

Multiple R-Squared: 0.9697, Adjusted R-squared: 0.9659

F-statistic: 255.8 on 6 and 48 DF, p-value: < 2.2e-16

Note the important point that this is not a different model to `wm`: just a different parameterization of the same model. We can immediately see that wool type B at high tension is estimated to give the lowest break rate. But given the uncertainty can we really be sure that this combination is better than wool A at medium or high tension? Such questions may be important if there are other considerations to take into account when choosing between wool types and tension levels (e.g. cost, wear rate of machinery, etc.)

One obvious thing to do would be to find a confidence interval for the difference in break rate between wool B at high tension, and wool A at medium tension. i.e. a confidence interval for $\gamma_{12} - \gamma_{23}$. To do this we need to find the standard deviation of $\hat{\gamma}_{12} - \hat{\gamma}_{23}$. The estimated covariance matrix for the γ_{ij} s is easily obtained

```
> vcov(wma)
```

	tensionL:woolA	tensionM:woolA	tensionH:woolA	tensionL:woolB
tensionL:woolA	0.1066880	0.0000000	0.0000000	0.0000000
tensionM:woolA	0.0000000	0.1066880	0.0000000	0.0000000
tensionH:woolA	0.0000000	0.0000000	0.1066880	0.0000000
tensionL:woolB	0.0000000	0.0000000	0.0000000	0.1066880
tensionM:woolB	0.0000000	0.0000000	0.0000000	0.0000000
tensionH:woolB	0.0000000	0.0000000	0.0000000	0.0000000

	tensionM:woolB	tensionH:woolB
tensionL:woolA	0.0000000	0.0000000
tensionM:woolA	0.0000000	0.0000000
tensionH:woolA	0.0000000	0.0000000
tensionL:woolB	0.0000000	0.0000000
tensionM:woolB	0.1066880	0.0000000
tensionH:woolB	0.0000000	0.1066880

It's diagonal!: the parameter estimators are independent! So $\hat{\sigma}_{\hat{\gamma}_{12}-\hat{\gamma}_{23}}^2 = 2 \times .106688 \simeq 0.2134$. Hence the required 95% CI is

$$4.8259 - 4.3022 \pm t_{.975,48} \times \sqrt{0.2134} = 0.52 \pm 0.93$$

So we really can't tell apart wool B at high tension and wool A at medium (or high) tension. Since the width of the confidence interval would be the same for any similar comparison, the interval half width (0.93) is known as the *least significant difference*: any difference below this is certainly not significant at the 5% level.

Actually, these follow up comparisons are a little bit tricky to do in a statistically rigorous manner. If you perform a large number of hypothesis tests then you'll eventually reject a null hypothesis by chance, even if it's true (in fact, testing at the 5% level you *must* reject 5% of correct null hypotheses). To avoid this problem people sometimes correct the significance level that they work at by dividing it by the number of hypotheses being tested. For example if you want to make 5 comparisons, testing at the 5% level, you actually test at the 1% level: the idea is to reduce the chances of incorrectly rejecting at least one null to 5%. This method is known as *Bonferroni correction*. In theory it sounds OK, but it implicitly assumes that the multiple tests are all independent, which they usually are not, in a linear modelling setting. It can also be awkward to decide how many tests you are really conducting: in the warpbreaks example we only looked at one test: but we did so after examining the results — this means that what we really did was rather like testing all 15 possible comparisons, but these can't possibly be independent. In general, therefore...

If at all possible: decide which comparisons are of interest in advance!

In the above example it was easy to get R to use a parameterization that was convenient for extracting the main result of interest. Sometimes it is necessary to work a little harder in order to get a convenient parameterization. The key is to choose the identifiability constraints in order to obtain an interpretable set of coefficients. This can be done in statistical software by selecting different *contrasts*. We will not go further into this here, see the course textbooks or `?contrasts` for more information. The `relevel` function in R is also very useful!

7.7 ANOVA tables

In R the `anova` function produces tabular output related to the F-ratio comparison of alternative models. It is important to know exactly what the entries in these *ANOVA tables*[†] mean, since they are a standard way of displaying results.

`anova` does different things depending on whether you call it with a single model argument, or several models. Consider the multiple model case first, using two of the warpbreaks models.

```
> anova(wm1,wm)
Analysis of Variance Table

Model 1: breaks^0.5 ~ wool + tension
Model 2: breaks^0.5 ~ wool + tension + tension:wool
  Res.Df    RSS Df Sum of Sq    F    Pr(>F)
1      50 53.291
2      48 46.089   2     7.201 3.75 0.03067 *
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Working across the columns of the ANOVA table...

`Res.Df` gives the residual degrees of freedom for each model: i.e. the number of data (n) less the number of model parameters (p_0 and p_1 , say).

`RSS` gives the residual sum of squares for each model (i.e. $RSS_0 = \|\mathbf{y} - \mathbf{X}_0\hat{\beta}_0\|^2$ and $RSS_1 = \|\mathbf{y} - \mathbf{X}_1\hat{\beta}_1\|^2$).

`Df` gives the *difference* in degrees of freedom between each model and the model above it in the table (from 2nd model onwards). i.e. $p_1 - p_0$ in the current case.

`Sum of Sq` gives the *difference* in residual sum of squares between each model and the model above it in the table (from 2nd model). In terms of section 4.3.2 this is $RSS_0 - RSS_1$.

`F` is the F-ratio test statistic for comparing each model to the model above it in the table. That is $(RSS_0 - RSS_1)/\{(p_1 - p_0)\hat{\sigma}^2\}$. Note that $\hat{\sigma}^2$ is always calculated from the largest model in the table.

[†] ANOVA stands for ANalysis Of VAriance.

Pr(>F) is the probability of getting an F-ratio at least as large as that in the F column if the model corresponding to the next row up in the table is correct. i.e. this is the p-value for testing the null hypothesis that the next model up is correct, against the larger alternative described by this row's model.

Only two models are compared in the example, but the definitions apply to any number of models from 2 upwards.

The second form of the ANOVA table occurs when only one model is supplied to `anova`. Here is an example.

```
> anova(wm)
Analysis of Variance Table

Response: breaks^0.5
          Df Sum Sq Mean Sq F value    Pr(>F)
wool       1  2.902    2.902   3.0222 0.088542 .
tension    2 15.892    7.946   8.2752 0.000817 ***
wool:tension 2  7.201    3.601   3.7500 0.030674 *
Residuals 48 46.089    0.960
```

In this case the table should be read from the bottom up.

- The `Residuals` row gives the residual degrees of freedom, residual sum of squares and residual variance estimate, $\hat{\sigma}^2$, for the full model (`breaks~wool+tension+wool:tension`).
- The `wool:tension` row gives the *difference* in degrees of freedom and residual sum of squares between the full model and a model without the interaction (i.e. `breaks~wool+tension`). In some sense these numbers are the degrees of freedom and sum of squares associated with the dropped interaction term. The `Mean Sq` column gives the difference in sum of squares divided by the difference in degrees of freedom (an example of an $(\text{RSS}_0 - \text{RSS}_1)/(p_1 - p_0)$ term in section 4.3.2 notation). The `F value` is then the `Mean Sq` over $\hat{\sigma}^2$: the F-ratio for testing the null hypothesis that the model without the interaction is adequate, against the alternative that the full model is correct. The corresponding p-value is the last entry on the row.
- The `tension` row compares the model with wool and tension main effects to a model with just a wool effect. (i.e compares `breaks~wool+tension` to `breaks~wool`). The interpretation of the entries is as for the `wool:tension` row, but note that all the differences are taken between the two models being compared, not between the wool only model and the full model. $\hat{\sigma}^2$ in the F-ratio statistic still comes from the full model/ last row of the table, however.
- The `wool` row compares a model with wool only to a model with just an intercept (i.e `breaks~wool` to `breaks~1`). The interpretation of the entries is as for the previous row.

When interpreting such tables it is important to note two things:

- If you conclude that the interaction of two factors is significant and therefore can not be dropped from the model, then that means that the two factors concerned are significant, no matter what the p-values are for the rows concerning the factors separately. This is because the p-values for the main effects are only meaningful if their interaction is not significant (more on this shortly).
- For most models, the ANOVA table will differ depending on the order in which terms are removed from the model: this is because estimators for model terms are generally not independent. Notable exceptions are models involving factor variables fitted to *balanced* data. In these cases the terms estimators are generally independent, so that the order in which the the model is reduced is immaterial: the `warppbreaks` data are an example of this. Data are *balanced* w.r.t. a factor or interaction if there are the same number of observations per level of the factor or interaction.

7.7.1 Example of erroneous dropping of a main effect

To emphasize the importance of reading ANOVA table from the bottom up, and of not trying to drop main effects while leaving in interactions, here is what happens if I attempt to remove the `wool` main effect, while leaving in the `wool:tension` interaction...

```
> wm2 <- lm(breaks^0.5~tension+tension:wool,warpbreaks)
> anova(wm2,wm)
Analysis of Variance Table
```

```
Model 1: breaks^0.5 ~ tension + tension:wool
Model 2: breaks^0.5 ~ wool + tension + tension:wool
  Res.Df    RSS Df Sum of Sq  F Pr(>F)
1     48 46.089
2     48 46.089  0 -4.974e-14
```

The models have exactly the same RSS and residual degrees of freedom, so `anova` can't do anything sensible. Actually they are really the same model, so there is nothing to test. The following demonstrates the equivalence:

```
> range(fitted(wm2)-fitted(wm))
[1] -6.217249e-15  7.105427e-15
```

all that differs between them is the identifiability constraints, and hence the interpretation of the parameters. By removing the `wool` main effect all we have done is to remove the need for one of the constraints on the interaction, so an extra parameter of the interaction now does the work of the missing main effect parameter.

7.7.2 `drop1` again

Actually `drop1` is often a more sensible function to use than `anova` with a single argument. `drop1` will never drop main effects if their interaction is present, and is always comparing the omission of a single term, to the full model, rather than successively dropping more and more terms. Here it is applied to the `wm` model ...

```
> drop1(wm,test="F")
Single term deletions

Model:
breaks^0.5 ~ wool + tension + tension:wool
              Df Sum of Sq    RSS    AIC F value    Pr(F)
<none>                46.089   3.446
wool:tension    2      7.201 53.291   7.286     3.75 0.03067 *
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

7.8 An agricultural field trial

Consider one more complicated example. An agricultural trial was conducted using 3 varieties of oats (“Golden Rain”, “Marvellous” and “Victory”) each grown at 4 levels of nitrogen fertiliser treatment (0.0, 0.2, 0.4 and 0.6 cwt/acre), with 6 replicates of each combination of variety and fertiliser treatment. To help reduce the problems with random differences in soil fertility, the experiment was arranged in 6 blocks. The experimental layout is shown schematically here:

		Block																	
		I		II		III		IV		V		VI							
Variety	V	0.2	0.6	M	0.0	0.2	V	0.4	0.0	G	0.0	0.2	M	0.4	0.0				
		0.0	0.4			0.6		0.4			0.6	0.4			0.6	0.2		0.2	0.6
	M	0.6	0.4		G	0.4		0.2	M		0.2	0.4		V	0.0	0.2	G	0.0	0.2
		0.0	0.2			0.6		0.0			0.0	0.6			0.6	0.4		0.6	0.0
	G	0.0	0.4		V	0.2		0.6	G		0.2	0.6		V	0.4	0.2	G	0.4	0.0
		0.6	0.2			0.0		0.4			0.4	0.0			0.4	0.6		0.2	0.6

Each block has been split into 3 ‘plots’ each containing one oat variety (order chosen randomly). Each plot is then divided into 4 ‘sub-plots’ each allocated one of the four fertiliser treatments (again in random order). One reason for utilising this design, rather than a randomised block design[‡] is practical, it is often easier to plant varieties out in large plots, and then apply the fertiliser treatments to the subplots. Another reason might be if the differences between nitrogen treatments are of particular interest. Since the different nitrogen treatments for any block/variety combination are so close together, random differences in soil fertility should be kept low, enabling quite precise comparisons to be made.

An appropriate model for the data from this experiment would involve main effects for nitrogen treatment and variety as well as a nitrogen-variety interaction, to account for the fact that response to nitrogen may vary with variety. A block effect would also be included, to account for the spatial variations in soil fertility that might be expected between blocks[§]. In fact the exact amount of nitrogen added to each plot is a known numeric quantity (one of 0, 0.2, 0.4 or 0.6), so there is an argument for treating it as a continuous predictor variable. On the other hand, if we don’t want to assume anything about the form of the response to nitrogen then we might also treat it as a factor variable. For the moment we’ll do the latter.

So, if y_{ijk} is the yield measured for the i^{th} variety with the j^{th} nitrogen level in the k^{th} block, then:

$$y_{ijk} = \mu + \alpha_i + \beta_j + \gamma_{ij} + \delta_k + \epsilon_{ijk}$$

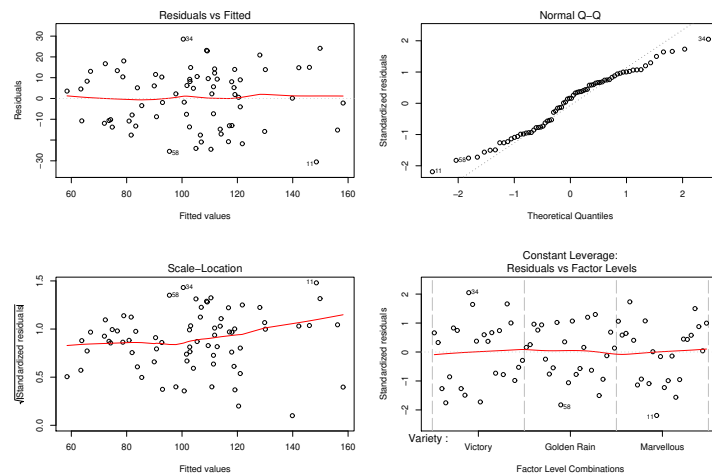
where the ϵ_{ijk} are i.i.d. $N(0, \sigma^2)$. Notice the implications of the structure of this model. Assuming that the block effect matters, the model would not be useful for predicting absolute yields, except in the particular blocks where the experiment was carried out, but it is useful for predicting the *differences* in yields between varieties and nitrogen levels, provided that the block effect really doesn’t interact with the other effects.

The following fits the model in R and produces the default checking plots.

```
> library(nlme) ## to access data
> oats <- Oats ## make copy of data to modify
> oats$nitro <- factor(oats$nitro) ## make nitro into a factor
> ## fit full model ...
> oml <- lm(yield~Variety*nitro+Block,oats)
> par(mfrow=c(2,2))
> plot(oml)
```

[‡]in which each combination of variety and fertiliser would be randomly allocated to one of 12 plots in a block

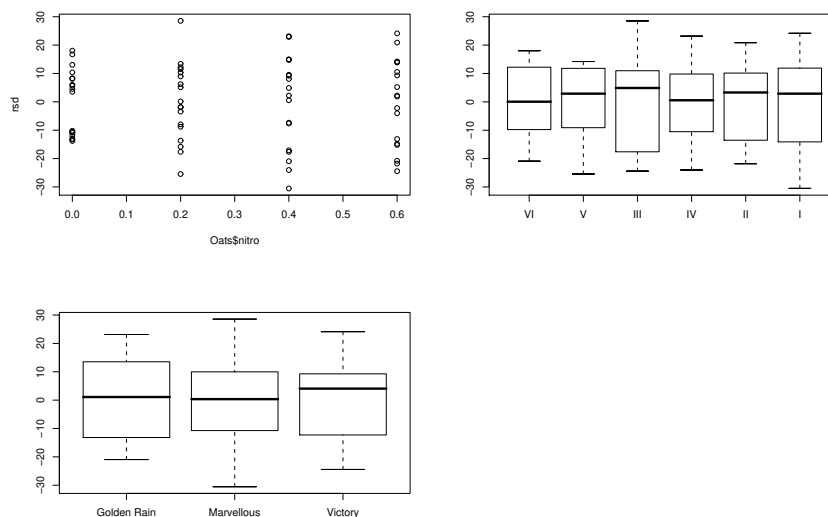
[§]Actually there is a strong argument for including a Block-Variety interaction, as a sort of surrogate for smaller scale spatial variability in soil conditions, but we really need some extensions of the linear model to handle this in a useful way.



...not too bad, but perhaps the QQ - plot is not quite right. However, simple transformations don't seem to fix the problem, so we'll have to live with this.

As with all models, residuals should also be plotted against the predictors.

```
> par(mfrow=c(2,2))
> rsd <- residuals(om1)
> plot(Oats$nitro, rsd)
> plot(oats$Block, rsd)
> plot(oats$Variety, rsd)
```



...again no cause for concern here (note the default plot when a factor is on the x axis).

Given that the linear model assumptions appear OK, we can consider dropping terms from the model...

```
> drop1(om1, test="F")
Single term deletions
```

Model:

```
yield ~ Variety * nitro + Block
      Df Sum of Sq    RSS      AIC F value    Pr(F)
<none>                13982.1    413.4
Block      5    15875.3 29857.3    458.0 12.4894 4.093e-08 ***
Variety:nitro 6      321.8 14303.8    403.0  0.2109    0.9719
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

So there is no evidence of a variety - nitrogen interaction; all the varieties seem to have the same response to nitrogen. There is strong evidence for a Block effect.

Dropping the interaction from the model...

```
> om2 <- lm(yield~Variety +nitro+Block,oats)
> drop1(om2,test="F")
Single term deletions
```

```
Model:
yield ~ Variety + nitro + Block
      Df Sum of Sq    RSS      AIC F value    Pr(F)
<none>                14304    403
Variety  2      1786 16090    407  3.8091    0.02762 *
nitro    3     20020 34324    460 28.4598 1.239e-11 ***
Block    5     15875 30179    447 13.5403 6.906e-09 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

So there is strong evidence for Block and nitrogen effects, and reasonable evidence for a variety effect as well.

Finally, consider the effect estimates.

```
> summary(om2)
...
Coefficients:
                Estimate Std. Error t value Pr(>|t|)
(Intercept)      79.917      4.420   18.079 < 2e-16 ***
VarietyMarvellous  5.292      4.420    1.197 0.235908
VarietyVictory    -6.875      4.420   -1.555 0.125058
nitro0.2          19.500      5.104    3.820 0.000315 ***
nitro0.4          34.833      5.104    6.824 4.64e-09 ***
nitro0.6          44.000      5.104    8.620 3.80e-12 ***
...
Residual standard error: 15.31 on 61 degrees of freedom
Multiple R-Squared:  0.7249,    Adjusted R-squared:  0.6797
F-statistic: 16.07 on 10 and 61 DF,  p-value: 1.153e-13
```

(I've edited out the block parameter estimates as they are of no real interest.) Clearly yield increases with nitrogen, although the effect seems not to be linear, so the initial decision to treat `nitro` as a factor was probably worthwhile. Marvellous seems to be a significantly better variety than Victory, although we can't really say whether its actually better than Golden Rain, given the Standard errors (you could work out confidence intervals for the differences, if you wanted).

8 How to approach an analysis

By now we have seen almost all the theory we need for practical analysis, and how to put much of it into practice. It's time to stand back, take stock, and think about some general principles for approaching any modelling task. It is not possible, or desirable, to produce a recipe that all analyses should follow, but here are some guidelines.

1. Always start by identifying the questions that you are trying to answer by analysing the data.
2. Always look at the data before fitting any model. Plot the data to get a feel for how variables are related. Check for obvious errors. If you can think of simple methods (e.g. plots) that will give you informal answers to your questions, use them before starting the formal model based analysis.
3. Now think about how you can use linear models to answer the questions of interest.
 - Sometimes at least the main features of the models and analysis are obvious from the question. For example, analysing clinical trial data to test whether a new drug is better than the standard treatment will obviously involve comparing models with and without the treatment factor, by hypothesis testing, and then estimating the size of any effect differences.
 - Other-times the question is much less prescriptive: as in the tyre wear example, answering the question of interest may involve finding a linear model that adequately captures the relationship between the response and predictors, but the question and wider context do not specify much about what model to use. In such cases you usually have to think about broad classes of models that might be sensible, and then use statistical model selection approaches to choose between them.
 - When formulating a model, ensure that you are clear what is the response, and what are the predictors.
 - Sometimes it is clear in advance that not all potential predictors should be included in the model. But be careful. Don't include variables that you know are irrelevant, but don't pre-exclude variables unless you are pretty certain about this. For example, not including a variable because economic theory suggests that it is unimportant is a recipe for losing a lot of money.
 - Try to avoid excluding predictors on the basis of statistical analysis of the predictors alone (i.e. without consideration of the response). For example, ancient textbooks, from the days before efficient and stable computation of linear models, sometimes suggest excluding a predictor in advance if it is highly correlated with another predictor that is included. When this advice was written it was computationally pragmatic, nowadays it is bizarre: the correlation between two predictors can't tell you about their effect on the response. So if possible, include both and let model selection decide which, if either, to remove. Of course this advice does not extend to the case when two predictors actually measure the same thing: then it is sensible to either combine them, or decide which one to drop.
4. Once you start fitting the models that are part of your analysis, make sure that you check that the modelling assumptions are met. Hypothesis tests, confidence intervals and so on don't work if they are not.
5. Always make sure that you interpret the results of your modelling in terms of the original question, and think carefully about any limitations that apply to the answer.

Finally, some principles for writing up a statistical analysis

1. Always clearly explain the context, and the questions being addressed.
2. Make sure that your analysis is repeatable by any statistician with access to the data, using the software of their choice. This means presenting models and results in universal mathematical language, not as R code and output.
3. Always explain the 'why' of your analysis as well as the 'what'.
4. Relate your conclusions back to the questions, and be careful to discuss the limitations of the approach taken.
5. Aim to be concise and useful, and at all costs avoid the sort of legalistic back covering approach in which every analysis report lists every possible thing that could ever go wrong with a linear model. Such an approach tells the reader nothing about what might be a real issue with the analysis actually being reported.

9 Beyond linear models

It is time to move on and consider some more general models inspired by the linear model.

1. The linear models covered so far admit a rich range of possibilities for modelling the expected value of the response variable, while permitting only the simplest possible model of variability about that mean. Many situations demand a richer structure to model the randomness in the response, and linear mixed models provide this by allowing a linear structure for the randomness that is similar to that allowed for the mean. Specifically,

$$\mathbf{y} = \mathbf{X}\boldsymbol{\beta} + \mathbf{Z}\mathbf{b} + \boldsymbol{\epsilon} \text{ where } \mathbf{b} \sim N(0, \psi(\boldsymbol{\theta})) \text{ and } \boldsymbol{\epsilon} \sim N(\mathbf{0}, \mathbf{I}\sigma^2).$$

\mathbf{Z} is a model matrix constructed in a similar way to \mathbf{X} , but \mathbf{b} is a vector of *random effects*. Their covariance matrix $\psi(\boldsymbol{\theta})$ typically has some relatively simple structure and is dependent on a parameter vector $\boldsymbol{\theta}$ which is the target of inference, alongside $\boldsymbol{\beta}$ and σ^2 . For example, suppose we want to model the time course of repeated blood pressure measurements of patients starting on different blood pressure treatments. As well as the dependence of blood pressure on treatment and time, included in $\mathbf{X}\boldsymbol{\beta}$ we should also allow for the differences between individual patients, and might model these by including a random effect for each patient, and perhaps also some random slope with respect to time for each patient: these random components are included in $\mathbf{Z}\mathbf{b}$. Our existing linear model theory is insufficient for inference with these *linear mixed models*.

2. In many situations a Gaussian distribution is inappropriate as a model for a response variable of interest, although a linear model structure might be appropriate for the expected value of the response, leading to a *generalized linear model* (GLM). For example a discrete response variable might follow a Poisson or binomial distribution, or a positive continuous response variable might be better described by a gamma distribution. Our existing theory is then insufficient: for a start all of these distributions break the linear model constant variance assumption.

So without actually straying very far from the linear model (these models all still have a univariate ‘response variable’, for example), we have two obvious motivations to develop inference methods that apply beyond the linear model. The general theory of *maximum likelihood estimation* provides the key to this.

9.1 Maximum likelihood estimation

The key idea of *maximum likelihood estimation* is simply this:

Parameter values that make the observed data appear relatively probable are more *likely* to be correct than parameter values that make the observed data appear relatively improbable.

For example, we would much prefer an estimate of parameter vector, $\boldsymbol{\theta}$, that assigned a probability density of 0.1 to our observed \mathbf{y} , according to the model, to an estimate for which the density was 0.00001.

So the idea is to judge the *likelihood* of parameter values using $f_{\boldsymbol{\theta}}(\mathbf{y})$, the model p.d.f. according to the given value of $\boldsymbol{\theta}$, evaluated at the observed data. Because \mathbf{y} is now fixed and we are considering the likelihood as a function of $\boldsymbol{\theta}$, it is usual to write the likelihood as $L(\boldsymbol{\theta}) \equiv f_{\boldsymbol{\theta}}(\mathbf{y})$. In fact, for theoretical and practical purposes it is usual to work with the log likelihood $l(\boldsymbol{\theta}) = \log L(\boldsymbol{\theta})$. The *maximum likelihood estimate* (MLE) of $\boldsymbol{\theta}$ is then

$$\hat{\boldsymbol{\theta}} = \underset{\boldsymbol{\theta}}{\operatorname{argmax}} l(\boldsymbol{\theta}).$$

In general, numerical methods will be needed to actually find $\hat{\boldsymbol{\theta}}$, but reliable and general methods are available.

There is more to maximum likelihood estimation than just its intuitive appeal. Under some regularity conditions and in the large sample limit as $n \rightarrow \infty$,

$$\hat{\boldsymbol{\theta}} \sim N(\boldsymbol{\theta}, \boldsymbol{\mathcal{I}}^{-1}), \tag{12}$$

where $\boldsymbol{\mathcal{I}} = -\mathbb{E}(\partial^2 l / \partial \boldsymbol{\theta} \partial \boldsymbol{\theta}^T)$ — the expected second derivative matrix of the negative log likelihood (also known as the *information matrix*). This turns out to be the best that can be achieved for an unbiased estimator (although MLEs are only unbiased in the large sample limit). These results are closely related to the fact that $\partial l / \partial \boldsymbol{\theta}$ has

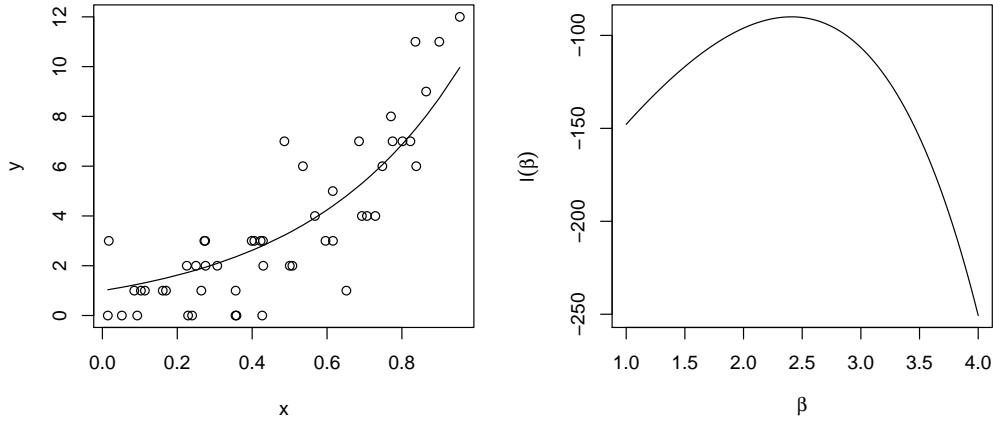


Figure 6: Left: 50 Poisson data, y , dependent on predictor data, x , and described by the model in section 9.1.1. The black curve is the maximum likelihood estimate of the Poisson mean as a function of x . Right: the log likelihood function for the model parameter β , computed using the data in the left panel and as described in the text.

expected value $\mathbf{0}$ and covariance matrix \mathcal{I} . In fact a similar result applies in terms of the observed (rather than expected) second derivative matrix. If $\hat{\mathcal{I}} = -\partial^2 l / \partial \boldsymbol{\theta} \partial \boldsymbol{\theta}^T$ (evaluated at $\hat{\boldsymbol{\theta}}$) then in the $n \rightarrow \infty$ limit,

$$\hat{\boldsymbol{\theta}} \sim N(\boldsymbol{\theta}, \hat{\mathcal{I}}^{-1}).$$

Another large sample result is also useful. Suppose that we want to compare two *nested* models, meaning that one model can be written as the other model subject to r restrictions on its parameters, $\boldsymbol{\theta}$, that can be written as $\mathbf{R}(\boldsymbol{\theta}) = \mathbf{0}$. Let $\hat{\boldsymbol{\theta}}_0$ denote the MLEs under the restriction. Then we can test $H_0 : \mathbf{R}(\boldsymbol{\theta}) = \mathbf{0}$ (i.e. test the null hypothesis that the restricted model is correct), by using the large sample result that

$$2\{l(\hat{\boldsymbol{\theta}}) - l(\hat{\boldsymbol{\theta}}_0)\} \sim \chi_r^2 \quad (13)$$

if the null hypothesis is correct. If the null hypothesis is not correct then the log likelihood ratio statistic on the left hand side will tend to be too large for consistency with a χ_r^2 distribution. To actually judge compatibility of data and null hypothesis we compute a p-value — the probability of a χ_r^2 random variable being at least as large as the observed test statistic. As usual a small p-value is evidence against the null hypothesis. Tests conducted using this result are known as *generalized likelihood ratio tests* (GLRT).

Hypothesis tests like the GLRT offer one way to choose between models, essentially favouring the simpler model unless the data offer sufficient evidence that this is not tenable and a more complicated model is needed. An alternative is to try to estimate how close the model is to the ‘true’ model and choose the model that is closest. This approach leads to the selecting the model that has the smallest value of the Akaike Information Criterion[¶],

$$\text{AIC} = -2l(\hat{\boldsymbol{\theta}}) + 2p \quad (14)$$

among the candidate models. p is the number of model parameters - the dimension of $\boldsymbol{\theta}$.

Deriving these results is somewhat beyond the scope of this course, but e.g. chapter 4 of Wood (2015) *Core Statistics* provides compact derivations.

9.1.1 A simple example

Consider the one parameter Poisson model:

$$y_i \underset{\text{ind}}{\sim} \text{Poisson}\{\exp(\beta x_i)\}$$

[¶]Akaike, with perhaps not entirely sincere modesty, called it ‘An Information Criterion’.

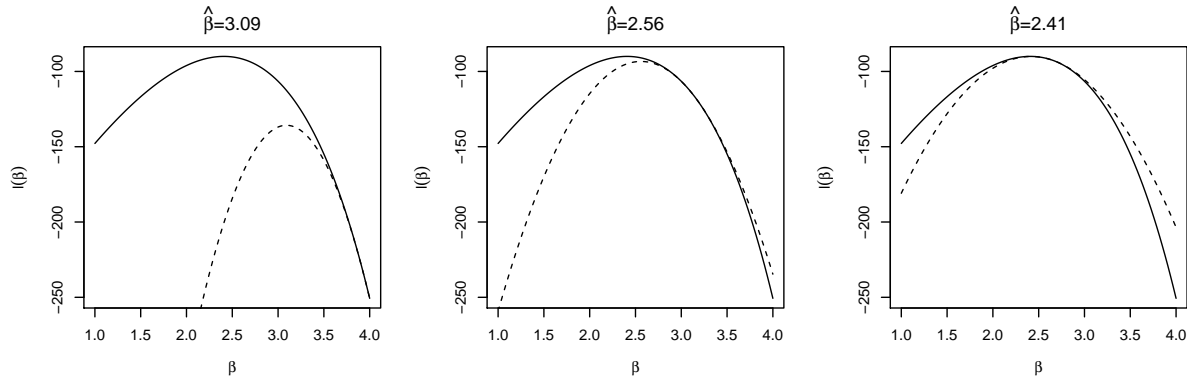


Figure 7: Three steps of Newton's method applied to the log likelihood for the Poisson example of section 9.1.1, starting on the left from $\beta = 4$. The dashed curve shows the approximating quadratic matching the value and first two derivatives of the log likelihood at the previous best estimate of β . The panel headings give the value, $\hat{\beta}$ maximising the approximating quadratic at each step. In this case no step length control is needed and the second derivative is always negative.

and suppose that we have n pairs of x_i, y_i data. Generally the probability function for a Poisson random variable with mean λ_i is $f(y_i) = \lambda_i^{y_i} \exp(-\lambda_i)/y_i!$. So for our model

$$f(y_i) = \exp(\beta x_i)^{y_i} \exp\{-\exp(\beta x_i)\}/y_i!$$

To compute the likelihood we need to compute the joint probability of all the data. Because our model says that the data are independent, this joint probability is just the product of the individual probabilities for each y_i , i.e. the likelihood function is just

$$L(\beta) = \prod_{i=1}^n \exp(\beta x_i)^{y_i} \exp\{-\exp(\beta x_i)\}/y_i!.$$

and repeatedly using the facts that $\log(a^b) = b \log(a)$ and $\log(ab) = \log(a) + \log(b)$, the log likelihood is therefore

$$l(\beta) = \sum_{i=1}^n y_i \beta x_i - \exp(\beta x_i) - \log y_i!$$

Figure 6 shows some x, y data (left) for which this model is appropriate, and plots the log-likelihood function for a range of β values (right). $\hat{\beta} = 2.41$ is the MLE, and the continuous curve overlaid on the left panel is the Poisson mean that this implies.

The key idea then, is that the log likelihood function provides us with a generic way of measuring how well a model fits data, and which parameter values do the best job at explaining the data. The simple Poisson example only considered a single parameter, but in principle we can compute the log likelihood for a model with as many parameters as we like, although we do need to be sure that we have sufficient data if we are intending to estimate large numbers of parameters.

9.1.2 Practical likelihood maximization

It is rarely possible to find explicit expressions for the maximum likelihood estimates, and instead we must maximize the log likelihood numerically. Newton's method is the gold standard for doing this, and works by optimizing successive quadratic approximations to the log likelihood. We start with an initial parameter value guess, and try to obtain a quadratic approximation to the log-likelihood that behaves like the log-likelihood itself in the vicinity of that guess. To do this we can use a Taylor approximation to the log-likelihood, discarding the terms above second

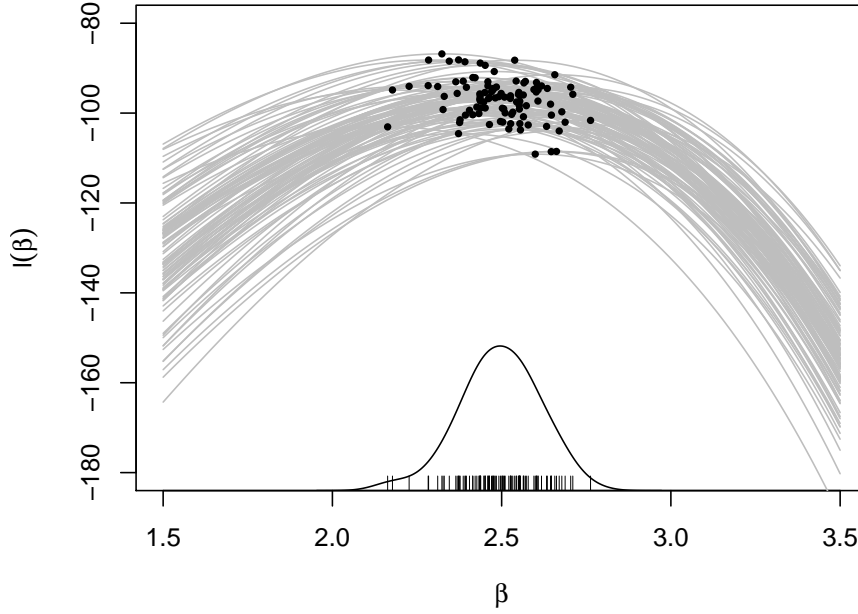


Figure 8: Log likelihood functions (grey) for 100 replicates of the section 9.1.1 Poisson dataset, when $\beta = 2.5$. The black dots show the maximum of each curve. The corresponding MLEs, $\hat{\beta}$, are shown as vertical bars on the x-axis (a rug plot). The continuous black curve is a (rescaled and vertically shifted) kernel density estimate of the distribution of $\hat{\beta}$ based on the shown 100 estimates. Notice how the distribution of $\hat{\beta}$ is approximately normal, with mean at the true value, $\beta = 2.5$, although there is zero probability that $\hat{\beta} = 2.5$ exactly.

order. That is we evaluate the log likelihood and its first two derivatives[‡] at the parameter value, and then find the unique quadratic function which also has this value and derivatives at the parameter value. Hopefully the approximation has a maximum close to the maximum of the log-likelihood itself, and it is easy to write down an explicit form for the maximizer of the approximating quadratic, using this as our updated parameter guess. Iterating this procedure we eventually reach the maximum of the log likelihood itself.

There are two modifications required to guarantee that Newton's method converges to the MLE. Firstly we need to ensure that the approximating quadratic is one that actually has a maximum (as opposed to a minimum, or even, in more than one dimension, a saddlepoint). In one dimension this is a matter of checking that the second derivative is negative and replacing it with a negative number if it isn't. In more dimensions there is also an equivalent fix (check that the negative of the second derivative matrix is positive definite, and if necessary modify it to force this condition to hold). The second modification is to check that the proposed change in parameter values actually increases the log likelihood itself. If it doesn't we just move the parameter back towards the previous parameter guess, until the log likelihood is increased at the new values. With these modifications Newton's method will find a maximum of the log likelihood, provided it has one.

Figure 7 illustrates Newton's method for the case of the single parameter Poisson regression model. In this case the maximum is reached in 3 steps (to graphical accuracy). Notice how the final quadratic matches the log likelihood in the vicinity of the MLE.

9.1.3 Illustrating $\hat{\theta}$ uncertainty

Figure 8 shows log likelihood curves for 100 replicates of the data shown in the left panel of figure 6, when the true β is 2.5, along with the corresponding MLEs, $\hat{\beta}$, and the corresponding non-parametric estimate of the distribution of $\hat{\beta}$. Notice how the likelihood function and $\hat{\beta}$ vary from replicate to replicate, but how the distribution of $\hat{\beta}$ is approximately normal and centred on the true β value of 2.5. This is exactly what (12) implies will happen.

[‡]first derivative vector and second derivative matrix in usual case of a parameter vector.

10 Introducing GLMs

A linear model is a statistical model that can be written

$$y_i = \mathbf{X}_i\boldsymbol{\beta} + \epsilon_i, \quad \epsilon_i \underset{\text{i.i.d.}}{\sim} N(0, \sigma^2)$$

where y_i is a *response variable*, \mathbf{X}_i is the i th row of a model matrix with elements usually depending on some *predictor variables*^{**}, the ϵ_i are random variables. $\boldsymbol{\beta}$ is a vector of unknown parameters, and the purpose of statistical inference with a linear model is to learn about $\boldsymbol{\beta}$ from the data.

An exactly equivalent way of writing the linear model is,

$$\mathbb{E}(y_i) \equiv \mu_i = \mathbf{X}_i\boldsymbol{\beta}, \quad y_i \underset{\text{indep.}}{\sim} N(\mu_i, \sigma^2).$$

Generalized linear models extend linear models by allowing some non-linearity in the model structure and much more flexibility in the specification of the distribution of the response variable y_i . Specifically, a GLM is a statistical model that can be written as

$$\mathbb{E}(y_i) \equiv \mu_i = \gamma(\mathbf{X}_i\boldsymbol{\beta}), \quad y_i \underset{\text{indep.}}{\sim} \text{EF}(\mu_i, \phi),$$

where γ is any smooth monotonic function and $\text{EF}(\mu_i, \phi)$ is some *Exponential family* distribution : examples include the Poisson, Gaussian (normal), binomial and gamma distributions. A feature of exponential family distributions is that their shape is largely determined by their mean, μ_i , and possibly one other *scale* parameter, usually denoted ϕ (e.g. for the normal distribution $\phi = \sigma^2$, for the Poisson, $\phi = 1$). For such distributions it is always possible to find a *variance function* V of μ_i such that

$$\text{var}(y_i) = V(\mu_i)\phi.$$

It is also possible to avoid specifying a distribution and simply specify V , using the theory of *quasi-likelihood*.

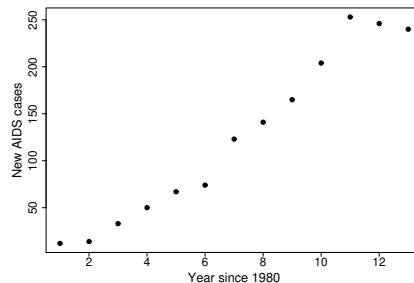
For historical reasons it is usual to write GLMs in terms of the (smooth monotonic) *link function*, g , which is the inverse function of γ . i.e.

$$g(\mu_i) = \mathbf{X}_i\boldsymbol{\beta}, \quad y_i \underset{\text{indep.}}{\sim} \text{EF}(\mu_i, \phi).$$

Examples of commonly used link functions are the log, square root and ‘logit’ (log odds ratio) functions (see later). The model is written this way because statisticians were (and are) used to thinking about models for transformed response data (i.e. transformed y_i). However it is important to realize that modelling some data using a log link (for example) is *very* different to modelling $\log(y_i)$ using a linear model. Note that $\mathbf{X}\boldsymbol{\beta}$ is known as the *linear predictor* (and is often given the symbol η).

10.1 Simple single covariate examples of GLMs

Example 1: AIDS in Belgium.



^{**}also referred to as *explanatory variables* or *covariates*.

The above figure shows new AIDS cases each year in Belgium, at the start of the epidemic. In the early stages of an epidemic an exponential increase model is often appropriate, and a GLM can be used to fit such a model. If y_i is the number of new AIDS cases per year and t_i denotes the number of years since 1980, then a suitable model for the data might be,

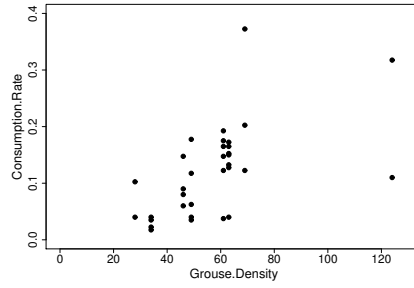
$$\mathbb{E}(y_i) \equiv \mu_i = \gamma e^{\alpha t_i}, \quad y_i \underset{\text{indep.}}{\sim} \text{Poi}(\mu_i).$$

Taking logs of both sides of the above equation and defining $\beta_1 \equiv \log(\gamma)$ and $\beta_2 \equiv \alpha$, we can re-write the model as

$$\log(\mu_i) = \beta_1 + t_i \beta_2, \quad y_i \underset{\text{indep.}}{\sim} \text{Poi}(\mu_i),$$

which is clearly a GLM with a log link and a model matrix whose i^{th} row is $\mathbf{X}_i = [1, t_i]$.

Example 2: Hen harriers and Grouse



The above plot shows the daily consumption rate of Grouse by Hen Harriers (a type of bird of prey) plotted against the density of Grouse on various Grouse moors. If c_i denotes consumption rate and d_i is grouse density, then ecological theory suggests a ‘saturating’ model for the data

$$\mathbb{E}(c_i) \equiv \mu_i = \frac{\alpha d_i^3}{\delta + d_i^3}, \quad c_i \underset{\text{indep.}}{\sim} \text{Gamma},$$

where α and δ are parameters to be estimated. Using the ‘inverse’ link function we get

$$\frac{1}{\mu_i} = \frac{1}{\alpha} + \frac{\delta}{\alpha} \frac{1}{d_i^3}, \quad c_i \underset{\text{indep.}}{\sim} \text{Gamma}.$$

So defining new parameters $\beta_1 = 1/\alpha$ and $\beta_2 = \delta/\alpha$ we get the GLM

$$\frac{1}{\mu_i} = \beta_1 + \beta_2 \frac{1}{d_i^3}, \quad c_i \underset{\text{indep.}}{\sim} \text{Gamma}.$$

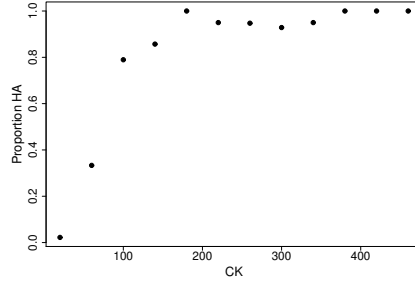
a GLM with an inverse link, Gamma distribution for the response, and model matrix with i^{th} row $\mathbf{X}_i = [1, d_i^{-3}]$.

Example 3: Heart Attacks and Creatine Kinase

The following data are from a study examining the efficacy of blood creatine kinase levels as a diagnostic when patients present with symptoms that may indicate a heart attack.

CK level	20	60	100	140	180	220	260	300	340	380	420	460
Heart Attack	2	13	30	30	21	19	18	13	19	15	7	8
Not Heart Attack	88	26	8	5	0	1	1	1	1	0	0	0

Here is a plot of the proportion of patients who subsequently turned out to have had a heart attack, against their blood CK levels on admission to hospital.



A convenient model that captures the saturating nature of the relationship between the observed proportions, p_i , and the CK levels, x_i is the ‘logistic’ model

$$\mathbb{E}(p_i) = \frac{e^{\beta_1 + \beta_2 x_i}}{1 + e^{\beta_1 + \beta_2 x_i}}$$

If y_i is the number of heart attack victims observed out of N_i patients with CK level x_i then

$$\mu_i \equiv \mathbb{E}(y_i) = N_i \mathbb{E}(p_i) = \frac{N_i e^{\beta_1 + \beta_2 x_i}}{1 + e^{\beta_1 + \beta_2 x_i}}$$

and treating the patients as independent we have $y_i \sim \text{bin}(\mu_i/N_i, N_i)$. This model doesn’t look linear, but if we apply the ‘logit’ link function to both sides it becomes,

$$\log\left(\frac{\mu_i}{N_i - \mu_i}\right) = \beta_1 + \beta_2 x_i,$$

which is clearly a GLM.

Example 4: Linear models!

Any linear model is just a special case of a GLM. The link function is the ‘identity’ link and the response distribution is Gaussian.

11 Inference with GLMs

Inference with GLMs is based on the theory of maximum likelihood estimation. This section shows how this works in general for any GLM, from which emerges a nice link back to linear models.

11.1 The exponential family of distributions

The response variable in a GLM can have any distribution from the *exponential family*. A distribution belongs to the exponential family of distributions if its probability density function, or probability mass function, can be written as

$$f_\theta(y) = \exp[\{y\theta - b(\theta)\}/a(\phi) + c(y, \phi)],$$

where b , a and c are arbitrary functions, ϕ an arbitrary *scale parameter*, and θ is known as the *canonical parameter* of the distribution. For GLMs, θ is completely determined by β . For example,

1. the normal distribution $N(\mu, \sigma^2)$ is a member of the exponential family with $\theta = \mu$, $b(\theta) = \theta^2/2 \equiv \mu^2/2$, $a(\phi) = \phi = \sigma^2$ and $c(\phi, y) = -y^2/(2\phi) - \log(\sqrt{\phi}2\pi) \equiv -y^2/(2\sigma^2) - \log(\sigma\sqrt{2\pi})$.
2. the Poisson distribution $\text{Poi}(\mu)$ is an exponential family distribution with $\theta = \log(\mu)$, $a(\phi) = \phi = 1$, $b(\theta) = \exp(\theta) = \mu$ and $c(y, \phi) = -\log(y!)$.

In both these cases $a(\phi) = \phi$ and in fact generally we only need the case where a is a linear function so that $a(\phi) = \phi/\omega$, for known constant ω – so let's assume this going forward.

Let the log likelihood for a single observation y be $l(\theta) = \log f_\theta(y)$. Then because $\mathbb{E}(\partial l / \partial \theta) = 0$

$$\frac{\partial l}{\partial \theta} = \omega \{y - b'(\theta)\} / \phi \Rightarrow \mu = \mathbb{E}(y) = b'(\theta). \quad (15)$$

$b'(\theta)$ is one-to-one, so there is a one to one correspondence between μ and θ . Further, from general likelihood theory the variance of $\partial l / \partial \theta$ is \mathcal{I} , so that $\mathbb{E}(\partial^2 l / \partial \theta^2) = -\mathbb{E}[(\partial l / \partial \theta)^2]$. Hence on differentiating $\partial l / \partial \theta$ again,

$$\omega b''(\theta) / \phi = \mathbb{E}[\{Y - b'(\theta)\}^2] \omega^2 / \phi^2, \Rightarrow \text{var}(y) = \phi b''(\theta) / \omega.$$

Given the one-to-one correspondence between μ and θ we can also define *variance function* $V(\mu) = b''(\theta) / \omega$, and write $\text{var}(y) = V(\mu)\phi$. For example in the Normal case, $V(\mu) = 1$ and in the Poisson case $V(\mu) = \mu$.

11.2 GLM fitting

To fit GLMs, consider the log likelihood as function of the model parameter β (which control the expected value of the response, y_i via $g(\mu_i) = \mathbf{X}_i \beta$, with μ_i in turn controlling the canonical parameter θ_i of y_i 's distribution),

$$l(\beta) = \sum_{i=1}^n \omega_i \{y_i \theta_i - b_i(\theta_i)\} / \phi + c_i(\phi, y_i).$$

Differentiating and recalling (15) we get

$$\frac{\partial l}{\partial \beta_j} = \frac{1}{\phi} \sum_{i=1}^n \omega_i \left(y_i \frac{\partial \theta_i}{\partial \beta_j} - \mu_i \frac{\partial \theta_i}{\partial \beta_j} \right).$$

By the chain rule

$$\frac{\partial \theta_i}{\partial \beta_j} = \frac{d\theta_i}{d\mu_i} \frac{\partial \mu_i}{\partial \beta_j}.$$

Differentiating (15) again implies $d\theta_i / d\mu_i = 1 / b''(\theta_i)$, while $\partial \mu_i / \partial \beta_j$ follows directly from $g(\mu_i) = \mathbf{X}_i \beta$, so

$$\frac{\partial l}{\partial \beta_j} = \frac{1}{\phi} \sum_{i=1}^n \frac{[y_i - b'_i(\theta_i)]}{b''_i(\theta_i) / \omega_i} \frac{\partial \mu_i}{\partial \beta_j} = \frac{1}{\phi} \sum_{i=1}^n \frac{(y_i - \mu_i)}{V(\mu_i) g'(\mu_i)} X_{ij}.$$

To maximize $l(\beta)$, we find the values $\hat{\beta}$ for which these derivative are 0. Note that $\hat{\beta}$ will not depend on ϕ .

In practice maximization by Newton's method also requires the second derivatives of the log likelihood. So differentiating again we get,

$$\frac{\partial^2 l}{\partial \beta_j \partial \beta_k} = -\frac{1}{\phi} \sum_{i=1}^n \left\{ \frac{X_{ik} X_{ij}}{g'(\mu_i)^2 V(\mu_i)} + \frac{(y_i - \mu_i) X_{ij} X_{ik}}{g'(\mu_i)^2 V(\mu_i)} \left(\frac{g''(\mu_i)}{g'(\mu_i)} + \frac{V'(\mu_i)}{V(\mu_i)} \right) \right\}$$

and hence

$$\mathbb{E} \left(\frac{\partial^2 l}{\partial \beta_j \partial \beta_k} \right) = -\frac{1}{\phi} \sum_{i=1}^n \frac{X_{ik} X_{ij}}{g'(\mu_i)^2 V(\mu_i)}.$$

Defining $w_i^{-1} = g'(\mu_i)^2 V(\mu_i)$ and $\mathbf{W} = \text{diag}(w_i)$, we therefore have that $\mathcal{I} = -\mathbb{E}(\partial^2 l / \partial \beta \partial \beta^\top) = \mathbf{X}^\top \mathbf{W} \mathbf{X} / \phi$. Hence the general result (12) becomes

$$\hat{\beta} \sim N(\beta, (\mathbf{X}^\top \mathbf{W} \mathbf{X})^{-1} \phi). \quad (16)$$

Applying Newton's method to find $\hat{\beta}$ amounts to iterating (from some initial guess $\hat{\beta}^0$)

$$\hat{\beta}^{k+1} = \hat{\beta}^k - \left(\frac{\partial^2 l}{\partial \beta \partial \beta^\top} \right)^{-1} \frac{\partial l}{\partial \beta}$$

until successive β^k do not change (all derivatives are evaluated at $\hat{\beta}^k$). A useful feature of Newton's method is that it converges just as well with the second derivative matrix replaced by its expectation. So if we define $\mathbf{G} = \text{diag}\{g'(\mu_i)\}$ then the Newton iteration becomes (evaluating \mathbf{W} , \mathbf{G} and μ at $\hat{\beta}^k$)

$$\begin{aligned}\hat{\beta}^{k+1} &= \hat{\beta}^k - (\mathbf{X}^\top \mathbf{W} \mathbf{X})^{-1} \mathbf{X}^\top \mathbf{W} \mathbf{G} (\mathbf{y} - \mu) \\ &= (\mathbf{X}^\top \mathbf{W} \mathbf{X})^{-1} \mathbf{X}^\top \mathbf{W} \{ \mathbf{G} (\mathbf{y} - \mu) + \mathbf{X} \beta^k \} \\ &= (\mathbf{X}^\top \mathbf{W} \mathbf{X})^{-1} \mathbf{X}^\top \mathbf{W} \mathbf{z}\end{aligned}$$

where $z_i = g'(\mu_i)(y_i - \mu_i) + \eta_i$ and $\eta_i = \mathbf{X}_i \beta$ (again evaluated at β^k). So $\hat{\beta}^{k+1}$ is recognisable as the minimizer of the weighted sum of squares, $\sum_i w_i (z_i - \mathbf{X}_i \beta)^2$.

Hence maximum likelihood estimation for a GLM is achieved using the iteratively re-weighted least squares (IRLS) algorithm.

1. Set $k = 0$, $\mu_i^0 = y_i + \Delta_i$ and $\eta_i^0 = g(\mu_i^0)$, where $\Delta_i = 0$ usually, but may be a small constant if needed to ensure that η_i^0 is well defined and finite. Then iterate the following two steps to convergence.
2. Form $z_i = g'(\mu_i^k)(y_i - \mu_i^k) + \eta_i^k$ and $w_i = \{g'(\mu_i^k)^2 V(\mu_i^k)\}^{-1}$.
3. Find $\hat{\beta}^{k+1} = \text{argmin}_{\beta} \sum_i w_i (z_i - \mathbf{X}_i \beta)^2$, by standard linear model methods. Increment k .

11.3 GLM checking and inference

Model checking for GLMs is performed using residuals, in the same way as for linear models. The difference is that the distribution of GLM residuals will depend on the response distribution used, which makes raw residuals difficult to interpret. For this reason residuals are usually standardized, so that they behave somewhat like the residuals from a linear model. For example, a simple standardization is to divide each residuals by its model predicted scaled standard deviation, so that all standardized residuals should have the same variance, if the model is correct. i.e.

$$\hat{\epsilon}_i = \frac{y_i - \hat{\mu}_i}{\sqrt{V(\hat{\mu}_i)}}$$

— these are called ‘Pearson residuals’.

When ϕ is known (16) can be used directly for finding confidence intervals for β_i s or testing hypotheses about them. When ϕ is unknown then an estimator is needed, and basing this on the sum of squares of the Pearson residuals results in the Pearson estimator,

$$\hat{\phi} = \frac{1}{n-p} \sum_{i=1}^n \frac{(y_i - \hat{\mu}_i)^2}{V(\hat{\mu}_i)},$$

where p is the number of parameters (dimension of β). This can then be plugged in to (16).

Another quantity that plays a similar role to the residual sum of squares in a linear model is the model *deviance*. This is defined as

$$D = 2\{l_{\text{sat}} - l(\hat{\beta})\}\phi$$

where the *saturated log likelihood*, l_{sat} , is the highest value that the log likelihood could have taken for the current data if there was one free parameter for each observation, y_i . Like the residual sum of squares for a linear model, the deviance gets smaller the more closely the model fits the data and does not depend on ϕ (which cancels). The deviance can provide alternative scaled residuals to the Pearson residuals, by writing it as a sum of terms relating to each y_i , $D = \sum_i d_i$. By analogy with the relationship between linear model residuals and the residual sum of squares, we can define *deviance residuals* whose sum of squares is the deviance: $\hat{e}^d = \text{sign}(y_i - \hat{\mu}_i) \sqrt{d_i}$.

The related *scaled deviance* is defined as $D^* = D/\phi$. If we want to compare two nested GLMs using the GLRT (13), then the test statistic can be re-written in terms of D^* to obtain the approximate result that under the null hypothesis that the simpler model is correct,

$$D_0^* - D_1^* \sim \chi_{p_1 - p_0}^2,$$

where D_j^* and p_j are the scaled deviance and number of parameters for model j . If ϕ is unknown we can use this result with $\hat{\phi}$ plugged in, but a slightly better approximation is that under the null

$$\frac{(D_0 - D_1)/(p_1 - p_0)}{D_1/(n - p_1)} \sim F_{p_1 - p_0, n - p_1}. \quad (17)$$

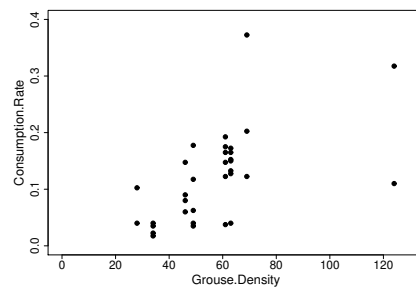
(the same result holds substituting the scaled deviance since the scale parameter cancels). AIC is obviously usable as an alternative model selection approach.

12 glm in R

R^{††} has a function called `glm` for fitting GLMs to data. `glm` functions much like `lm`, except that the rhs of the model formula now defines the way in which the link function of the expected response depends on the predictor variables. In addition you have to tell `glm` what response variable distribution to use, and what link function: this is done using the `family` argument, as we will see.

To see `glm` in action, consider the hen harrier data again. The data are stored in a data frame called `harrier`. First plot them.

```
harrier <- harrier[harrier$Consumption.Rate!=0,] ## not interesting
with(harrier, plot(Grouse.Density, Consumption.Rate, ylim=c(0, .4), xlim=c(0, 130)))
```



Now fit the model discussed previously:

```
hm <- glm(Consumption.Rate ~ I(1/Grouse.Density^3), Gamma(link="inverse"), data=harrier)
```

Notice how the identity function `I()` is used on the rhs of the model formula in order to use Grouse density to the power minus 3 as a predictor: this is necessary because of the special meaning of various ordinary arithmetic operators within model formulae (e.g. `+` means ‘and’ not ‘the sum of’ in a model formula). The second argument to `glm` specifies the distribution (here `Gamma`) and the link function to use (here `"inverse"`). The final argument indicates where to find the columns of data referred to in the formula.

`glm` returns a *fitted model object* (which has been stored in `hm`). Typing its name will cause a small summary of the object to be displayed.

```
> hm
```

```
Call: glm(formula = Consumption.Rate ~ I(1/Grouse.Density^3),
  family = Gamma(link = "inverse"), data = harrier)
```

```
Coefficients:
```

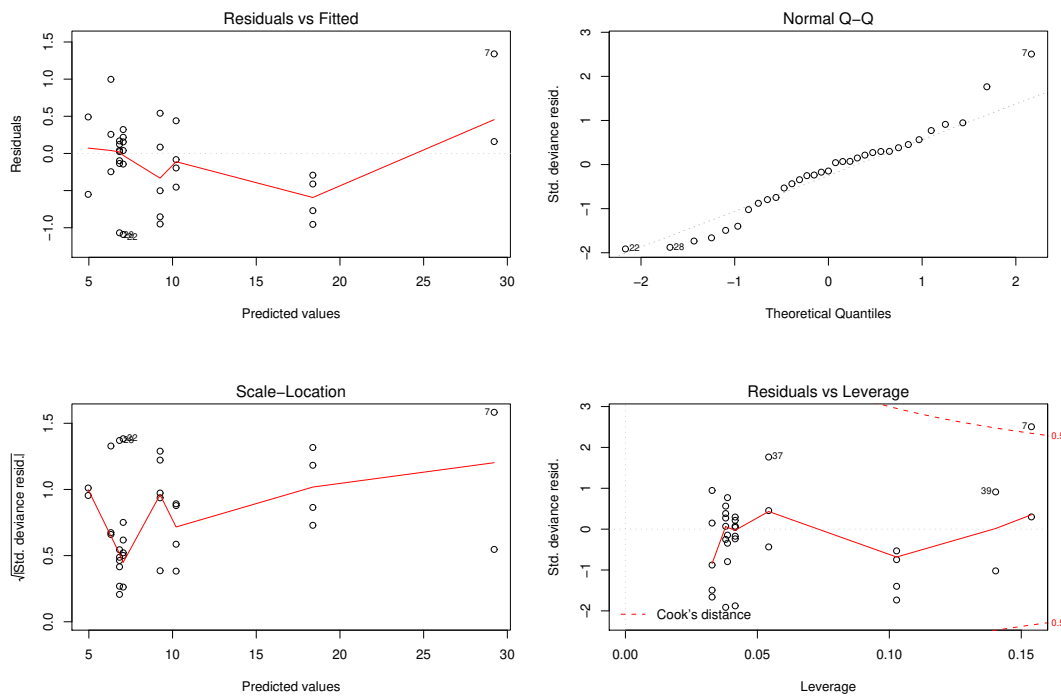
```
(Intercept)    I(1/Grouse.Density^3)
  4.676e+00         5.386e+05
```

^{††}R is available free from <http://cran.r-project.org>

```
Degrees of Freedom: 32 Total (i.e. Null); 31 Residual
Null Deviance:      16.17
Residual Deviance:  10.47      AIC: -92.38
```

As with any model, we should check residuals before proceeding.

```
par(mfrow=c(2,2)) ## get all plots on one page
plot(hm) ## plot residuals in various ways
```



These residual plots are interpreted in much the same way as the equivalent plots for a linear model, with two differences: i) If the response distribution is not normal/Gaussian then we don't expect the normal QQ plot to show an exact straight line relationship and (ii) if our response is binary then checking is more difficult, with many plots being effectively useless.

For the harrier model there are some clear patterns in the residuals, with the data for some densities being entirely above or below the fitted line. For this simple example, it can help to plot model predictions and data on the same plot. To make predictions at a series of grouse densities, we can use the `predict` function.

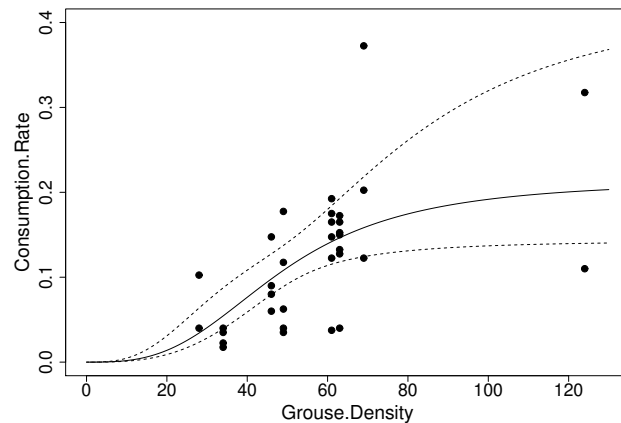
```
pd <- data.frame(Grouse.Density=0:130) ## data at which to predict
fv <- predict(hm,pd,type="response")    ## get model predictions
```

Note the `type` argument to `predict`, indicating that we want the predictions to be made on the original response scale, not the link transformed scale (the default). Now we can produce a plot

```
par(mfrow=c(1,1))
with(harrier,plot(Grouse.Density,Consumption.Rate,ylim=c(0,.4),xlim=c(0,130)))
lines(0:130,fv)
```

Actually it would be good to add approximate 95% CIs to the plot. Again `predict` is used, but this time standard errors for each prediction are also produced (on the link transformed scale). The following adds CIs to the plot.

```
lp <- predict(hm, pd, se=TRUE)
lines(0:130, 1/(lp$fit+2*lp$se.fit), lty=2)
lines(0:130, 1/(lp$fit-2*lp$se.fit), lty=2)
```



From this plot it is clear why we have patterns in the residuals, and only a very complex dependence of consumption rate on density would cure it. Since the different densities actually relate to different grouse moors, it is probable that there are some other moor-to-moor differences that should really be included in this model — perhaps relating to the alternative hen-harrier food available at each moor. Clearly some more work is required here.

Is the cubic dependence on density really the best model for these data? There are various ways of addressing this question, but one of the simplest is to try alternative powers, and compare the AICs of the alternative model fits. For example.

```
> hm1 <- glm(Consumption.Rate~I(1/Grouse.Density^2), Gamma(link="inverse"),
+ data=harrier)
> hm2 <- glm(Consumption.Rate~I(1/Grouse.Density), Gamma(link="inverse"),
+ data=harrier)
> AIC(hm, hm1, hm2)
      df      AIC
hm      3 -92.38289
hm1     3 -94.17016
hm2     3 -92.63887
```

So AIC actually supports the model with a quadratic dependence on density. Such a model saturates more slowly than the original model, but also has problematic residual plots. Finally, for a larger summary of a model, use the `summary` function...

```
> summary(hm1)
```

Call:

```
glm(formula = Consumption.Rate ~ I(1/Grouse.Density^2),
     family = Gamma(link = "inverse"), data = harrier)
```

Deviance Residuals:

Min	1Q	Median	3Q	Max
-1.05990	-0.43304	-0.01186	0.20319	1.12450

Coefficients:

	Estimate	Std. Error	t value	Pr(> t)
(Intercept)	2.627	1.315	1.998	0.054558 .

```
I(1/Grouse.Density^2) 17674.876    4359.132    4.055 0.000314 ***
---
```

```
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
(Dispersion parameter for Gamma family taken to be 0.3044807)
```

```
Null deviance: 16.1657  on 32  degrees of freedom
Residual deviance:  9.9461  on 31  degrees of freedom
AIC: -94.17
```

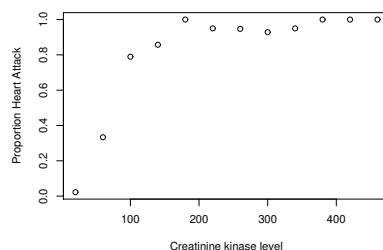
```
Number of Fisher Scoring iterations: 5
```

The output here is much the same as that for a linear model, except that residual sums of squares are replaced by *deviances*, of which more later. Note also that an estimate of the scale parameter of the gamma distribution is provided: for some other distributions this parameter is just the known constant 1.

12.1 Heart attack example

Again consider the heart attack data and model from section 10.1. First read in the data and plot them.

```
ck<- c(20 , 60 , 100 , 140 , 180 , 220 , 260 , 300 , 340 , 380 , 420 , 460)
ha <- c(2 , 13 , 30 , 30 , 21 , 19 , 18 , 13 , 19 , 15 , 7 , 8 )
ok <- c( 88 , 26 , 8 , 5 , 0 , 1 , 1 , 1 , 1 , 0 , 0 , 0)
heart <- data.frame(ck=ck,ha=ha,ok=ok)
p<-heart$ha/(heart$ha+heart$ok)
plot(heart$ck,p,xlab="Creatinine kinase level",
     ylab="Proportion Heart Attack")
```



Recall that our basic model for these data is that, if y_i is the number of heart attack victims out of N_i patients at CK level x_i , then

$$y_i \sim \text{binom}(\mu_i/N_i, N_i)$$

where $\mathbb{E}(y_i) \equiv \mu_i$, $g(\mu_i) = \beta_0 + \beta_1 x_i$ and g is the ‘logit-link’

$$g(\mu_i) = \log \left(\frac{\mu_i}{N_i - \mu_i} \right),$$

When using binomial models, we need to somehow supply the model fitting function with information about N_i as well as y_i . R offers two ways of doing this with `glm`.

1. The response variable can be the observed proportion of successful binomial trials, in which case an array giving the number of trials must be supplied as the `weights` argument to `glm`. For binary data, no weights vector need be supplied, as the default weights of 1 suffice.
2. The response variable can be supplied as a two column array, in which the first column gives the number of binomial ‘successes’, and the second column is the number of binomial ‘failures’.

For the current example the second method will be used. Supplying 2 arrays on the r.h.s. of the model formula involves using `cbind`. Here is a `glm` call which will fit the heart attack model:

```
> mod.0 <- glm(cbind(ha,ok)~ck,family=binomial(link=logit),
+ data=heart)
```

or we could have used

```
mod.0 <- glm(cbind(ha,ok)~ck,family=binomial,data=heart)
```

since the logit link is the default for the binomial. Here is the default information printed about the model:

```
> mod.0
```

```
Call: glm(formula=cbind(ha,ok)~ck, family=binomial, data=heart)
```

```
Coefficients:
```

```
(Intercept)          ck
   -2.75834      0.03124
```

```
Degrees of Freedom: 11 Total (i.e. Null);  10 Residual
```

```
Null Deviance:      271.7
```

```
Residual Deviance: 36.93      AIC: 62.33
```

In the output, the Null deviance is the deviance for a model with just a constant term, while the Residual deviance is the deviance of the fitted model. These can be combined to give the *proportion deviance explained*, a generalization of r^2 , as follows:

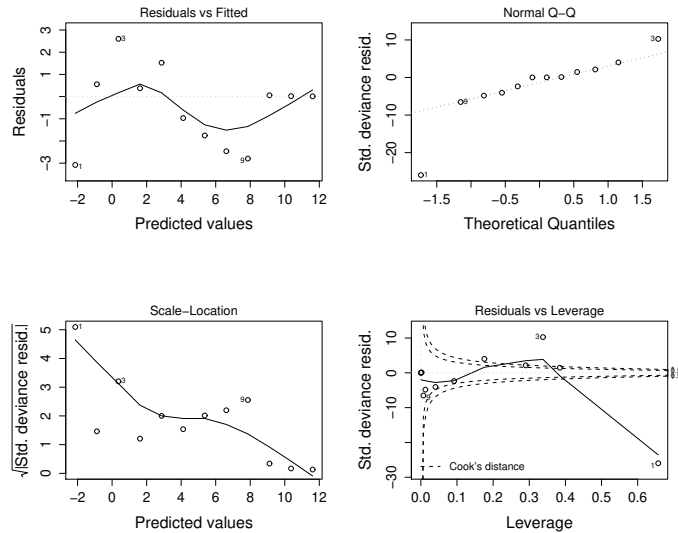
```
> (271.7-36.93)/271.7
[1] 0.864078
```

Notice that the deviance is quite high for the χ^2_{10} random variable that it should approximate if the model is fitting well. In fact

```
> 1-pchisq(36.93,10)
[1] 5.819325e-05
```

shows that there is a *very* small probability of a χ^2_{10} random variable being as large as 36.93. The residual plots also suggest a poor fit.

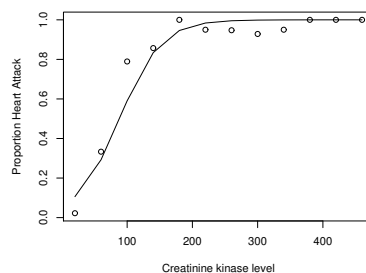
```
> op <- par(mfrow=c(2,2))
> plot(mod.0)
```



Again, the plots have much the same interpretation as the model checking plots for an ordinary linear model, except that it is now standardized residuals that are plotted (actually ‘deviance residuals — see later), the `Predicted values` are on the scale of the linear predictor rather than the response, and some departure from a straight line relationship in the Normal QQ plot is often to be expected. The plots are not easy to interpret when there are so few data, but there appears to be a trend in the mean of the residuals plotted against fitted value, which would cause concern. Furthermore, the first point has very high influence. Note that the interpretation of the residuals would be much more difficult for binary data (see later).

Notice how the problems do not stand out so clearly from a plot of the fitted values overlayed on the raw estimated probabilities:

```
> plot(heart$ck,p,xlab="Creatinine kinase level",
+ ylab="Proportion Heart Attack")
> lines(heart$ck,fitted(mod.0))
```



Note also that the fitted values provided by `glm` for binomial models are the estimated p_i 's, rather than the estimated μ_i 's.

The trend in the mean of the residuals suggests trying a cubic model, rather than the initial straight line.

```
> mod.2 <- glm(cbind(ha,ok)~ck+I(ck^2)+I(ck^3),family=binomial,
+ data=heart)
> mod.2
```

```
Call: glm(formula=cbind(ha,ok)~ck+I(ck^2)+I(ck^3),
```

```
family=binomial,data=heart)
```

Coefficients:

```
(Intercept)      ck      I(ck^2)      I(ck^3)
-5.786e+00    1.102e-01  -4.648e-04    6.448e-07
```

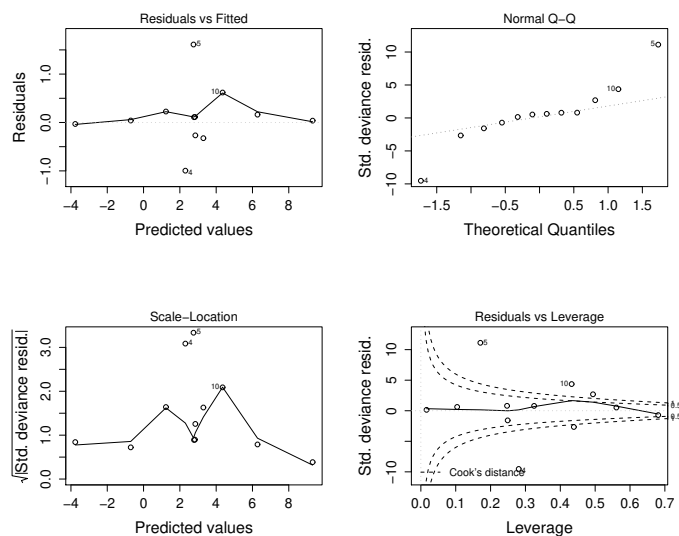
Degrees of Freedom: 11 Total (i.e. Null); 8 Residual

Null Deviance: 271.7

Residual Deviance: 4.252 AIC: 33.66

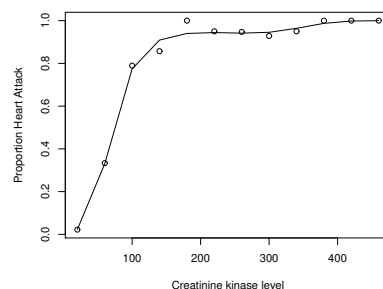
```
> par(mfrow=c(2,2))
```

```
> plot(mod.2)
```



Clearly 4.252 is not too large for consistency with a χ^2_8 distribution (it is less than the expected value, in fact) and the AIC has improved substantially. The residual plots now show less clear patterns than for the previous model, although if we had more data then such a departure from constant variance would be a cause for concern. Furthermore the fit is clearly closer to the data now:

```
par(mfrow=c(1,1))
plot(heart$ck,p,xlab="Creatinine kinase level",
      ylab="Proportion Heart Attack")
lines(heart$ck,fitted(mod.2))
```



We can also get R to test the null hypothesis that mod.0 is correct against the alternative that mod.2 is required. Somewhat confusingly the `anova` function is used to do this, although it is a generalized likelihood ratio test that is being performed, and not an analysis of variance.


```
> anova(mod.0,mod.2,test="Chisq")
Analysis of Deviance Table

Model 1: cbind(ha, ok) ~ ck
Model 2: cbind(ha, ok) ~ ck + I(ck^2) + I(ck^3)
   Resid. Df Resid. Dev Df Deviance P(>|Chi|)
1         10      36.929
2          8       4.252  2    32.676 8.025e-08
```

A p-value this low indicates very strong evidence against the null hypothesis: we really do need model 2.

12.2 Melanoma case-control study

Case-control studies are an important type of study, in which a group of patients with some disease (the *cases*) are compared to a randomly selected group of healthy subjects from the same population as the cases (the *controls*). Variables that might be associated with the disease are also collected for all subjects. If a variable is really associated with the disease then it ought to be predictive of whether a randomly selected patient in the study is a case or a control. Such predictivity can be assessed using GLMs, of the ‘logistic regression type’.

For example consider a study looking at 143 cases of malignant melanoma (a serious skin cancer) in white male patients aged 25 to 55, classified according to skin type (A, B or C for ‘celtic’, ‘middle european type’ or ‘Mediterranean’), compared to 356 white male controls aged 25-55 (selected without further reference to age or skin type). Patients were divided into 3 groups according to an ‘age’ factor variable, as well as being classified into 3 groups by the ‘skin’ factor variable (so there are 9 groups in total). The data are in a data frame `mdl`:

	mel	age	skin	n
1	15	25–35	A	54
2	8	25–35	B	52
3	7	25–35	C	44
4	26	35–45	A	75
5	18	35–45	B	52
6	6	35–45	C	42
7	30	45–55	A	67
8	25	45–55	B	66
9	8	45–55	C	47

Consider testing the null hypothesis that skin type is not associated with melanoma, against the alternative that it is. Neglecting the possibility of an interaction term, we could do this by comparing the null model

$$\text{logit}(p_i) = \beta_0 + \beta_1 \text{age}_i, \quad y_i \sim \text{binom}(p_i, n_i)$$

to the alternative

$$\text{logit}(p_i) = \beta_0 + \beta_1 \text{age}_i + \gamma_{\text{skin}(i)}, \quad y_i \sim \text{binom}(p_i, n_i)$$

where `skin(i)` is the skin type factor level for group i .

Fitting these models and comparing them using a GLRT (aka analysis of deviance) is straightforward

```
> mel0 <- glm(mel/n~age, family=binomial, data=mdl, weights=n)
> mel1 <- glm(mel/n~age+skin, family=binomial, data=mdl, weights=n)
> anova(mel0,mel1,test="Chisq")
Analysis of Deviance Table
```

```
Model 1: mel/n ~ age
Model 2: mel/n ~ age + skin
   Resid. Df Resid. Dev Df Deviance P(>|Chi|)
1         6      20.5062
2         4       3.4389  2    17.0673  0.0002
```

`test="Chisq"` specifies that a generalized likelihood ratio test is to be performed using (13). Here the p-value is very low: there is a very small probability of observing this large a difference in deviance between the two models if `mel0` really generated the data. This strongly suggests that `mel0` is incorrect. In other words, there is strong evidence in favour of `mel1` and an effect of skin type on melanoma risk. The next step would be to examine the model coefficients to ascertain the nature and size of the effect.

Note that these case-control studies can only be used to look at the relative risk of melanoma given different risk factors. The study tells us nothing about the absolute risk of melanoma, because we have chosen the ratio of cases to controls, rather than observing it in the population of interest.

12.3 Insurance claims testing example

The dataset `motori` is derived from dataset `motorins` from R library `faraway`^{††}. It contains insurance company data from Sweden, on payouts (`Payment`) in relation to number `Insured`, `km` travelled (a numeric variable with 5 discrete values), `Make` of car (a factor variable with 9 levels) and number of years no claims `Bonus`. An initial model for the data is:

$$\mathbb{E}(\text{Payment}_i) = \text{Insured}_i \times \text{risk}_i, \quad \text{where } \text{Payment} \sim \text{gamma}$$

so

$$\log\{\mathbb{E}(\text{Payment}_i)\} = \log(\text{Insured}_i) + \log(\text{risk}_i).$$

$\log(\text{Insured}_i)$ is an example of a model *offset* — a predictor variable whose coefficient is fixed at 1. $\log(\text{risk}_i)$ can be modelled using a linear model structure to give:

$$\log\{\mathbb{E}(\text{Payment}_i)\} = \log(\text{Insured}_i) + \beta_{\text{Make}(i)} \text{km}_i + \gamma_{\text{Make}(i)} + \alpha \text{Bonus}_i, .$$

Consider testing $H_0 : \beta_1 = \beta_2 = \dots = \beta_9$ against the alternative that the β_j are not all equal. First fit models embodying the two hypotheses:

```
g1 <- glm(Payment~offset(log(Insured))+km*Make+Bonus, Gamma(link=log), motori)
g0 <- glm(Payment~offset(log(Insured))+km+Make+Bonus, Gamma(link=log), motori)
```

Note that `g0` has replaced the 9 $\beta_{\text{Make}(i)}$ parameters with a single parameter β applying to all makes of car. Since ϕ is not known for the gamma, we now perform an F-ratio test comparing the models using (17)

```
> anova(g0,g1,test="F")
Analysis of Deviance Table
```

```
Model 1: Payment ~ offset(log(Insured)) + km + Make + Bonus
Model 2: Payment ~ offset(log(Insured)) + km * Make + Bonus
  Resid. Df Resid. Dev  Df Deviance    F Pr(>F)
1      284    155.056
2      276    151.890    8     3.166 0.752 0.6455
```

It appears that the dependence of claim rate on `km` travelled can be assumed not to vary with car make.

12.4 Semiconductor wafer model selection example

As an example of backwards selection consider the semiconductor electrical resistance data given in Faraway (2005) as the `wafer` data frame. Four factors (`x1` to `x4`) in the manufacturing process were believed to influence semiconductor resistance, `resist`, and an experiment was conducted to try out all combinations of two levels of each. The data are as follows:

^{††}available from `cran.r-project.org` to accompany *Extending the linear model with R* by Julian Faraway.

```
> wafer
      x1 x2 x3 x4 resist
1      - - - -  193.4
2      + - - -  247.6
3      - + - -  168.2
4      + + - -  205.0
5      - - + -  303.4
6      + - + -  339.9
7      - + + -  226.3
8      + + + -  208.3
9      - - - +  220.0
10     + - - +  256.4
11     - + - +  165.7
12     + + - +  203.5
13     - - + +  285.0
14     + - + +  268.0
15     - + + +  169.1
16     + + + +  208.5
```

An initial model is fitted with all interactions of the factors up to 3rd order, and assuming a Gamma distribution with log link.

```
wm <- glm(resist ~ (x1+x2+x3+x4)^3, Gamma(link="log"), data=wafer)
```

Applying backward selection, we now want to calculate the p-values associated with dropping each 3 way interaction from the full model (one at a time!). Recall that the `drop1` function in R is a very convenient way of doing this. It goes through all the terms in the model that can be dropped on their own, refits without each of them in turn and compares each resulting fit with the original model...

```
> drop1(wm, test="F")
```

Single term deletions

Model:

```
resist ~ x1 * x2 * x3 + x1 * x3 * x4 + x1 * x2 * x4 + x2 * x3 *
          x4
```

	Df	Deviance	AIC	F value	Pr(F)
<none>		0.008	129.726		
x1:x2:x3	1	0.009	127.764	0.0380	0.8775
x1:x3:x4	1	0.011	128.035	0.3094	0.6769
x1:x2:x4	1	0.029	130.144	2.4173	0.3639
x2:x3:x4	1	0.011	128.012	0.2867	0.6871

The rows of the table are labelled with the names of the dropped terms. The reported p-value in the final row is for testing the null hypothesis that the model without the dropped term is adequate (against the alternative that the full model is needed). The AIC for each model under consideration is also reported. The p-values suggest dropping the interaction `x1:x2:x3`, and

```
wm1 <- glm(resist ~ (x1+x2+x3+x4)^3 - x1:x2:x3,
            Gamma(link="log"), data=wafer)
```

is one way to achieve this. The p-values are now re-calculated for the re-fitted model.

```
> drop1(wm1, test="F")
```

Single term deletions

Model:

```

resist ~ x1 * x3 * x4 + x1 * x2 * x4 + x2 * x3 * x4
      Df Deviance      AIC F value  Pr(F)
<none>      0.009 128.322
x1:x3:x4   1      0.011 126.918   0.5961 0.5208
x1:x4:x2   1      0.029 130.981   4.6577 0.1636
x3:x4:x2   1      0.011 126.874   0.5523 0.5348

```

Notice how they have changed from the previous set of p-values. Now the $x3:x4:x2$ interaction is the one to drop. Repeating these steps we eventually end up with

```

> wm7 <- glm(resist~x1*x3+ x3*x4 +x2*x3,Gamma(link="log"),data=wafer)
> drop1(wm7,test="F")
Single term deletions

```

```

Model:
resist ~ x1 * x3 + x3 * x4 + x2 * x3
      Df Deviance      AIC F value  Pr(F)
<none>      0.036 139.199
x1:x3   1      0.060 142.540   5.3311 0.04977 *
x3:x4   1      0.069 144.510   7.2970 0.02702 *
x3:x2   1      0.067 144.061   6.8491 0.03079 *
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

```

So if 0.05 is the threshold for retaining a term then this is the final model. i.e. All main effects are present, along with 3 two way interactions.

Once a model has been selected, then the coefficient estimates would be examined and interpreted, possibly with the aid of confidence intervals. Note however that inference performed with the final fitted model tends to overstate what can really be concluded from the data, since it does not allow for the uncertainty in model selection.

12.4.1 AIC based selection

As an example of using AIC let's redo the wafer model selection example using AIC for backwards selection.

```

> library(faraway)
> data(wafer)
> wm <- glm(resist~x1*x2*x3+x1*x3*x4+x1*x2*x4+x2*x3*x4,
+           Gamma(link="log"),data=wafer)
> drop1(wm)
Single term deletions

```

```

Model:
resist ~ x1 * x2 * x3 + x1 * x3 * x4 + x1 * x2 * x4 + x2 * x3 *
      x4
      Df Deviance      AIC
<none>      0.008 129.726
x1:x2:x3   1      0.009 127.764
x1:x3:x4   1      0.011 128.035
x1:x2:x4   1      0.029 130.144
x2:x3:x4   1      0.011 128.012

```

Since no test was specified `drop1` simply evaluates the AIC for the full model (`wm` in this case) and versions of it omitting all possible single terms. The model with the lowest AIC is then selected. In this instance it is the model that omits $x1:x2:x3$, so that term would be dropped. The easiest way to refit a model omitting some terms is to use the `update` function, as follows...

```
> wm1 <- update(wm, .~.-x1:x2:x3)
```

`update` refits a model identical to `wm`, *except* that it has no `x1:x2:x3` interaction term, and returns the resulting fitted model object, here stored in `wm1`. The ‘.’ on the lhs of the `update` model formula indicates that the response variable is as for `wm`, while the ‘.’ on the rhs indicates that the linear predictor should be exactly as for `wm`, except for the term explicitly omitted by `-x1:x2:x3`.

Now `wm1` can be used like any other fitted `glm` object.

```
> drop1(wm1)
```

Single term deletions

Model:

```
resist ~ x1 + x2 + x3 + x4 + x1:x2 + x1:x3 + x2:x3 + x1:x4 +
        x3:x4 + x2:x4 + x1:x3:x4 + x1:x2:x4 + x2:x3:x4
```

	Df	Deviance	AIC
<none>		0.009	128.322
x1:x3:x4	1	0.011	126.918
x1:x2:x4	1	0.029	130.981
x2:x3:x4	1	0.011	126.874

Notice one wrinkle: the AIC reported for `wm1` is 128.322, but when we used `drop1` before on `wm`, it suggested that the AIC for `wm1` would be 127.764. This happens because we need a scale parameter estimate in order to evaluate the AIC, and `drop1` always uses the same estimate for all the models it compares, based on the largest model it is considering. Hence the two calls to `drop1` give different AIC estimates for the same model, because the different calls are using different scale parameter estimates. (This is nothing to do with having used `update`, by the way.) Strictly speaking the AIC should be evaluated using the MLE of ϕ , in which case this problem would not occur, but it makes the computation much more expensive (and less reliable) if we do this. Of course the problem does not arise if ϕ is known.

Continuing ...

```
> wm2 <- update(wm1, .~.-x2:x3:x4)
```

```
> drop1(wm2)
```

Single term deletions

Model:

```
resist ~ x1 + x2 + x3 + x4 + x1:x2 + x1:x3 + x2:x3 + x1:x4 +
        x3:x4 + x2:x4 + x1:x3:x4 + x1:x2:x4
```

	Df	Deviance	AIC
<none>		0.011	130.224
x2:x3	1	0.043	136.823
x1:x3:x4	1	0.014	128.925
x1:x2:x4	1	0.031	133.701

```
> wm3 <- update(wm2, .~.-x1:x3:x4)
```

```
> drop1(wm3)
```

Single term deletions

Model:

```
resist ~ x1 + x2 + x3 + x4 + x1:x2 + x1:x3 + x2:x3 + x1:x4 +
        x3:x4 + x2:x4 + x1:x2:x4
```

	Df	Deviance	AIC
<none>		0.014	131.582
x1:x3	1	0.038	136.723
x2:x3	1	0.045	138.879
x3:x4	1	0.047	139.386
x1:x2:x4	1	0.034	135.503

...at which point we would select `wm3` and proceed to examine its coefficients and interpret the model fit. Notice that the AIC selected model is quite a bit more complex than the model selected by hypothesis testing. This is typical.

Given the rather algorithmic nature of the selection process, it is possible to automate it entirely. The `step` function will perform the whole backwards selection-by-AIC process for you, with one function call...

```
> wm3a <- step(wm)
Start:  AIC= 129.73
      resist ~ x1 * x2 * x3 + x1 * x3 * x4 + x1 * x2 * x4 + x2 * x3 *
            x4

      Df Deviance    AIC
- x1:x2:x3  1    0.009 127.764
- x2:x3:x4  1    0.011 128.012
- x1:x3:x4  1    0.011 128.035
<none>      0.008 129.726
- x1:x2:x4  1    0.029 130.144

Step:  AIC= 128.32
      resist ~ x1 + x2 + x3 + x4 + x1:x2 + x1:x3 + x2:x3 + x1:x4 +
            x3:x4 + x2:x4 + x1:x3:x4 + x1:x2:x4 + x2:x3:x4

      Df Deviance    AIC
- x2:x3:x4  1    0.011 126.874
- x1:x3:x4  1    0.011 126.918
<none>      0.009 128.322
- x1:x2:x4  1    0.029 130.981

Step:  AIC= 130.22
      resist ~ x1 + x2 + x3 + x4 + x1:x2 + x1:x3 + x2:x3 + x1:x4 +
            x3:x4 + x2:x4 + x1:x3:x4 + x1:x2:x4

Call:  glm(formula = resist ~ x1 + x2 + x3 + x4 + x1:x2 + x1:x3 + x2:x3 +
      x1:x4 + x3:x4 + x2:x4 + x1:x3:x4 + x1:x2:x4,
      family = Gamma(link = "log"), data = wafer)

Coefficients:
(Intercept)          x1+          x2+          x3+          x4+          x1+:x2+
    5.2586         0.2843        -0.1273         0.4626         0.1502        -0.1228
      x1+:x3+      x2+:x3+      x1+:x4+      x3+:x4+      x2+:x4+      x1+:x3+:x4+
    -0.2071       -0.1783       -0.1852       -0.2339       -0.1863         0.1018
x1+:x2+:x4+
    0.2845

Degrees of Freedom: 15 Total (i.e. Null);  3 Residual
Null Deviance:      0.6978
Residual Deviance: 0.01108      AIC: 130.2
```

Notice that the finally selected model is a little different to the one that was selected using `drop1`. This is again down to how the scale parameter is handled: it's done differently in `step` and `drop1`, and for this model, this has made a slight difference to the finally selected model.

Finally note that another possibility with AIC is *all subsets* model selection, in which every possible sub-model of the most complex model is considered, and the one with the lowest AIC is finally selected.

12.5 Remarks on model selection

Finally let's review the reasons for model selection.

1. We do model selection because we are often uncertain about the exact form that a model should take, even though it is often possible to write down a model that we expect to be 'complicated enough', so that that for some parameter values it should be a reasonable approximation to the truth.
2. Selection is important for interpretational reasons: simpler models are easy to interpret than complex ones.
3. Model selection also tends to improve the precision of estimates and the accuracy of model predictions. If a model contains more terms than necessary then we will inevitably 'use up' information in the data in estimating the associated coefficients, which in turn means that the important terms are less precisely estimated.

Whether model selection is performed using AIC or model selection depends on the purposes of the analysis. If we want to develop a model for prediction purposes then it makes sense to use AIC, but if our interest lies in trying to understand relationships between the predictors and the response, it may be preferable to use hypothesis testing based methods to try and avoid including model terms unless there is good evidence that they are needed.

Finally note that there is a difficult problem associated with model selection:

- It is common practice to use model selection methods to choose one model from a large set of potential models, but then to treat the selected model exactly as if it were the only model we ever considered, when it comes to calculating confidence intervals etc. In doing this we neglect the uncertainty associated with model selection, and will therefore tend to overstate how precisely we know the coefficients of the selected model (and how precise its predictions are). This issue is an active area of current statistical research.

12.6 Interpreting model coefficients

Once a model is selected and checked, then usually you will want to examine and interpret its estimated coefficients. For many of the examples that we have met the interpretation of parameters is obvious, but for complex models it can be less easy, especially with factor variables when identifiability constraints are needed. The failsafe way to check the meaning of each coefficient in practical modelling is to examine the model matrix. For example, the summary for model `wm3` from the previous section is.

```
> summary(wm3)
...
Coefficients:
              Estimate Std. Error t value Pr(>|t|)
(Intercept)   5.27133     0.05060 104.176 5.09e-08 ***
x1+            0.25858     0.06532   3.958  0.01670 *
x2+           -0.12711     0.06532  -1.946  0.12354
x3+            0.43720     0.05843   7.483  0.00171 **
x4+            0.12466     0.06532   1.908  0.12899
x1+:x2+       -0.12222     0.08263  -1.479  0.21321
x1+:x3+       -0.15622     0.05843  -2.674  0.05559 .
x2+:x3+       -0.17825     0.05843  -3.051  0.03800 *
x1+:x4+       -0.13430     0.08263  -1.625  0.17941
x3+:x4+       -0.18306     0.05843  -3.133  0.03508 *
x2+:x4+       -0.18610     0.08263  -2.252  0.08743 .
x1+:x2+:x4+   0.28450     0.11686   2.435  0.07162 .
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
...
```

This can look intimidating, although in fact the parameter names are pretty helpful here. They basically tell you the circumstance under which the coefficient of a factor will be added to the model. For example if `x1` is in the `+` state for some response measurement, then we include the `x1+` term in the model (which just amounts to adding .25858 to the linear predictor in this case, since `x1` is a factor). If `x1` and `x2` are in the `+` state for some response measurement, then terms `x1+`, `x2+` and `x1+:x2+` are included, and so on.

To make things completely clear, however, look at the model matrix (and original data frame).

```
> wafer
  x1 x2 x3 x4 resist
1  - - - -  193.4
2  + - - -  247.6
3  - + - -  168.2
4  + + - -  205.0
5  - - + -  303.4
6  + - + -  339.9
7  - + + -  226.3
8  + + + -  208.3
9  - - - +  220.0
10 + - - +  256.4
11 - + - +  165.7
12 + + - +  203.5
13 - - + +  285.0
14 + - + +  268.0
15 - + + +  169.1
16 + + + +  208.5

> model.matrix(wm3)
      (Intercept) x1+ x2+ x3+ x4+ x1+:x2+ x1+:x3+ x2+:x3+ x1+:x4+ x3+:x4+ x2+:x4+ x1+:x2+:x4+
1             1    0    0    0    0         0         0         0         0         0         0         0
2             1    1    0    0    0         0         0         0         0         0         0
3             1    0    1    0    0         0         0         0         0         0         0
4             1    1    1    0    0         1         0         0         0         0         0
5             1    0    0    1    0         0         0         0         0         0         0
6             1    1    0    1    0         0         1         0         0         0         0
7             1    0    1    1    0         0         0         1         0         0         0
8             1    1    1    1    0         1         1         1         0         0         0
9             1    0    0    0    1         0         0         0         0         0         0
10            1    1    0    0    1         0         0         0         1         0         0
11            1    0    1    0    1         0         0         0         0         0         1
12            1    1    1    0    1         1         0         0         1         0         1
13            1    0    0    1    1         0         0         0         0         1         0
14            1    1    0    1    1         0         1         0         1         1         0
15            1    0    1    1    1         0         0         1         0         1         1
16            1    1    1    1    1         1         1         1         1         1         1
```

It is now clear that the `intercept` is the expected resistance for a wafer where none of the factors are in the `+` state. The coefficients `x1+` to `x4+` give the expected increase in resistivity when just one of the factors is in the `+` state (referring back to the summary, `x1` and `x3` seem to lead to a significant increase in resistance, on their own). So what about the interactions? Look at `x1+:x2+` as an example — it's an adjustment that is added on when `x1` and `x2` are *both* in the `+` state together: i.e. it is how much different the expected resistivity is to what you would expect if the effects of `x1` and `x2` both being `+` simply added to each other. Referring back to the summary, it seems that when two factors are in the `+` state, the resistivity is lower than you would expect from just looking at their effects on their own (although not all the interaction coefficients are significantly different from 0).

13 Mixed models

Statistical models describe how data were sampled from a population about which we want to learn. Sometimes the population is very concrete, such as the population of trees in Scotland, or the population of patients in NHS hospitals. Other time it is more abstract, such as the population of Cairngorm windspeeds that could have occurred under current climatic conditions, or the population of light speeds that could have been observed in a physics experiment. But the point is always that we want to learn about the whole population from observations of only a sample of its members. Statistics allows us to do this, provided that the sample is in some way a random sample.

So far when considering models we have considered only data where the sampling process led to quite simple models for the randomness in the data. In the linear model case randomness is described by simple independent zero mean constant variance additive 'errors'. For GLMs we generalized this somewhat, but still assumed that

random variability was independent between observations and depended only on the non-random model predicted expected value of each observation. To see why we might need more than this, consider the example of a study to look at the relationship between pulse rate, fitness and weight at different exercise levels in UK adults. For the results to be applicable to UK adults we need to select a random sample of adults from the UK population. Clearly to obtain pulse rate at several exercise levels we will need multiple measurements for each adult. There are obviously two sources of uncertainty in this setup that we will have to account for in order to make valid inferences:

1. There will be individual variability between pulse rates of different people in the study, unaccounted for by the variables we are looking at. We can view this as random variability arising from having randomly selected individuals from the whole population, and should model it as such: i.e. as a random subject effect.
2. There will be within person random variability in pulse measurements: i.e. an 'error' term that applies to each measurement.

So we have two levels of randomness here: person to person randomness, and measurement to measurement randomness. Each person's measurements are affected by one random component that is constant for that person, and additional randomness on each measurement. In the context of linear models, we might want to use a model structure something like this:

$$\text{pulse}_i = \beta_0 + \beta_1 \text{fitness}_i + \beta_2 \text{weight}_i + b_{\text{person}(i)} + \epsilon_i, \quad b_j \sim N(0, \sigma_b^2), \quad \epsilon_i \sim N(0, \sigma^2)$$

where $\text{person}(i)$ is the label for the person that measurement i relates to. b_j is the *random effect* for person j , it is a normal random deviate, but is constant for all measurements on person j , while the measurement error term ϵ_i varies randomly from measurement to measurement.

At this point you might ask: why make all this fuss, can't we just treat b_j as the fixed parameter of an ordinary linear model in which person is a factor variable? We could, but then we would not really have any basis for making inferences about the UK population generally. The given model explicitly says that each b_j is a random draw from the population's distribution of individual effects: so clearly the model can tell us something about that population. In contrast, if we treat the b_j as parameters then the model puts no structure on them at all: each individual's effect could take *any value whatsoever*, so that knowing the b_j values for the sample can tell us nothing at all about the b_j values in the rest of the population.

The pulse rate model is an example of a *linear mixed model* (LMM), with general structure

$$\mathbf{y} = \mathbf{X}\boldsymbol{\beta} + \mathbf{Z}\mathbf{b} + \boldsymbol{\epsilon}, \quad \mathbf{b} \sim N(\mathbf{0}, \boldsymbol{\Omega}_\theta), \quad \boldsymbol{\epsilon} \sim N(\mathbf{0}, \mathbf{I}\sigma^2) \quad (18)$$

where \mathbf{Z} is a model matrix for the random effects \mathbf{b} , which have a covariance matrix $\boldsymbol{\Omega}_\theta$, parameterized in terms of parameters θ . To distinguish them from the random effects, the elements of $\boldsymbol{\beta}$ are referred to as *fixed effects* in such model. For the pulse rate example $\boldsymbol{\Omega}_\theta = \mathbf{I}\sigma_b^2$, so σ_b^2 , or more likely $\log \sigma_b$, is the only θ parameter. \mathbf{Z} would be a matrix of indicator variables for the levels of the person factor. Note that there are no identifiability constraints needed for random factors: the zero mean normality assumption is sufficient to ensure identifiability (provided that $\boldsymbol{\Omega}_\theta$ is finite). The LMM is essentially a linear model in which we have allowed the sort of linear structure for the randomness in the data that we already allowed for modelling the non-random mean.

Unsurprisingly we might also want to use *generalized linear mixed models* (GLMM) of the form

$$g(\mu_i) = \mathbf{X}_i\boldsymbol{\beta} + \mathbf{Z}_i\mathbf{b}, \quad \mathbf{b} \sim N(\mathbf{0}, \boldsymbol{\Omega}_\theta), \quad y_i \sim \text{EF}(\mu_i, \phi) \quad (19)$$

So μ_i is now random, because of the randomness in \mathbf{b} , but the distribution of y_i given μ_i is modelled in the same way as for a GLM.

13.1 Mixed model theory

The simplest way to approach inference with mixed models is to borrow one simple idea from Bayesian statistics: that a fixed effect is equivalent to a random effect with a constant improper density over the whole real line. If we are doing this, we might as well define one vector containing both fixed and random effects: $\mathbf{d}^\top = (\boldsymbol{\beta}^\top, \mathbf{b}^\top)$. Now consider the joint density of \mathbf{y} and \mathbf{d}

$$\log f_\theta(\mathbf{y}, \mathbf{d}) = \log f(\mathbf{y}|\mathbf{d}) + \log f_\theta(\mathbf{d}) = l(\mathbf{d}) - \frac{1}{2}\mathbf{d}^\top \begin{pmatrix} \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \boldsymbol{\Omega}_\theta^{-1} \end{pmatrix} \mathbf{d} + c$$

where c is an irrelevant constant and $l(\mathbf{d}) = \log f(\mathbf{y}|\mathbf{d})$ is the log likelihood implied by the distribution of \mathbf{y} given $\boldsymbol{\mu}$ (hence given \mathbf{d}). The quadratic term comes from the normality of \mathbf{b} (and improper uniform distribution assumed for $\boldsymbol{\beta}$). For notational convenience let's re-write it as $\mathbf{d}^\top \mathbf{S}_\theta \mathbf{d}$.

Since $f_\theta(\mathbf{d}|\mathbf{y}) = f_\theta(\mathbf{d}, \mathbf{y})/f_\theta(\mathbf{y})$, then for given $\boldsymbol{\theta}$ we can compute the *Maximum A Posteriori* (MAP) estimates as those maximizing $f_\theta(\mathbf{d}, \mathbf{y})$ and hence $f_\theta(\mathbf{d}|\mathbf{y})$:

$$\hat{\mathbf{d}} = \underset{\mathbf{d}}{\operatorname{argmax}} l(\mathbf{d}) - \frac{1}{2} \mathbf{d}^\top \mathbf{S}_\theta \mathbf{d}. \quad (20)$$

The resulting $\hat{\boldsymbol{\beta}}$ is the estimate of parameter vector $\boldsymbol{\beta}$, while the $\hat{\mathbf{b}}$ are usually referred to as ‘predictions’ of the random effects. Now let \mathbf{H}_θ be the negative second derivative matrix of $\log f_\theta(\mathbf{y}, \mathbf{d})$, that is

$$\mathbf{H}_\theta = - \left. \frac{\partial^2 l}{\partial \mathbf{d} \partial \mathbf{d}^\top} \right|_{\hat{\mathbf{d}}} + \mathbf{S}_\theta.$$

Writing $\mathbb{X} = [\mathbf{X}, \mathbf{Z}]$, the negative second derivative matrix of the log likelihood is $\mathbb{X}^\top \mathbb{X}$ for the LMM (Gaussian likelihood) case, and $\mathbb{X}^\top \mathbf{W} \mathbb{X}$ for a GLMM, where \mathbf{W} is defined in terms of $\boldsymbol{\mu}$ exactly as for a GLM. Now consider a second order Taylor approximation of $\log f_\theta(\mathbf{y}, \mathbf{d})$ about $\hat{\mathbf{d}}$,

$$\log f_\theta(\mathbf{y}, \mathbf{d}) \simeq \log f_\theta(\mathbf{y}, \hat{\mathbf{d}}) - \frac{1}{2} (\mathbf{d} - \hat{\mathbf{d}})^\top \mathbf{H}_\theta (\mathbf{d} - \hat{\mathbf{d}}), \quad (21)$$

(the first derivative term is zero by definition of $\hat{\mathbf{d}}$). This is exact for a Gaussian likelihood, since all higher order derivatives are then zero. Because $f_\theta(\mathbf{d}|\mathbf{y}) \propto f_\theta(\mathbf{d}, \mathbf{y})$, then on exponentiating (21) we can recognize that

$$\mathbf{d}|\mathbf{y} \sim N(\hat{\mathbf{d}}, \mathbf{H}_\theta^{-1}) \quad (22)$$

exactly in the Gaussian likelihood case, and approximately otherwise. Although this result is Bayesian, it can be shown that the Bayesian covariance matrix for $\boldsymbol{\beta}$ (the upper left block of \mathbf{H}_θ^{-1}) exactly corresponds to the covariance matrix of $\hat{\boldsymbol{\beta}}$ under a purely frequentist approach.

We also need to be able to estimate the random effect parameters $\boldsymbol{\theta}$. This can be done by maximising the marginal likelihood of $\boldsymbol{\theta}$. From basic conditional probability $L(\boldsymbol{\theta}) = f_\theta(\mathbf{y}) = f_\theta(\hat{\mathbf{d}}, \mathbf{y})/f_\theta(\hat{\mathbf{d}}|\mathbf{y})$. On substituting the p.d.f. for (22) for the denominator and taking logs we get the log marginal likelihood

$$l(\boldsymbol{\theta}) = \log f_\theta(\hat{\mathbf{d}}, \mathbf{y}) - \frac{1}{2} \log |\mathbf{H}_\theta| + \frac{p}{2} \log(2\pi)$$

where p is the dimension of \mathbf{d} . This result is obviously exact for a Gaussian likelihood and approximate otherwise. The approximation is known as a *Laplace approximation*, and is not the only possibility. The marginal likelihood is the integral of $f_\theta(\hat{\mathbf{d}}, \mathbf{y})$ w.r.t. \mathbf{d} , and for some model structures this factorizes into a product of low dimensional integrals that can be accurately evaluated numerically, offering an alternative to Laplace approximation.

However it is computed, the marginal likelihood can be optimized numerically to find $\boldsymbol{\theta}$, with each $\boldsymbol{\theta}$ tried by the optimization method requiring an ‘inner’ optimization to find the corresponding $\hat{\mathbf{d}}$, and the corresponding \mathbf{H}_θ . Really efficient optimization of $l(\boldsymbol{\theta})$ requires its derivatives w.r.t. $\boldsymbol{\theta}$. These can be computed by computing $\partial \hat{\mathbf{d}}/\partial \boldsymbol{\theta}$, which requires implicit differentiation methods. But we have covered as much theory as we need here.

13.2 Computer intensive inference via resampling

Some aspects of inference with mixed models are difficult. For example, we can not use the generalized likelihood ratio test to compare two mixed models in which one constrains some variance components of the other to be zero. This is because a crucial regularity condition for (13) is that the null hypothesis must not restricting some parameters to the edge of the parameter space. In addition computations using (22) can be both costly and inaccurate in situations in which the number of random effects is of the same order as the number of data. For this reason it is common to turn to computer intensive *resampling* methods instead. The most widely used is the *bootstrap*.

The idea is that uncertainty about parameter values is inherited from the uncertainty in the data sampling process. When we calculate the uncertainties associated with parameters we are essentially computing the uncertainty

that would result from repeating the data gathering and analysis process a large number of times. To see this at its simplest consider inference about parameter μ in the model $x_i \sim N(\mu, \sigma^2)$ for $i = 1, \dots, 50$. In this case $\hat{\mu} = \bar{x}$. To characterise the uncertainty in $\hat{\mu}$ we could imagine gathering replicate data sets again and again and again, computing \bar{x} for each, and thereby building up an empirical distribution for $\hat{\mu} = \bar{x}$.

In most cases such replication of the data is not possible, and even if we could do it, we would then prefer to combine all the data to get a larger sample size. However we can *simulate* repeated replication of the data gathering process, by simply repeatedly *resampling* from the one data set that we have. In particular, we can randomly resample the original data with replacement, to create simulated replicate datasets of the same size as the original. Unknown parameters can then be estimated from each of these datasets, in the same way as was done for the original dataset, in order to assess the uncertainty of the parameter estimators. This process is known as *bootstrapping*. Here it is applied to the simple example above (using simulated data)...

```
set.seed(1); n <- 50
x <- rnorm(n) + 4 ## simulate data
n.rep <- 1000
mubs <- rep(0, n.rep); mubs[1] <- mean(x)
for (i in 2:n.rep) {
  xb <- sample(x, n, replace=TRUE) ## bootstrap resample data
  mubs[i] <- mean(xb) ## compute mu.hat for this resample
}
quantile(mubs, c(.025, .975)) ## quantiles of bootstrap distribution give CI
      2.5%      97.5%
3.867025 4.315665
```

For comparison here is the standard theoretical confidence interval using the same data.

```
mu.hat <- mean(x)
mu.sd <- 1/sqrt(n)
c(mu.hat - 1.96*mu.sd, mu.hat + 1.96*mu.sd)
[1] 3.823262 4.377634
```

Careful consideration of the bootstrapping process might lead you to worry about the way these bootstrap ‘percentile’ intervals have been computed - that the intervals are somehow backwards around $\hat{\mu}$ and are only OK if the distribution of $\hat{\mu}$ is symmetric. In fact various approaches can be justified here, but for the moment, let’s move on. The key point to take from this example is the notion of quantifying uncertainty by resampling from your data to simulate replication of the data gathering process.

Note that the idea of resampling generalizes easily to multivariate data - in R we simply resample whole data frame rows with replacement, and apply whatever analysis was applied to the original data frame to each resampled version. For example if `dat` is a data frame, the following code snippet generates one resampled version:

```
nd <- nrow(dat)
i <- sample(1:nd, nd, replace=TRUE) ## resample row indices
datbs <- dat[i,] ## bootstrap data frame
```

Resampling the data is known as *non-parametric bootstrapping*. For model based analysis *parametric bootstrapping* is the alternative. In this case, rather than re-sampling the original data, we simulate new data from the model, using the parameters estimated from the original data. Here it is in action for the simple example above.

```
mu <- mean(x); sd <- sd(x)
mubs <- rep(0, n.rep); mubs[1] <- mean(x)
for (i in 2:n.rep) {
  xb <- rnorm(n, mu, sd) ## parametric bootstrap sample
  mubs[i] <- mean(xb) ## compute mu.hat for this resample
}
quantile(mubs, c(.025, .975)) ## quantiles of bootstrap distribution give CI
      2.5%      97.5%
3.850921 4.319388
```

Much more could be said about this topic, but the basic concept suffices for what follows.

14 Mixed models in R

There are several mixed modelling packages available in R. `nlme` is shipped with all versions of R, and offers a rich variety of linear mixed models. Function `glmmPQL` for the `MASS` library (also a default R package) is a wrapper that uses `nlme` to fit GLMMs using an approximation to the Laplace approximate marginal likelihood. A better option for GLMMs is `lme4` which uses full Laplace approximation, and a rather elegant model specification language, as well as covering LMMs. For simple independent random effects package `mgcv` supplied with R, can also fit LMMs and GLMMs via full Laplace approximation. Here we will mainly focus on `lme4`.

14.1 Simple example

The `Rail` data frame supplied with R package `nlme` contains measurements of the `travel` time in nanoseconds of ultrasonic waves along 6 railway `Rails`. Hairline fractures in the rails slow the travel time. There are 3 measurements per rail. A very simple LMM is appropriate for these data:

$$\text{travel}_i = \alpha + b_{\text{Rail}(i)} + \epsilon_i \text{ where } b_j \sim N(0, \sigma_b^2) \text{ and } \epsilon_i \sim N(0, \sigma^2)$$

`Rail(i)` is the indicator of which rail observation i relates to (so there are six b_j values). First consider fitting this model using `lme4`.

LLMs are estimated in `lme4` using the `lmer` function. This is very much like `lm` to use, except that the model formula may involve random effect terms, which are indicated by a bracket notation of the form $(x|g)$ where x specifies the random effect and g is a factor variable indicating groups which each have their own draw from the random effect distribution. x might in fact consist of several terms. For example $(x+z+h|g)$ would specify a term $b_{0j} + b_{1j}x_i + b_{2j}z_i + b_{3j}h_i$ for each level j of g . If we didn't want the b_{0j} term it would be suppressed using `-1`. A single random intercept for each level of g is specified by $(1|g)$. So here it is in use:

```
library(lme4)
library(nlme) ## for Rail data
rm <- lmer(travel ~ (1|Rail), Rail)
rm ## default print method
Linear mixed model fit by REML ['lmerMod']
Formula: travel ~ (1 | Rail)
Data: Rail
REML criterion at convergence: 122.177
Random effects:
Groups   Name              Std.Dev.
Rail     (Intercept)  24.805
Residual                    4.021
Number of obs: 18, groups: Rail, 6
Fixed Effects:
(Intercept)
        66.5

confint(rm) ## get confidence intervals for parameters
Computing profile confidence intervals ...
          2.5 %      97.5 %
.sig01    13.929422 45.546746
.sigma     2.824557  6.377695
(Intercept) 44.960664 88.039333
```

Profile confidence intervals are obtained by using a GLRT to test whether a parameter could take a particular value, and then searching numerically for the range of such values that the test would accept - here at the 5% level. This is a somewhat slow process.

For models with simple random effects like this, package `mgcv` can also be used. It is mainly used for fitting *Generalized Additive Models*, so the model specification is less obvious.

```
library(mgcv)
rm2 <- gam(travel ~ s(Rail,bs="re"),data=Rail,method="REML")
summary(rm2)

Family: gaussian
Link function: identity

Formula:
travel ~ s(Rail, bs = "re")

Parametric coefficients:
              Estimate Std. Error t value Pr(>|t|)
(Intercept)    66.50      10.17    6.538 2.73e-05 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Approximate significance of smooth terms:
              edf Ref.df      F p-value
s(Rail)  4.957      5 114.2  <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

R-sq.(adj) =  0.971   Deviance explained =   98%
-REML = 61.089   Scale est. = 16.167      n = 18

gam.vcomp(rm2) ## get confidence intervals for variances

Standard deviations and 0.95 confidence intervals:

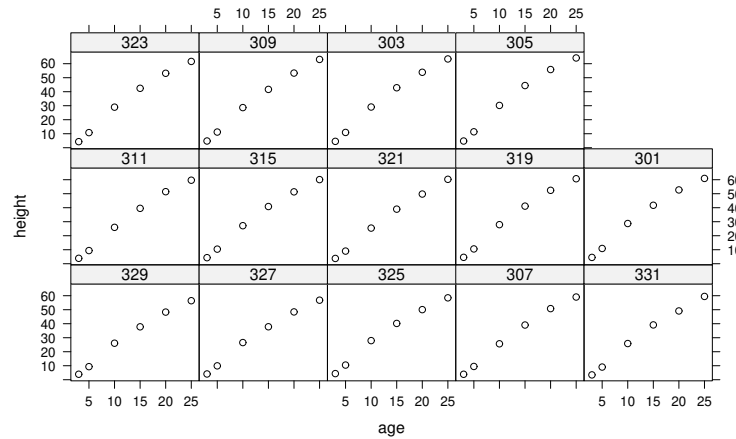
              std.dev      lower      upper
s(Rail) 24.805465 13.274315 46.353510
scale    4.020779  2.695004  5.998753
```

Notice how the estimates are the same as for `lmer` and the intervals very close, but not identical because profile likelihood is not being used. We'll use `mgcv` again later, but stick to `lme4` for now.

14.2 Loblolly pines: a more complicated example

The data frame `Loblolly` contains growth data for Loblolly pine trees over 25 years. Data are supplied on height, age for each tree. Tree identity is given by the factor variable `Seed`. Interest is in modelling the mean growth trajectory. First examine the data.

```
library(lattice) ## useful package for visualizing grouped data
xyplot(height ~ age | Seed, data = Loblolly)
```



So the trajectories vary from tree to tree, and are clearly curved, rather than linear. An appropriate model for the i^{th} measurement on the j^{th} tree might be,

$$\text{height}_{ji} = \beta_0 + \beta_1 \text{age}_{ji} + \beta_2 \text{age}_{ji}^2 + \beta_3 \text{age}_{ji}^3 + b_0 + b_{j1} \text{age}_{ji} + b_{j2} \text{age}_{ji}^2 + b_{j3} \text{age}_{ji}^3 + \epsilon_{j,i}$$

where the $\epsilon_{j,i}$ are i.i.d. zero mean normal random variables. As usual β denotes the fixed effects and $\mathbf{b}_j \sim N(0, \Omega(\theta))$ denotes the random effects. We will suppose that these random effects are independent between trees, but correlated within each tree (so most elements of $\Omega(\theta)$ will be zero, and if we arrange the data one tree after another then $\Omega(\theta)$ will be block diagonal).

The model has a fixed cubic of age giving the overall average growth curve, and cubic functions of age with random coefficients given each tree's deviation from the average trajectory. The expected tree specific deviation is zero. It can be estimated using `lmer`, but to do so it helps to specify the cubic polynomials using the `poly` function. This does not change the model, but uses a more computationally stable representation of the polynomials than we would get by using, e.g. `~ age + I(age^2) + I(age^3)`, although the individual parameter interpretations of course change when we do this.

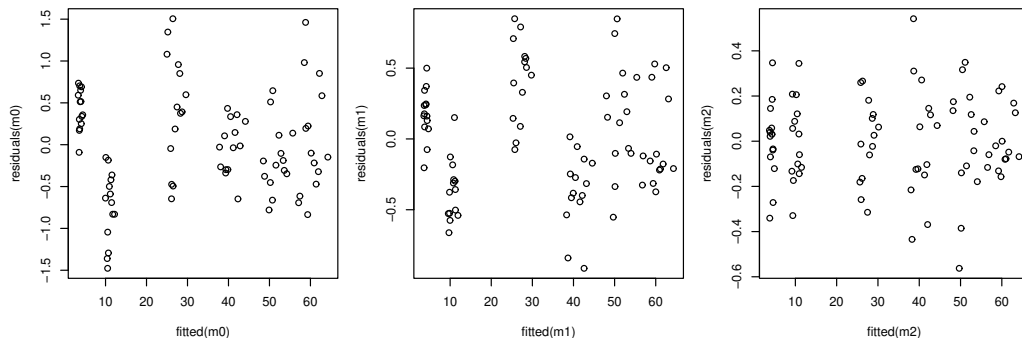
```
m0 <- lmer(height~poly(age,3)+(poly(age,3)|Seed),data=Loblolly)
m0
Linear mixed model fit by REML ['lmerMod']
Formula: height ~ poly(age, 3) + (poly(age, 3) | Seed)
Data: Loblolly
REML criterion at convergence: 217.2254
Random effects:
Groups   Name                Std.Dev. Corr
Seed     (Intercept)          1.4236
         poly(age, 3)1    6.1589    1.00
         poly(age, 3)2    2.0838   -1.00 -1.00
         poly(age, 3)3    0.1068   -1.00 -1.00  1.00
Residual                    0.6886
Number of obs: 84, groups: Seed, 14
Fixed Effects:
(Intercept) poly(age, 3)1 poly(age, 3)2 poly(age, 3)3
 32.36440    186.44570    -21.84656     0.05782
```

Notice how the standard deviations and correlations of the random effects are reported, along with the estimates of the fixed effects. As with any model we should check residuals before proceeding, and in this case they are clearly unsatisfactory, implying that the fixed effects model should be more flexible. So try

```

m1 <- lmer(height~poly(age,4)+(poly(age,3)|Seed),data=Loblolly)
m2 <- lmer(height~poly(age,5)+(poly(age,3)|Seed),data=Loblolly)
par(mfrow=c(1,3))
plot(fitted(m0),residuals(m0))
plot(fitted(m1),residuals(m1))
plot(fitted(m2),residuals(m2))

```

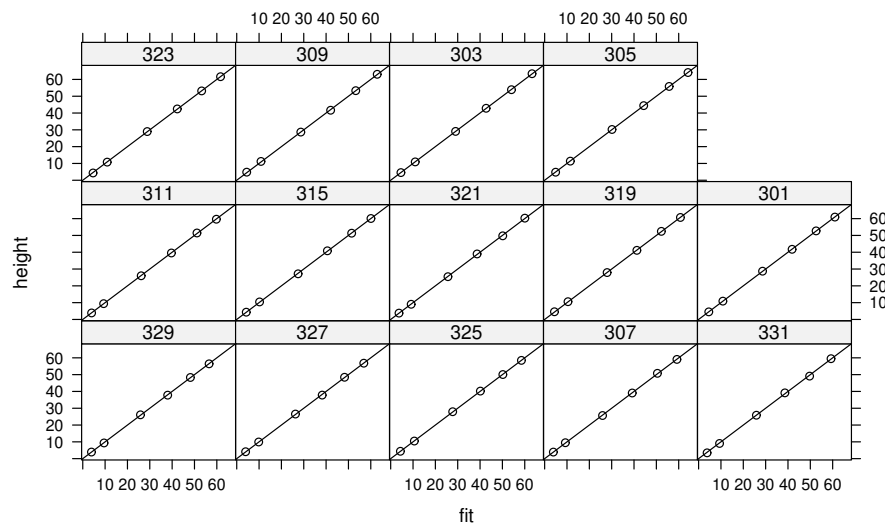


Clearly we need m_2 with a 5th order fixed effect term to get reasonable residuals. See `?plot.merMod` for many more residual plot options. An obvious check is to see how well the model is predicting by tree. The following plots height against its model fitted value, with the line of perfect fit overlaid:

```

Lob <- Loblolly; Lob$fit <- fitted(m2) ## copy fitted to data
xyplot(height ~ fit | Seed, data = Lob,
panel = function(x, y) { panel.xyplot(x, y); panel.abline(0,1)})

```



The fit is very close.

All three models assumed that the random effects were correlated within each tree. We could also try modelling them as independent. Double conditioning bars for a random effect achieve this:

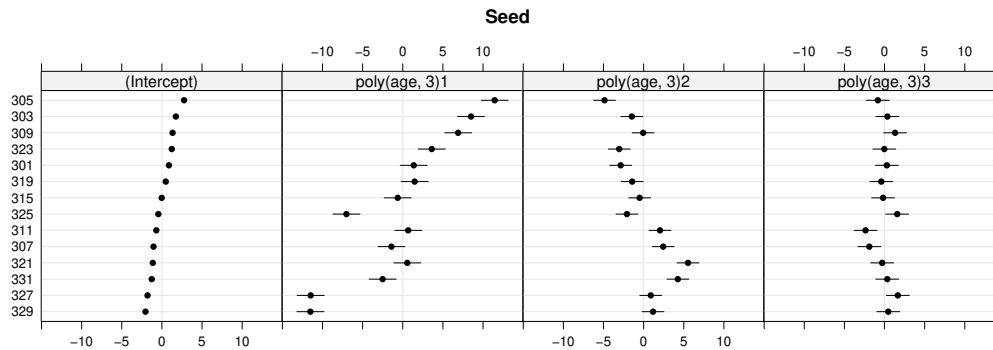
```

m3 <- lmer(height~poly(age,5)+(poly(age,3)||Seed),data=Loblolly)
AIC(m2,m3) ## use AIC to compare models
df      AIC
m2 17 173.4263
m3 14 208.0855

```

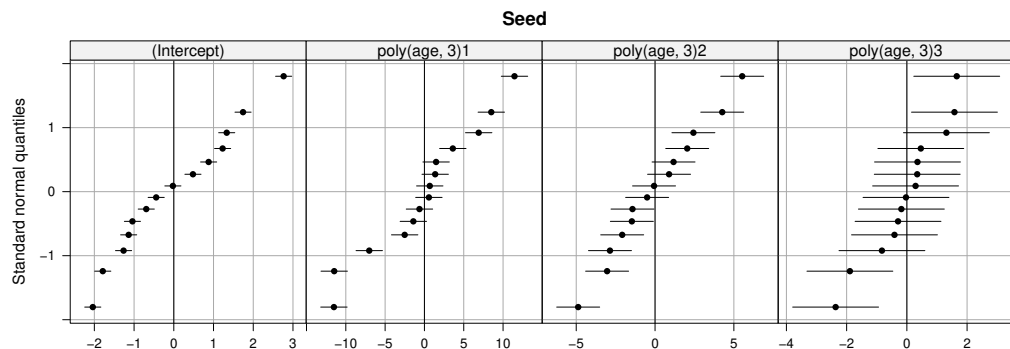
Independent random effects do not look plausible here: `m2` has a substantially lower AIC than `m3`. Another sensible check is to look at the predictions of the random effects, partly to judge the reasonableness of the normality assumption for the random effects. `lme4` offers functions to do this. For example, here are the random polynomial coefficients by tree:

```
rr <- ranef(m2) ## extract predicted random effects
dotplot(rr)
```



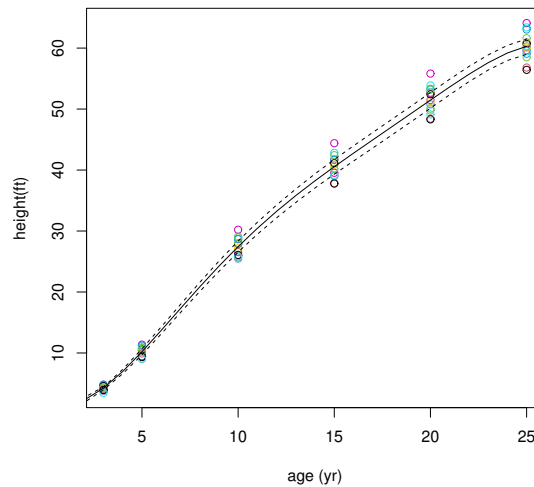
and here are the corresponding normal QQ-plots

```
qqmath(rr)
```



...not exactly normal, but not massively far out either. So, given that the model seems reasonable, we can continue and examine the mean growth curve that is the thing of interest here. The following code overlays the predicted mean growth trajectory and 95% CI over the original data. As mentioned previously, computing confidence intervals in mixed model contexts can be challenging, and the authors of `lme4` currently favour a bootstrapping approach, so that is what is used here. As arguments the `bootMer` function takes a fitted model and a function defining the quantity of interest to be computed from the fitted model. Parametric bootstrapping is then used to assess the sampling variability of quantity of interest, and from this CIs can be computed.

```
fv <- predict(m2, re.form = ~0, newdata=data.frame(age=1:25))
bs <- bootMer(m2, function(x)
  predict(x, re.form = ~0, newdata=data.frame(age=1:25)), nsim=100)
ci <- confint(bs) ## get CI from bootstrap output
with(lob, plot(age, height, col=Seed, xlab="age (yr)", ylab="height (ft)"))
lines(1:25, fv, ylim=range(ci))
lines(1:25, ci[,1], lty=2); lines(1:25, ci[,2], lty=2)
```

Looking at this plot raises the immediate question of whether the constant residual variance assumption of the LMM was most appropriate, or whether a GLMM using a gamma distribution, with a log link to ensure positivity might be better. `glmer` allows us to fit this: again it works like `glm`, but with random effect terms allowed in the model formula.

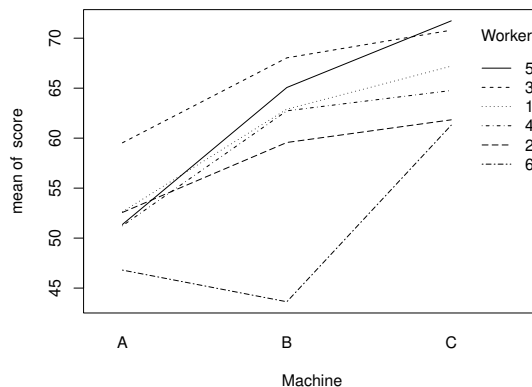
```
m4 <- glmer(height~poly(age,5)+(poly(age,3)|Seed),data=Loblolly,
            family=Gamma(link=log))
m4
Generalized linear mixed model fit by maximum likelihood (Laplace
Approximation) [glmerMod]
Family: Gamma ( log )
Formula: height ~ poly(age, 5) + (poly(age, 3) | Seed)
Data: Loblolly
      AIC      BIC    logLik deviance df.resid
58.7420 100.0659 -12.3710  24.7420      67
Random effects:
Groups   Name                      Std.Dev. Corr
Seed     (Intercept)                0.02919
         poly(age, 3)1              0.16542  -0.46
         poly(age, 3)2              0.09081   0.22 -0.48
         poly(age, 3)3              0.07791  -0.14  0.09 -0.53
Residual                      0.01349
Number of obs: 84, groups: Seed, 14
Fixed Effects:
      (Intercept) poly(age, 5)1 poly(age, 5)2 poly(age, 5)3 poly(age, 5)4
           3.1347          8.0301          -3.1912           1.2679          -0.5061
poly(age, 5)5
           0.1324
```

The residuals versus fitted values plots for this look fine, and the estimates average growth is very similar. Unfortunately at time of writing the `bootMer` function fails for this model, which appears to be a bug, rather than something wrong with the model or principle of bootstrapping.

14.3 Several levels of nesting

The `Machines` data frame (from package `nlme`) contains data on the productivity score of 6 Workers on each of 3 Machines. Each worker has three scores on each of the three machines. Interest is in establishing which of the machines leads to highest productivity. First let's plot the mean score for each worker on each machine.

```
with(Machines, interaction.plot(Machine, Worker, score))
```



Clearly any model will require a fixed effect for `Machine`, and random effects for `Worker`, but from the plot it looks like there is a `Machine Worker` interaction as well. Since the `Worker` effect is random, meaning that we view the workers as random samples from the population of all suitable workers, then the interaction will need to be random as well. That is a model for the score of the k th score of the i th work on the j th machine something like:

$$\text{score}_{ijk} = \beta_j + b_i + a_{ij} + \epsilon_{ijk}$$

where b_i and a_{ij} are independent Gaussian random effects. This can be estimated using.

```
a1 <- lmer(score ~ Machine + (1|Worker) + (1|Worker:Machine), data=Machines)
```

We might also want to try the slightly more complicated model in which the interaction terms are correlated for each worker, which can be achieved by this call:

```
a2 <- lmer(score ~ Machine + (1|Worker) + (Machine-1|Worker), data=Machines)
```

For completeness let's also try the implausible model without an interaction, and compare all three models by AIC

```
a0 <- lmer(score ~ Machine + (1|Worker), data=Machines)
```

```
AIC(a0, a1, a2)
      df      AIC
a0    5 296.8782
a1    6 227.6876
a2   11 230.3112
```

The interaction is clearly needed, but it seems that treating the interaction random effects as independent is fine here. Residual plots produced by `plot(a1)` are OK, so we can compute confidence intervals suitable for addressing the original question.

```
confint(a1)
Computing profile confidence intervals ...
      2.5 %    97.5 %
.sig01  2.3528033  5.431501
.sig02  1.9514581  9.410584
.sigma   0.7759507  1.234966
(Intercept) 47.3951611 57.315949
MachineB    3.7380904 12.195243
MachineC    9.6880904 18.145243
```

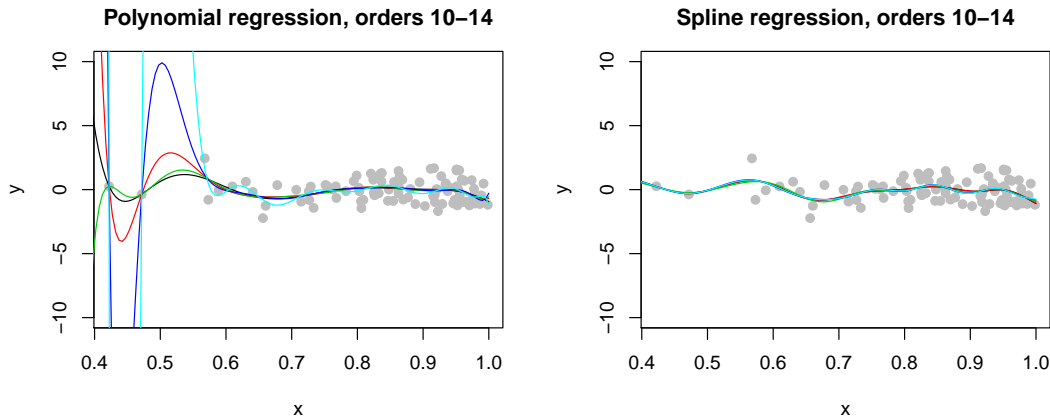


Figure 9: Left: least squares fits of polynomials of order 10-14 to the data shown as grey dots. Right: fits of splines of order 10-14 to the same data. Clearly the spline fit offers much greater statistical stability.

Note that the model is parameterized in terms of an intercept, corresponding to the expected score on Machine A, and parameters corresponding to the difference between the other 2 machines and A. Both B and C clearly give significantly higher scores than A. It appears that C is probably better, but the extent of the overlap in CIs suggests that this effect is unlikely to be statistically significant: i.e. we can not be sure that there is really any difference between the expected scores on B and C. A second experiment with more workers would probably be needed to resolve this. We know we would need more workers rather than more replicates per worker, because of how large the random effect standard deviations are relative to the residual standard deviation.

15 Generalized Additive Models

Previously we have several times considered models in which polynomials were used to model the smooth relationship between the response variable and one or more predictors. For example in section 14.2 polynomials of tree age were used, and in section 5.2 polynomials were used to produce a smooth relationship between tyre rubber loss and the tensile strength and hardness of the rubber. In both examples there was no special reason to use polynomials, we simply needed some way of representing the conceptual model that *the response is related to some smooth function of these predictors*, and polynomials provided an easy way to do so.

But if we want to use models built from smooth functions, are polynomials really the best way to go about it? For the tyre wear example, selecting the appropriate model was a somewhat clumsy process, despite only considering terms up to third order. It would have been very unwieldy if more complexity was called for. And what is the justification for using polynomials anyway? The most obvious is provided by Taylor's theorem: any smooth unknown function can be approximated by a polynomial expansion, with accuracy improving the more terms we include. But this approximation accuracy is in the vicinity of the single point we expanded about: in a statistical model we are usually interested in accuracy across the whole range of observed covariate values.

This latter point is not merely theoretical, as their order increases polynomials become an ever less stable basis for representing smooth functions in a model. Figure 9 shows several polynomial fits to the data shown as grey dots in the figure, in the left panel. The polynomial fits oscillate wildly and vary widely as the order changes in the region where data are limited. On the right are shown fits using an alternative way of representing smooth functions: *cubic splines*. The splines shown have the same degrees of freedom as the polynomials, but display much more stable behaviour, and change much less erratically with order.

You might be concerned that this stability is because the spline of a given order is simply inflexible relative to the equivalent polynomial. i.e. that splines buy statistical stability at the expense of accuracy. That is not the case as figure 10 shows. The spline of a given order can closely approximate a polynomial of the same order over the range of the data being modelled. That's not accidental. Splines arise by considering how best to approximate an unknown function based on a finite set of observations of its value. The answer turns out to be that the function should be represented using *piecewise polynomials*: sections of polynomial joined together to be continuous at the joins only for a limited number of derivatives. For example a *cubic spline* is made up of sections of cubic joined together in such a way that they are continuous only up to second derivative. That's quite a drop in continuity

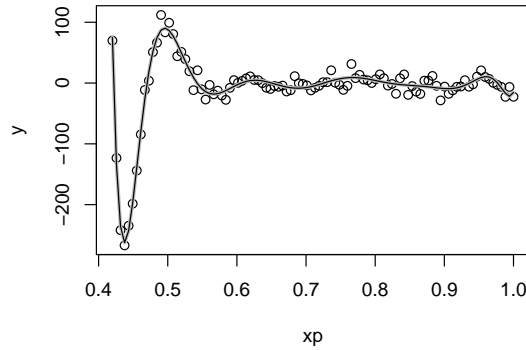


Figure 10: Noisy sample (open circles) were taken for the order 14 polynomial fit shown in figure 9. The grey curve shows the order 14 polynomial re-fit to these data, while the black curve is the order 14 spline fit. Clearly the splines more stable behaviour in figure 9 is not the result of inflexibility.

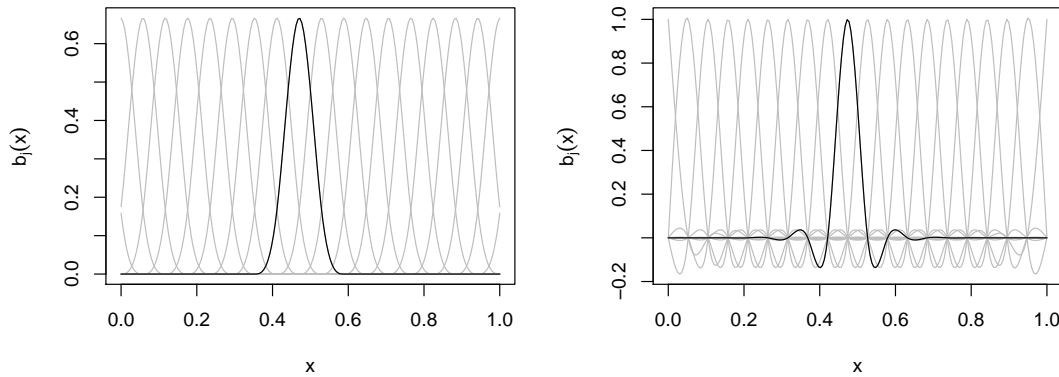


Figure 11: Two equivalent bases for cubic splines. Left: B-spline basis - the grey curves are the full set, with one picked out in black. B-splines are non zero only over a finite interval. Right: Cardinal basis.

relative to polynomials which have all derivatives continuous, but it's what causes the better performance.

Full mathematical details of how splines are derived and computed would take us too far from the main themes of this course, all we need here are some basic principles. A smooth function in a model can be represented using a *basis expansion*

$$f(x_i) = \sum_{j=1}^k \beta_j b_j(x_i)$$

where the β_j are parameters/coefficients to be estimated, and the $b_j(x)$ are fixed known *basis functions*. If $b_j(x) = x^{j-1}$ then we have the simplest polynomial basis. Of course bases are not unique, any invertable linear combination of the basis functions gives another basis for the same function space. The spline bases have the good properties discussed above, and figure 11 illustrates two alternative sets of basis functions for cubic splines. Spline bases for smooth functions of several variables can also be developed.

One possibility is to use such bases exactly as we used polynomials in regression models, but we then have to choose the number of basis functions k , and this becomes unwieldy for complex terms, or smooth functions of several variables. An alternative is to choose k generously enough that we are fairly sure that we are going to overfit rather than underfit, but to control the actual degree of flexibility of the spline using a tunable penalty term applied to likelihood. An example of such a penalty is the cubic spline penalty,

$$\int f''(x)^2 dx = \beta^T \mathbf{S} \beta.$$

This measures the wiggleness[†] of $f(x)$ in terms of its integrated squared second derivative. A straight line fit would

[†] sometimes referred to as *roughness*, but that is misleading for anyone with a physics or engineering background, or familiar with sandpaper

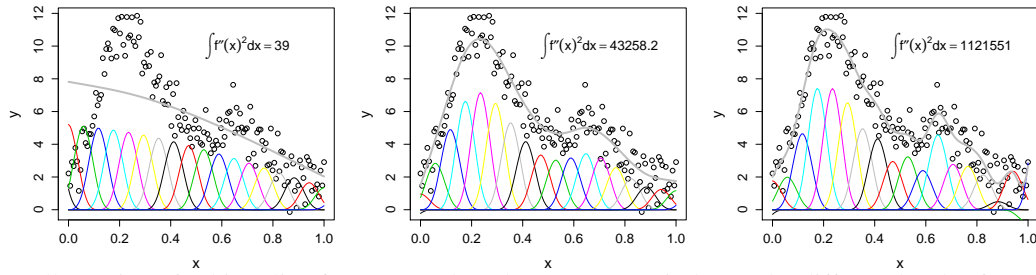


Figure 12: Illustration of cubic spline fits to some data shown as open circles, under different levels of penalization. In each panel the coloured curves show the B-spline basis functions ($b_j(x)$), each multiplied by the corresponding coefficient (β_j): these are summed pointwise, to give the grey curve, fitting the data. In the left panel a high λ was used in fitting, forcing the penalty to be low. This has led to a very smooth curve that does not fit well. In the right panel a very low smoothing parameter is used, so that the penalty is allowed to be very high. This leads to a wiggly curve that overfits the data. In the middle is a better choice.

have zero penalty, and a very wiggly curve a much higher value. That the penalty can be written as $\beta^T \mathbf{S} \beta$, where \mathbf{S} is a known positive semi-definite matrix, follows directly from the fact that $f(x)$ has a linear basis expansion.

So if we were interested in the simple model

$$g(\mu_i) = f(x_i), \quad y_i \sim \text{EF}(\mu_i, \phi)$$

then we would represent $f(x)$ using a spline basis expansion and estimate the model coefficients as

$$\hat{\beta} = \underset{\beta}{\operatorname{argmax}} \quad l(\beta) - \frac{\lambda}{2} \beta^T \mathbf{S} \beta. \quad (23)$$

The fitting objective here balances model goodness of fit as measured by high $l(\beta)$ against model simplicity as measured by $-\beta^T \mathbf{S} \beta$. The *smoothing parameter*, λ , controls the trade-off between the two. A low λ favours a close fit to the data and a wiggly model. A high λ favours a smooth model at the expense of fitting the data closely. Before considering the choice of λ , figure 12 illustrates its effect, as well as how the basis expansion adds up to a fitted function for different λ values.

So how should λ be estimated? A good approach is to recognise that (23) has exactly the same form as (20) in section 13.1. So if we are prepared to view the smoothing penalty as being induced by a *smoothing prior* $\beta \sim N(\mathbf{0}, \mathbf{S}^- / \lambda)$ (\mathbf{S}^- is a pseudoinverse), then we can use the mixed model methods of section 13.1 to estimate the model, including the smoothing parameter, which now plays the role of a precision (inverse variance) parameter.

Now the methods of section 13.1 are not limited to estimating just one precision parameter, so there is nothing to stop us working with models constructed from a sum of several smooth functions of several predictors, resulting in the *generalized additive model* (GAM)

$$g(\mu_i) = \alpha + \sum_j f_j(x_{ij}), \quad y_i \sim \text{EF}(\mu_i, \phi) \quad (24)$$

Unsurprisingly, we can also add in all the sorts of parametric terms usable in a regular GLM, and random effects as in a GLMM. Functions of more than one covariate can also be used. The only extra detail needed for GAMs is the requirement for identifiability constraints on the smooth functions. The problem is that the f_j are only identifiable up to an additive constant. The smoothing penalties/priors do not remove the problem, because the smoothing penalties never act on the intercept of the function: for example $f(x)$ and $f(x) + 100$ have identical cubic spline penalty. This means that an identifiability constraint has to be applied for fitting. $\sum_i f_j(x_{ij}) = 0 \forall j$ does the job.

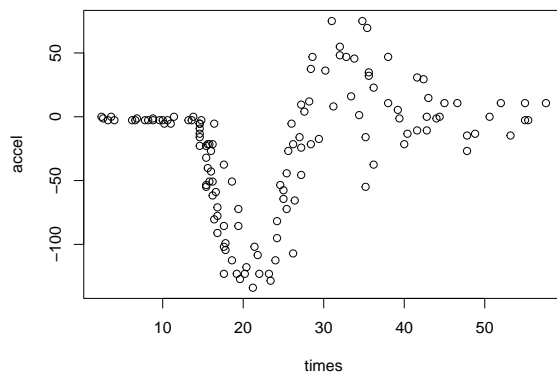
16 GAMs in R

R package `mgcv` supplies a function `gam` for fitting GAMs and their mixed model extensions. Many different smoothers are built in, and distributions well beyond the exponential family are also available. Model building and checking operates in much the same way as with GLMs or GLMMs, although to some extent smoothing parameter estimation takes over part of the model selection process. A new aspect of checking is the need to check that the basis dimension used for representing a smooth term is not too small, so that it might be causing underfit/overfitting.

16.1 A simple smoothing example

To start consider a model involving a smooth function of one variable. The MASS library contains data frame `mcycle` from a motorcycle crash experiment. The data give acceleration of the head of a crash test dummy (g-force) at different times (milliseconds) during a simulated crash. The crash occurs at around 14ms and the data are from multiple experiments (not all from one).

```
library(MASS)
with(mcycle, plot(times, accel))
```



A simple model is $\text{accel}_i = f(\text{times}_i) + \epsilon_i$ where f is a smooth function. Let's try this

```
library(mgcv)
b <- gam(accel ~ s(times), data=mcycle, method="REML")
b
```

```
Family: gaussian
Link function: identity
```

```
Formula:
accel ~ s(times)
```

```
Estimated degrees of freedom:
8.62 total = 9.62
```

```
REML score: 616.142
```

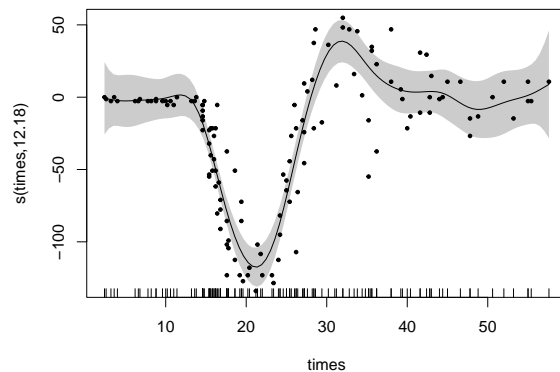
`gam` is similar to `glm` in use, except that the model formula can contain terms specifying smooth functions, such as `s(times)`. `method="REML"` specifies that smoothing parameters are to be estimated by marginal likelihood maximisation[‡]. An alternative is to estimate smoothing parameters by cross validation, i.e. to optimize the average ability of the model to predict data when they are omitted from the fitting data. Cross validation is actually the default for `gam`, but only for historical reasons.

The default print method for a fitted `gam` object reports a small amount of information about it, including the *effective degrees of freedom* for the smooth terms. This requires explanation. Clearly if we impose smoothing penalties, then we are restricting the freedom of the coefficients to adjust to the data, and can no longer view the degrees of freedom of a model as being simply the number of coefficients that it has. In the extreme case, a cubic spline term with a very large smoothing parameter will always produce a straight line fit, with hence 2 degrees of freedom, irrespective of how many basis functions and coefficients are used to represent it. A notion of effective degrees of freedom can be developed by thinking about the average shrinkage of the β_j as a result of penalization, and this is what is reported.

[‡]the RE means 'restricted' and relates to a terminology difference between Bayesian and frequentist approaches that is unimportant here

Now in this case the default basis dimension was used, which is just 10 (see `?s`). That means that the smooth term will have a maximum of 9 degrees of freedom, once the sum to zero identifiability constraint is accounted for (the model includes an intercept by default). The estimated degrees of freedom are very close to this, so we might worry that the basis dimension is too low. To be sure it is better to refit, increasing it. The following does this, setting the basis dimension to 20 and plotting the result.

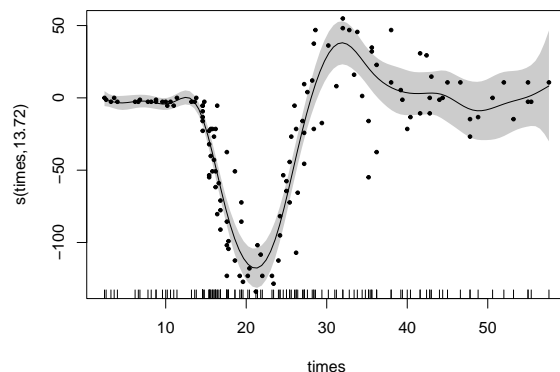
```
b <- gam(accel~s(times,k=20),data=mcycle,method="REML")
plot(b,shift=coef(b)[1],scheme=1)
points(mcycle$times,mcycle$accel,pch=19,cex=.5)
```



The `plot` function uses the default plot method for fitted `gam` objects, which plots all the smooth terms in a model - note the EDF reported in the y axis label. These will have had sum to zero constraints imposed, which shifts the curve by the amount of the model intercept in this case: the `shift` argument has been used to reverse this shift here, so that the scale matches the original data scale. `scheme=1` selects the plotting scheme showing the confidence interval for the smooth as a grey band.

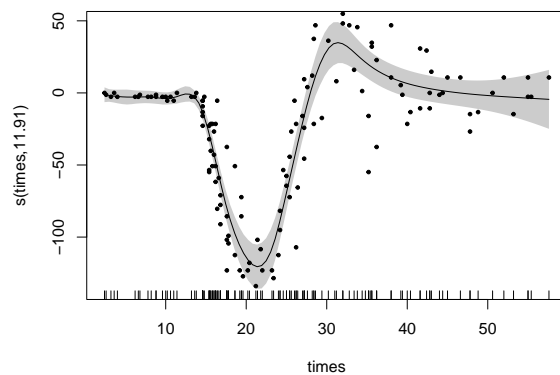
An obvious problem with this model is that it does not reflect the low variability in the data pre-impact at 14ms. One possibility would be to allow the standard deviation of the acceleration to also be modelled as a smooth function of time. That can be done (see `?gaulss`) but is overkill in this case. It is better to simply weight the data before and after impact differently, to reflect the change in residual variance apparent from the initial fit.

```
rsd <- residuals(b)
ii <- mcycle$times < 14 ## index of times before 14
v0 <- var(rsd[ii]); v1 <- var(rsd[!ii]) ## residual vars
w <- c(rep(1/v0,sum(ii)),rep(1/v1,sum(!ii))) ## weights
bw <- gam(accel~s(times,k=20),data=mcycle,method="REML",weights=w)
plot(bw,shift=coef(b)[1],scheme=1)
points(mcycle$times,mcycle$accel,pch=19,cex=.5)
```



This is clearly an improvement, but the curve seems rather wiggly at times when the data don't really suggest this, presumably to allow enough flexibility to capture the rapid variation just after impact. One possible fix is to allow a more complicated penalty, in which the degree of smoothing is also allowed to vary with time. The "ad" adaptive splne smoother does this (default basis dimension 40).

```
baw <- gam(accel~s(times,bs="ad"),data=mcycle,method="REML",weights=w)
plot(baw,shift=coef(b)[1],scheme=1)
points(mcycle$times,mcycle$accel,pch=19,cex=.5)
```



This looks better, and notice how the effective degrees of freedom have dropped a little as the spurious variation at low and high times has now been suppressed. However, although visually pleasing, the adaptive smoother has made little real statistical improvement, despite being quite expensive to fit: such smooths should be used sparingly.

Residual checking is clearly as important for GAMs as for any GLM. The `plot` method for `gam` does not produce residual plots by default, so you need to construct them yourself, but there is one helper function that produces basic plots.

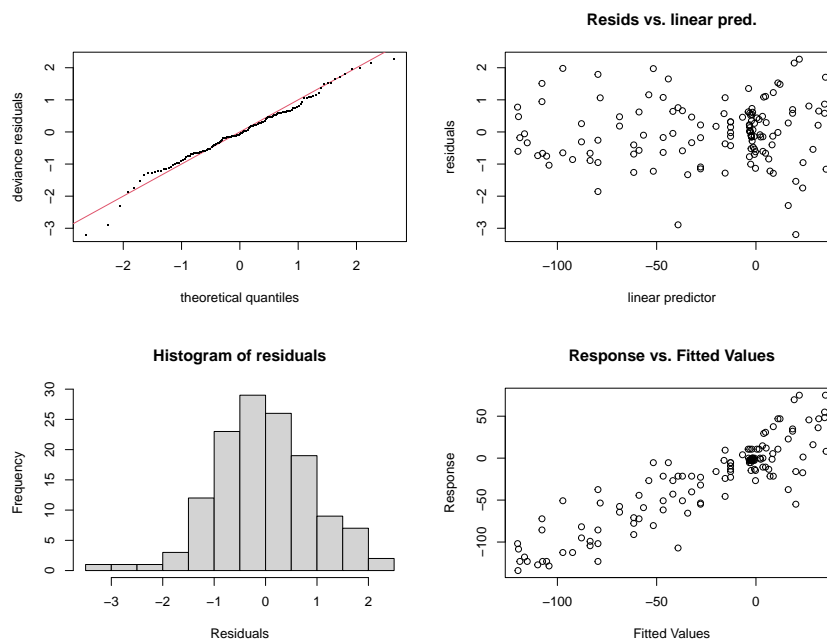
```
gam.check(baw)
```

```
Method: REML   Optimizer: outer newton
full convergence after 13 iterations.
Gradient range [-0.002453041,0.001114406]
(score 578.1708 & scale 0.9795562).
Hessian positive definite, eigenvalue range [0.0001244763,65.70927].
Model rank = 40 / 40
```

```
Basis dimension (k) checking results. Low p-value (k-index<1) may
indicate that k is too low, especially if edf is close to k'.
```

	k'	edf	k-index	p-value
s(times)	39.0	11.9	1.18	0.98

The initial information is technical stuff about algorithm convergence. The model rank is also reported: 40/40 means that the model was fully numerically identifiable. The final table is an informal check of whether the basis dimension is high enough. k' is the maximum effective degrees of freedom for the term, so if `edf` is very close to this, you might need to try increasing the basis dimension (k), especially if the reported p-value is very low. The p-value is for a randomization test that looks for autocorrelation in the residuals when they are ordered w.r.t. the argument of the smooth term. Such auto-correlation might arise because the basis dimension is too small, but it might also arise because of short range auto-correlation that the model can not capture - so whether or not to increase the basis dimension requires thought. Some basic residual plots are also produced.



A summary method for gam objects is also available.

```
summary(baw)
```

```
Family: gaussian
```

```
Link function: identity
```

```
Formula:
```

```
accel ~ s(times, bs = "ad")
```

```
Parametric coefficients:
```

```
              Estimate Std. Error t value Pr(>|t|)
(Intercept)  -25.002      1.796   -13.92  <2e-16 ***
---
```

```
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
Approximate significance of smooth terms:
```

```
              edf Ref.df      F p-value
s(times) 11.91   14.2 40.06  <2e-16 ***
---
```

```
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
R-sq.(adj) =  0.784   Deviance explained = 82.9%
```

```
-REML = 578.17   Scale est. = 0.97956   n = 133
```

All the p-values are for testing whether the corresponding terms could be zero. For the smooth terms these are only approximate. AIC is also available, and of course we can predict just as from a `glm`. Suppose, for example that I want predictions of acceleration at times 22 and 30, with standard errors:

```
predict(baw, newdata = data.frame(times = c(22, 30)), se = TRUE)
```

```
$fit
```

```
      1      2
-118.88652  29.72794
```

```
$se.fit
```

```
      1      2
 8.228211  7.838858
```

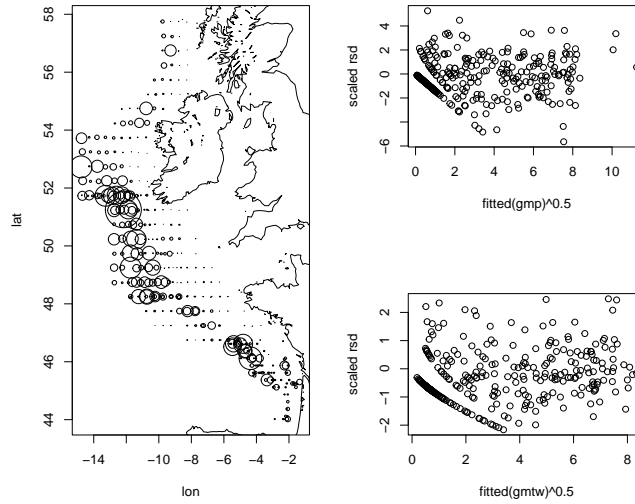


Figure 13: Left: location of mackerel egg survey stations, with symbol size proportional to egg density. Top right scaled residuals against square root fitted values for Poisson GAM model fit - there is clear overdispersion. Lower right: as above for the Tweedie distribution.

16.2 A GAM for fish egg survey data

The `mack` data frame from the `gamair` package contains data from a marine survey of mackerel eggs. Such surveys are undertaken to ascertain the number of eggs in an area, in order to infer how many adult fish there must have been to produce them. This sounds roundabout, but it makes sense because it's much easier to get a proper representative sample of eggs than it is of adult fish. Adult fish tend to avoid sampling nets - eggs do not. The left panel of figure 13 shows the density of eggs per square metre of water surface, by sampling location. Sampling was done by hauling a net vertically through the water column. The figure was produced by

```
library(gamair); data(mack); data(coast)
layout(matrix(c(1,1,2,3),2,2))
with(mack,plot(lon,lat,cex=egg.dens/100))
with(coast,lines(lon,lat))
```

In formulating a model for these data that tries to account for the variation in egg density, there are a number of predictor variables to which the adult fish might be reacting when spawning: depth of the water (`b.depth`), temperature at the surface and at 20m down (`temp.surf` and `temp.20m`), salinity of the water and proximity to the continental shelf edge: a reasonable proxy for this is distance to the 200m depth contour (`c.dist`). The sampling process actually used nets of different areas in different locations, so when trying to model the counts of eggs in the samples we must account for this. A sensible model has the general form $\mathbb{E}(\text{count}_i) = \text{net.area}_i d_i$ where d_i is the egg density. On the log scale this becomes $\log \mathbb{E}(\text{count}_i) = \log(\text{net.area}_i) + \log(d_i)$. We treat the $\log(\text{net.area}_i)$ as a fixed *offset* term - basically a predictor variable with associated coefficient fixed to 1. $\log d_i$ can then be modelled using a smooth additive structure. So an initial model might be something like

$$\log(\mu_i) = \log(\text{net.area}_i) + f_1(\text{lon}_i, \text{lat}_i) + f_2(\text{c.dist}_i) + f_3(\text{salinity}_i) + f_4(\text{temp.surf}_i) + f_5(\text{temp.20m}_i) + f_6(\text{b.depth}_i^{1/2})$$

where $\mu_i = \mathbb{E}(\text{count}_i)$. The f_j are all smooth functions. The square root of `b.depth` reduces the otherwise extreme skew in this variable. In the case of f_1 it is a function of two variables - a *thin plate spline* can be used for this. Since we have count data, a Poisson distribution is natural. The following fits the model and then produces a plot of its scaled Pearson residuals against square root fitted values.

```
mack$log.net.area <- log(mack$net.area) ## create offset
gmp <- gam(egg.count ~ s(lon,lat,k=100) + s(I(b.depth^.5)) +
  s(c.dist) + s(salinity) + s(temp.surf) + s(temp.20m) +
```

```

offset(log.net.area), data=mack, family=poisson, method="REML",
select=TRUE)
plot(fitted(gmp)^.5, residuals(gmp)/gmp$scale^.5, ylab="scaled rsd")

```

The residual plot is at the top right of figure 13. The residuals are substantially *overdispersed* relative to the Poisson assumption. They should really lie much closer to the interval $[-2, 2]$ if the Poisson assumption is valid. Since this is not the case, it is worth trying another distribution. The Tweedie distribution assumes response variance $\phi \mu_i^p$ where ϕ is a scale parameter and p a parameter between 1 and 2, to be estimated. For fixed p it is in the exponential family.

```

gmtw <- gam(egg.count ~ s(lon, lat, k=100) + s(I(b.depth^.5)) +
s(c.dist) + s(salinity) + s(temp.surf) + s(temp.20m) +
offset(log.net.area), data=mack, family=tw, method="REML",
select=TRUE)
plot(fitted(gmtw)^.5, residuals(gmtw)/gmtw$scale^.5, ylab="scaled rsd")

```

The residual plot is shown at lower right of figure 13 - this looks fine now. Notice the `select=TRUE` option to `gam`. This implements one approach to deciding which smooth terms are needed in the model. Each smooth term has an extra quadratic penalty imposed on it, which penalizes its otherwise unpenalized component. If both the smoothing parameters are then estimated to be very large, the term will be penalized to zero - out of the model effectively. Printing the model, we see the effect of this:

```

> gmtw

Family: Tweedie(p=1.333)
Link function: log

Formula:
egg.count ~ s(lon, lat, k = 100) + s(I(b.depth^0.5)) + s(c.dist) +
s(salinity) + s(temp.surf) + s(temp.20m) + offset(log.net.area)

Estimated degrees of freedom:
39.0321  2.3761  0.8347  0.0012  0.0010  4.2384  total = 47.48

REML score: 927.7768

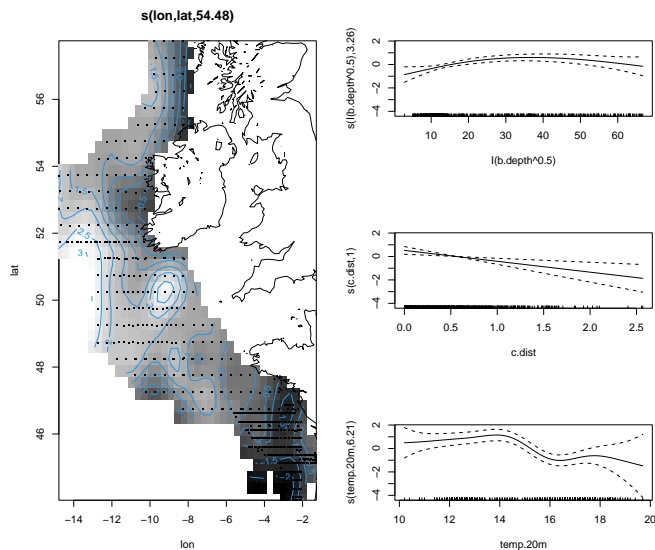
```

The degree of freedom for each smooth are reported, and for salinity and surface temperature they are basically zero. These terms can be dropped. For these data it is worth re-fitting without the two dropped terms: salinity measurements are not available for all the sampling stations, which means that they have to be dropped from fitting if salinity is included in the model. Dropping salinity allows more data to be used. So let's refit, and plot the resulting effects

```

gmtw2 <- gam(egg.count ~ s(lon, lat, k=100) + s(I(b.depth^.5)) +
s(c.dist) + s(temp.20m) + offset(log.net.area), data=mack,
family=tw, method="REML")
layout(matrix(c(1,1,1,2,3,4), 3, 2)) ## pretty graphics layout
plot(gmtw2, scheme=3, select=1, too.far=.03) ## select just spatial
with(coast, lines(lon, lat)) ## so coast can be added
for (i in 2:4) plot(gmtw2, scheme=3, select=i) ## the rest

```



The spatial effect is doing much of the work here. For this sort of model to be useful, we need to be able to predict from it over the spatial area of interest. That means that we need the covariates over a grid covering the spatial area, not just at the sample stations. The `mackp` data frame provides such gridded covariate values, but only at the spatial grid cells where egg population estimates are of interest. We can predict at these locations using `predict`, but will need to copy the predictions for these locations into the relevant cells of a regular grid for plotting. All a bit fiddly, as real data analysis often is, but here it is:

```
data(mackp)
par(mfrow=c(1,2)) ## dividing the plot window
mackp$log.net.area <- rep(0,nrow(mackp)) ## setting the prediction offset to 0
lon <- seq(-15,-1,1/4); lat <- seq(44,58,1/4) ## the lon and lat axes
zz <- array(NA,57*57) ## a vector for predictions
zz[mackp$area.index] <- predict(gmtw2,mackp) ## fill in the interesting cells
image(lon,lat,matrix(zz,57,57),col=gray(0:32/32), ## plot
      cex.lab=1.5,cex.axis=1.4)
contour(lon,lat,matrix(zz,57,57),add=TRUE) ## add contours
with(coast,lines(lon,lat)) ## and the coast
```

The result is shown in the left panel of figure 14.

An interesting alternative model is this

$$\log(\mu_i) = \log(\text{net.area}_i) + f_1(\text{lon}_i, \text{lat}_i) + f_2(\text{lon}_i, \text{lat}_i)\text{temp.20m}_i + f_3(\text{lon}_i, \text{lat}_i)\text{b.depth}_i^{1/2}$$

...essentially a linear regression on temp.20m and $\text{b.depth}_i^{1/2}$ in which the regression coefficients vary smoothly over space. Such models are sometimes known as *geographic regression* or more generally as *varying coefficient* models. The following fits this, with the `by` argument to `s()` achieving the required multiplication of smooth term and covariate.

```
gmgr <- gam(egg.count ~s(lon,lat,k=100)+s(lon,lat,by=temp.20m)
           +s(lon,lat,by=I(b.depth^.5)) +offset(log.net.area),
           data=mack,family=tw,method="REML")
zz[mackp$area.index]<-predict(gmgr,mackp)
image(lon,lat,matrix(zz,57,57),col=gray(0:32/32),
      cex.lab=1.5,cex.axis=1.4)
contour(lon,lat,matrix(zz,57,57),add=TRUE)
with(coast,(lines(lon,lat)))
```

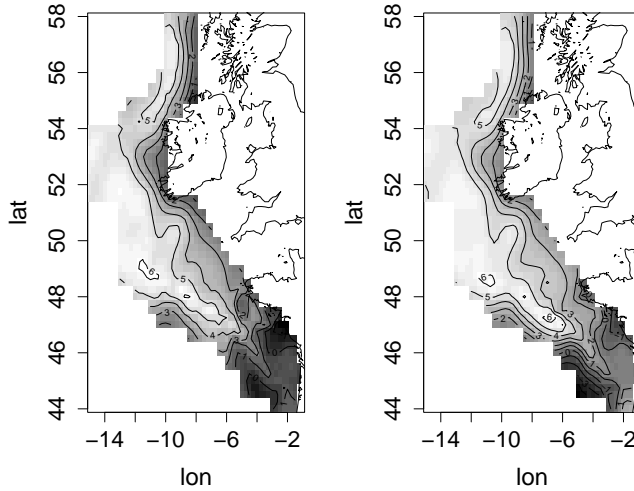


Figure 14: Left: Egg per square metre predictions from the GAM `gmtw2`. Right: the same for the geographic regression model.

The resulting predictions are shown on the right of figure 14. Interestingly this model has a lower AIC than the previous one.

16.3 Smooth interactions and smooth ANOVA decompositions

In the previous example the smooth of spatial location was *isotropic*. Wiggleness was penalized equally in all directions. That is not always appropriate. For example if you want to smooth w.r.t. spatial location and time, what would it even mean to penalize wiggleness equally with respect to location and time? They are not even measured in comparable units. The solution to this is to think about the general notion of a statistical interaction and generalize it to the smooth function case. An interaction is when the effect of one variable is itself modified by variation in another variable. So a smooth interaction of x and z , would require a smooth function $f_x(x)$ to change its form smoothly as z changes its value. We can achieve this by writing the coefficients of the spline representing $f_x(x)$ as smooth functions of z , by allowing each to be represented by a spline of z . We then attach two penalties to the smooth - one for smoothness in the x direction and another for smoothness in the z direction. They each have their own smoothing parameter. Doing this results in a *tensor product* spline smoother that is invariant to linear rescaling of x and z , so that it is immaterial whether they have the same units or not.

`gam` implements tensor product smoothers using `te()` terms in the model formula. Of course we might also be interested in models that decompose smooth effects into main effects and pure interactions of the form:

$$f_x(x) + f_z(z) + f_{xz}(x, z).$$

In that case it is important the basis for f_{xz} excludes functions of the form $f_x(x) + f_z(z)$, or there will be identifiability problems. `ti()` terms are the way to specify smooth interactions in a `gam` formula, constructed to exclude lower order interactions and main effects.

To see smooth interactions in action, consider the `wesdr` data frame, which contains results from a clinical study of diabetic patients examining the development of diabetic retinopathy. The data frame contains a binary indicator, `ret`, of whether the patient developed retinopathy or not, and three predictors whose influence on the probability of retinopathy is of interest: body mass index (`bmi`), duration of disease at enrolment in the study (`dur`) and percentage glycosylated haemoglobin in the blood (`gly` - about 3% in non diabetics). Obviously $\text{ret}_i \sim \text{binom}(1, \mu_i)$ is appropriate, and we might try the model

$$\text{logit}(\mu_i) = f_1(\text{dur}_i) + f_2(\text{gly}_i) + f_3(\text{bmi}_i) + f_4(\text{dur}_i, \text{gly}_i) + f_5(\text{dur}_i, \text{bmi}_i) + f_6(\text{bmi}_i, \text{gly}_i)$$

where f_1 to f_3 are smooth ‘main effects’, while f_4 to f_6 are smooth ‘interactions’ constructed to exclude the main effects. Here is the code to fit the model.

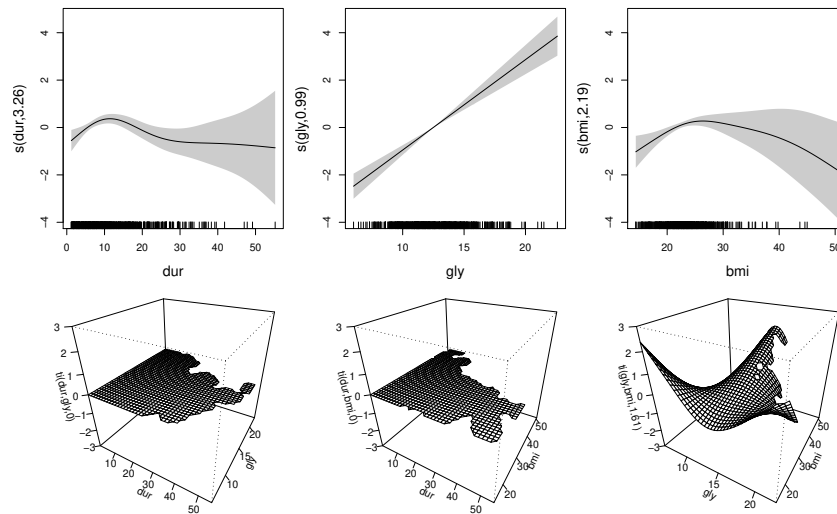


Figure 15: Estimated effects for the smooth ANOVA logistic regression model for retinopathy risk. Notice how only the gly-bmi interaction is estimated to be non-zero.

```
library(gamair); data(wesdr)
k <- 7
b <- gam(ret ~ s(dur,k=k) + s(gly,k=k) + s(bmi,k=k) +
          ti(dur,gly,k=k) + ti(dur,bmi,k=k) + ti(gly,bmi,k=k),
          select=TRUE, data=wesdr, family=binomial(), method="ML")
b
Family: binomial
Link function: logit

Formula:
ret ~ s(dur, k = k) + s(gly, k = k) + s(bmi, k = k) + ti(dur,
  gly, k = k) + ti(dur, bmi, k = k) + ti(gly, bmi, k = k)
The TheT
Estimated degrees of freedom:
3.2553 0.9892 2.1866 0.0003 0.0001 1.6118 total = 9.04

ML score: 385.7904
```

The estimated effects from this model are shown in figure 15. The interaction of gly and bmi seems to be important, but the others not. Hence the effect of dur can be interpreted from its main effect only, but for gly and bmi, we probably want to interpret their combined effect from main effects and interactions. Figure 16 does this, using `vis.gam` to visualize the combined effect conveniently.

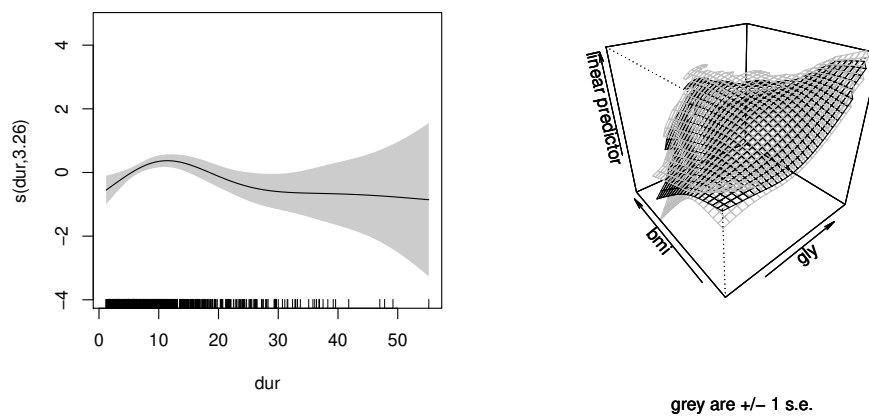


Figure 16: Estimated duration effect and combined effect of BMI and percent glycosylated hemoglobin from the retinopathy model. On the right, the grey surfaces are at plus or minus one standard error from the estimate. Notice how a higher level of glycosylated hemoglobin seems to be tolerated when body mass index is low.