7

Statistical Modelling

Fitting models to data is the central function of R. The process is essentially one of exploration; there are no fixed rules and no absolutes. The object is to determine a minimal adequate model from the large set of potential models that might be used to describe the given set of data. In this book we discuss five types of model:

- the null model,
- the minimal adequate model,
- the current model,
- the maximal model, and
- the saturated model.

The step-wise progression from the saturated model (or the maximal model, whichever is appropriate) through a series of simplifications to the minimal adequate model is made on the basis of **deletion tests** – F-tests or chi-squared tests that assess the significance of the increase in deviance that results when a given term is removed from the current model.

Models are representations of reality that should be both accurate and convenient. However, it is impossible to maximize a model's realism, generality and holism simultaneously, and the principle of parsimony (or Occam's razor; see p. 7) is a vital tool in helping to choose one model over another. Thus, we would only include an explanatory variable in a model if it significantly improved the fit of a model. Just because we went to the trouble of measuring something, does not mean we have to have it in our model. Parsimony says that, other things being equal, we prefer:

- a model with $n - 1$ parameters to a model with n parameters,
- a model with $k - 1$ explanatory variables to a model with k explanatory variables,
- a linear model to a model which is curved,
- a model without a hump to a model with a hump, and
- a model without interactions to a model containing interactions between factors.

Statistics: An Introduction using R M. J. Crawley
© 2005 John Wiley & Sons, Ltd ISBNs: 0-470-02298-1 (PBK); 0-470-02297-3 (PPC)

Other considerations include a preference for models containing explanatory variables that are easy to measure over variables that are difficult or expensive to measure. Also, we prefer models that are based on a sound mechanistic understanding of the process over purely empirical functions.

Parsimony requires that the model should be as simple as possible. This means that the model should not contain any redundant parameters or factor levels. We achieve this by fitting a maximal model then simplifying it by following one or more of these steps:

- remove non-significant interaction terms,
- remove non-significant quadratic or other non-linear terms,
- remove non-significant explanatory variables,
- group together factor levels that do not differ from one another, and
- in Ancova, set non-significant slopes of continuous explanatory variables to zero.

All the above are subject, of course, to the caveats that the simplifications make good scientific sense, and do not lead to significant reductions in explanatory power.

Just as there is no perfect model, so there may be no optimal scale of measurement for a model. Suppose, for example, we had a process that had Poisson errors with multiplicative effects amongst the explanatory variables. Then, one must choose between three different scales, each of which optimizes one of three different properties:

- the scale of $\sqrt{y}$ would give constancy of variance;
- the scale of $y^{\frac{2}{3}}$ would give approximately normal errors;
- the scale of $\ln(y)$ would give additivity.

Thus, any measurement scale is always going to be a compromise, and you should choose the scale that gives the best overall performance of the model.

Model	Interpretation
Saturated model	One parameter for every data point Fit: perfect Degrees of freedom: none Explanatory power of the model: none
Maximal model	Contains all (p) factors, interactions and covariates that might be of any interest. Many of the model's terms are likely to be insignificant Degrees of freedom: $n - p - 1$ Explanatory power of the model: it depends
Minimal adequate model	A simplified model with $0 \leq p' \leq p$ parameters Fit: less than the maximal model, but not significantly so Degrees of freedom: $n - p' - 1$

	Explanatory power of the model: $r^2 = SSR/SSY$
Null model	Just one parameter, the overall mean $\bar{y}$ Fit: none; $SSE = SSY$ Degrees of freedom: $n - 1$ Explanatory power of the model: none

The Steps Involved in Model Simplification

There are no hard and fast rules, but the procedure laid out below works well in practice. With large numbers of explanatory variables, and many interactions and non-linear terms, the process of model simplification can take a very long time. However, this is time well spent because it reduces the risk of overlooking an important aspect of the data. It is important to realize that there is no guaranteed way of finding all the important structures in a complex dataframe.

Step	Procedure	Explanation
1	Fit the maximal model	Fit all the factors, interactions and covariates of interest. Note the residual deviance. If you are using Poisson or binomial errors, check for overdispersion and rescale if necessary.
2	Begin model simplification	Inspect the parameter estimates using **summary**. Remove the least significant terms first, using **update -**, starting with the highest-order interactions.
3	If the deletion causes an insignificant increase in deviance	Leave that term out of the model. Inspect the parameter values again. Remove the least significant term remaining.
4	If the deletion causes a significant increase in deviance	Put the term back in the model using **update** +. These are the statistically significant terms as assessed by deletion from the maximal model.
5	Keep removing terms from the model	Repeat steps 3 or 4 until the model contains nothing but significant terms. This is the minimal adequate model. If none of the parameters is significant, then the minimal adequate model is the null model.

Caveats

Model simplification is an important process but it should not be taken to extremes. For example, the interpretation of deviances and standard errors produced with fixed parameters that have been estimated from the data, should be undertaken with caution. Again, the search for 'nice numbers' should not be pursued uncritically. Sometimes there are good scientific reasons for using a particular number (e.g. a power of 0.66 in an allometric relationship between respiration and body mass). It is much more straightforward, for example, to say that yield increases by 2 kg per hectare for every extra unit of fertilizer, than to say that it increases by 1.947 kg. Similarly, it may be preferable to say that the odds of infection increase ten-fold under a given treatment, than to say that the logits increase by 2.321; without model simplification this is equivalent to saying that there is a 10.186-fold increase in the odds. It would be absurd, however, to fix on an estimate of 6 rather than 6.1 just because 6 is a whole number.

Order of Deletion

Remember that **order matters**. If your explanatory variables are correlated with each other, then the significance you attach to a given explanatory variable will depend upon whether you delete it from a maximal model or add it to the null model. If you always test by model simplification then you won't fall into this trap.

The fact that you have laboured long and hard to include a particular experimental treatment does not justify the retention of that factor in the model if the analysis shows it to have no explanatory power. Anova tables are often published containing a mixture of significant and non-significant effects. This is not a problem in orthogonal designs, because sums of squares can be unequivocally attributed to each factor and interaction term. However, as soon as there are missing values or unequal weights, then it is impossible to tell how the parameter estimates and standard errors of the significant terms would have been altered if the non-significant terms had been deleted. The best practice is:

- say whether your data are orthogonal or not,
- present a minimal adequate model,
- give a list of the non-significant terms that were omitted, and the deviance changes that resulted from their deletion.

The reader can then judge for themselves the relative magnitude of the non-significant factors, and the importance of correlations between the explanatory variables.

The temptation to retain terms in the model that are 'close to significance' should be resisted. The best way to proceed is this. If a result would have been **important** if it had been statistically significant, then it is worth repeating the experiment with higher replication and/or more efficient blocking, in order to demonstrate the importance of the factor in a convincing and statistically acceptable way.

Model Formulae in R

The structure of the model is specified in the model formula like this:

$$\textbf{response variable} \sim \textbf{explanatory variable(s)},$$

where the **tilde** symbol ∼ reads ‘is modelled as a function of’. So a simple linear regression of *y* on *x* would be written like this

y ~ x

and a one-way Anova where sex is a two-level factor would be written like this

y ~ sex.

The right-hand side of the model formula shows

- the number of explanatory variables and their identities–their attributes (e.g. continuous or categorical) are usually defined prior to the model fit,
- the interactions between the explanatory variables (if any),
- non-linear terms is the explanatory variables.

On the right of the tilde, one also has the option to specify offsets or error terms in some special cases. As with the response variable, the explanatory variables can appear as transformations, or as powers or polynomials.

It is very important to note that symbols are used differently in model formulae than in arithmetic expressions. In particular:

+ indicates inclusion of an explanatory variable in the model (not addition);

− indicates deletion of an explanatory variable from the model (not subtraction);

* indicates inclusion of explanatory variables and interactions (not multiplication);

/ indicates nesting of explanatory variables in the model (not division);

| indicates conditioning (e.g. $y \sim x \mid z$ is read as ‘*y* as a function of *x* given *z*’).

There are several other symbols that have special meanings in model formulae, in particular

: colon means an interaction, so that A:B means the two-way interaction between A and B, and N:P:K:Mg means the four-way interaction between N, P, K and Mg.

Some terms can be written in an expanded form. Thus:

A*B*C is the same as A + B + C + A:B + A:C + B:C + A:B:C

A/B/C is the same as A + B%in%A + C%in%B%in%A

(A + B + C)^3 is the same as A*B*C

(A + B + C)^2 is the same as A*B*C – A:B:C.

Interactions Between Explanatory Variables

Interactions between two two-level categorical variables A*B mean that two main effect means and one interaction mean are evaluated. On the other hand, if factor A has three levels and factor B has four levels, then seven parameters are estimated for the main effects (three means for A and four means for B). The number of interaction terms is $(a-1)(b-1)$ where a and b are the numbers of levels of the factors A and B respectively. So in this case, R would estimate $(3-1)(4-1)=6$ parameters for the interaction.

Interactions between two continuous variables are fitted differently. If x and z are two continuous explanatory variables, then x^*z means fit $x+z+x:z$ and the interaction term $x:z$ behaves as if a new variable had been computed that was the point-wise product of the two vectors x and z. The same effect could be obtained by calculating the product explicitly

```
product.xz <- x * z
```

then using the model formula `y ~ x + z + product.xz`. Note that the representation of the interaction by the **product** of the two continuous variables is an assumption, not a fact. The real interaction might be of an altogether different functional form (e.g. `x * z^2`).

Interactions between a categorical variable and a continuous variable are interpreted as an analysis of covariance; a separate slope and intercept are fitted for each level of the categorical variable. So $y \sim \mathrm{A}^*x$ would fit three regression equations if the factor A had three levels; this would estimate six parameters from the data, three slopes and three intercepts.

The slash operator is used to denote nesting. Thus, with categorical variables A and B

```
y~A/B
```

means fit ‘A plus B within A’. This could be written in two other equivalent ways:

```
y~A + A:B
y~A + B %in% A
```

both of which alternatives emphasize that there is no point in attempting to estimate a main effect for B (it is probably just a factor label like ‘tree number 1’ that is of no scientific interest; see p. 185).

Some functions for specifying non-linear terms and higher-order interactions are useful. To fit a polynomial regression in x and z, we could write

```
y~poly(x,3) + poly(z,2)
```

to fit a cubic polynomial in x and a quadratic polynomial in z.

To fit interactions, but only up to a certain level, the ^ operator is useful. This formula

```
y~(A + B + C)^2
```

fits all the main effects and two-way interactions (i.e. it excludes the three-way interaction that A*B*C would have included).

The **I** function (capital letter i) stands for 'as is'. It overrides the interpretation of a model symbol as a formula operator when the intention is to use it as an arithmetic operator. Suppose you wanted to fit $1/x$ as an explanatory variable in a regression, you might try this:

```
y ~ 1/x
```

but this actually does something very peculiar. It fits *x* nested within the intercept! When it appears in a model formula, the slash operator is assumed to imply nesting. To obtain the effect we want, we use **I** to write

```
y ~ I(1/x).
```

We also need to use **I** when we want * to represent multiplication and ^ to mean 'to the power' rather than an interaction model expansion: thus to fit x and x^2 in a quadratic regression we would write

```
y ~ x + I(x^2).
```

Multiple Error Terms

When there is nesting (e.g. split plots in a designed experiment; see p. 177) or temporal pseudoreplication (see p. 13) you can include an error function as part of the model formula. Suppose you had a three-factor factorial experiment with categorical variables A, B and C. The twist is that each treatment is applied to plots of different sizes. A is applied to replicated whole fields, B is applied at random to half fields and C is applied to smaller split–split plots within each field. This is shown in a model formula like this:

```
y ~ A*B*C + Error(A/B/C).
```

Note that the terms within the model formula are separated by asterisks to show that it is a full factorial with all interaction terms included, whereas the terms are separated by slashes in the error statement. There are as many terms in the error statement as there are different sizes of plots – three in this case, although the smallest plot size (C in this example) can be omitted from the list – and the terms are listed left to right from the largest to the smallest plots; see p. 176 for details and examples.

The Intercept as Parameter 1

The simple command

```
y ~ 1
```

causes the null model to be fit. This works out the grand mean (the overall average) of all the data and works out the total deviance (or the total sum of squares, *SSY*, in models with normal errors and the identity link). In some cases, this may be the minimal adequate model; it is possible that none of the explanatory variables we have measured contribute anything significant to our understanding of the variation in the response variable. This is normally what you don't want to happen at the end of your three-year research project.

To remove the intercept (parameter 1) from a regression model (i.e. to force the regression line through the origin, you fit '−1' like this:

```
y~x–1.
```

You should not do this unless you know exactly what you are doing, and exactly why you are doing it (see p. 135 for details). Removing the intercept from an Anova model where all the variables are categorical has a different effect:

```
y~gender–1.
```

This gives the mean for males and the mean for females in the summary table, rather than the overall mean and the difference in mean for males (see Contrasts, p. 209).

Update in Model Simplification

In the update function used during model simplification, the dot '.' is used to specify 'what is there already' on either side of the tilde. So if your original model said

```
model<-lm(y~A*B)
```

then the update function to remove the interaction term A:B could be written like this:

```
model2<-update(model,~. - A:B)
```

Note that there is no need to repeat the name of the response variable, and the punctuation 'tilde dot' means take model as it is, and remove from it ('minus') the interaction term A:B.

Examples of R Model Formulae

Model	Model formula	Comments
Null	y~1	1 is the intercept in regression models, but here it is the overall mean *y*
Regression	y~x	*x* is a continuous explanatory variable
One-way Anova	y~gender	Gender is a two-level categorical variable
Two-way Anova	y~gender + genotype	Genotype is a four-level categorical variable
Factorial Anova	y~N * P * K	N, P and K are two-level factors to be fit along with all their interactions
Three-way Anova	y~N*P*K-N:P:K	As above, but don't fit the three-way interaction

Model	Model formula	Comments
Analysis of covariance	y~x+gender	A common slope for y against x but with two intercepts, one for each gender
Analysis of covariance	y~x * gender	Two slopes and two intercepts
Nested Anova	y~a/b/c	Factor c nested within factor b within factor a
Split-plot Anova	y~a*b*c+Error(a/b/c)	A factorial experiment but with three plots sizes and three different error variances, one for each plot size
Multiple regression	y~x + z	Two continuous explanatory variables, flat surface fit
Multiple regression	y~x * z	Fit an interaction term as well $(x+ z + x : z)$
Multiple regression	y~x+I(xˆ2)+z+I(zˆ2)	Fit a quadratic term for both x and z
Multiple regression	y <- poly(x,2)+z	Fit a quadratic polynomial for x and linear z
Multiple regression	y~(x+z+w)^2	Fit three variables plus all their two-way interactions
Non-parametric model	y~s(x)+lo(z)	y is a function of smoothed x and loess z
Transformed response and explanatory variables	log(y)~I(1/x)+sqrt(z)	All three variables are transformed in the model

Model Formulae for Regression

The important point to grasp is that model formulae look very like equations but there are important differences. Our simplest useful equation looks like this:

$$y = a + bx.$$

It is a two-parameter model with one parameter for the intercept, a, and another for the slope, b, of the graph of the continuous response variable y against a continuous explanatory variable x. The model formula for the same relationship looks like this:

y~x

The equals sign is replaced by a tilde, and all of the parameters are left out. If we had a multiple regression with two continuous explanatory variables x and z the equation would look like this

$$y = a + bx + cz,$$

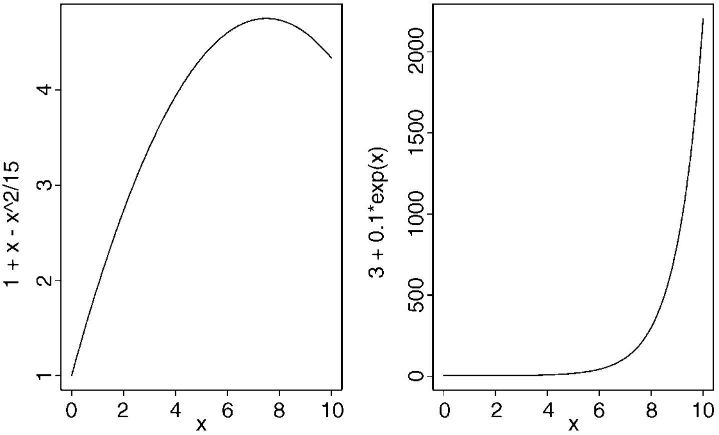
but the model formula is this

```
y~x + z
```

It is all wonderfully simple – but just a minute. How does R know what parameters we want to estimate from the data? We have only told it the names of the explanatory variables. We have said nothing about how to fit them, or what sort of equation we want to fit to the data. The key to this is to understand **what kind of explanatory variable is being fit** to the data. If the explanatory variable x specified on the right of the tilde is a continuous variable, then R **assumes** that you want to do a regression, and hence that you want to estimate two parameters in a linear regression whose equation is $y = a + bx$.

A common misconception is that linear models involve a straight-line relationship between the response variable and the explanatory variables. This is **not** the case, as you can see from these two linear models.

```
par(mfrow=c(1,2))
x<-seq(0,10,0.1)
plot(x,1+x-x^2/15,type="l")
plot(x,3+0.1*exp(x),type="l")
```



The definition of a linear model is an equation that contains mathematical variables, parameters and random variables that is **linear in the parameters and in the random variables**. What this means is that if a, b and c are parameters then obviously

$$y = a + bx$$

is a linear model, but so is

$$y = a + bx - cx^2$$

because x^2 can be replaced by z which gives a linear relationship

$$y = a + bx + cz$$

and so is

$$y = a + be^x$$

because we can create a new variable $z = \exp(x)$, so that

$$y = a + bz.$$

Some models are non-linear but can be readily linearized by transformation. For example:

$$y = \exp(a + bx)$$

is non-linear, but on taking logs of both sides, it becomes

$$\ln(y) = a + bx.$$

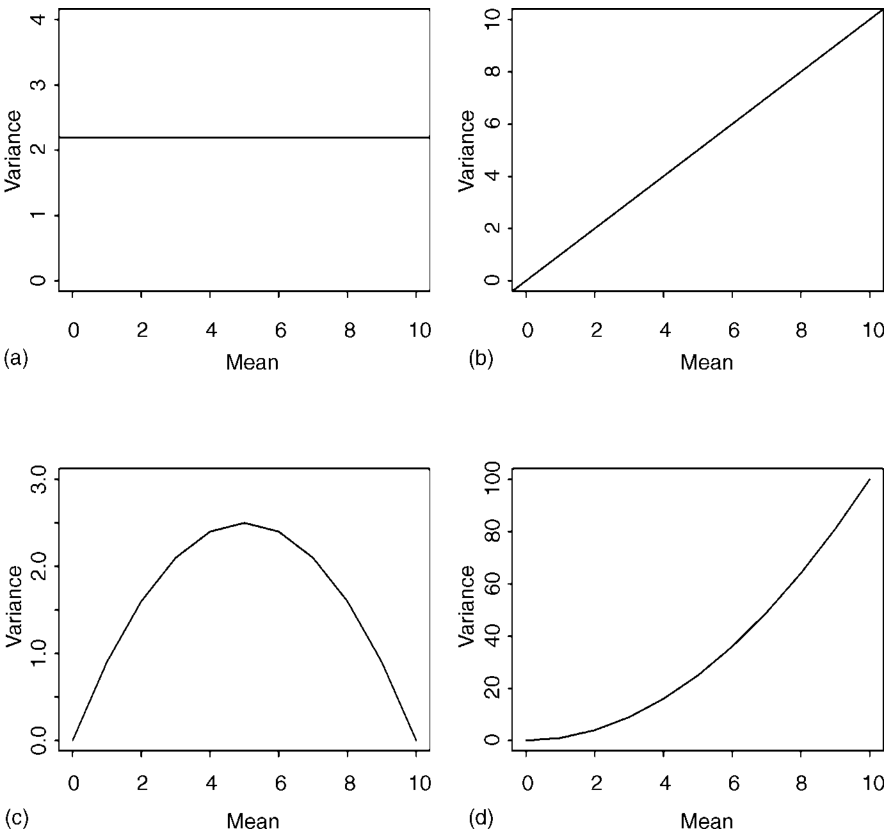
If the equation you want to fit is more complicated than this, then you need to specify the form of the equation, and use non-linear methods (nls or nlme) to fit the model to the data (see p. 149).

GLMs: Generalized Linear Models

We can use glms (pronounced 'glims') when the variance is not constant, and/or when the errors are not normally distributed. Certain kinds of response variables invariably suffer from these two important contraventions of the standard assumptions, and glms are excellent at dealing with them. Specifically, we might consider using glms when the response variable is:

- count data expressed as proportions (e.g. logistic regressions),
- count data that are not proportions (e.g. log linear models of counts),
- binary response variables (e.g. dead or alive), or
- data on time-to-death where the variance increases faster than linearly with the mean (e.g. time data with gamma errors).

The central assumption that we have made up to this point is that variance was constant (a). In count data, however, where the response variable is an integer and there are often lots of zero's in the dataframe, the variance may increase linearly with the mean (b). With proportion data, where we have a count of the number of failures of an event as well as the number of successes, the variance will be a ∩-shaped function of the mean (c). Where the response variable follows a gamma distribution (as in data on time-to-death) the



variance increases faster than linearly with the mean (d). Many of the basic statistical methods like regression and Student's *t*-test assume that variance is constant, but in many applications this assumption is untenable. Hence the great utility of glms.

A generalized linear model has three important properties:

- the **error structure**,
- the **linear predictor**,
- the **link function**.

These are all likely to be unfamiliar concepts. The ideas behind them are straightforward, however, and it is worth learning what each of the concepts involves.

The Error Structure

Up to this point, we have dealt with the statistical analysis of data with normal errors. In practice, however, many kinds of data have non-normal errors. For example:

- errors that are strongly skewed,
- errors that are kurtotic,

- errors that are strictly bounded (as in proportions),
- errors that cannot lead to negative fitted values (as in counts).

In the past, the only tools available to deal with these problems were transformation of the response variable or the adoption of non-parametric methods. A **glm** allows the specification of a variety of different error distributions:

- Poisson errors, useful with count data,
- binomial errors, useful with data on proportions,
- gamma errors, useful with data showing a constant coefficient of variation, and
- exponential errors, useful with data on time-to-death (survival analysis).

The error structure is defined by means of the **family** directive, used as part of the model formula like this:

```
glm(y~z, family = poisson)
```

which means that the response variable *y* has Poisson errors. Or

```
glm(y~z, family = binomial)
```

which means that the response is binary, and the model has binomial errors. As with previous models, the explanatory variable *z* can be continuous (leading to a regression analysis) or categorical (leading to an Anova-like procedure called analysis of deviance, as described below).

The Linear Predictor

The structure of the model relates each observed *y*-value to a predicted value. The predicted value is obtained **by transformation of the value emerging from the linear predictor**. The linear predictor, η (eta), is a linear sum of the effects of one or more explanatory variables, x_j:

$$\eta_i = \sum_{j=1}^{p} x_{ib}\beta_j,$$

where the x's are the values of the p different explanatory variables, and the β's are the (usually) unknown parameters to be estimated from the data. The right-hand side of the equation is called the **linear structure**.

There are as many terms in the linear predictor as there are parameters, p, to be estimated from the data. Thus with a simple regression, the linear predictor is the sum of two terms whose parameters are the intercept and the slope. With a one-way Anova with four treatments, the linear predictor is the sum of four terms leading to the estimation of the mean for each level of the factor. If there are covariates in the model, they add one term each to the linear predictor (the slope of each relationship). Interaction terms in a factorial Anova add one or more parameters to the linear predictor, depending upon the degrees of freedom of each factor (e.g. there would be three extra parameters for the interaction between a two-level factor and a four-level factor, because $(2-1)\times(4-1)=3$).

Fitted Values

To determine the fit of a given model, a **glm** evaluates the linear predictor for each value of the response variable, then compares the predicted value with a **transformed** value of y. The transformation to be employed is specified in the link function, as explained below. The fitted value is computed by applying the reciprocal of the link function, in order to get back to the original scale of measurement of the response variable. Thus, with a log link, the fitted value is the antilog of the linear predictor, and with the reciprocal link, the fitted value is the reciprocal of the linear predictor.

The Link Function

One of the difficult things to grasp about **glm** is the relationship between the values of the response variable (as measured in the data and predicted by the model in fitted values) and the linear predictor. The thing to remember is that the **link function relates the mean value of *y* to its linear predictor**. In symbols, this means that:

$$\eta = g(\mu)$$

which is simple, but needs thinking about. The linear predictor, η (eta), emerges from the linear model as a sum of the terms for each of the p parameters. **This is not a value of *y*** (except in the special case of the **identity link** that we have been using (implicitly) up to now). The value of η is obtained by transforming the value of y by the link function, and the predicted value of y is obtained by applying the inverse link function to η.

The most frequently used link functions are shown below. An important criterion in the choice of link function is to ensure that the fitted values stay within reasonable bounds. We would want to ensure, for example, that counts were all greater than or equal to zero (negative count data would be nonsense). Similarly, if the response variable was the proportion of individuals that died, then the fitted values would have to lie between zero and one (fitted values greater than one or less than zero would be meaningless). In the first case, a log link is appropriate because the fitted values are antilogs of the linear predictor, and all antilogs are greater than or equal to zero. In the second case, the logit link is appropriate because the fitted values are calculated as the antilogs of the log-odds, $\log(p/q)$.

By using different link functions, the performance of a variety of models can be compared directly. The total deviance is the same in each case and we can investigate the consequences of altering our assumptions about precisely how a given change in the linear predictor brings about a response in the fitted value of y. The most appropriate link function is the one which produces the minimum residual deviance.

Canonical Link Functions

The canonical link functions are the default options employed when a particular error structure is specified in the **family** directive in the model formula. Omission of a **link** directive means that the following settings are used:

Error	Canonical link
Normal	*identity*
poisson	*log*
binomial	*logit*
Gamma	*reciprocal*

You should try to memorize these canonical links and to understand why each is appropriate to its associated error distribution. Note that only Gamma errors have a capital initial letter.

Choosing between using a link function (e.g. log link) and transforming the response variable (i.e. having log(y) as the response variable rather than y) takes a certain amount of experience. The decision is usually based on **whether the variance is constant** on the original scale of measurement. If the variance was constant, you would use a link function. If the variance increased with the mean, you would be more likely to log transform the response.

Proportion Data and Binomial Errors

Proportion data have three important properties that affect the way the data should be analysed:

- the data are strictly bounded,
- the variance is non-constant,
- errors are non-normal.

You cannot have a proportion greater than one or less than zero. This has obvious implications for the kinds of functions fitted and for the distributions of residuals around these fitted functions. For example, it makes no sense to have a linear model with a negative slope for proportion data because there would come a point, with high levels of the x variable, that negative proportions would be predicted. Likewise, it makes no sense to have a linear model with a positive slope for proportion data because there would come a point, with high levels of the x variable, that proportions greater than one would be predicted.

With proportion data, if the probability of success is zero, then there will be no successes in repeated trials, all the data will be zeros and hence the variance will be zero.

Likewise, if the probability of success is one, then there will be as many successes as there are trials, and again the variance will be zero. For proportion data, therefore, the variance increases with the mean up to a maximum (when the probability of success is one half) then declines again towards zero as the mean approaches one. The variance mean relationship is humped, rather than constant as assumed in the classical tests.

The final assumption is that the errors (the differences between the data and the fitted values estimated by the model) are normally distributed. This cannot be so in a linear model because the data are bounded above and below: no matter how big a negative residual at high predicted values, $\hat{y}$, a positive residual cannot be bigger than $1 - \hat{y}$. Similarly, no matter how big a positive residual might be for low predicted values $\hat{y}$, a negative residual cannot be greater than $\hat{y}$ (because you cannot have negative proportions). This means that confidence intervals must be asymmetric whenever $\hat{y}$ takes large values (close to one) or small values (close to zero).

All these issues (boundedness, non-constant variance, non-Normal errors) are dealt with by using a generalized linear model with a binomial error structure. It could not be simpler to deal with this. Instead of using a linear model and writing

```
lm(y~x)
```

we use a generalized linear model (glm) and specify that the error family is binomial like this:

```
glm(y~x,family=binomial).
```

That's all there is to it. In fact, it is even easier than that, because we don't need to write 'family='

```
glm(y~x,binomial).
```

Count Data and Poisson Errors

Count data have a number of properties that need to be considered during modelling:

- count data are bounded below (you can't have counts less than zero),
- variance is not constant (variance increases with the mean),
- errors are not normally distributed, and
- the fact that the data are whole numbers (integers) affects the error distribution.

It is very simple to deal with all these issues by using a glm. All we need to write is

```
glm(y~x,poisson)
```

and the model is fitted with a log link (to ensure that the fitted values are bounded below) and Poisson errors (to account for the non-normality).

GAMs: Generalized Additive Models

These models are like glms in that they can have different error structures and different link functions to deal with count data or proportion data. What makes them different is that the shape of the relationship between *y* and a continuous variable *x* is not specified by some explicit functional form. Instead, non-parametric smoothers are used to describe the relationship. This is especially useful for relationships that exhibit complicated shapes, like hump-shaped curves (see p. 195). The model looks just like a glm, except that the relationships we want to be smoothed are prefixed by s: thus, if we had a three-variable multiple regression (three continuous explanatory variables *w*, *x* and *z*) on count data and we wanted to smooth all three explanatory variables, we would write:

```
model<-gam(y~s(w)+s(x)+s(z),poisson)
```

Model Criticism

There is a temptation to become personally attached to a particular model. Statisticians call this 'falling in love with your model'. It is as well to remember the following truths about models:

- all models are wrong,
- some models are better than others,
- the correct model can never be known with certainty, and
- the simpler the model, the better it is.

There are several ways that we can improve things if it turns out that our present model is inadequate:

- transform the response variable,
- transform one or more of the explanatory variables,
- try fitting different explanatory variables if you have any,
- use a different error structure,
- use non-parametric smoothers instead of parametric functions,
- use different weights for different *y* values.

All of these are investigated in the coming chapters. In essence, you need a set of tools to establish whether, and how, your model is inadequate. For example, the model might

- predict some of the *y* values poorly,
- show non-constant variance,
- show non-Normal errors,

- be strongly influenced by a small number of influential data points,
- show some sort of systematic pattern in the residuals, or
- exhibit overdispersion.

Summary of Statistical Models in R

Models are fitted using one of the model-fitting functions as follows.

- **lm**: fits a linear model with normal errors and constant variance; generally this is used for regression analysis using continuous explanatory variables.
- **aov**: fits analysis of variance with normal errors, constant variance and the identity link; generally used for categorical explanatory variables or Ancovas with a mix of categorical and continuous explanatory variables.
- **glm**: fits generalized linear models to data using categorical or continuous explanatory variables, by specifying one of a family of **error structures** (e.g. Poisson for count data or binomial for proportion data) and a particular **link function**.
- **gam**: fits generalized additive models to data with one of a family of error structures (e.g. Poisson for count data or binomial for proportion data) in which the continuous explanatory variables can (optionally) be fitted as arbitrary smoothed functions using non-parametric smoothers rather than specific parametric functions.
- **lme**: fits linear mixed effects models with specified mixtures of fixed effects and random effects and allows for the specification of correlation structure amongst the explanatory variables and autocorrelation of the response variable (e.g. time series effects with repeated measures).
- **nls**: fits a non-linear regression model via least squares, estimating the parameters of a specified non-linear function.
- **nlme**: fits a specified non-linear function in a mixed effects model where the parameters of the non-linear function are assumed to be random effects; allows for the specification of correlation structure amongst the explanatory variables and autocorrelation of the response variable (e.g. time series effects with repeated measures).
- **loess**: fits a local regression model with one or more continuous explanatory variables using non-parametric techniques to produce a smoothed model surface.
- **tree**: fits a regression tree model using binary recursive partitioning whereby the data are successively split along coordinate axes of the explanatory variables so that at any node, the split is chosen that maximally distinguishes the response variable in the left and the right branches. With a categorical response variable, the tree is called a classification tree, and the model used for classification assumes that the response variable follows a multinomial distribution.

For most of these models, a range of **generic functions** can be used to obtain information about the model. The most important and most frequently used are given below.

summary produces parameter estimates and standard errors from **lm**, and Anova tables from **aov**; this will often determine your choice between **lm** and **aov.** For either **lm** or **aov** you can choose **summary.aov** or **summary** · **lm** to get the alternative form of output (an Anova table or a table of parameter estimates and standard errors; see p. 212).

plot produces diagnostic plots for model checking, including residuals against fitted values, influence tests, etc.

anova a wonderfully useful function for comparing different models and producing Anova tables.

update used to modify the last model fit; it saves both typing effort and computing time.

Other useful generics include:

coef the coefficients (estimated parameters) from the model,

fitted the fitted values, predicted by the model for the values of the explanatory variables included,

resid the residuals (the differences between measured and predicted values of y),

predict uses information from the fitted model to produce smooth functions for plotting a line through the scatterplot of your data.

Model Checking

After fitting a model to data we need to investigate how well the model describes the data. In particular, we should look to see if there are any systematic trends in the goodness of fit. For example, does the goodness of fit increase with the observation number, or is it a function of one or more of the explanatory variables? We can work with the raw residuals:

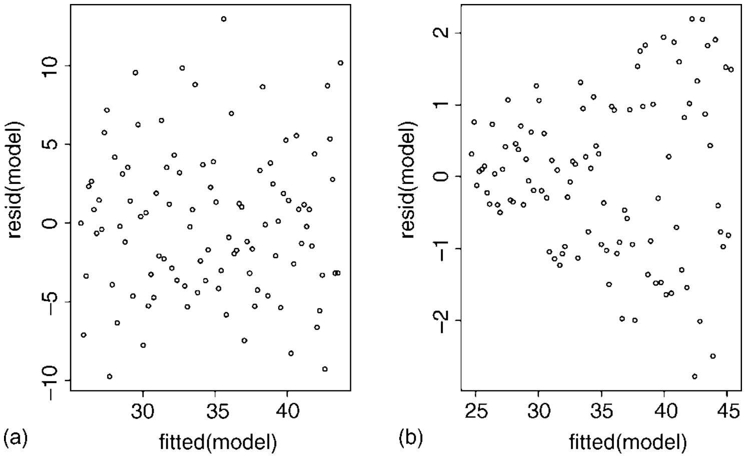
$$\text{residuals} = y - \text{fitted values}.$$

For instance, we should routinely plot the residuals against:

- the fitted values (to look for non-constancy of variance: heteroscedasticity),
- the explanatory variables (to look for evidence of curvature),
- the sequence of data collection (to look for temporal correlation),
- standard normal deviates (to look for non-normality of errors).

Non-constant Variance: Heteroscedasticity

A good model must also account for the variance–mean relationship adequately and produce additive effects on the appropriate scale (as defined by the link function). A plot of standardized residuals against fitted values should look like the sky at night (points scattered at random over the whole plotting region), with no trend in the size or degree of scatter of the residuals. A common problem is that the variance increases with the mean, so that we obtain an expanding, fan-shaped pattern of residuals.



The plot on the left (a) is what we want to see with no trend in the residuals with the fitted values. The plot on the right (b) is a problem. There is a clear pattern of increasing residuals as the fitted values get larger. This is a picture of what **heteroscedasticity** looks like.

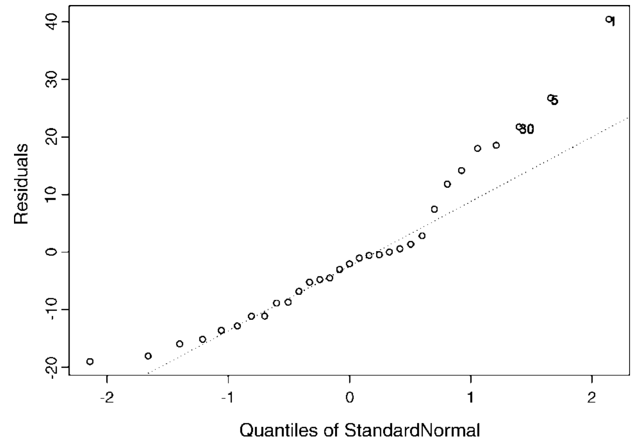
Non-Normality of Errors

Errors may be non-Normal for several reasons. They may be skew, with long tails to the left or right. Or they may be kurtotic, with a flatter or more pointy top to their distribution. In any case, the theory is based on the assumption of Normal errors, and if the errors are **not** Normally distributed, then we shall not know how this affects our interpretation of the data or the inferences we make from it.

It takes considerable experience to interpret the Normal error plots. Here we generate a series of data sets where we introduce different but known kinds of non-Normal errors. Then we plot them using a simple home-made function called mcheck (first developed by John Nelder in the original GLIM language; the name stands for model checking). The idea is to see what patterns are generated in Normal plots by the different kinds of non-Normality. In real applications we would use the generic plot(model) rather than mcheck (see below). First, we write the function mcheck. The idea is to produce two plots, side by side – a plot of the residuals against the fitted values on the left, and a plot of the ordered residuals against the quantiles of the Normal distribution on the right.

```
mcheck <- function (obj,...) {
rs<-obj$resid
fv<-obj$fitted
par(mfrow=c(1,2))
plot(fv,rs,xlab="Fitted values",ylab="Residuals")
abline(h=0, lty=2)
qqnorm(rs,xlab="Normal scores",ylab="Ordered residuals")
qqline(rs,lty=2)
par(mfrow=c(1,1))
invisible(NULL)   }
```

Note the use of $ (**component selection**) to extract the residuals and fitted values from the model object which is passed to the function as obj (the expression x$name is the name **component** of x). The functions qqnorm and qqline are built-in functions to produce Normal probability plots. It is good programming practice to set the graphics parameters back to their default settings before leaving the function.



This is an example of 'banana-shaped' type of non-Normal errors (see p. 227). Other models might produce S-shaped plots of qqnorm (see p. 64).

Influence

One of the commonest reasons for a lack of fit is through the existence of outliers in the data. It is important to understand, however, that a point may **appear** to be an outlier because of mis-specification of the model, and not because there is anything wrong with the data. It is important to understand that analysis of residuals is a very poor way of looking for influence. Precisely because a point is highly influential, means that it forces the regression line close to it, and hence the influential point may have a very small residual.

Leverage

Points increase in influence to the extent that they lie on their own, a long way from the mean value of *x* (either left or right). To account for this, measures of leverage for a given data point y are proportional to $(x - \bar{x})^2$. The commonest measure of leverage is

$$h_i = \frac{1}{n} + \frac{(x_i - \bar{x})^2}{\sum (x_j - \bar{x})^2},$$

where the denominator is *SSX* (see p. 133). A good rule of thumb is that a point is highly influential if its

$$h_i > \frac{2p}{n},$$

where p is the number of parameters in the model. We could easily calculate the leverage value of each point in x. It is more efficient, perhaps, to write a general function that could carry out the calculation of **h** for any vector of x values

```
leverage<-function(x){ 1/length(x)+(x-mean(x))^2/sum((x-mean(x))^2) }.
```

Mis-specified Model

The model may have the wrong terms in it, or the terms may be included in the model in the wrong way. Here we simply note that **transformation of the explanatory variables** often produces improvements in model performance. The most frequently used transformations are logs, powers and reciprocals.

When both the error distribution and functional form of the relationship are unknown, there is no single specific rationale for choosing any given transformation in preference to another. The aim is pragmatic, namely to find a transformation that gives:

- constant error variance,
- approximately normal errors,
- additivity,
- a linear relationship between the response variables and the explanatory variables, or
- straightforward scientific interpretation.

The choice is bound to be a compromise, and as such, is best resolved by quantitative comparison of the deviance produced under different model forms. Again, in testing for non-linearity in the relationship between y and x we might add a term in x^2 to the model; a significant parameter in the x^2 term indicates curvilinearity in the relationship between y and x.