Chapter 6

Linear Statistical Models

Linear models form the core of classical statistics and are still the basis of much of statistical practice; many modern modelling and analytical techniques build on the methodology developed for linear models.

In S most modelling exercises are conducted in a fairly standard way. The dataset is usually held in a single *data frame* object. A primary model is fitted using a *model fitting function*, for which a *formula* specifying the form of the model and the data frame specifying the variables to be used are the basic arguments. The resulting *fitted model object* can be interrogated, analysed and even modified in various ways using generic functions. The important point to note is that the fitted model object carries with it the information that the fitting process has revealed.

Although most modelling exercises conform to this rough paradigm some features of linear models are special. The *formula* for a linear model specifies the response variable and the explanatory variables (or factors) used to model the mean response by a version of the Wilkinson–Rogers notation (Wilkinson and Rogers, 1973) for specifying models that we discuss in Section 6.2.

We begin with an example to give a feel for the process and to present some of the details.

6.1 An Analysis of Covariance Example

The data frame `whiteside` contains a dataset collected in the 1960s by Mr Derek Whiteside of the UK Building Research Station and reported in the collection of small datasets edited by Hand *et al*. (1994, No. 88, p. 69). Whiteside recorded the weekly gas consumption and average external temperature at his own house in south-east England during two 'heating seasons'[1] one before and one after cavity-wall insulation was installed. The object of the exercise was to assess the effect of the insulation on gas consumption.

The variables in data frame `whiteside` are `Insul`, a factor with levels `Before` and `After`, `Temp`, for the weekly average external temperature in degrees Celsius and `Gas`, the weekly gas consumption in 1 000 cubic feet units. We begin by plotting the data in two panels showing separate least-squares lines.

[1] We are grateful to Dr Kevin McConway for clarification.

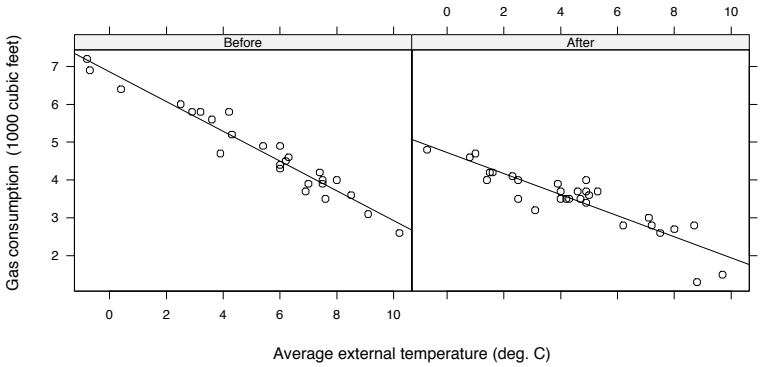

Figure 6.1: Whiteside's data showing the effect of insulation on household gas consumption.

```
xyplot(Gas ~ Temp | Insul, whiteside, panel =
  function(x, y, ...) {
    panel.xyplot(x, y, ...)
    panel.lmline(x, y, ...)
  }, xlab = "Average external temperature (deg. C)",
  ylab = "Gas consumption  (1000 cubic feet)", aspect = "xy",
  strip = function(...) strip.default(..., style = 1))
```

The result is shown in Figure 6.1. Within the range of temperatures given a straight line model appears to be adequate. The plot shows that insulation reduces the gas consumption for equal external temperatures, but it also appears to affect the slope, that is, the rate at which gas consumption increases as external temperature falls.

To explore these issues quantitatively we will need to fit linear models, the primary function for which is `lm`. The main arguments to `lm` are

```
lm(formula, data, weights, subset, na.action)
```

where

`formula` is the model formula (the only required argument),

`data` is an optional data frame,

`weights` is a vector of positive weights, if non-uniform weights are needed,

`subset` is an index vector specifying a subset of the data to be used (by default all items are used),

`na.action` is a function specifying how missing values are to be handled (by default, missing values are not allowed in S-PLUS but cause cases to be omitted in R.).

R

If the argument `data` is specified, it gives a data frame from which variables are selected ahead of the search path. Working with data frames and using this argument is strongly recommended.

It should be noted that setting `na.action = na.omit` will allow models to be fitted omitting cases that have missing components on a required variable. If any cases are omitted the fitted values and residual vector will no longer match the original observation vector in length; use `na.action = na.exclude` if the fitted values (and so on) should include `NA`s.

Formulae have been discussed in outline in Section 3.7 on page 56. For `lm` the right-hand side specifies the explanatory variables. Operators on the right-hand side of linear model formulae have the special meaning of the Wilkinson–Rogers notation and not their arithmetical meaning.

To fit the separate regressions of gas consumption on temperature as shown in Figure 6.1 we may use

```
gasB <- lm(Gas ~ Temp, data = whiteside, subset = Insul=="Before")
gasA <- update(gasB, subset = Insul=="After")
```

The first line fits a simple linear regression for the 'before' temperatures. The right-hand side of the formula needs only to specify the variable `Temp` since an intercept term (corresponding to a column of unities of the model matrix) is always implicitly included. It may be explicitly included using `1 + Temp`, where the `+` operator implies *inclusion* of a term in the model, not addition.

The function `update` is a convenient way to modify a fitted model. Its first argument is a fitted model object that results from one of the model-fitting functions such as `lm`. The remaining arguments of `update` specify the desired changes to arguments of the call that generated the object. In this case we simply wish to switch subsets from `Insul=="Before"` to `Insul=="After"`; the formula and data frame remain the same. Notice that variables used in the `subset` argument may also come from the data frame and need not be visible on the (global) search path.

Fitted model objects have an appropriate *class*, in this case `"lm"`. Generic functions to perform further operations on the object include

`print` for a simple display,

`summary` for a conventional regression analysis output,

`coef` (or `coefficients`) for extracting the regression coefficients,

`resid` (or `residuals`) for residuals,

`fitted` (or `fitted.values`) for fitted values,

`deviance` for the residual sum of squares,

`anova` for a sequential analysis of variance table, or a comparison of several hierarchical models,

`predict` for predicting means for new data, optionally with standard errors, and

`plot` for diagnostic plots.

Many of these method functions are very simple, merely extracting a component of the fitted model object. The only component likely to be accessed for which no extractor function is supplied[2] is `df.residual`, the residual degrees of freedom.

[2] In S-PLUS: there is a `df.residual` function in R.

The output from `summary` is self-explanatory. Edited results for our fitted models are

```
> summary(gasB)
    ....
Coefficients:
             Value Std. Error t value Pr(>|t|)
(Intercept)  6.854     0.118    57.876   0.000
       Temp -0.393     0.020   -20.078   0.000

Residual standard error: 0.281 on 24 degrees of freedom

> summary(gasA)
    ....
Coefficients:
             Value Std. Error t value Pr(>|t|)
(Intercept)  4.724     0.130    36.410   0.000
       Temp -0.278     0.025   -11.036   0.000

Residual standard error: 0.355 on 28 degrees of freedom
```

The difference in residual variances is relatively small, but the formal textbook F-test for equality of variances could easily be done. The sample variances could be extracted in at least two ways, for example

```
varB <- deviance(gasB)/gasB$df.resid    # direct calculation
varB <- summary(gasB)$sigma^2           # alternative
```

It is known this F-test is highly non-robust to non-normality (see, for example, Hampel *et al*. (1986, pp. 55, 188)) so its usefulness here would be doubtful.

To fit both regression models in the same `"lm"` model object we may use

```
> gasBA <- lm(Gas ~ Insul/Temp - 1, data = whiteside)
> summary(gasBA)
    ....
Coefficients:
                  Value Std. Error t value Pr(>|t|)
    InsulBefore   6.854     0.136    50.409   0.000
     InsulAfter   4.724     0.118    40.000   0.000
InsulBeforeTemp  -0.393     0.022   -17.487   0.000
 InsulAfterTemp  -0.278     0.023   -12.124   0.000

Residual standard error: 0.323 on 52 degrees of freedom
    ....
```

Notice that the estimates are the same but the standard errors are different because they are now based on the pooled estimate of variance.

Terms of the form `a/x`, where `a` is a factor, are best thought of as "separate regression models of type `1 + x` within the levels of `a`." In this case an intercept is not needed, since it is replaced by two separate intercepts for the two levels of insulation, and the formula term `- 1` removes it.

We can check for curvature in the mean function by fitting separate quadratic rather than linear regressions in the two groups. This may be done as

```
> gasQ <- lm(Gas ~ Insul/(Temp + I(Temp^2)) - 1, data = whiteside)
> summary(gasQ)$coef
                          Value Std. Error  t value   Pr(>|t|)
         InsulBefore  6.7592152  0.1507868  44.8263 0.0000e+00
          InsulAfter  4.4963739  0.1606679  27.9855 0.0000e+00
     InsulBeforeTemp -0.3176587  0.0629652  -5.0450 6.3623e-06
      InsulAfterTemp -0.1379016  0.0730580  -1.8876 6.4896e-02
InsulBeforeI(Temp^2) -0.0084726  0.0066247  -1.2789 2.0683e-01
 InsulAfterI(Temp^2) -0.0149795  0.0074471  -2.0114 4.9684e-02
```

The 'identity' function `I(...)` is used in this context and with `data.frame` (see page 18). It evaluates its argument with operators having their arithmetical meaning and returns the result. Hence it allows arithmetical operators to be used in linear model formulae, although if any function call is used in a formula their arguments are evaluated in this way.

The separate regression coefficients show that a second-degree term is possibly needed for the `After` group only, but the evidence is not overwhelming.[3] We retain the separate linear regressions model on the grounds of simplicity.

An even simpler model that might be considered is one with parallel regressions. We can fit this model and test it within the separate regression model using

```
> # R: options(contrasts = c("contr.helmert", "contr.poly"))
> gasPR <- lm(Gas ~ Insul + Temp, data = whiteside)
> anova(gasPR, gasBA)
Analysis of Variance Table
    ....
          Terms Resid. Df    RSS    Test Df Sum of Sq F Value
1   Insul + Temp        53 6.7704
2 Insul/Temp - 1        52 5.4252 1 vs. 2  1    1.3451  12.893

       Pr(F)
1
2 0.00073069
```

When `anova` is used with two or more nested models it gives an analysis of variance table for those models. In this case it shows that separate slopes are indeed necessary. Note the unusual layout of the analysis of variance table. Here we could conduct this test in a simpler and more informative way. We now fit the model with separate slopes using a different parametrization:

```
> options(contrasts = c("contr.treatment", "contr.poly"))
> gasBA1 <- lm(Gas ~ Insul*Temp, data = whiteside)
> summary(gasBA1)$coef
              Value Std. Error  t value   Pr(>|t|)
(Intercept)  6.85383   0.135964  50.4091 0.0000e+00
      Insul -2.12998   0.180092 -11.8272 2.2204e-16
       Temp -0.39324   0.022487 -17.4874 0.0000e+00
 Insul:Temp  0.11530   0.032112   3.5907 7.3069e-04
```

[3] Notice that when the quadratic terms are present first-degree coefficients mean 'the slope of the curve at temperature zero', so a non-significant value does not mean that the linear term is not needed. Removing the non-significant linear term for the 'after' group, for example, would be unjustified.

The call to `options` is explained more fully in Section 6.2; for now we note that it affects the way regression models are parametrized when factors are used. The formula `Insul*Temp` expands to `1 + Insul + Temp + Insul:Temp` and the corresponding coefficients are, in order, the intercept for the 'before' group, the *difference* in intercepts, the slope for the 'before' group and the *difference* in slopes. Since this last term is significant we conclude that the two separate slopes are required in the model. Indeed note that the F-statistic in the analysis of variance table is the square of the final t-statistic and that the tail areas are identical.

6.2 Model Formulae and Model Matrices

This section contains some rather technical material and might be skimmed at first reading.

A linear model is specified by the response vector $\boldsymbol{y}$ and by the matrix of explanatory variables, or *model matrix*, X. The model formula conveys both pieces of information, the left-hand side providing the response and the right-hand side instructions on how to generate the model matrix according to a particular convention.

A multiple regression with three quantitative determining variables might be specified as `y ~ x1 + x2 + x3`. This would correspond to a model with a familiar algebraic specification

$$y_i = \beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \beta_3 x_{i3} + \epsilon_i, \qquad i = 1, 2, \ldots, n$$

The model matrix has the partitioned form

$$X = \begin{bmatrix} \mathbf{1} & \boldsymbol{x}_1 & \boldsymbol{x}_2 & \boldsymbol{x}_3 \end{bmatrix}$$

The intercept term (β_0 corresponding to the leading column of ones in X) is implicitly present; its presence may be confirmed by giving a formula such as `y ~ 1 + x1 + x2 + x3`, but wherever the `1` occurs in the formula the column of ones will always be the first column of the model matrix. It may be omitted and a regression through the origin fitted by giving a `- 1` term in the formula, as in `y ~ x1 + x2 + x3 - 1`.

Factor terms in a model formula are used to specify classifications leading to what are often called analysis of variance models. Suppose `a` is a factor. An analysis of variance model for the one-way layout defined by `a` might be written in the algebraic form

$$y_{ij} = \mu + \alpha_j + \epsilon_{ij} \qquad i = 1, 2, \ldots, n_j; \quad j = 1, 2, \ldots, k$$

where there are k classes and the n_j is the size of the jth. Let $n = \sum_j n_j$. This specification is over-parametrized, but we could write the model matrix in the form

$$X = \begin{bmatrix} \mathbf{1} & X_{\mathtt{a}} \end{bmatrix}$$

where $X_{\mathtt{a}}$ is an $n \times k$ binary incidence (or 'dummy variable') matrix where each row has a single unity in the column of the class to which it belongs.

The redundancy comes from the fact that the columns of $X_{\mathtt{a}}$ add to $\mathbf{1}$, making X of rank k rather than $k+1$. One way to resolve the redundancy is to remove the column of ones. This amounts to setting $\mu = 0$, leading to an algebraic specification of the form

$$y_{ij} = \alpha_j + \epsilon_{ij} \qquad i = 1, 2, \ldots, n_j;\ j = 1, 2, \ldots, k$$

so the α_j parameters are the class means. This formulation may be specified by `y ~ a - 1`.

If we do not break the redundancy by removing the intercept term it must be done some other way, since otherwise the parameters are not identifiable. The way this is done in S is most easily described in terms of the model matrix. The model matrix generated has the form

$$X^\star = \left[\mathbf{1}\ X_{\mathtt{a}} C_{\mathtt{a}}\right]$$

where $C_{\mathtt{a}}$, the *contrast matrix* for `a`, is a $k \times (k-1)$ matrix chosen so that $X^\star$ has rank k, the number of columns. A necessary (and usually sufficient) condition for this to be the case is that the square matrix $[\mathbf{1}\, C_{\mathtt{a}}]$ be non-singular.

The reduced model matrix $X^\star$ in turn defines a linear model, but the parameters are often not directly interpretable and an algebraic formulation of the precise model may be difficult to write down. Nevertheless, the relationship between the newly defined and original (redundant) parameters is clearly given by

$$\boldsymbol{\alpha} = C_{\mathtt{a}} \boldsymbol{\alpha}^\star \tag{6.1}$$

where $\boldsymbol{\alpha}$ are the original α parameters and $\boldsymbol{\alpha}^\star$ are the new.

If $\boldsymbol{c}_{\mathtt{a}}$ is a non-zero vector such that $\boldsymbol{c}_{\mathtt{a}}^T C_{\mathtt{a}} = \mathbf{0}$ it can be seen immediately that using $\boldsymbol{\alpha}^\star$ as parameters amounts to estimating the original parameters, $\boldsymbol{\alpha}$ subject to the *identification constraint* $\boldsymbol{c}_{\mathtt{a}}^T \boldsymbol{\alpha} = 0$ which is usually sufficient to make them unique. Such a vector (or matrix) $\boldsymbol{c}_{\mathtt{a}}$ is called an *annihilator* of $C_{\mathtt{a}}$ or a basis for the orthogonal complement of the range of $C_{\mathtt{a}}$.

If we fit the one-way layout model using the formula

```
y ~ a
```

the coefficients we obtain will be estimates of μ and $\boldsymbol{\alpha}^\star$. The corresponding constrained estimates of the $\boldsymbol{\alpha}$ may be obtained by multiplying by the contrasts matrix or by using the function `dummy.coef`. Consider an artificial example:

```
> dat <- data.frame(a = factor(rep(1:3, 3)),
                    y = rnorm(9, rep(2:4, 3), 0.1))
> obj <- lm(y ~ a, dat)
> (alf.star <- coef(obj))
 (Intercept)      a1      a2
      2.9719 0.51452 0.49808
> Ca <- contrasts(dat$a)      # contrast matrix for 'a'
> drop(Ca %*% alf.star[-1])
       1        2       3
 -1.0126 0.016443 0.99615
```

```
> dummy.coef(obj)
$"(Intercept)":
[1] 2.9719

$a:
       1        2       3
 -1.0126 0.016443 0.99615
```

Notice that the estimates of $\boldsymbol{\alpha}$ sum to zero because the contrast matrix used here implies the identification constraint $\mathbf{1}^T\boldsymbol{\alpha} = 0$.

Contrast matrices

S+ By default S-PLUS uses so-called *Helmert* contrast matrices for unordered factors and orthogonal polynomial contrast matrices for ordered factors. The forms of these can be deduced from the following artificial example:

```
> N <- factor(Nlevs <- c(0,1,2,4))
> contrasts(N)
  [,1] [,2] [,3]
0   -1   -1   -1
1    1   -1   -1
2    0    2   -1
4    0    0    3
> contrasts(ordered(N))
        .L   .Q       .C
0 -0.67082  0.5 -0.22361
1 -0.22361 -0.5  0.67082
2  0.22361 -0.5 -0.67082
4  0.67082  0.5  0.22361
```

For the `poly` contrasts it can be seen that the corresponding parameters $\boldsymbol{\alpha}^\star$ can be interpreted as the coefficients in an orthogonal polynomial model of degree $r-1$, *provided* the ordered levels are equally spaced (which is not the case for the example) *and* the class sizes are equal. The $\boldsymbol{\alpha}^\star$ parameters corresponding to the Helmert contrasts also have an easy interpretation, as we see in the following. Since both the Helmert and polynomial contrast matrices satisfy $\mathbf{1}^T C = \mathbf{0}$, the implied constraint on $\boldsymbol{\alpha}$ will be $\mathbf{1}^T\boldsymbol{\alpha} = 0$ in both cases.

The default contrast matrices can be changed by resetting the `contrasts` option. This is a character vector of length two giving the names of the functions that generate the contrast matrices for unordered and ordered factors respectively. For example,

```
options(contrasts = c("contr.treatment", "contr.poly"))
```

sets the default contrast matrix function for factors to `contr.treatment` and for
R ordered factors to `contr.poly` (the original default). (This *is* the default in R.) Four supplied contrast functions are as follows:

`contr.helmert` for the Helmert contrasts.

`contr.treatment` for contrasts such that each coefficient represents a comparison of that level with level 1 (omitting level 1 itself). This corresponds to the constraint $\alpha_1 = 0$. Note that in this parametrization the coefficients are *not* contrasts in the usual sense.

`contr.sum` where the coefficients are constrained to add to zero; that is, in this case the components of $\boldsymbol{\alpha}^\star$ are the same as the first $r-1$ components of $\boldsymbol{\alpha}$, with the latter constrained to add to zero.

`contr.poly` for the equally spaced, equally replicated orthogonal polynomial contrasts.

Others can be written using these as templates (as we do with our function `contr.sdif`, used on pages 293 and 294). We recommend the use of the treatment contrasts for unbalanced layouts, including generalized linear models and survival models, because the unconstrained coefficients obtained directly from the fit are then easy to interpret.

Notice that the `helmert`, `sum` and `poly` contrasts ensure the rank condition on C is met by choosing C so that the columns of $[\,\mathbf{1}\,C\,]$ are mutually orthogonal, whereas the `treatment` contrasts choose C so that $[\,\mathbf{1}\,C\,]$ is in echelon form.

Contrast matrices for particular factors may also be set as an attribute of the factor itself. This can be done either by the `contrasts` replacement function or by using the function `C` which takes three arguments: the factor, the matrix from which contrasts are to be taken (or the abbreviated name of a function that will generate such a matrix) and the number of contrasts. On some occasions a p-level factor may be given a contrast matrix with fewer than $p-1$ columns, in which case it contributes fewer than $p-1$ degrees of freedom to the model, or the unreduced parameters $\boldsymbol{\alpha}$ have additional constraints placed on them apart from the one needed for identification. An alternative method is to use the replacement form with a specific number of contrasts as the second argument. For example, suppose we wish to create a factor `N2` that would generate orthogonal linear and quadratic polynomial terms, only. Two equivalent ways of doing this would be

```
> N2 <- N
> contrasts(N2, 2) <- poly(Nlevs, 2)
> N2 <- C(N, poly(Nlevs, 2), 2)       # alternative
> contrasts(N2)
          1        2
0 -0.591608  0.56408
1 -0.253546 -0.32233
2  0.084515 -0.64466
4  0.760639  0.40291
```

In this case the constraints imposed on the $\boldsymbol{\alpha}$ parameters are not merely for identification but actually change the model subspace.

Parameter interpretation

The `poly` contrast matrices lead to $\boldsymbol{\alpha}^\star$ parameters that are sometimes interpretable as coefficients in an orthogonal polynomial regression. The `treatment`

contrasts set $\alpha_1 = 0$ and choose the remaining αs as the $\alpha^\star$s. Other cases are often not so direct, but an interpretation is possible.

To interpret the $\boldsymbol{\alpha}^\star$ parameters in general, consider the relationship (6.1). Since the contrast matrix C is of full column rank it has a unique left inverse C^+, so we can reverse this relationship to give

$$\boldsymbol{\alpha}^\star = C^+\boldsymbol{\alpha} \quad \text{where} \quad C^+ = (C^TC)^{-1}C^T \tag{6.2}$$

The pattern in the matrix C^+ then provides an interpretation of each unconstrained parameter as a linear function of the (usually) readily appreciated constrained parameters. For example, consider the Helmert contrasts for $r = 4$. To exhibit the pattern in C^+ more clearly we use the function `fractions` from `MASS` for rational approximation and display.

```
> fractions(ginv(contr.helmert(n = 4)))
      [,1]  [,2]  [,3]  [,4]
[1,]  -1/2   1/2     0     0
[2,]  -1/6  -1/6   1/3     0
[3,] -1/12 -1/12 -1/12   1/4
```

Hence $\alpha_1^\star = \frac{1}{2}(\alpha_2 - \alpha_1)$, $\alpha_2^\star = \frac{1}{3}\{\alpha_3 - \frac{1}{2}(\alpha_1 + \alpha_2)\}$ and in general $\alpha_j^\star$ is a comparison of α_{j+1} with the average of all preceding αs, divided by $j + 1$. This is a comparison of the (unweighted) mean of class $j + 1$ with that of the preceding classes.

It can sometimes be important to use contrast matrices that give a simple interpretation to the fitted coefficients. This can be done by noting that $(C^+)^+ = C$. For example, suppose we wished to choose contrasts so that the $\alpha_j^\star = \alpha_{j+1} - \alpha_j$, that is, the successive differences of class effects. For $r = 5$, say, the C^+ matrix is then given by

```
> Cp <- diag(-1, 4, 5);  Cp[row(Cp) == col(Cp) - 1] <- 1
> Cp
     [,1] [,2] [,3] [,4] [,5]
[1,]   -1    1    0    0    0
[2,]    0   -1    1    0    0
[3,]    0    0   -1    1    0
[4,]    0    0    0   -1    1
```

Hence the contrast matrix to obtain these linear functions as the estimated coefficients is

```
> fractions(ginv(Cp))
     [,1] [,2] [,3] [,4]
[1,] -4/5 -3/5 -2/5 -1/5
[2,]  1/5 -3/5 -2/5 -1/5
[3,]  1/5  2/5 -2/5 -1/5
[4,]  1/5  2/5  3/5 -1/5
[5,]  1/5  2/5  3/5  4/5
```

Note that again the columns have zero sums, so the implied constraint is that the effects add to zero. (If it were not obvious we could find the induced constraint

using our function `Null` (page 63) to find a basis for the null space of the contrast matrix.)

The pattern is obvious from this example and a contrast matrix function for the general case can now be written. To be usable as a component of the `contrasts` option such a function has to conform with a fairly strict convention, but the key computational steps are

```
....
    contr <- col(matrix(nrow = n, ncol = n - 1))
    upper.tri <- !lower.tri(contr)
    contr[upper.tri] <- contr[upper.tri] - n
    contr/n
....
```

The complete function is supplied as `contr.sdif` in `MASS`. We make use of it on pages 293 and 294.

Higher-way layouts

Two- and higher-way layouts may be specified by two or more factors and formula operators. The way the model matrix is generated is then an extension of the conventions for a one-way layout.

If `a` and `b` are r- and s-level factors, respectively, the model formula `y ~ a+b` specifies an additive model for the two-way layout. Using the redundant specification the algebraic formulation would be

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

and the model matrix would be

$$X = \begin{bmatrix}\mathbf{1} & X_{\mathtt{a}} & X_{\mathtt{b}}\end{bmatrix}$$

The reduced model matrix then has the form

$$X^\star = \begin{bmatrix}\mathbf{1} & X_{\mathtt{a}}C_{\mathtt{a}} & X_{\mathtt{b}}C_{\mathtt{b}}\end{bmatrix}$$

However, if the intercept term is explicitly removed using, say, `y ~ a + b - 1`, the reduced form is

$$X^\star = \begin{bmatrix}X_{\mathtt{a}} & X_{\mathtt{b}}C_{\mathtt{b}}\end{bmatrix}$$

Note that this is asymmetric in `a` and `b` and order-dependent.

A two-way non-additive model has a redundant specification of the form

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

The model matrix can be written as

$$X = \begin{bmatrix}\mathbf{1} & X_{\mathtt{a}} & X_{\mathtt{b}} & X_{\mathtt{a}}{:}X_{\mathtt{b}}\end{bmatrix}$$

where we use the notation $A{:}B$ to denote the matrix obtained by taking each column of A and multiplying it element-wise by each column of B. In the example $X_{\mathtt{a}}{:}X_{\mathtt{b}}$ generates an incidence matrix for the sub-classes defined jointly by `a` and `b`. Such a model may be specified by the formula

```
y ~ a + b + a:b
```

or equivalently by `y ~ a*b`. The reduced form of the model matrix is then

$$X^\star = \left[\mathbf{1}\ X_{\mathtt{a}}C_{\mathtt{a}}\ X_{\mathtt{b}}C_{\mathtt{b}}\ (X_{\mathtt{a}}C_{\mathtt{a}}){:}(X_{\mathtt{b}}C_{\mathtt{b}})\right]$$

It may be seen that $(X_{\mathtt{a}}C_{\mathtt{a}}){:}(X_{\mathtt{b}}C_{\mathtt{b}}) = (X_{\mathtt{a}}{:}X_{\mathtt{b}})(C_{\mathtt{b}} \otimes C_{\mathtt{a}})$, where $\otimes$ denotes the Kronecker product, so the relationship between the γ parameters and the corresponding $\gamma^\star$s is

$$\boldsymbol{\gamma} = (C_{\mathtt{b}} \otimes C_{\mathtt{a}})\boldsymbol{\gamma}^\star$$

The identification constraints can be most easily specified by writing $\boldsymbol{\gamma}$ as an $r \times s$ matrix. If this is done, the relationship has the form $\boldsymbol{\gamma} = C_{\mathtt{a}}\boldsymbol{\gamma}^\star C_{\mathtt{b}}^T$ and the constraints have the form $\boldsymbol{c}_{\mathtt{a}}^T\boldsymbol{\gamma} = \mathbf{0}^T$ and $\boldsymbol{\gamma}\boldsymbol{c}_{\mathtt{b}} = \mathbf{0}$, separately.

If the intercept term is removed, however, such as by using `y ~ a*b - 1`, the form is different, namely,

$$X^\star = \left[X_{\mathtt{a}}\ X_{\mathtt{b}}C_{\mathtt{b}}\ X_{\mathtt{a}}^{(-r)}{:}(X_{\mathtt{b}}C_{\mathtt{b}})\right]$$

where (somewhat confusingly) $X_{\mathtt{a}}^{(-r)}$ is a matrix obtained by removing the *last* column of $X_{\mathtt{a}}$. Furthermore, if a model is specified as `y ~ - 1 + a + a:b` the model matrix generated is $\left[X_{\mathtt{a}}\ X_{\mathtt{a}}{:}(X_{\mathtt{b}}C_{\mathtt{b}})\right]$. In general addition of a term `a:b` extends the previously constructed design matrix to a complete non-additive model in some non-redundant way (unless the design is deficient, of course).

Even though `a*b` expands to `a + b + a:b`, it should be noted that `a + a:b` is not always the same[4] as `a*b - b` or even `a + b - b + a:b`. When used in model-fitting functions the last two formulae construct the design matrix for `a*b` and only then remove any columns corresponding to the `b` term. (The result is not a statistically meaningful model.) Model matrices are constructed within the fitting functions by arranging the positive terms in order of their complexity, sequentially adding columns to the model matrix according to the redundancy resolution rules and then removing any generated columns corresponding to negative terms. The exception to this rule is the intercept term which is always removed initially. (With `update`, however, the formula is expanded and all negative terms are removed *before* the model matrix is constructed.)

The model `a + a:b` generates the same matrix as `a/b`, which expands in S-PLUS to `a + b %in% a`. There is no compelling reason for the additional operator,[5] `%in%`, but it does serve to emphasize that the slash operator should be thought of as specifying separate submodels of the form `1 + b` for each level of `a`. The operator behaves like the colon formula operator when the second main effect term is not given, but is conventionally reserved for nested models.

Star products of more than two terms, such as `a*b*c`, may be thought of as expanding `(1 + a):(1 + b):(1 + c)` according to ordinary algebraic rules and may be used to define higher-way non-additive layouts. There is also a power operator, `^`, for generating models up to a specified degree of interaction term.

[4] In S-PLUS: it is always the same in R.

[5] Which R does not have.

For example, `(a+b+c)^3` generates the same model as `a*b*c` but `(a+b+c)^2` has the highest order interaction absent.

Combinations of factors and non-factors with formula operators are useful in an obvious way. We have seen already that `a/x - 1` generates separate simple linear regressions on `x` within the levels of `a`. The same model may be specified as `a + a:x - 1`, whereas `a*x` generates an equivalent model using a different resolution of the redundancy. It should be noted that `(x + y + z)^3` does *not* generate a general third-degree polynomial regression in the three variables, as might be expected. This is because terms of the form `x:x` are regarded as the same as `x`, not as `I(x^2)`. However, in S-PLUS a single term such as `x^2` is silently promoted to `I(x^2)` and interpreted as a power. S+

6.3 Regression Diagnostics

The message in the Whiteside example is relatively easy to discover and we did not have to work hard to find an adequate linear model. There is an extensive literature (for example Atkinson, 1985) on examining the fit of linear models to consider whether one or more points are not fitted as well as they should be or have undue influence on the fitting of the model. This can be contrasted with the robust regression methods we discuss in Section 6.5, which automatically take account of anomalous points.

The basic tool for examining the fit is the residuals, and we have already looked for patterns in residuals and assessed the normality of their distribution. The residuals are not independent (they sum to zero if an intercept is present) and they do not have the same variance. Indeed, their variance-covariance matrix is

$$\operatorname{var}(e) = \sigma^2[I - H] \tag{6.3}$$

where $H = X(X^TX)^{-1}X^T$ is the orthogonal projector matrix onto the model space, or *hat* matrix. If a diagonal entry h_{ii} of H is large, changing y_i will move the fitted surface appreciably towards the altered value. For this reason h_{ii} is said to measure the *leverage* of the observation y_i. The trace of H is p, the dimension of the model space, so 'large' is taken to be greater than two or three times the average, p/n.

Having large leverage has two consequences for the corresponding residual. First, its variance will be lower than average from (6.3). We can compensate for this by rescaling the residuals to have unit variance. The *standardized residuals* are

$$e'_i = \frac{e_i}{s\sqrt{1 - h_{ii}}}$$

where as usual we have estimated σ^2 by s^2, the residual mean square. Second, if one error is very large, the variance estimate s^2 will be too large, and this deflates all the standardized residuals. Let us consider fitting the model omitting observation i. We then get a prediction for the omitted observation, $\hat{y}_{(i)}$, and an

estimate of the error variance, $s^2_{(i)}$, from the reduced sample. The *studentized residuals* are

$$e^*_i = \frac{y_i - \hat{y}_{(i)}}{\sqrt{\mathrm{var}\big(y_i - \hat{y}_{(i)}\big)}}$$

but with σ replaced by $s_{(i)}$. Fortunately, it is not necessary to re-fit the model each time an observation is omitted, since it can be shown that

$$e^*_i = e'_i \Big/ \left[\frac{n-p-e'^2_i}{n-p-1}\right]^{1/2}$$

Notice that this implies that the standardized residuals, e_i, must be bounded by $\pm\sqrt{n-p}$.

The terminology used here is not universally adopted; in particular studentized residuals are sometimes called *jackknifed* residuals.

It is usually better to compare studentized residuals rather than residuals; in particular we recommend that they be used for normal probability plots.

We have provided functions `studres` and `stdres` to compute studentized and standardized residuals. There is a function `hat`, but this expects the model matrix as its argument. (There is a useful function, `lm.influence`, for most of the fundamental calculations. The diagonal of the hat matrix can be obtained by `lm.influence(lmobject)$hat`.)

Scottish hill races

As an example of regression diagnostics, let us return to the data on 35 Scottish hill races in our data frame `hills` considered in Chapter 1. The data come from Atkinson (1986) and are discussed further in Atkinson (1988) and Staudte and Sheather (1990). The columns are the overall race distance, the total height climbed and the record time. In Chapter 1 we considered a regression of `time` on `dist`. We can now include `climb`:

```
> (hills.lm <- lm(time ~ dist + climb, data = hills))
Coefficients:
 (Intercept)  dist    climb
      -8.992 6.218 0.011048

Degrees of freedom: 35 total; 32 residual
Residual standard error: 14.676
> frame(); par(fig = c(0, 0.6, 0, 0.55))
> plot(fitted(hills.lm), studres(hills.lm))
> abline(h = 0, lty = 2)
> identify(fitted(hills.lm), studres(hills.lm),
           row.names(hills))
> par(fig = c(0.6, 1, 0, 0.55), pty = "s")
> qqnorm(studres(hills.lm))
> qqline(studres(hills.lm))
> hills.hat <- lm.influence(hills.lm)$hat
> cbind(hills, lev = hills.hat)[hills.hat > 3/35, ]
```

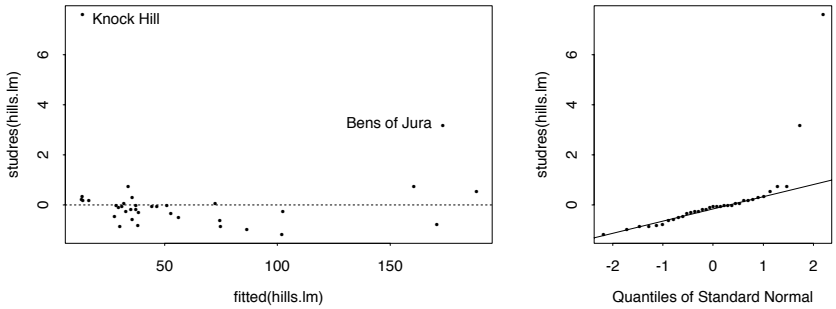

Figure 6.2: Diagnostic plots for Scottish hills data, unweighted model.

```
              dist climb    time     lev
 Bens of Jura   16  7500 204.617 0.42043
  Lairig Ghru   28  2100 192.667 0.68982
    Ben Nevis   10  4400  85.583 0.12158
Two Breweries   18  5200 170.250 0.17158
 Moffat Chase   20  5000 159.833 0.19099
```

so two points have very high leverage, two points have large residuals, and `Bens of Jura` is in both sets. (See Figure 6.2.)

If we look at `Knock Hill` we see that the prediction is over an hour less than the reported record:

```
> cbind(hills, pred = predict(hills.lm))["Knock Hill", ]
           dist climb  time   pred
Knock Hill    3   350 78.65 13.529
```

and Atkinson (1988) suggests that the record is one hour out. We drop this observation to be safe:

```
> (hills1.lm <- update(hills.lm, subset = -18))
Coefficients:
 (Intercept)   dist    climb
      -13.53 6.3646 0.011855

Degrees of freedom: 34 total; 31 residual
Residual standard error: 8.8035
```

Since `Knock Hill` did not have a high leverage, deleting it did not change the fitted model greatly. On the other hand, `Bens of Jura` had both a high leverage and a large residual and so does affect the fit:

```
> update(hills.lm, subset = -c(7, 18))
Coefficients:
 (Intercept)   dist     climb
    -10.362 6.6921 0.0080468

Degrees of freedom: 33 total; 30 residual
Residual standard error: 6.0538
```

If we consider this example carefully we find a number of unsatisfactory features. First, the prediction is negative for short races. Extrapolation is often unsafe, but on physical grounds we would expect the model to be a good fit with a zero intercept; indeed hill-walkers use a prediction of this sort (3 miles/hour plus 20 minutes per 1 000 feet). We can see from the summary that the intercept is significantly negative:

```
> summary(hills1.lm)
    ....
Coefficients:
             Value Std. Error t value Pr(>|t|)
(Intercept) -13.530   2.649     -5.108   0.000
       dist   6.365   0.361     17.624   0.000
      climb   0.012   0.001      9.600   0.000
    ....
```

Furthermore, we would not expect the predictions of times that range from 15 minutes to over 3 hours to be equally accurate, but rather that the accuracy be roughly proportional to the time. This suggests a log transform, but that would be hard to interpret. Rather we weight the fit using distance as a surrogate for time. We want weights inversely proportional to the variance:

```
> summary(update(hills1.lm, weights = 1/dist^2))
    ....
Coefficients:
             Value Std. Error t value Pr(>|t|)
(Intercept)  -5.809   2.034     -2.855   0.008
       dist   5.821   0.536     10.858   0.000
      climb   0.009   0.002      5.873   0.000

Residual standard error: 1.16 on 31 degrees of freedom
```

The intercept is still significantly non-zero. If we are prepared to set it to zero on physical grounds, we can achieve the same effect by dividing the prediction equation by distance, and regressing inverse speed (time/distance) on gradient (climb/distance):

```
> lm(time ~ -1 + dist + climb, hills[-18, ], weights = 1/dist^2)
Coefficients:
 dist     climb
  4.9 0.0084718

Degrees of freedom: 34 total; 32 residual
Residual standard error (on weighted scale): 1.2786
> hills <- hills   # make a local copy (needed in S-PLUS)
> hills$ispeed <- hills$time/hills$dist
> hills$grad <- hills$climb/hills$dist
> (hills2.lm <- lm(ispeed ~ grad, data = hills[-18, ]))
Coefficients:
 (Intercept)      grad
         4.9 0.0084718
```

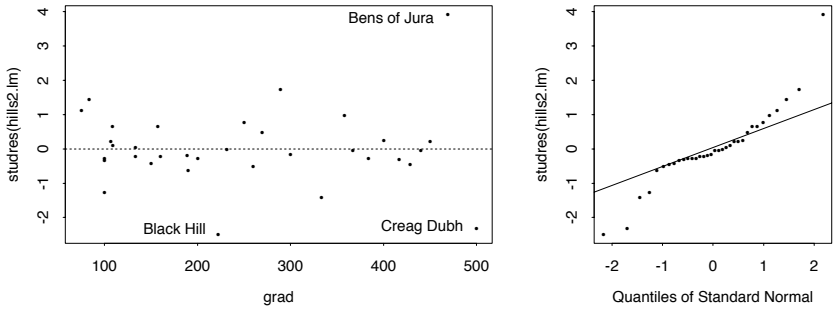

Figure 6.3: Diagnostic plots for Scottish hills data, weighted model.

```
Degrees of freedom: 34 total; 32 residual
Residual standard error: 1.2786
> frame(); par(fig = c(0, 0.6, 0, 0.55))
> plot(hills$grad[-18], studres(hills2.lm), xlab = "grad")
> abline(h = 0, lty = 2)
> identify(hills$grad[-18], studres(hills2.lm),
           row.names(hills)[-18])
> par(fig = c(0.6, 1, 0, 0.55), pty = "s")    # Figure 6.3
> qqnorm(studres(hills2.lm))
> qqline(studres(hills2.lm))
> hills2.hat <- lm.influence(hills2.lm)$hat
> cbind(hills[-18,], lev = hills2.hat)[hills2.hat > 1.8*2/34, ]
             dist climb    time  ispeed   grad     lev
Bens of Jura   16  7500 204.617 12.7886 468.75 0.11354
  Creag Dubh    4  2000  26.217  6.5542 500.00 0.13915
```

The two highest-leverage cases are now the steepest two races, and are outliers pulling in opposite directions. We could consider elaborating the model, but this would be to fit only one or two exceptional points; for most of the data we have the formula of 5 minutes/mile plus 8 minutes per 1 000 feet. We return to this example on page 162 where robust fits do support a zero intercept.

6.4 Safe Prediction

A warning is needed on the use of the `predict` method function when polynomials are used (and also splines, see Section 8.8). We illustrate this by the dataset `wtloss`, for which a more appropriate analysis is given in Chapter 8. This has a weight loss `Weight` against `Days`. Consider a quadratic polynomial regression model of `Weight` on `Days`. This may be fitted by either of

```
quad1 <- lm(Weight ~ Days + I(Days^2), data = wtloss)
quad2 <- lm(Weight ~ poly(Days, 2), data = wtloss)
```

The second uses orthogonal polynomials and is the preferred form on grounds of numerical stability.

Suppose we wished to predict future weight loss. The first step is to create a new data frame with a variable `x` containing the new values, for example,

```
new.x <- data.frame(Days = seq(250, 300, 10),
                    row.names = seq(250, 300, 10))
```

The `predict` method may now be used:

```
> predict(quad1, newdata = new.x)
    250    260    270    280    290    300
 112.51 111.47 110.58 109.83 109.21 108.74
> predict(quad2, newdata = new.x) # from S-PLUS 6.0
    250    260    270    280   290    300
 244.56 192.78 149.14 113.64 86.29 67.081
```

The first form gives correct answers but the second does not in S-PLUS!

The reason for this is as follows. The `predict` method for `lm` objects works by attaching the estimated coefficients to a new model matrix that it constructs using the formula and the new data. In the first case the procedure will work, but in the second case the columns of the model matrix are for a *different* orthogonal polynomial basis, and so the old coefficients do not apply. The same will hold for any function used to define the model that generates mathematically different
R bases for old and new data, such as spline bases using `bs` or `ns`. R retains enough information to predict from the old data.

S+ The remedy in S-PLUS is to use the method function `predict.gam`:

```
>  predict.gam(quad2, newdata = new.x) # S-PLUS only
    250    260    270    280    290    300
 112.51 111.47 110.58 109.83 109.21 108.74
```

This constructs a new model matrix by putting old and new data together, re-estimates the regression using the old data only and predicts using these estimates of regression coefficients. This can involve appreciable extra computation, but the results will be correct for polynomials, but not exactly so for splines since the knot positions will change. As a check, `predict.gam` compares the predictions with the old fitted values for the original data. If these are seriously different, a warning is issued that the process has probably failed.

In our view this is a serious flaw in `predict.lm`. It would have been better to use the safe method as the default and provide an `unsafe` argument for the faster method as an option.

6.5 Robust and Resistant Regression

There are a number of ways to perform robust regression in S-PLUS, but many have drawbacks and are not mentioned here. First consider an example. Rousseeuw and Leroy (1987) give data on annual numbers of Belgian telephone calls, given in our dataset `phones`.

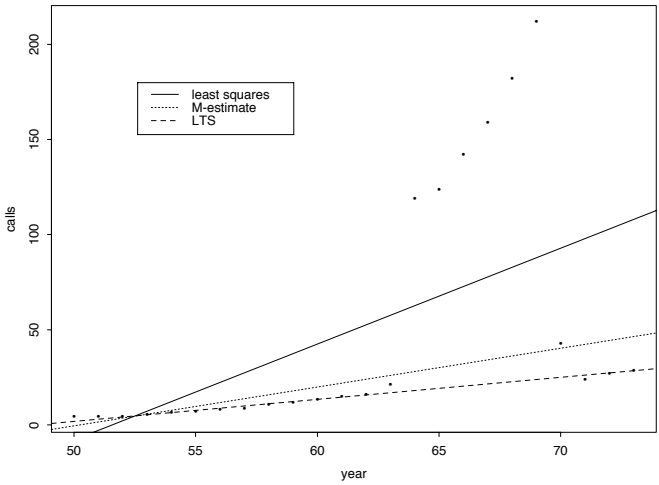

Figure 6.4: Millions of phone calls in Belgium, 1950–73, from Rousseeuw and Leroy (1987), with three fitted lines.

```
# R: library(lqs)
phones.lm <- lm(calls ~ year, data = phones)
attach(phones); plot(year, calls); detach()
abline(phones.lm$coef)
abline(rlm(calls ~ year, phones, maxit=50), lty = 2, col = 2)
abline(lqs(calls ~ year, phones), lty = 3, col = 3)
legend(locator(1), lty = 1:3, col = 1:3,
       legend = c("least squares", "M-estimate", "LTS"))
```

Figure 6.4 shows the least squares line, an M-estimated regression and the least trimmed squares regression (Section 6.5). The `lqs` line is $-56.16 + 1.16\,$`year`. Rousseeuw & Leroy's investigations showed that for 1964–9 the total length of calls (in minutes) had been recorded rather than the number, with each system being used during parts of 1963 and 1970.

Next some theory. In a regression problem there are two possible sources of errors, the observations y_i and the corresponding row vector of p regressors $\boldsymbol{x}_i$. Most robust methods in regression only consider the first, and in some cases (designed experiments?) errors in the regressors can be ignored. This is the case for M-estimators, the only ones we consider in this section.

Consider a regression problem with n cases $(y_i, \boldsymbol{x}_i)$ from the model

$$y = \boldsymbol{x}\beta + \epsilon$$

for a p-variate row vector $\boldsymbol{x}$.

M-estimators

If we assume a scaled pdf $f(e/s)/s$ for ϵ and set $\rho = -\log f$, the maximum likelihood estimator minimizes

$$\sum_{i=1}^{n} \rho\left(\frac{y_i - \boldsymbol{x}_i b}{s}\right) + n \log s \tag{6.4}$$

Suppose for now that s is known. Let $\psi = \rho'$. Then the MLE b of β solves

$$\sum_{i=1}^{n} \boldsymbol{x}_i \psi\left(\frac{y_i - \boldsymbol{x}_i b}{s}\right) = 0 \tag{6.5}$$

Let $r_i = y_i - \boldsymbol{x}_i b$ denote the residuals.

The solution to equation (6.5) or to minimizing over (6.4) can be used to define an M-estimator of β.

A common way to solve (6.5) is by iterated re-weighted least squares, with weights

$$w_i = \psi\left(\frac{y_i - \boldsymbol{x}_i b}{s}\right) \Big/ \left(\frac{y_i - \boldsymbol{x}_i b}{s}\right) \tag{6.6}$$

The iteration is guaranteed to converge only for *convex* ρ functions, and for re-descending functions (such as those of Tukey and Hampel; page 123) equation (6.5) may have multiple roots. In such cases it is usual to choose a good starting point and iterate carefully.

Of course, in practice the scale s is not known. A simple and very resistant scale estimator is the MAD about some centre. This is applied to the residuals about zero, either to the current residuals within the loop or to the residuals from a very resistant fit (see the next subsection).

Alternatively, we can estimate s in an MLE-like way. Finding a stationary point of (6.4) with respect to s gives

$$\sum_i \psi\left(\frac{y_i - \boldsymbol{x}_i b}{s}\right)\left(\frac{y_i - \boldsymbol{x}_i b}{s}\right) = n$$

which is not resistant (and is biased at the normal). As in the univariate case we modify this to

$$\sum_i \chi\left(\frac{y_i - \boldsymbol{x}_i b}{s}\right) = (n-p)\gamma \tag{6.7}$$

Our function `rlm`

Our `MASS` library section introduces a new class `rlm` and model-fitting function `rlm`, building on `lm`. The syntax in general follows `lm`. By default Huber's M-estimator is used with tuning parameter $c = 1.345$. By default the scale s is estimated by iterated MAD, but Huber's proposal 2 can also be used.

```
> summary(lm(calls ~ year, data = phones), cor = F)
               Value Std. Error  t value Pr(>|t|)
(Intercept) -260.059  102.607     -2.535    0.019
       year    5.041    1.658      3.041    0.006
Residual standard error: 56.2 on 22 degrees of freedom
> summary(rlm(calls ~ year, maxit = 50, data = phones), cor = F)
               Value Std. Error  t value
(Intercept) -102.622   26.608     -3.857
       year    2.041    0.430      4.748
Residual standard error: 9.03 on 22 degrees of freedom
> summary(rlm(calls ~ year, scale.est = "proposal 2",
              data = phones), cor = F)
Coefficients:
               Value Std. Error  t value
(Intercept) -227.925  101.874     -2.237
       year    4.453    1.646      2.705
Residual standard error: 57.3 on 22 degrees of freedom
```

As Figure 6.4 shows, in this example there is a batch of outliers from a different population in the late 1960s, and these should be rejected completely, which the Huber M-estimators do not. Let us try a re-descending estimator.

```
> summary(rlm(calls ~ year, data = phones, psi = psi.bisquare),
          cor = F)
Coefficients:
              Value Std. Error t value
(Intercept) -52.302   2.753    -18.999
       year   1.098   0.044     24.685
Residual standard error: 1.65 on 22 degrees of freedom
```

This happened to work well for the default least-squares start, but we might want to consider a better starting point, such as that given by `init = "lts"`.

Resistant regression

M-estimators are not very resistant to outliers unless they have redescending ψ functions, in which case they need a good starting point. A succession of more resistant regression estimators was defined in the 1980s. The first to become popular was

$$\min_b \operatorname*{median}_i |y_i - \boldsymbol{x}_i b|^2$$

called the *least median of squares* (LMS) estimator. The square is necessary if n is even, when the central median is taken. This fit is very resistant, and needs no scale estimate. It is, however, very inefficient, converging at rate $1/\sqrt[3]{n}$. Furthermore, it displays marked sensitivity to central data values; see Hettmansperger and Sheather (1992) and Davies (1993, §2.3).

Rousseeuw suggested least trimmed squares (LTS) regression:

$$\min_b \sum_{i=1}^{q} |y_i - \boldsymbol{x}_i b|^2_{(i)}$$

as this is more efficient, but shares the same extreme resistance. The recommended sum is over the smallest $q = \lfloor (n+p+1)/2 \rfloor$ squared residuals. (Earlier accounts differed.)

This was followed by *S-estimation*, in which the coefficients are chosen to find the solution to

$$\sum_{i=1}^{n} \chi\Big(\frac{y_i - \boldsymbol{x}_i b}{c_0\, s}\Big) = (n-p)\beta$$

with smallest scale s. Here χ is usually chosen to be the integral of Tukey's bisquare function

$$\chi(u) = u^6 - 3u^4 + 3u^2,\ |u| \leqslant 1, \qquad 1,\ |u| \geqslant 1$$

$c_0 = 1.548$ and $\beta = 0.5$ is chosen for consistency at the normal distribution of errors. This gives efficiency 28.7% at the normal, which is low but better than LMS and LTS.

In only a few special cases (such as LMS for univariate regression with intercept) can these optimization problems be solved exactly, and approximate search methods are used.

S *implementation*

Various versions of S-PLUS have (different) implementations of LMS and LTS regression in functions `lmsreg` and `ltsreg`[6], but as these are not fully documented and give different results in different releases, we prefer our function `lqs`.[7] The default method is LTS.

```
> lqs(calls ~ year, data = phones)
Coefficients:
 (Intercept)   year
 -56.2         1.16
Scale estimates 1.25 1.13

> lqs(calls ~ year, data = phones, method = "lms")
Coefficients:
 (Intercept)   year
 -55.9         1.15
Scale estimates 0.938 0.909

> lqs(calls ~ year, data = phones, method = "S")
Coefficients:
 (Intercept)  year
 -52.5         1.1
Scale estimates 2.13
```

Two scale estimates are given for LMS and LTS: the first comes from the fit criterion, the second from the variance of the residuals of magnitude no more than 2.5 times the first scale estimate. All the scale estimates are set up to be

[6] In S-PLUS this now uses 10% trimming.

[7] Adopted by R in its package `lqs` .

consistent at the normal, but measure different things for highly non-normal data (as here).

MM-estimation

It is possible to combine the resistance of these methods with the efficiency of M-estimation. The MM-estimator proposed by Yohai, Stahel and Zamar (1991) (see also Marazzi, 1993, §9.1.3) is an M-estimator starting at the coefficients given by the S-estimator and with fixed scale given by the S-estimator. This retains (for $c > c_0$) the high-breakdown point of the S-estimator and the high efficiency at the normal. At considerable computational expense, this gives the best of both worlds.

Our function `rlm` has an option to implement MM-estimation.

```
> summary(rlm(calls ~ year, data=phones, method="MM"), cor = F)
Coefficients:
             Value Std. Error t value
(Intercept) -52.423   2.916    -17.978
       year   1.101   0.047     23.367

Residual standard error: 2.13 on 22 degrees of freedom
```

S-PLUS has a function `lmRob` in library section `robust` that implements a slightly different MM-estimator with similar properties, and comes with a full set of method functions, so it can be used routinely as a replacement for `lm`. Let us try it on the `phones` data. S+

```
> library(robust, first = T) # S-PLUS only
> phones.lmr <- lmRob(calls ~ year, data = phones)
> summary(phones.lmr, cor = F)
Coefficients:
             Value Std. Error t value Pr(>|t|)
(Intercept) -52.541   3.625    -14.493   0.000
       year   1.104   0.061     18.148   0.000

Residual scale estimate: 2.03 on 22 degrees of freedom
Proportion of variation in response explained by model: 0.494

Test for Bias:
            Statistics P-value
 M-estimate      1.401   0.496
LS-estimate      0.243   0.886
> plot(phones.lmr)
```

This works well, rejecting all the spurious observations. The 'test for bias' is of the M-estimator against the initial S-estimator; if the M-estimator appears biased the initial S-estimator is returned.

Library section `robust` provides a wide range of robust techniques.

Scottish hill races revisited

We return to the data on Scottish hill races studied in the introduction and Section 6.3. There we saw one gross outlier and a number of other extreme observations.

```
> hills.lm
Coefficients:
 (Intercept)  dist    climb
      -8.992 6.218 0.011048
Residual standard error: 14.676

> hills1.lm # omitting Knock Hill
Coefficients:
 (Intercept)   dist    climb
      -13.53 6.3646 0.011855
Residual standard error: 8.8035

> rlm(time ~ dist + climb, data = hills)
Coefficients:
 (Intercept)   dist     climb
     -9.6067 6.5507 0.0082959
Scale estimate: 5.21

> summary(rlm(time ~ dist + climb, data = hills,
              weights = 1/dist^2, method = "MM"), cor = F)
Coefficients:
              Value Std. Error t value
(Intercept)  -1.802   1.664     -1.083
       dist   5.244   0.233     22.549
      climb   0.007   0.001      9.391
Residual standard error: 4.84 on 32 degrees of freedom

> lqs(time ~ dist + climb, data = hills, nsamp = "exact")
Coefficients:
 (Intercept)     dist    climb
 -1.26        4.86     0.00851
Scale estimates 2.94 3.01
```

Notice that the intercept is no longer significant in the robust weighted fits. By default `lqs` uses a random search, but here exhaustive enumeration is possible, so we use it.

If we move to the model for inverse speed:

```
> summary(hills2.lm) # omitting Knock Hill
Coefficients:
             Value Std. Error t value Pr(>|t|)
(Intercept)  4.900  0.474     10.344   0.000
       grad  0.008  0.002      5.022   0.000

Residual standard error: 1.28 on 32 degrees of freedom
```

```
> summary(rlm(ispeed ~ grad, data = hills), cor = F)
Coefficients:
             Value Std. Error t value
(Intercept)  5.176  0.381     13.585
       grad  0.007  0.001      5.428

Residual standard error: 0.869 on 33 degrees of freedom
# method="MM" results are very similar.
> # S: summary(lmRob(ispeed ~ grad, data = hills))
             Value Std. Error t value Pr(>|t|)
(Intercept)  5.082  0.403     12.612   0.000
       grad  0.008  0.002      5.055   0.000

Residual scale estimate: 0.819 on 33 degrees of freedom

> lqs(ispeed ~ grad, data = hills)
Coefficients:
 (Intercept)    grad
 4.75        0.00805
Scale estimates 0.608 0.643
```

The results are in close agreement with the least-squares results after removing `Knock Hill`.

6.6 Bootstrapping Linear Models

In frequentist inference we have to consider what might have happened but did not. Linear models can arise exactly or approximately in a number of ways. The most commonly considered form is

$$Y = X\beta + \epsilon$$

in which only ϵ is considered to be random. This supposes that in all (hypothetical) repetitions the same x points would have been chosen, but the responses would vary. This is a plausible assumption for a designed experiment such as the N, P, K experiment on page 165 and for an observational study such as Quine's with prespecified factors. It is less clearly suitable for the Scottish hill races, and clearly not correct for Whiteside's gas consumption data.

Another form of regression is sometimes referred to as the *random regressor* case in which the pairs (x_i, y_i) are thought of as a random sample from a population and we are interested in the regression function $f(x) = \mathrm{E}\big(Y \mid X = x\big)$ which is assumed to be linear. This seems appropriate for the gas consumption data. However, it is common to perform conditional inference in this case and condition on the observed xs, converting this to a fixed-design problem. For example, in the hill races the inferences drawn depend on whether certain races, notably Bens of Jura, were included in the sample. As they were included, conclusions conditional on the set of races seems most pertinent. (There are other

ways that linear models can arise, including calibration problems and where both x and y are measured with error about a true linear relationship.)

These considerations are particularly relevant when we consider bootstrap resampling. The most obvious form of bootstrapping is to randomly sample pairs (x_i, y_i) with replacement,[8] which corresponds to randomly weighted regressions. However, this may not be appropriate in not mimicking the assumed random variation and in some examples in producing singular fits with high probability. The main alternative, *model-based resampling*, is to resample the residuals. After fitting the linear model we have

$$y_i = x_i\widehat{\beta} + e_i$$

and we create a new dataset by $y_i = x_i\widehat{\beta} + e_i^*$ where the (e_i^*) are resampled with replacement from the residuals (e_i). There are a number of possible objections to this procedure. First, the residuals need not have mean zero if there is no intercept in the model, and it is usual to subtract their mean. Second, they do not have the correct variance or even the same variance. Thus we can adjust their variance by resampling the *modified residuals* $r_i = e_1/\sqrt{1 - h_{ii}}$, which have variance σ^2 from (6.3).

We see bootstrapping as having little place in least-squares regression. If the errors are close to normal, the standard theory suffices. If not, there are better methods of fitting than least-squares, or perhaps the data should be transformed as in the `quine` dataset on page 171.

The distribution theory for the estimated coefficients in robust regression is based on asymptotic theory, so we could use bootstrap estimates of variability as an alternative. Resampling the residuals seems most appropriate for the `phones` data.

```
library(boot)
fit <- lm(calls ~ year, data = phones)
ph <- data.frame(phones, res = resid(fit), fitted = fitted(fit))
ph.fun <- function(data, i) {
  d <- data
  d$calls <- d$fitted + d$res[i]
  coef(update(fit, data=d))
}

(ph.lm.boot <- boot(ph, ph.fun, R = 999))
    ....
     original    bias    std. error
t1* -260.0592  0.210500     95.3262
t2*    5.0415 -0.011469      1.5385

fit <- rlm(calls ~ year, method = "MM", data = phones)
ph <- data.frame(phones, res = resid(fit), fitted = fitted(fit))
(ph.rlm.boot <- boot(ph, ph.fun, R = 999))
    ....
```

[8] Davison and Hinkley (1997) call this *case-based resampling*.

Table 6.1: Layout of a classic N, P, K fractional factorial design. The response is yield (in lbs/(1/70)acre-plot).

pk	np	—	nk
49.5	62.8	46.8	57.0
n	npk	k	p
59.8	58.5	55.5	56.0
p	npk	n	k
62.8	55.8	69.5	55.0

n	npk	k	p
62.0	48.8	45.5	44.2
np	—	nk	pk
52.0	51.5	49.8	48.8
nk	np	pk	—
57.2	59.0	53.2	56.0

```
     original      bias  std. error
t1* -52.4230  2.354793    26.98130
t2*   1.1009 -0.014189     0.37449
```

(The `rlm` bootstrap runs took about fifteen minutes,[9] and readers might like to start with a smaller number of resamples.) These results suggest that the asymptotic theory for `rlm` is optimistic for this example, but as the residuals are clearly serially correlated the validity of the bootstrap is equally in doubt. Statistical inference really does depend on what one considers might have happened but did not.

The bootstrap results can be investigated further by using `plot`, and `boot.ci` will give confidence intervals for the coefficients. The robust results have very long tails.

6.7 Factorial Designs and Designed Experiments

Factorial designs are powerful tools in the design of experiments. Experimenters often cannot afford to perform all the runs needed for a complete factorial experiment, or they may not all be fitted into one experimental block. To see what can be achieved, consider the following N, P, K (*nitrogen, phosphate, potassium*) factorial experiment on the growth of peas which was conducted on six blocks shown in Table 6.1.

Half of the design (technically a fractional factorial design) is performed in each of six blocks, so each half occurs three times. (If we consider the variables to take values ± 1, the halves are defined by even or odd parity, equivalently product equal to $+1$ or -1.) Note that the NPK interaction cannot be estimated as it is confounded with block differences, specifically with $(b_2 + b_3 + b_4 - b_1 - b_5 - b_6)$. An ANOVA table may be computed by

```
> (npk.aov <- aov(yield ~ block + N*P*K, data = npk))
    ....
Terms:
```

[9] Using S-PLUS under Linux; R took 90 seconds.

```
                 block      N      P      K    N:P    N:K
 Sum of Squares 343.29 189.28   8.40  95.20  21.28  33.14
Deg. of Freedom      5      1      1      1      1      1
                   P:K Residuals
 Sum of Squares   0.48    185.29
Deg. of Freedom      1        12

Residual standard error: 3.9294
1 out of 13 effects not estimable
Estimated effects are balanced
> summary(npk.aov)
          Df Sum of Sq Mean Sq F Value   Pr(F)
    block  5    343.29   68.66   4.447 0.01594
        N  1    189.28  189.28  12.259 0.00437
        P  1      8.40    8.40   0.544 0.47490
        K  1     95.20   95.20   6.166 0.02880
      N:P  1     21.28   21.28   1.378 0.26317
      N:K  1     33.14   33.14   2.146 0.16865
      P:K  1      0.48    0.48   0.031 0.86275
Residuals 12    185.29   15.44

> alias(npk.aov)
    ....
Complete
      (Intercept) block1 block2 block3 block4 block5 N P K
N:P:K               1      0.33   0.17  -0.3   -0.2
      N:P N:K P:K
N:P:K
> coef(npk.aov)
 (Intercept) block1 block2  block3  block4 block5
      54.875 1.7125 1.6792 -1.8229 -1.0137  0.295
      N        P       K      N:P    N:K     P:K
 2.8083 -0.59167 -1.9917 -0.94167 -1.175 0.14167
```

Note how the `N:P:K` interaction is silently omitted in the summary, although its absence is mentioned in printing `npk.aov`. The `alias` command shows which effect is missing (the particular combinations corresponding to the use of Helmert contrasts for the factor `block`).

Only the `N` and `K` main effects are significant (we ignore blocks whose terms are there precisely because we expect them to be important and so we must allow for them). For two-level factors the Helmert contrast is the same as the sum contrast (up to sign) giving -1 to the first level and $+1$ to the second level. Thus the effects of adding nitrogen and potassium are 5.62 and -3.98, respectively. This interpretation is easier to see with treatment contrasts:

```
> options(contrasts = c("contr.treatment", "contr.poly"))
> npk.aov1 <- aov(yield ~ block + N + K, data = npk)
> summary.lm(npk.aov1)
    ....
Coefficients:
```

```
            Value Std. Error t value Pr(>|t|)
    ....
      N   5.617   1.609      3.490   0.003
      K  -3.983   1.609     -2.475   0.025

Residual standard error: 3.94 on 16 degrees of freedom
```

Note the use of `summary.lm` to give the standard errors. Standard errors of contrasts can also be found from the function `se.contrast`. The full form is quite complex, but a simple use is:

```
> se.contrast(npk.aov1, list(N == "0", N == "1"), data = npk)
Refitting model to allow projection
[1] 1.6092
```

For highly regular designs such as this standard errors may also be found along with estimates of means, effects and other quantities using `model.tables`.

```
> model.tables(npk.aov1, type = "means", se = T)
    ....
 N
      0      1
 52.067 57.683
    ....
Standard errors for differences of means
         block       N       K
        2.7872  1.6092  1.6092
replic. 4.0000 12.0000 12.0000
```

Generating designs

The three functions[10] `expand.grid`, `fac.design` and `oa.design` can each be used to construct designs such as our example.

Of these, `expand.grid` is the simplest. It is used in a similar way to `data.frame`; the arguments may be named and the result is a data frame with those names. The columns contain all combinations of values for each argument. If the argument values are numeric the column is numeric; if they are anything else, for example, character, the column is a factor. Consider an example:

```
> mp <- c("-", "+")
> (NPK <- expand.grid(N = mp, P = mp, K = mp))
  N P K
1 - - -
2 + - -
3 - + -
4 + + -
5 - - +
6 + - +
7 - + +
8 + + +
```

[10] Only `expand.grid` is in R.

Note that the first column changes fastest and the last slowest. This is a single complete replicate.

Our example used three replicates, each split into two blocks so that the block comparison is confounded with the highest-order interaction. We can construct such a design in stages. First we find a half-replicate to be repeated three times and form the contents of three of the blocks. The simplest way to do this is to use `fac.design`:

```
blocks13 <- fac.design(levels = c(2, 2, 2),
     factor= list(N=mp, P=mp, K=mp), rep = 3, fraction = 1/2)
```

The first two arguments give the numbers of levels and the factor names and level labels. The third argument gives the number of replications (default 1). The `fraction` argument may only be used for 2^p factorials. It may be given either as a small negative power of 2, as here, or as a *defining contrast formula*. When `fraction` is numerical the function chooses a defining contrast that becomes the `fraction` attribute of the result. For half-replicates the highest-order interaction is chosen to be aliased with the mean. To find the complementary fraction for the remaining three blocks we need to use the defining contrast formula form for `fraction`:

```
blocks46 <- fac.design(levels = c(2, 2, 2),
    factor = list(N=mp, P=mp, K=mp), rep = 3, fraction = ~ -N:P:K)
```

(This is explained in the following.) To complete our design we put the blocks together, add in the `block` factor and randomize:

```
NPK <- design(block = factor(rep(1:6, each = 4)),
              rbind(blocks13, blocks46))
i <- order(runif(6)[NPK$block], runif(24))
NPK <- NPK[i,]  # Randomized
```

Using `design` instead of `data.frame` creates an object of class `design` that inherits from `data.frame`. For most purposes designs and data frames are equivalent, but some generic functions such as `plot`, `formula` and `alias` have useful `design` methods.

Defining contrast formulae resemble model formulae in syntax only; the meaning is quite distinct. There is no left-hand side. The right-hand side consists of colon products of factors only, separated by `+` or `-` signs. A plus (or leading blank) specifies that the treatments with *positive* signs for that contrast are to be selected and a minus those with *negative* signs. A formula such as `~A:B:C-A:D:E` specifies a quarter-replicate consisting of the treatments that have a positive sign in the ABC interaction and a negative sign in ADE.

Box, Hunter and Hunter (1978, §12.5) consider a 2^{7-4} design used for an experiment in riding up a hill on a bicycle. The seven factors are Seat (up or down), Dynamo (off or on), Handlebars (up or down), Gears (low or medium), Raincoat (on or off), Breakfast (yes or no) and Tyre pressure (hard or soft). A resolution III design was used, so the main effects are not aliased with each other.

Such a design cannot be constructed using a numerical fraction in `fac.design`, so the defining contrasts have to be known. Box *et al*. use the design relations:

$$D = AB, \quad E = AC, \quad F = BC, \quad G = ABC$$

which mean that ABD, ACE, BCF and $ABCG$ are all aliased with the mean, and form the defining contrasts of the fraction. Whether we choose the positive or negative halves is immaterial here.

```
> lev <- rep(2, 7)
> factors <- list(S=mp, D=mp, H=mp, G=mp, R=mp, B=mp, P=mp)
> (Bike <- fac.design(lev, factors,
      fraction = ~ S:D:G + S:H:R + D:H:B + S:D:H:P))
  S D H G R B P
1 - - - - - - -
2 - + + + + - -
3 + - + + - + -
4 + + - - + + -
5 + + + - - - +
6 + - - + + - +
7 - + - + - + +
8 - - + - + + +

Fraction:  ~ S:D:G + S:H:R + D:H:B + S:D:H:P
```

(We chose `P` for pressure rather than `T` for tyres since `T` and `F` are reserved identifiers.)

We may check the symmetry of the design using `replications`:

```
> replications(~.^2, data = Bike)
 S D H G R B P S:D S:H S:G S:R S:B S:P D:H D:G D:R D:B D:P H:G
 4 4 4 4 4 4 4   2   2   2   2   2   2   2   2   2   2   2   2
 H:R H:B H:P G:R G:B G:P R:B R:P B:P
   2   2   2   2   2   2   2   2   2
```

Fractions may be specified either in a call to `fac.design` or subsequently using the `fractionate` function.

The third function, `oa.design`, provides some resolution III designs (also known as *main effect plans* or *orthogonal arrays*) for factors at two or three levels. Only low-order cases are provided, but these are the most useful in practice.

6.8 An Unbalanced Four-Way Layout

Aitkin (1978) discussed an observational study of S. Quine. The response is the number of days absent from school in a year by children from a large town in rural New South Wales, Australia. The children were classified by four factors, namely,

`Age`	4 levels: primary, first, second or third form
`Eth`	2 levels: aboriginal or non-aboriginal
`Lrn`	2 levels: slow or average learner
`Sex`	2 levels: male or female.

The dataset is included in the paper of Aitkin (1978) and is available as data frame `quine` in `MASS`. This has been explored several times already, but we now consider a more formal statistical analysis.

There were 146 children in the study. The frequencies of the combinations of factors are

```
> attach(quine)
> table(Lrn, Age, Sex, Eth)
, , F, A                    , , F, N
   F0 F1 F2 F3                 F0 F1 F2 F3
AL  4  5  1  9              AL  4  6  1 10
SL  1 10  8  0              SL  1 11  9  0

, , M, A                    , , M, N
   F0 F1 F2 F3                 F0 F1 F2 F3
AL  5  2  7  7              AL  6  2  7  7
SL  3  3  4  0              SL  3  7  3  0
```

(The output has been slightly rearranged to save space.) The classification is unavoidably very unbalanced. There are no slow learners in form `F3`, but all 28 other cells are non-empty. In his paper Aitkin considers a normal analysis on the untransformed response, but in the reply to the discussion he chooses a transformed response, $\log(\text{Days} + 1)$.

A casual inspection of the data shows that homoscedasticity is likely to be an unrealistic assumption on the original scale, so our first step is to plot the cell variances and standard deviations against the cell means.

```
Means <- tapply(Days, list(Eth, Sex, Age, Lrn), mean)
Vars  <- tapply(Days, list(Eth, Sex, Age, Lrn), var)
SD <- sqrt(Vars)
par(mfrow = c(1, 2))
plot(Means, Vars, xlab = "Cell Means", ylab = "Cell Variances")
plot(Means, SD, xlab = "Cell Means", ylab = "Cell Std Devn.")
```

Missing values are silently omitted from the plot. Interpretation of the result in Figure 6.5 requires some caution because of the small and widely different degrees of freedom on which each variance is based. Nevertheless the approximate linearity of the standard deviations against the cell means suggests a logarithmic transformation or something similar is appropriate. (See, for example, Rao, 1973, §6g.)

Some further insight on the transformation needed is provided by considering a model for the transformed observations

$$y^{(\lambda)} = \begin{cases} (y^\lambda - 1)/\lambda & \lambda \neq 0 \\ \log y & \lambda = 0 \end{cases}$$

where here $y = \text{Days} + \alpha$. (The positive constant α is added to avoid problems with zero entries.) Rather than include α as a second parameter we first consider Aitkin's choice of $\alpha = 1$. Box and Cox (1964) show that the profile likelihood function for λ is

$$\widehat{L}(\lambda) = \text{const} - \tfrac{n}{2} \log \text{RSS}(z^{(\lambda)})$$

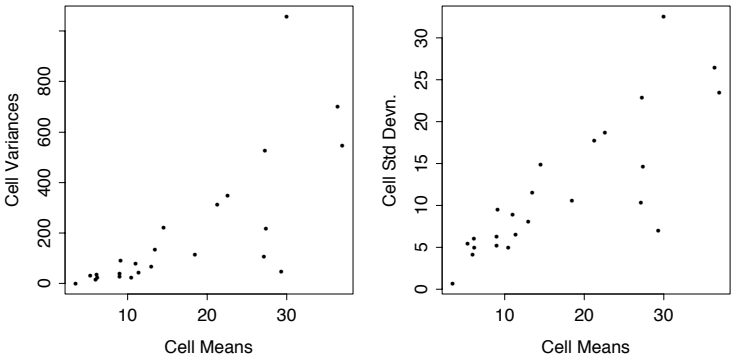

Figure 6.5: Two diagnostic plots for the Quine data.

where $z^{(\lambda)} = y^{(\lambda)}/\dot{y}^{\lambda-1}$, $\dot{y}$ is the geometric mean of the observations and $\mathrm{RSS}(z^{(\lambda)})$ is the residual sum of squares for the regression of $z^{(\lambda)}$.

Box & Cox suggest using the profile likelihood function for the largest linear model to be considered as a guide in choosing a value for λ, which will then remain fixed for any remaining analyses. Ideally other considerations from the context will provide further guidance in the choice of λ, and in any case it is desirable to choose easily interpretable values such as square-root, log or inverse.

Our `MASS` function `boxcox` calculates and (optionally) displays the Box–Cox profile likelihood function, together with a horizontal line showing what would be an approximate 95% likelihood ratio confidence interval for λ. The function is generic and several calling protocols are allowed but a convenient one to use here is with the same arguments as `lm` together with an additional (named) argument, `lambda`, to provide the sequence at which the marginal likelihood is to be evaluated. (By default the result is extended using a spline interpolation.)

Since the dataset has four empty cells the full model `Eth*Sex*Age*Lrn` has a rank-deficient model matrix. Hence in S-PLUS we must use `singular.ok = T` S+ to fit the model.

```
boxcox(Days+1 ~ Eth*Sex*Age*Lrn, data = quine, singular.ok = T,
       lambda = seq(-0.05, 0.45, len = 20))
```

(Alternatively the first argument may be a fitted model object that supplies all needed information apart from `lambda`.) The result is shown on the left-hand panel of Figure 6.6 which suggests strongly that a log transformation is not optimal when $\alpha = 1$ is chosen. An alternative one-parameter family of transformations that could be considered in this case is

$$t(y, \alpha) = \log(y + \alpha)$$

Using the same analysis as presented in Box and Cox (1964) the profile log likelihood for α is easily seen to be

$$\widehat{L}(\alpha) = \text{const} - \tfrac{n}{2} \log \mathrm{RSS}\{\log(y + \alpha)\} - \sum \log(y + \alpha)$$

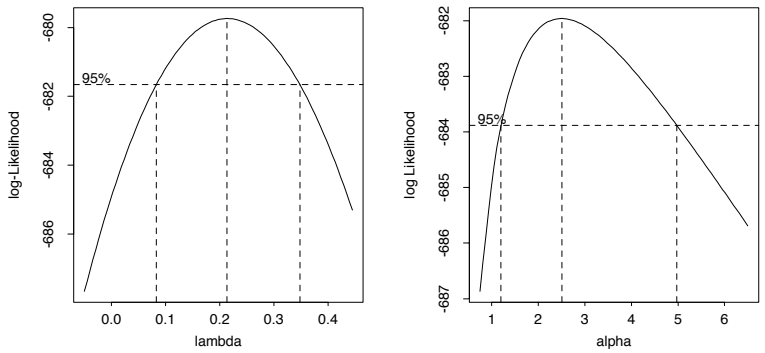

Figure 6.6: Profile likelihood for a Box–Cox transformation model with displacement $\alpha = 1$, left, and a displaced log transformation model, right.

It is interesting to see how this may be calculated directly using low-level tools, in particular the functions `qr` for the QR-decomposition and `qr.resid` for orthogonal projection onto the residual space. Readers are invited to look at our functions `logtrans.default` and `boxcox.default`.

```
logtrans(Days ~ Age*Sex*Eth*Lrn, data = quine,
    alpha = seq(0.75, 6.5, len = 20), singular.ok = T)
```

The result is shown in the right-hand panel of Figure 6.6. If a displaced log transformation is chosen a value $\alpha = 2.5$ is suggested, and we adopt this in our analysis. Note that $\alpha = 1$ is outside the notional 95% confidence interval. It can also be checked that with $\alpha = 2.5$ the log transform is well within the range of reasonable Box–Cox transformations to choose.

Model selection

The complete model, `Eth*Sex*Age*Lrn`, has a different parameter for each identified group and hence contains all possible simpler models for the mean as special cases, but has little predictive or explanatory power. For a better insight into the mean structure we need to find more parsimonious models. Before considering tools to prune or extend regression models it is useful to make a few general points on the process itself.

Marginality restrictions

In regression models it is usually the case that not all terms are on an equal footing as far as inclusion or removal is concerned. For example, in a quadratic regression on a single variable x one would normally consider removing only the highest-degree term, x^2, first. Removing the first-degree term while the second-degree one is still present amounts to forcing the fitted curve to be flat at $x = 0$, and unless there were some good reason from the context to do this it would be an arbitrary imposition on the model. Another way to view this is to note that if we write a polynomial regression in terms of a new variable $x^\star = x - \alpha$ the

model remains in predictive terms the same, but only the highest-order coefficient remains invariant. If, as is usually the case, we would like our model selection procedure not to depend on the arbitrary choice of origin we must work only with the highest-degree terms at each stage.

The linear term in x is said to be *marginal* to the quadratic term, and the intercept term is marginal to both. In a similar way if a second-degree term in two variables, $x_1 x_2$, is present, any linear terms in either variable or an intercept term are marginal to it.

There are circumstances where a regression through the origin does make sense, but in cases where the origin is arbitrary one would normally only consider regression models where for each term present all terms marginal to it are also present.

In the case of factors the situation is even more clear-cut. A two-factor interaction `a:b` is marginal to any higher-order interaction that contains `a` and `b`. Fitting a model such as `a + a:b` leads to a model matrix where the columns corresponding to `a:b` are extended to compensate for the absent marginal term, `b`, and the fitted values are the same as if it were present. Fitting models with marginal terms removed such as with `a*b - b` generates a model with no readily understood statistical meaning[11] but updating models specified in this way using `update` changes the model matrix so that the absent marginal term again has no effect on the fitted model. In other words, removing marginal factor terms from a fitted model is either statistically meaningless or futile in the sense that the model simply changes its parametrization to something equivalent.

Variable selection for the Quine data

The `anova` function when given a single fitted-model object argument constructs a *sequential* analysis of variance table. That is, a sequence of models is fitted by expanding the formula, arranging the terms in increasing order of marginality and including one additional term for each row of the table. The process is order-dependent for non-orthogonal designs and several different orders may be needed to appreciate the analysis fully if the non-orthogonality is severe. For an orthogonal design the process is not order-dependent provided marginality restrictions are obeyed.

To explore the effect of adding or dropping terms from a model our two functions `addterm` and `dropterm` are usually more convenient. These allow the effect of, respectively, adding or removing individual terms from a model to be assessed, where the model is defined by a fitted-model object given as the first argument. For `addterm` a second argument is required to specify the scope of the terms considered for inclusion. This may be a formula or an object defining a formula for a larger model. Terms are included or removed in such a way that the marginality principle for factor terms is obeyed; for purely quantitative regressors this has to be managed by the user.

[11] Marginal terms are sometimes removed in this way in order to calculate what are known as 'Type III sums of squares' but we have yet to see a situation where this makes compelling statistical sense. If they are needed, they can be computed by `summary.aov` in S-PLUS.

Both functions are generic and compute the change in AIC (Akaike, 1974)

$$\text{AIC} = -2\,\text{maximized log-likelihood} + 2\,\#\,\text{parameters}$$

Since the log-likelihood is defined only up to a constant depending on the data, this is also true of AIC. For a regression model with n observations, p parameters and normally-distributed errors the log-likelihood is

$$L(\beta, \sigma^2; \boldsymbol{y}) = \text{const} - \tfrac{n}{2}\log\sigma^2 - \tfrac{1}{2\sigma^2}\|\boldsymbol{y} - X\boldsymbol{\beta}\|^2$$

and on maximizing over β we have

$$L(\widehat{\boldsymbol{\beta}}, \sigma^2; \boldsymbol{y}) = \text{const} - \tfrac{n}{2}\log\sigma^2 - \tfrac{1}{2\sigma^2}\text{RSS}$$

Thus if σ^2 is *known*, we can take

$$\text{AIC} = \frac{\text{RSS}}{\sigma^2} + 2p + \text{const}$$

but if σ^2 is *unknown*,

$$L(\widehat{\boldsymbol{\beta}}, \widehat{\sigma}^2; \boldsymbol{y}) = \text{const} - \tfrac{n}{2}\log\widehat{\sigma}^2 - \tfrac{n}{2}, \qquad \widehat{\sigma}^2 = \text{RSS}/n$$

and so

$$\text{AIC} = n\log(\text{RSS}/n) + 2p + \text{const}$$

For *known* σ^2 it is conventional to use Mallows' C_p,

$$C_p = \text{RSS}/\sigma^2 + 2p - n$$

(Mallows, 1973) and in this case `addterm` and `dropterm` label their output as C_p.

For an example consider removing the four-way interaction from the complete model and assessing which three-way terms might be dropped next.

```
> quine.hi <- aov(log(Days + 2.5) ~ .^4, quine)
> quine.nxt <- update(quine.hi, . ~ . - Eth:Sex:Age:Lrn)
> dropterm(quine.nxt, test = "F")
Single term deletions
....
             Df Sum of Sq    RSS     AIC F Value   Pr(F)
     <none>               64.099 -68.184
Eth:Sex:Age  3    0.9739 65.073 -71.982  0.6077 0.61125
Eth:Sex:Lrn  1    1.5788 65.678 -66.631  2.9557 0.08816
Eth:Age:Lrn  2    2.1284 66.227 -67.415  1.9923 0.14087
Sex:Age:Lrn  2    1.4662 65.565 -68.882  1.3725 0.25743
```

Clearly dropping `Eth:Sex:Age` most reduces AIC but dropping `Eth:Sex:Lrn` would increase it. Note that only non-marginal terms are included; none are significant in a conventional F-test.

Alternatively we could start from the simplest model and consider adding terms to reduce C_p; in this case the choice of scale parameter is important, since the simple-minded choice is inflated and may over-penalize complex models.

```
> quine.lo <- aov(log(Days+2.5) ~ 1, quine)
> addterm(quine.lo, quine.hi, test = "F")
Single term additions
....
       Df Sum of Sq    RSS     AIC F Value   Pr(F)
<none>              106.79 -43.664
   Eth  1    10.682  96.11 -57.052  16.006 0.00010
   Sex  1     0.597 106.19 -42.483   0.809 0.36981
   Age  3     4.747 102.04 -44.303   2.202 0.09048
   Lrn  1     0.004 106.78 -41.670   0.006 0.93921
```

It appears that only `Eth` and `Age` might be useful, although in fact all factors are needed since some higher-way interactions lead to large decreases in the residual sum of squares.

Automated model selection

Our function `stepAIC` may be used to automate the process of stepwise selection. It requires a fitted model to define the starting process (one somewhere near the final model is probably advantageous), a list of two formulae defining the upper (most complex) and lower (most simple) models for the process to consider and a scale estimate. If a large model is selected as the starting point, the `scope` and `scale` arguments have generally reasonable defaults, but for a small model where the process is probably to be one of adding terms, they will usually need both to be supplied. (A further argument, `direction`, may be used to specify whether the process should only add terms, only remove terms, or do either as needed.)

By default the function produces a verbose account of the steps it takes which we turn off here for reasons of space, but which the user will often care to note. The `anova` component of the result shows the sequence of steps taken and the reduction in AIC or C_p achieved.

```
> quine.stp <- stepAIC(quine.nxt,
    scope = list(upper = ~Eth*Sex*Age*Lrn, lower = ~1),
    trace = F)
> quine.stp$anova
....
           Step Df Deviance Resid. Df Resid. Dev     AIC
1                                 120     64.099 -68.184
2 - Eth:Sex:Age  3   0.9739       123     65.073 -71.982
3 - Sex:Age:Lrn  2   1.5268       125     66.600 -72.597
```

At this stage we might want to look further at the final model from a significance point of view. The result of `stepAIC` has the same class as its starting point argument, so in this case `dropterm` may be used to check each remaining non-marginal term for significance.

```
> dropterm(quine.stp, test = "F")
            Df Sum of Sq    RSS     AIC F Value   Pr(F)
     <none>              66.600 -72.597
    Sex:Age  3    10.796 77.396 -56.663  6.7542 0.00029
Eth:Sex:Lrn  1     3.032 69.632 -68.096  5.6916 0.01855
Eth:Age:Lrn  2     2.096 68.696 -72.072  1.9670 0.14418
```

The term `Eth:Age:Lrn` is not significant at the conventional 5% significance level. This suggests, correctly, that selecting terms on the basis of AIC can be somewhat permissive in its choice of terms, being roughly equivalent to choosing an F-cutoff of 2. We can proceed manually

```
> quine.3 <- update(quine.stp, . ~ . - Eth:Age:Lrn)
> dropterm(quine.3, test = "F")
            Df Sum of Sq    RSS     AIC F Value   Pr(F)
     <none>              68.696 -72.072
    Eth:Age  3     3.031 71.727 -71.768  1.8679 0.13833
    Sex:Age  3    11.427 80.123 -55.607  7.0419 0.00020
    Age:Lrn  2     2.815 71.511 -70.209  2.6020 0.07807
Eth:Sex:Lrn  1     4.696 73.391 -64.419  8.6809 0.00383
> quine.4 <- update(quine.3, . ~ . - Eth:Age)
> dropterm(quine.4, test = "F")
            Df Sum of Sq    RSS     AIC F Value    Pr(F)
     <none>              71.727 -71.768
    Sex:Age  3    11.566 83.292 -55.942   6.987 0.000215
    Age:Lrn  2     2.912 74.639 -69.959   2.639 0.075279
Eth:Sex:Lrn  1     6.818 78.545 -60.511  12.357 0.000605
> quine.5 <- update(quine.4, . ~ . - Age:Lrn)
> dropterm(quine.5, test = "F")

Model:
log(Days + 2.5) ~ Eth + Sex + Age + Lrn + Eth:Sex + Eth:Lrn
                  + Sex:Age + Sex:Lrn + Eth:Sex:Lrn
            Df Sum of Sq    RSS     AIC F Value     Pr(F)
     <none>              74.639 -69.959
    Sex:Age  3    9.9002 84.539 -57.774   5.836 0.0008944
Eth:Sex:Lrn  1    6.2988 80.937 -60.130  11.140 0.0010982
```

or by setting `k = 4` in `stepAIC`. We obtain a model equivalent to `Sex/(Age + Eth*Lrn)` which is the same as that found by Aitkin (1978), apart from his choice of $\alpha = 1$ for the displacement constant. (However, when we consider a negative binomial model for the same data in Section 7.4 a more extensive model seems to be needed.)

Standard diagnostic checks on the residuals from our final fitted model show no strong evidence of any failure of the assumptions, as the reader may wish to verify.

It can also be verified that had we started from a very simple model and worked forwards we would have stopped much sooner with a much simpler model, even using the same scale estimate. This is because the major reductions in the residual sum of squares only occur when the third-order interaction `Eth:Sex:Lrn` is included.

There are other tools in S-PLUS for model selection called `stepwise` and `leaps`,[12] but these only apply for quantitative regressors. There is also no possibility of ensuring that marginality restrictions are obeyed.

[12] There are equivalent functions in the R package `leaps` on CRAN.

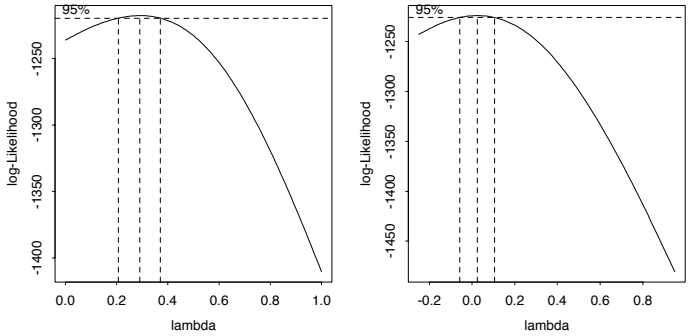

Figure 6.7: Box–Cox plots for the `cpus` data. Left: original regressors. Right: discretized regressors.

6.9 Predicting Computer Performance

Ein-Dor and Feldmesser (1987) studied data on the performance on a benchmark of a mix of minicomputers and mainframes. The measure was normalized relative to an IBM 370/158–3. There were six machine characteristics: the cycle time (nanoseconds), the cache size (Kb), the main memory size (Kb) and number of channels. (For the latter two there are minimum and maximum possible values; what the actual machine tested had is unspecified.) The original paper gave a linear regression for the square root of performance, but log scale looks more intuitive.

We can consider the Box–Cox family of transformations, Figure 6.7.

```
boxcox(perf ~ syct + mmin + mmax + cach + chmin + chmax,
       data = cpus, lambda = seq(0, 1, 0.1))
```

which tends to suggest a power of around 0.3 (and excludes both 0 and 0.5 from its 95% confidence interval). However, this does not allow for the regressors to be transformed, and many of them would be most naturally expressed on log scale. One way to allow the variables to be transformed is to discretize them; we show a more sophisticated approach in Section 8.8.

```
cpus1 <- cpus
attach(cpus)
for(v in names(cpus)[2:7])
  cpus1[[v]] <- cut(cpus[[v]], unique(quantile(cpus[[v]])),
                    include.lowest = T)
detach()
boxcox(perf ~ syct + mmin + mmax + cach + chmin + chmax,
       data = cpus1, lambda = seq(-0.25, 1, 0.1))
```

which does give a confidence interval including zero.

The purpose of this study is to predict computer performance. We randomly select 100 examples for fitting the models and test the performance on the remaining 109 examples.

```
> set.seed(123)
> cpus2 <- cpus[, 2:8]  # excludes names, authors' predictions
> cpus.samp <- sample(1:209, 100)
> cpus.lm <- lm(log10(perf) ~ ., data = cpus2[cpus.samp, ])
> test.cpus <- function(fit)
    sqrt(sum((log10(cpus2[-cpus.samp, "perf"]) -
              predict(fit, cpus2[-cpus.samp,]))^2)/109)
> test.cpus(cpus.lm)
[1] 0.21295
> cpus.lm2 <- stepAIC(cpus.lm, trace = F)
> cpus.lm2$anova
    Step Df Deviance Resid. Df Resid. Dev     AIC
1                           93     3.2108 -329.86
2 - syct  1 0.013177        94     3.2240 -331.45
> test.cpus(cpus.lm2)
[1] 0.21711
```

So selecting a smaller model does not improve the performance on this random split. We consider a variety of non-linear models for this example in later chapters.

6.10 Multiple Comparisons

As we all know, the theory of p-values of hypothesis tests and of the coverage of confidence intervals applies to pre-planned analyses. However, the only circumstances in which an adjustment is routinely made for testing after looking at the data is in multiple comparisons of contrasts in designed experiments. This is sometimes known as *post hoc* adjustment.

Consider the experiment on yields of barley in our dataset `immer`.[13] This has the yields of five varieties of barley at six experimental farms in both 1931 and 1932; we average the results for the two years. An analysis of variance gives

```
> immer.aov <- aov((Y1 + Y2)/2 ~ Var + Loc, data = immer)
> summary(immer.aov)
          Df Sum of Sq Mean Sq F Value     Pr(F)
       Var  4      2655   663.7   5.989 0.0024526
       Loc  5     10610  2122.1  19.148 0.0000005
Residuals 20      2217   110.8
```

The interest is in the difference in yield between varieties, and there is a statistically significant difference. We can see the mean yields by a call to `model.tables`.

```
> model.tables(immer.aov, type = "means", se = T, cterms = "Var")
    ....
 Var
     M      P      S      T      V
 94.39 102.54  91.13 118.20  99.18
```

[13] The Trellis dataset `barley` discussed in Cleveland (1993) is a more extensive version of the same dataset.

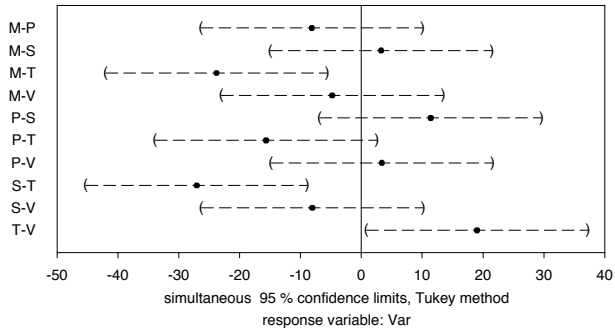

Figure 6.8: Simultaneous 95% confidence intervals for variety comparisons in the `immer` dataset.

```
Standard errors for differences of means
          Var
        6.078
replic. 6.000
```

This suggests that variety `T` is different from all the others, as a pairwise significant difference at 5% would exceed $6.078 \times t_{20}(0.975) \approx 12.6$; however the comparisons to be made have been selected after looking at the fit.

Function `multicomp`[14] allows us to compute *simultaneous* confidence intervals in this problem, that is, confidence intervals such that the probability that they cover the true values for all of the comparisons considered is bounded above by 5% for 95% confidence intervals. We can also plot the confidence intervals (Figure 6.8) by

```
> multicomp(immer.aov, plot = T)  # S-PLUS only
95 % simultaneous confidence intervals for specified
linear combinations, by the Tukey method

critical point: 2.9925
response variable: (Y1 + Y2)/2

intervals excluding 0 are flagged by '****'

    Estimate Std.Error Lower Bound Upper Bound
M-P    -8.15      6.08     -26.300       10.00
M-S     3.26      6.08     -14.900       21.40
M-T   -23.80      6.08     -42.000       -5.62 ****
M-V    -4.79      6.08     -23.000       13.40
P-S    11.40      6.08      -6.780       29.60
P-T   -15.70      6.08     -33.800        2.53
P-V     3.36      6.08     -14.800       21.50
S-T   -27.10      6.08     -45.300       -8.88 ****
```

[14] Available in S-PLUS, but not in R.

```
S-V    -8.05      6.08     -26.200       10.10
T-V    19.00      6.08       0.828       37.20 ****
```

This does not allow us to conclude that variety `T` has a significantly different yield than variety `P`.

We can do the Tukey multiple comparison test in R by

```
> (tk <- TukeyHSD(immer.aov, which = "Var"))
  Tukey multiple comparisons of means
    95% family-wise confidence level

Fit: aov(formula = (Y1 + Y2)/2 ~ Var + Loc, data = immer)
$Var
        diff      lwr      upr
P-M   8.1500 -10.0376 26.33759
S-M  -3.2583 -21.4459 14.92925
T-M  23.8083   5.6207 41.99592
V-M   4.7917 -13.3959 22.97925
S-P -11.4083 -29.5959  6.77925
T-P  15.6583  -2.5293 33.84592
V-P  -3.3583 -21.5459 14.82925
T-S  27.0667   8.8791 45.25425
V-S   8.0500 -10.1376 26.23759
V-T -19.0167 -37.2043 -0.82908

> plot(tk)
```

We may want to restrict the set of comparisons, for example to comparisons with a control treatment. The dataset `oats` is discussed on page 282; here we ignore the split-plot structure.

```
> oats1 <- aov(Y ~ N + V + B, data = oats)
> summary(oats1)
          Df Sum of Sq Mean Sq F Value    Pr(F)
        N  3     20020  6673.5  28.460 0.000000
        V  2      1786   893.2   3.809 0.027617
        B  5     15875  3175.1  13.540 0.000000
Residuals 61     14304   234.5

> multicomp(oats1, focus = "V")  # S-PLUS only

95 % simultaneous confidence intervals for specified
linear combinations, by the Tukey method

critical point: 2.4022
response variable: Y

intervals excluding 0 are flagged by '****'

                      Estimate Std.Error Lower Bound
Golden.rain-Marvellous    -5.29      4.42      -15.90
```

```
   Golden.rain-Victory     6.88      4.42       -3.74
    Marvellous-Victory    12.20      4.42        1.55
                        Upper Bound
Golden.rain-Marvellous         5.33
   Golden.rain-Victory        17.50
    Marvellous-Victory        22.80 ****

> # R: (tk <- TukeyHSD(oats1, which = "V"))
  Tukey multiple comparisons of means
    95% family-wise confidence level

Fit: aov(formula = Y ~ N + V + B, data = oats)
$V
                           diff       lwr     upr
Marvellous-Golden.rain   5.2917  -5.3273 15.9107
Victory-Golden.rain     -6.8750 -17.4940  3.7440
Victory-Marvellous     -12.1667 -22.7857 -1.5477

> plot(tk)

> multicomp(oats1, focus = "N", comparisons = "mcc", control = 1)
    ....
               Estimate Std.Error Lower Bound Upper Bound
0.2cwt-0.0cwt     19.5       5.1        7.24        31.8 ****
0.4cwt-0.0cwt     34.8       5.1       22.60        47.1 ****
0.6cwt-0.0cwt     44.0       5.1       31.70        56.3 ****
```

Note that we need to specify the control level; perversely by default the last level is chosen. We might also want to know if all the increases in nitrogen give significant increases in yield, which we can examine by

```
> lmat <- matrix(c(0,-1,1,rep(0, 11), 0,0,-1,1, rep(0,10),
                 0,0,0,-1,1,rep(0,9)), , 3,
        dimnames = list(NULL,
        c("0.2cwt-0.0cwt", "0.4cwt-0.2cwt", "0.6cwt-0.4cwt")))
> multicomp(oats1, lmat = lmat, bounds = "lower",
            comparisons = "none")
    ....
               Estimate Std.Error Lower Bound
0.2cwt-0.0cwt    19.50       5.1        8.43 ****
0.4cwt-0.2cwt    15.30       5.1        4.27 ****
0.6cwt-0.4cwt     9.17       5.1       -1.90
```

There is a bewildering variety of methods for multiple comparisons reflected in the options for `multicomp`. Miller (1981), Hsu (1996) and Yandell (1997, Chapter 6) give fuller details. Do remember that this tackles only part of the problem; the analyses here have been done after selecting a model and specific factors on which to focus. The allowance for multiple comparisons is only over contrasts of one selected factor in one selected model.