12

Linear models

Many data sets are inherently too complex to be handled adequately by standard procedures and thus require the formulation of ad hoc models. The class of *linear models* provides a flexible framework into which many — although not all — of these cases can be fitted.

You may have noticed that the `lm` function is applied to data classified into groups (Chapter 7) as well as to (multiple) linear regression (Chapters 6 and 11) problems, even though the theory for these procedures appears to be quite different. However, they are, in fact, special cases of the same general model.

The basic point is that a multiple regression model can describe a wide variety of situations if you choose the explanatory variables suitably. There is no requirement that the explanatory variables should follow a normal distribution, or any continuous distribution for that matter. One simple example (which we use without comment in Chapter 11) is that a grouping into two categories can be coded as a 0/1 variable and used in a regression analysis. The regression coefficient in that case corresponds to a difference between two groups rather than the slope of an actual line. To encode a grouping with more than two categories, you can use multiple 0/1 variables.

Generating these *dummy variables* becomes tedious, but it can be automated by the use of model formulas. Among other things, such formulas provide a convenient abstraction by treating classification variables (factors) and continuous variables symmetrically. You will need to learn

P. Dalgaard, *Introductory Statistics with R*,
DOI: 10.1007/978-0-387-79054-1_12,

exactly what model formulas do in order to become able to express your own modelling ideas.

This chapter contains a collection of models and their handling by `lm`, mainly in the form of relatively minor extensions and modifications of methods described earlier. It is meant only to give you a feel for the scope of possibilities and does not pretend to be complete.

12.1 Polynomial regression

One basic observation showing that multiple regression analysis can do more than meets the eye is that you can include second-order and higher powers of a variable in the model along with the original linear term. That is, you can have a model like

$$y = \alpha + \beta_1 x + \beta_2 x^2 + \cdots + \beta_k x^k + \epsilon$$

This obviously describes a nonlinear relation between y and x, but that does not matter; the model is still a linear model. What does matter is that the relation between the *parameters* and the expected observations is linear. It also does not matter that there is a deterministic relation between the regression variables $x, x^2, x^3, \ldots$, as long as there is no *linear* relation between them. However, fitting high-degree polynomials can be difficult because near-collinearity between terms makes the fit numerically unstable.

We return to the cystic fibrosis data set for an example. The plot of `pemax` and `height` in Figure 11.1 may suggest that the relation is not quite linear. One way to test this is to try to add a term that is the square of the height.

```
> attach(cystfibr)
> summary(lm(pemax~height+I(height^2)))
...
             Estimate Std. Error t value Pr(>|t|)
(Intercept) 615.36248  240.95580   2.554   0.0181 *
height       -8.08324    3.32052  -2.434   0.0235 *
I(height^2)   0.03064    0.01126   2.721   0.0125 *
...
```

Notice that the computed height^2 in the model formula needs to be "protected" by `I(...)`. This technique is often used to prevent any special interpretation of operators in a model formula. Such an interpretation will not take place inside a function call, and `I` is the *identity* function that returns its argument unaltered.

We find a significant deviation from linearity. However, considering the process that led to doing this particular analysis, the *p*-values have to be taken with more than a grain of salt. This is getting dangerously close to "data dredging", fishing expeditions in data. Consider it more an illustration of a technique than an exemplary data analysis.

To draw a plot of the fitted curve with prediction and confidence bands, we can use `predict`. To avoid problems caused by data not being sorted by height, we use `newdata`, which allows the prediction of values for a chosen set of predictors. Here we choose a set of heights between 110 and 180 cm in steps of 2 cm:

```
> pred.frame <- data.frame(height=seq(110,180,2))
> lm.pemax.hq <- lm(pemax~height+I(height^2))
> predict(lm.pemax.hq,interval="pred",newdata=pred.frame)
         fit      lwr      upr
1   96.90026 37.94461 155.8559
2   94.33611 36.82985 151.8424
3   92.01705 35.73077 148.3033
...
34 141.68922 88.70229 194.6761
35 147.21294 93.51117 200.9147
36 152.98174 98.36718 207.5963
```

Based on these predicted data, Figure 12.1 is obtained as follows:

```
> pp <- predict(lm.pemax.hq,newdata=pred.frame,interval="pred")
> pc <- predict(lm.pemax.hq,newdata=pred.frame,interval="conf")
> plot(height,pemax,ylim=c(0,200))
> matlines(pred.frame$height,pp,lty=c(1,2,2),col="black")
> matlines(pred.frame$height,pc,lty=c(1,3,3),col="black")
```

It is seen that the fitted curve is slightly decreasing for small heights. This is probably an artifact caused by the choice of a second-order polynomial to fit data. More likely, the reality is that `pemax` is relatively constant up to about 150 cm, after which it increases quickly with height. Note also that there seems to be a discrepancy between the prediction limits and the actual distribution of data for the smaller heights. The standard deviation might be larger for larger heights, but it is not impossible to obtain a similar distribution of points by coincidence, and there is also an issue with potential overfitting to the observed data. It is really not advisable to construct prediction intervals based on data as limited as these unless you are sure that the model is correct.

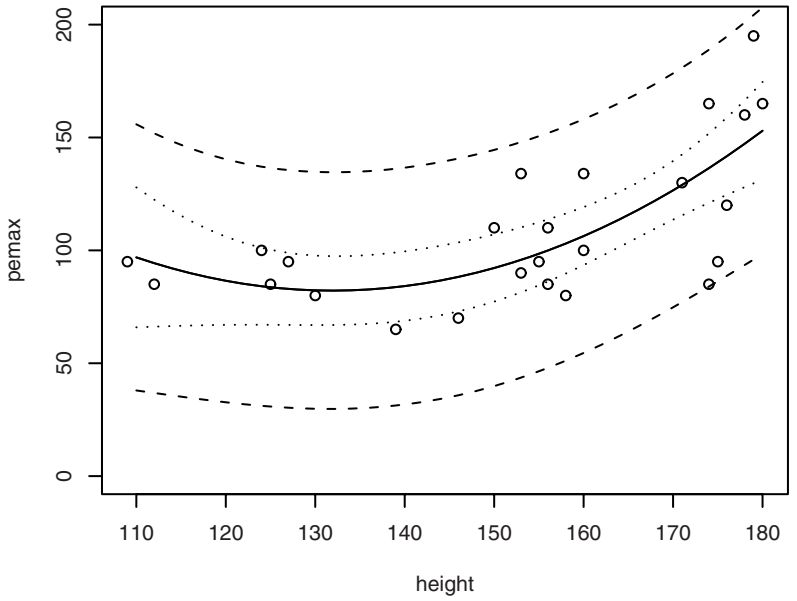

Figure 12.1. Quadratic regression with confidence and prediction limits.

12.2 Regression through the origin

It sometimes makes sense to assume that a regression line passes through $(0, 0)$ — that the intercept of the regression line is zero. This can be specified in the model formula by adding the term `-1` ("minus intercept") to the right-hand side: `y ~ x - 1`.

The logic of the notation can be seen by writing the linear regression model as $y = \alpha \times 1 + \beta \times x + \epsilon$. The intercept corresponds to having an extra descriptive variable, which is the constant 1. Removing this variable yields regression through the origin.

This is a simulated example of a linear relationship through the origin ($y = 2x + \epsilon$):

```
> x <- runif(20)
> y <- 2*x+rnorm(20,0,0.3)
> summary(lm(y~x))

Call:
lm(formula = y ~ x)
```

```
Residuals:
     Min       1Q   Median       3Q      Max
-0.50769 -0.08766  0.03802  0.14512  0.26358

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept) -0.14896    0.08812   -1.69    0.108
x            2.39772    0.15420   15.55 7.05e-12 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.2115 on 18 degrees of freedom
Multiple R-squared: 0.9307,   Adjusted R-squared: 0.9269
F-statistic: 241.8 on 1 and 18 DF,  p-value: 7.047e-12

> summary(lm(y~x-1))

Call:
lm(formula = y ~ x - 1)

Residuals:
     Min       1Q   Median       3Q      Max
-0.62178 -0.16855 -0.04019  0.12044  0.27346

Coefficients:
  Estimate Std. Error t value Pr(>|t|)
x  2.17778    0.08669   25.12 4.87e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.2216 on 19 degrees of freedom
Multiple R-squared: 0.9708,   Adjusted R-squared: 0.9692
F-statistic: 631.1 on 1 and 19 DF,  p-value: 4.873e-16
```

In the first analysis, the intercept is not significant, which is, of course, not surprising. In the second analysis we force the intercept to be zero, resulting in a slope estimate with a substantially improved accuracy.

Comparison of the R^2-values in the two analyses shows something that occasionally causes confusion: R^2 is much larger in the model with no intercept! This does *not*, however, mean that the relation is "more linear" when the intercept is not included or that more of the variation is explained. What is happening is that the definition of R^2 itself is changing. It is most easily seen from the ANOVA tables in the two cases:

```
> anova(lm(y~x))
Analysis of Variance Table

Response: y
          Df Sum Sq Mean Sq F value    Pr(>F)
x          1 10.8134 10.8134  241.80 7.047e-12 ***
Residuals 18  0.8050  0.0447
```

```
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
> anova(lm(y~x-1))
Analysis of Variance Table

Response: y
          Df  Sum Sq Mean Sq F value    Pr(>F)
x          1 30.9804 30.9804  631.06 4.873e-16 ***
Residuals 19  0.9328  0.0491
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Notice that the total sum of squares and the total number of degrees of freedom is not the same in the two analyses. In the model with an intercept, there are 19 DF in all and the total sum of squares is $\sum(y_i - \bar{y})^2$, while the model without an intercept has a total of 20 DF and the total sum of squares is defined as $\sum y_i^2$. Unless $\bar{y}$ is close to zero, the latter "total SS" will be much larger than the former, so if the residual variance is similar, R^2 will be much closer to 1.

The reason for defining the total sum of squares like this for models without intercepts is that it has to correspond to the residual sum of squares in a minimal model. The minimal model has to be a submodel of the regression model; otherwise the ANOVA table simply does not make sense. In an ordinary regression analysis, the minimal model is $y = \alpha + \epsilon$, but when the regression model does not include α, the only sensible minimal model is $y = 0 + \epsilon$.

12.3 Design matrices and dummy variables

The function `model.matrix` gives the *design matrix* for a given model. It can look like this:

```
> model.matrix(pemax~height+weight)
   (Intercept) height weight
1            1    109   13.1
2            1    112   12.9
3            1    124   14.1
4            1    125   16.2
...
24           1    175   51.1
25           1    179   71.5
attr(,"assign")
[1] 0 1 2
```

(The `cystfibr` data set was attached previously.)

You should not worry about the "assign" attribute at this stage, but the three columns are important. If you add them together, weighted by the corresponding regression coefficients, you get exactly the fitted values. Notice that the intercept enters as the coefficient to a column of ones.

If the same is attempted for a model containing a factor, the following happens. (We return to the anesthetic ventilation example from p. 129.)

```
> attach(red.cell.folate)
> model.matrix(folate~ventilation)
   (Intercept) ventilationN2O+O2,op ventilationO2,24h
1            1                    0                 0
2            1                    0                 0
...
16           1                    1                 0
17           1                    1                 0
18           1                    0                 1
19           1                    0                 1
20           1                    0                 1
21           1                    0                 1
22           1                    0                 1
attr(,"assign")
[1] 0 1 1
attr(,"contrasts")
attr(,"contrasts")$ventilation
[1] "contr.treatment"
```

The two columns of zeros and ones are sometimes called *dummy variables.* They are interpreted exactly as above: Multiplying them by the respective regression coefficients and adding the results yields the fitted value. Notice that, for example, the second column is 1 for observations in group 2 and 0 otherwise; that is, the corresponding regression coefficient describes something that is added to the intercept for observations in that particular group. Both columns have zeros for observations from the first group, the mean value of which is described by the intercept (β_0) alone. The regression coefficient β_1 thus describes the *difference* in means between groups 1 and 2, and β_2 between groups 1 and 3.

You may be confused by the use of the term "regression coefficients" even though no regression lines are present in models like that above. The point is that you *formally* rewrite a model for groups as a multiple regression model, so that you can use the same software. As is seen, there is a unique correspondence between the formal regression coefficients and the group means.

You can define dummy variables in several different ways to describe a grouping. This particular scheme is called *treatment contrasts* because if the first group is "no treatment" then the coefficients immediately give the treatment effects for each of the other groups. We do not discuss other

choices here; see Venables and Ripley (2002) for a much deeper discussion. Note only that contrast type can be set on a per-term basis and that this is what is reflected in the `"contrasts"` attribute of the design matrix.

For completeness, the `"assign"` attribute indicates which columns belong together. When, for instance, you request an analysis of variance using `anova`, the sum of squares for `ventilation` will have 2 degrees of freedom, corresponding to the removal of both columns simultaneously.

Removing the intercept from a model containing a factor term will not correspond to a model in which a particular group has mean zero since such models are usually nonsensical. Instead, R generates a simpler set of dummy variables, which are indicator variables of the levels of the factor. This corresponds to the same model as when the intercept is included (the fitted values are identical), but the regression coefficients have a different interpretation.

12.4 Linearity over groups

Sometimes data are grouped according to a division of a continuous scale (e.g., by age group), or an experiment was designed to take several measurements at each of a fixed set of x-values. In both cases it is relevant to compare the results of a linear regression with those of an analysis of variance.

In the case of grouped x-values, you might take a central value as representative for everyone in a given group, for instance formally letting everyone in a "20–29-year" category be 25 years old. If individual x-values are available, they may of course be used in a linear regression, but it makes the analysis a little more complicated, so we discuss only the situation where that is not the case.

We thus have two alternative models for the same data. Both belong to the class of linear models that `lm` is capable of handling. The linear regression model is a *submodel* of the model for one-way analysis of variance because the former can be obtained by placing restrictions on the parameters of the latter (namely that the true group means lie on a straight line).

It is possible to test whether or not a model reduction is allowable by comparing the reduction in the amount of variation explained to the residual variation in the larger model, resulting in an F test.

In the following example on trypsin concentrations in age groups (Altman, 1991, p. 212), data are given as the mean and SD within each of six groups. This is a kind of data that R is not quite prepared to handle, and it

has therefore been necessary to create "fake" data giving the same means and SDs. These can be obtained via

```
> attach(fake.trypsin)
```

The actual results of the analysis of variance depend only on the means and SDs and are therefore independent of the faking. Readers interested in how to perform the actual faking should take a look at the file `fake.trypsin.R` in the `rawdata` directory of the `ISwR` package.

The `fake.trypsin` data frame contains three variables, as seen by

```
> summary(fake.trypsin)
    trypsin            grp        grpf
 Min.   :-39.96   Min.   :1.000   1: 32
 1st Qu.:119.52   1st Qu.:2.000   2:137
 Median :167.59   Median :2.000   3: 38
 Mean   :168.68   Mean   :2.583   4: 44
 3rd Qu.:213.98   3rd Qu.:3.000   5: 16
 Max.   :390.13   Max.   :6.000   6:  4
```

Notice that there are both `grp`, which is a numeric vector, and `grpf`, which is a factor with six levels.

Performing a one-way analysis of variance on the fake data gives the following ANOVA table:

```
> anova(lm(trypsin~grpf))
Analysis of Variance Table

Response: trypsin
           Df Sum Sq Mean Sq F value    Pr(>F)
grpf        5 224103   44821  13.508 9.592e-12 ***
Residuals 265 879272    3318
```

If you had used `grp` instead of `grpf` in the model formula, you would have obtained a linear regression on the group number instead. In some circumstances, that would have been a serious error, but here it actually makes sense. The midpoints of the age intervals are equidistant, so the model is equivalent to assuming a linear development with age (the interpretation of the regression coefficient requires some care, though). The ANOVA table looks as follows:

```
> anova(lm(trypsin~grp))
Analysis of Variance Table

Response: trypsin
           Df Sum Sq Mean Sq F value    Pr(>F)
grp         1 206698  206698  62.009 8.451e-14 ***
Residuals 269 896677    3333
```

Notice that the residual mean squares did not change very much, indicating that the two models describe the data nearly equally well. If you want to have a formal test of the simple linear model against the model where there is a separate mean for each group, it can be done easily as follows:

```
> model1 <- lm(trypsin~grp)
> model2 <- lm(trypsin~grpf)
> anova(model1,model2)
Analysis of Variance Table

Model 1: trypsin ~ grp
Model 2: trypsin ~ grpf
  Res.Df    RSS  Df Sum of Sq      F Pr(>F)
1    269 896677
2    265 879272   4     17405 1.3114 0.2661
```

So we see that the model reduction has a nonsignificant p-value and hence that `model2` does not fit data significantly better than `model1`.

This technique works *only* when one model is a submodel of the other, which is the case here since the linear model is defined by a restriction on the group means.

Another way to achieve the same result is to add the two models together formally as follows:

```
> anova(lm(trypsin~grp+grpf))
Analysis of Variance Table

Response: trypsin
           Df Sum Sq Mean Sq F value    Pr(>F)
grp         1 206698  206698 62.2959 7.833e-14 ***
grpf        4  17405    4351  1.3114    0.2661
Residuals 265 879272    3318
```

This model is exactly the same as when only `grpf` was included. However, the ANOVA table now contains a subdivision of the model sum of squares in which the `grpf` line describes the *change* incurred by expanding the model from one to five parameters. The ANOVA table in Altman (1991, p. 213) is different, erroneously.

The plot in Figure 12.2 is made like this:

```
> xbar.trypsin <- tapply(trypsin,grpf,mean)
> stripchart(trypsin~grp, method="jitter",
+    jitter=.1, vertical=T, pch=20)
> lines(1:6,xbar.trypsin,type="b",pch=4,cex=2,lty=2)
> abline(lm(trypsin~grp))
```

The graphical techniques used here are essentially identical to those used for Figure 7.1, so we do not go into further details.

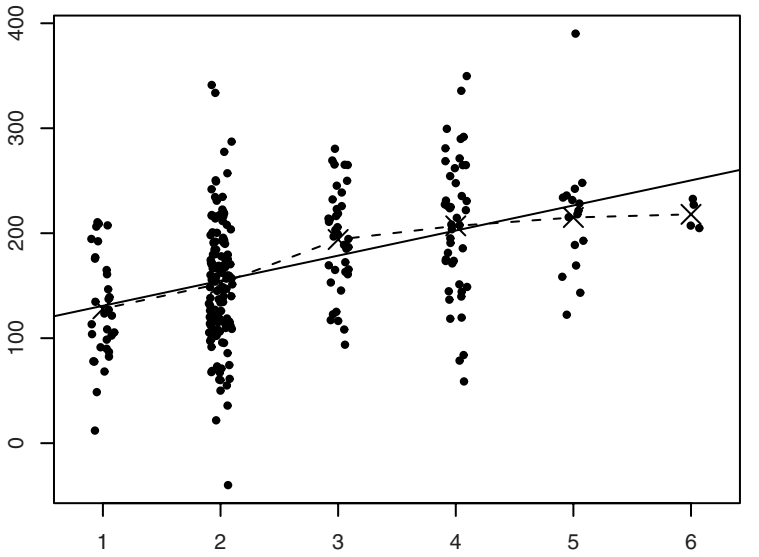

Figure 12.2. "Fake" data for the trypsin example with fitted line and empirical means.

Notice that the fakeness of the data is exposed by a point showing a negative trypsin concentration! The original data are unavailable but would likely show a distribution skewed slightly upwards.

Actually, it *is* possible to analyze the data in R without generating fake data. A weighted regression analysis of the group means, with weights equal to the number of observations in each group, will yield the first two lines of the ANOVA table, and the last one can be computed from the SDs. The details are as follows:

```
> n <- c(32,137, 38,44,16,4)
> tryp.mean <- c(128,152,194,207,215,218)
> tryp.sd <-c(50.9,58.5,49.3,66.3,60,14)
> gr<-1:6
> anova(lm(tryp.mean~gr+factor(gr),weights=n))
Analysis of Variance Table

Response: tryp.mean
           Df Sum Sq Mean Sq F value Pr(>F)
gr          1 206698  206698
factor(gr)  4  17405    4351
Residuals   0      0
```

Notice that the "Residuals" line is zero and that the F tests are not calculated. Omitting the `factor(gr)` term will cause that line to go into Residuals and be treated as an estimate of the error variation, but that is not what you want since it does not include the information about the variation within groups. Instead, you need to fill in the missing information computed from the group standard deviations and sizes. The following gives the residual sum of squares and the corresponding degrees of freedom and mean squares:

```
> sum(tryp.sd^2*(n-1))
[1] 879271.9
> sum(n-1)
[1] 265
> sum(tryp.sd^2*(n-1))/sum(n-1)
[1] 3318.007
```

There is no simple way of updating the ANOVA table with an external variance estimate, but it is easy enough to do the computations directly:

```
> 206698/3318.007 # F statistic for gr
[1] 62.29583
> 1-pf(206698/3318.007,1,265) # p-value
[1] 7.838175e-14
> 4351/3318.007   # F statistic for factor(gr)
[1] 1.311329
> 1-pf(4351/3318.007,4,265) # p-value
[1] 0.2660733
```

12.5 Interactions

A basic assumption in a multiple regression model is that terms act additively on the response. However, this does not mean that linear models cannot describe nonadditivity. You can add special *interaction terms* that specify that the effect of one term is modified according to the level of another. In the model formulas in R, such terms are generated using the colon operator; for example, `a:b`. Usually, you will also include the terms `a` and `b`, and R allows the notation `a*b` for `a+b+a:b`. Higher-order interactions among three or more variables are also possible.

The exact definition of the interaction terms and the interpretation of their associated regression coefficients can be elusive. Some peculiar things happen if an interaction term is present but one or more of the main effects are missing. The full details are probably best revealed through experimentation. However, depending on the nature of the terms `a` and `b` as factors or numeric variables, the overall effect of including interaction terms can be described as follows:

- *Interaction between two factors.* This is conceptually the simplest case. The model with interaction corresponds to having different levels for all possible combinations of levels of the two factors.
- *Interaction between a factor and a numeric variable.* In this case, the model with interaction contains linear effects of the continuous variable but with different slopes within each group defined by the factor.
- *Interaction between two continuous variables.* This gives a slightly peculiar model containing a new regression variable that is the product of the two. The interpretation is that you have a linear effect of varying one variable while keeping the other constant, but with a slope that changes as you vary the other variable.

12.6 Two-way ANOVA with replication

The `coking` data set comes from Johnson (1994, Section 13.1). The time required to make coke from coal is analyzed in a 2×3 experiment varying the oven temperature and the oven width. There were three replications at each combination.

```
> attach(coking)
> anova(lm(time~width*temp))
Analysis of Variance Table

Response: time
           Df  Sum Sq Mean Sq F value    Pr(>F)
width       2 123.143  61.572 222.102 3.312e-10 ***
temp        1  17.209  17.209  62.076 4.394e-06 ***
width:temp  2   5.701   2.851  10.283  0.002504 **
Residuals  12   3.327   0.277
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

We see that the interaction term is significant. If we take a look at the cell means, we can get an idea of why this happens:

```
> tapply(time,list(width,temp),mean)
        1600     1900
4   3.066667 2.300000
8   7.166667 5.533333
12 10.800000 7.333333
```

The difference between high and low temperatures increases with oven width, making an additive model inadequate. When this is the case, the individual tests for the two factors make no sense. If the interaction had

not been significant, then we would have been able to perform separate F tests for the two factors.

12.7 Analysis of covariance

As the example in this section, we use a data set concerning growth conditions of *Tetrahymena* cells, collected by Per Hellung-Larsen. Data are from two groups of cell cultures where glucose was either added or not added to the growth medium. For each culture, the average cell diameter (μ) and cell concentration (count per ml) were recorded. The cell concentration was set at the beginning of the experiment, and there is no systematic difference in cell concentration between the two glucose groups. However, it is expected that the cell diameter is affected by the presence of glucose in the medium.

Data are in the data frame `hellung`, which can be loaded and viewed like this:

```
> hellung
   glucose   conc diameter
1        1 631000     21.2
2        1 592000     21.5
3        1 563000     21.3
4        1 475000     21.0
...
49       2  14000     24.4
50       2  13000     24.3
51       2  11000     24.2
```

The coding of `glucose` is such that 1 and 2 mean yes and no, respectively. There are no missing values.

Summarizing the data frame yields

```
> summary(hellung)
     glucose           conc           diameter
 Min.   :1.000   Min.   : 11000   Min.   :19.20
 1st Qu.:1.000   1st Qu.: 27500   1st Qu.:21.40
 Median :1.000   Median : 69000   Median :23.30
 Mean   :1.373   Mean   :164325   Mean   :23.00
 3rd Qu.:2.000   3rd Qu.:243000   3rd Qu.:24.35
 Max.   :2.000   Max.   :631000   Max.   :26.30
```

Notice that the distribution of the concentrations is strongly right-skewed with a mean more than twice as big as the median. Note also that glucose is regarded as a numeric vector by `summary`, even though it has only two different values.

It will be more convenient to have `glucose` as a factor, so it is recoded as shown below. Recall that to change a variable inside a data frame, you use `$`-notation (p. 21) to specify the component you want to change:

```
> hellung$glucose <- factor(hellung$glucose, labels=c("Yes","No"))
> summary(hellung)
  glucose       conc           diameter
 Yes:32   Min.   : 11000   Min.   :19.20
 No :19   1st Qu.: 27500   1st Qu.:21.40
          Median : 69000   Median :23.30
          Mean   :164325   Mean   :23.00
          3rd Qu.:243000   3rd Qu.:24.35
          Max.   :631000   Max.   :26.30
```

It is convenient to be able to refer to the variables of `hellung` without the `hellung$` prefix, so we put `hellung` in the search path.

```
> attach(hellung)
```

12.7.1 Graphical description

First, we plot the raw data (Figure 12.3):

```
> plot(conc,diameter,pch=as.numeric(glucose))
```

By calculating `as.numeric(glucose)`, we convert the factor `glucose` to the underlying codes, 1 and 2. The specification of `pch` thus implies that group 1 ("Yes") is drawn using plotting character 1 (circles) and group 2 with plotting character 2 (triangles).

To get different plotting symbols, you must first create a vector containing the symbol numbers and give that as the `pch` argument. The following form yields open and filled circles: `c(1,16)[glucose]`. It looks a bit cryptic at first, but it is really just a consequence of R's way of indexing. For indexing purposes, a factor like `glucose` behaves as a vector of 1s and 2s, so you get the first element of `c(1,16)`, namely 1, whenever an observation is from group 1; when the observation is from group 2, you similarly get 16.

The explanatory text is inserted with `legend` like this:

```
> legend(locator(n=1),legend=c("glucose","no glucose"),pch=1:2)
```

Notice that both the function and one of its arguments are named `legend`.

The function `locator` returns the coordinates of a point on a plot. It works so that the function awaits a click with a mouse button and then returns the cursor position. You may want to call `locator()` directly from

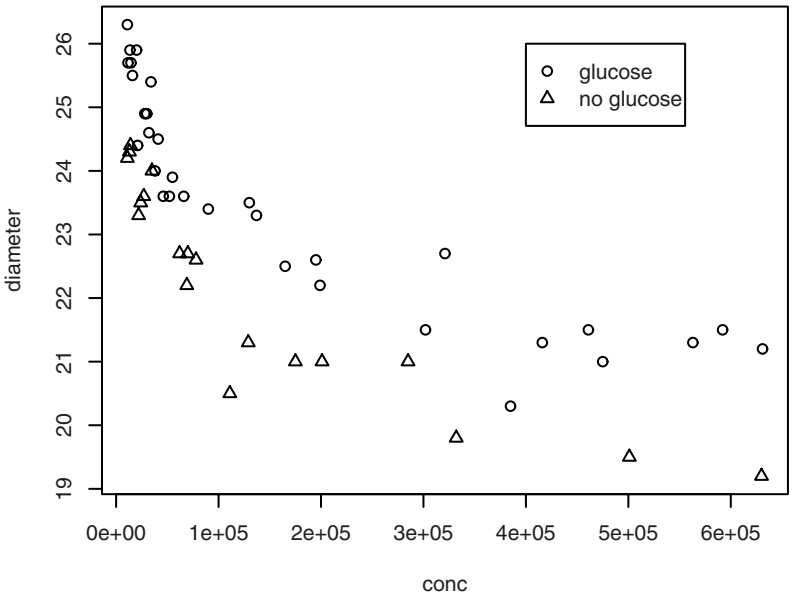

Figure 12.3. Plot of diameter versus concentration for *Tetrahymena* data.

the command line to see the effect. Notice that if you do not specify a value for `n`, then you need to right-click when you are done selecting points.

The plot shows a clear inverse and nonlinear relation between concentration and cell diameter. Further, it is seen that the cultures without glucose are systematically below cultures with added glucose.

You get a much nicer plot (Figure 12.4) by using a logarithmic x-axis:

```
> plot(conc,diameter,pch=as.numeric(glucose),log="x")
```

Now the relation suddenly looks linear!

You could also try a log-log plot (shown in Figure 12.5 with regression lines as described below):

```
> plot(conc,diameter,pch=as.numeric(glucose),log="xy")
```

As is seen, this really does not change much, but it was nevertheless decided to analyze data with both diameter and concentration log-transformed because a power-law relation was expected ($y = \alpha x^{\beta}$, which gives a straight line on a log-log plot).

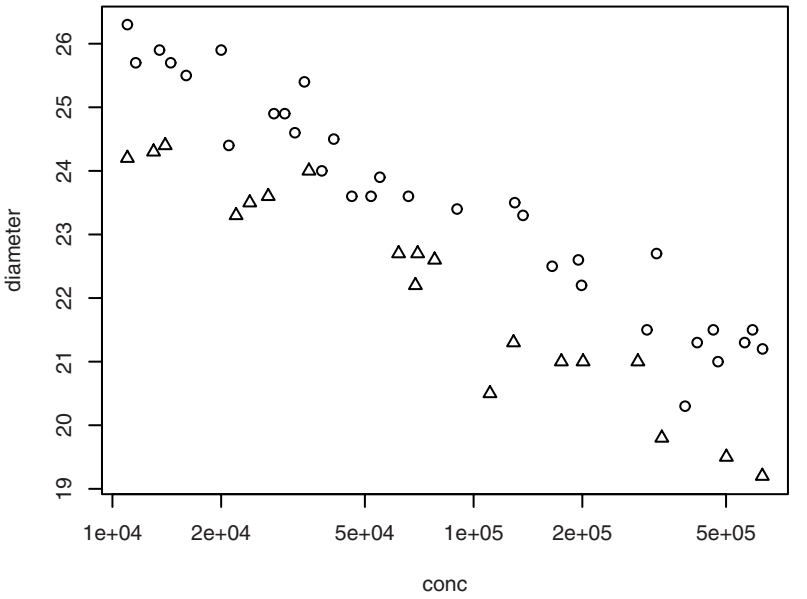

Figure 12.4. *Tetrahymena* data with logarithmic x-axis.

When adding regression lines to a log plot or log-log plot, you should notice that `abline` interprets them as lines in the coordinate system obtained *after* taking (base-10) logarithms. Thus, you can add a line for each group with `abline` applied to the results of a regression analysis of `log10(diameter)` on `log10(conc)`. First, however, it is convenient to define data frames corresponding to the two glucose groups:

```
> tethym.gluc <- hellung[glucose=="Yes",]
> tethym.nogluc <- hellung[glucose=="No",]
```

Notice that you have to use the names, not the numbers, of the factor levels.

Since we only need the two data frames for adding lines to the figure, it would be cumbersome to add them in turn to the search path with `attach`, do the plotting, and then use `detach` to remove them. It is easier to use the `data` argument to `lm`; this allows you to explicitly specify the data frame in which to look for variables. The two regression lines are drawn with

```
> lm.nogluc <- lm(log10(diameter)~ log10(conc),data=tethym.nogluc)
> lm.gluc <- lm(log10(diameter)~ log10(conc),data=tethym.gluc)
```

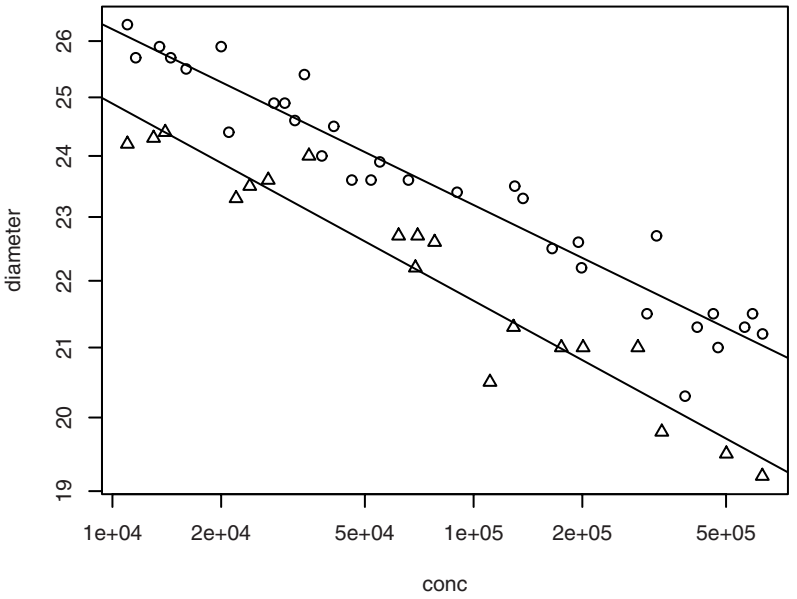

Figure 12.5. *Tetrahymena* data, log-log plot with regression lines.

```
> abline(lm.nogluc)
> abline(lm.gluc)
```

after which the plot looks like Figure 12.5. It is seen that the lines fit the data quite well and that they are almost, but not perfectly, parallel. The question is whether the difference in slope is statistically significant. This is the topic of the next section.

12.7.2 Comparison of regression lines

Corresponding to the two lines from before, we have the following regression analyses:

```
> summary(lm(log10(diameter)~ log10(conc), data=tethym.gluc))

Call:
lm(formula = log10(diameter) ~ log10(conc), data = tethym.gluc)

Residuals:
       Min         1Q     Median         3Q        Max
-0.0267219 -0.0043361  0.0006891  0.0035489  0.0176077
```

```
Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  1.63134    0.01345  121.29   <2e-16 ***
log10(conc) -0.05320    0.00272  -19.56   <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.008779 on 30 degrees of freedom
Multiple R-squared: 0.9273,    Adjusted R-squared: 0.9248
F-statistic: 382.5 on 1 and 30 DF,  p-value: < 2.2e-16

> summary(lm(log10(diameter)~ log10(conc), data=tethym.nogluc))

Call:
lm(formula = log10(diameter) ~ log10(conc), data = tethym.nogluc)

Residuals:
       Min         1Q     Median         3Q        Max
-2.192e-02 -4.977e-03  5.598e-05  5.597e-03  1.663e-02

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept)  1.634761   0.020209   80.89  < 2e-16 ***
log10(conc) -0.059677   0.004125  -14.47 5.48e-11 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.009532 on 17 degrees of freedom
Multiple R-squared: 0.9249,     Adjusted R-squared: 0.9205
F-statistic: 209.3 on 1 and 17 DF,  p-value: 5.482e-11
```

Notice that you can use arithmetic expressions in the model formula [here `log10(...)`]. There are limitations, though, because, for example, `z~x+y` means a model where `z` is described by an additive model in `x` and `y`, which is not the same as a regression analysis on the sum of the two. The latter may be specified using `z~I(x+y)` (`I` for "identity").

A quick assessment of the significance of the difference between the slopes of the two lines can be obtained as follows: The difference between the slope estimates is 0.0065, and the standard error of that is $\sqrt{0.0041^2 + 0.0027^2} = 0.0049$. Since $t = 0.0065/0.0049 = 1.3$, it would seem that we are allowed to assume that the slopes are the same.

It is, however, preferable to fit a model to the entire data set and test the hypothesis of equal slopes in that model. One reason that this approach is preferable is that it can be generalized to more complicated models. Another reason is that even though there is nothing seriously wrong with the simple test for equal slopes, that procedure gives you little information on how to proceed. If the slopes are the same, you would naturally want

to find an estimate of the common slope and the distance between the parallel lines.

First, we set up a model that allows the relation between concentration and cell diameter to have different slopes and intercepts in the two glucose groups:

```
> summary(lm(log10(diameter)~log10(conc)*glucose))

Call:
lm(formula = log10(diameter) ~ log10(conc) * glucose)

Residuals:
       Min         1Q     Median         3Q        Max
-2.672e-02 -4.888e-03  5.598e-05  3.767e-03  1.761e-02

Coefficients:
                       Estimate Std. Error t value Pr(>|t|)
(Intercept)            1.631344   0.013879 117.543   <2e-16 ***
log10(conc)           -0.053196   0.002807 -18.954   <2e-16 ***
glucoseNo              0.003418   0.023695   0.144    0.886
log10(conc):glucoseNo -0.006480   0.004821  -1.344    0.185
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.009059 on 47 degrees of freedom
Multiple R-squared: 0.9361,    Adjusted R-squared: 0.9321
F-statistic: 229.6 on 3 and 47 DF,  p-value: < 2.2e-16
```

These regression coefficients should be read as follows. The expected value of the log cell diameter for an observation with cell concentration C is obtained as the sum of the following four quantities:

1. The intercept, 1.6313
2. $-0.0532 \times \log_{10} C$
3. 0.0034, but only for a culture without glucose
4. $-0.0065 \times \log_{10} C$, but only for cultures without glucose

Accordingly, for cell cultures with glucose, we have the linear relation

$$\log_{10} D = 1.6313 - 0.0532 \times \log_{10} C$$

and for cultures without glucose we have

$$\log_{10} D = (1.6313 + 0.0034) - (0.0532 + 0.0065) \times \log_{10} C$$

Put differently, the first two coefficients in the joint model can be interpreted as the estimates for intercept and slope in group 1, whereas the latter two are the differences between group 1 and group 2 in intercept and slope, respectively. Comparison with the separate regression analyses

shows that slopes and intercepts are the same as in the joint analysis. The standard errors differ a little from the separate analyses because a pooled variance estimate is now used. Notice that the rough test of difference in slope outlined above is essentially the t test for the last coefficient.

Notice also that the `glucose` and `log10(conc).glucose` terms indicate items to be added for cultures *without* glucose. This is because the factor levels are ordered yes = 1 and no = 2, and the base level is the first group.

Fitting an additive model, we get

```
> summary(lm(log10(diameter)~log10(conc)+glucose))
...
Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept)  1.642132   0.011417  143.83  < 2e-16 ***
log10(conc) -0.055393   0.002301  -24.07  < 2e-16 ***
glucoseNo   -0.028238   0.002647  -10.67 2.93e-14 ***
...
```

Here the interpretation of the coefficients is that the estimated relation for cultures with glucose is

$$\log_{10} D = 1.6421 - 0.0554 \times \log_{10} C$$

and for cultures without glucose it is

$$\log_{10} D = (1.6421 - 0.0282) - 0.0554 \times \log_{10} C$$

That is, the lines for the two cultures are parallel, but the log diameters for cultures without glucose are 0.0282 below those with glucose. On the original (nonlogarithmic) scale, this means that the former are 6.3% lower (a constant absolute difference on a logarithmic scale corresponds to constant relative differences on the original scale and $10^{-0.0282} = 0.937$).

The joint analysis presumes that the variance around the regression line is the same in the two groups. This assumption should really have been tested before embarking on the analysis above. A formal test can be performed with `var.test`, which conveniently allows a pair of linear models as arguments instead of a model formula or two group vectors:

```
> var.test(lm.gluc,lm.nogluc)

        F test to compare two variances

data:  lm.gluc and lm.nogluc
F = 0.8482, num df = 30, denom df = 17, p-value = 0.6731
alternative hypothesis: true ratio of variances is not equal to 1
95 percent confidence interval:
 0.3389901 1.9129940
sample estimates:
```

```
ratio of variances
         0.8481674
```

When there are more than two groups, Bartlett's test can be used. It, too, allows linear models to be compared. The reservations about robustness against nonnormality apply here, too.

It is seen that it is possible to assume that the lines have the same slope and that they have the same intercept, but — as we will see below — not both at once. The hypothesis of a common intercept is silly anyway unless the slopes are also identical: The intercept is by definition the y-value at $x = 0$, which because of the log scale corresponds to a cell concentration of 1. That is far outside the region the data cover, and it is a completely arbitrary point that will change if the concentrations are measured in different units.

The ANOVA table for the model is

```
> anova(lm(log10(diameter)~ log10(conc)*glucose))
Analysis of Variance Table

Response: log10(diameter)
                    Df   Sum Sq  Mean Sq F value    Pr(>F)
log10(conc)          1 0.046890 0.046890 571.436 < 2.2e-16 ***
glucose              1 0.009494 0.009494 115.698  2.89e-14 ***
log10(conc):glucose  1 0.000148 0.000148   1.807    0.1853
Residuals           47 0.003857 0.000082
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

The model formula `a*b`, where in the present case `a` is `log10(conc)` and `b` is `glucose`, is a short form for `a + b + a:b`, which is read "effect of `a` plus effect of `b` plus interaction". The F test in the penultimate line of the ANOVA table is a test for the hypothesis that the last term (`a:b`) can be omitted, reducing the model to be additive in `log10(conc)` and `glucose`, which corresponds to the parallel regression lines. The F test one line earlier indicates whether you can *subsequently* remove `glucose` and the one in the first line to whether you can additionally remove `log10(conc)`, leaving an empty model.

Alternatively, you can read the table from top to bottom as adding terms describing more and more of the total sum of squares. To those familiar with the SAS system, this kind of ANOVA table is known as type I sums of squares.

The p-value for `log10(conc):glucose` can be recognized as that of the t test for the coefficient labelled `log10(conc).glucose` in the previous output. The F statistic is exactly the square of t as well. However, this is true only because there are just two groups. Had there been three or more, there would have been several regression coefficients and the F test would

have tested them all against zero simultaneously, just like when all groups are tested equal in a one-way analysis of variance.

Note that the test for removing `log10(conc)` does not make sense because you would have to remove `glucose` first, which is "forbidden" when `glucose` has a highly significant effect. It makes perfectly good sense to test `log10(conc)` *without* removing `glucose` — which corresponds to testing that the two parallel regression lines can be assumed horizontal — but that test is not found in the ANOVA table. You can get the right test by changing the order of terms in the model formula; compare, for instance, these two regression analyses:

```
> anova(lm(log10(diameter)~glucose+log10(conc)))
Analysis of Variance Table

Response: log10(diameter)
            Df   Sum Sq  Mean Sq F value    Pr(>F)
glucose      1 0.008033 0.008033  96.278 4.696e-13 ***
log10(conc)  1 0.048351 0.048351 579.494 < 2.2e-16 ***
Residuals   48 0.004005 0.000083
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
> anova(lm(log10(diameter)~log10(conc)+ glucose))
Analysis of Variance Table

Response: log10(diameter)
            Df   Sum Sq  Mean Sq F value    Pr(>F)
log10(conc)  1 0.046890 0.046890  561.99 < 2.2e-16 ***
glucose      1 0.009494 0.009494  113.78 2.932e-14 ***
Residuals   48 0.004005 0.000083
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

They both describe exactly the same model, as is indicated by the residual sum of squares being identical. The partitioning of the sum of squares is *not* the same, though — and the difference may be much more dramatic than it is here. The difference is whether `log10(conc)` is added to a model already containing `glucose` or vice versa. Since the second F test in both tables is highly significant, no model reduction is possible and the F test in the line above it is irrelevant.

If you go back and look at the regression coefficients in the model with parallel regression lines, you will see that the squares of the t tests are 579.49 and 113.8, precisely the last F test in the two tables above.

It is informative to compare the covariance analysis above with the simpler analysis in which the effect of cell concentration is ignored:

```
> t.test(log10(diameter)~glucose)
```

```
        Welch Two Sample t-test

data:  log10(diameter) by glucose
t = 2.7037, df = 36.31, p-value = 0.01037
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
 0.006492194 0.045424241
sample estimates:
mean in group Yes  mean in group No
         1.370046          1.344088
```

Notice that the p-value is much less extreme. It is still significant in this case, but in smaller data sets the statistical significance could easily disappear completely. The difference in means between the two groups is 0.026, which is comparable to the 0.028 that was the glucose effect in the analysis of covariance. However, the confidence interval goes from 0.006 to 0.045, whereas the analysis of covariance had 0.023 to 0.034 [$0.0282 \pm t_{.975}(48) \times 0.0026$], which is almost four times as narrow, obviously a substantial gain in efficiency.

12.8 Diagnostics

Regression diagnostics are used to evaluate the model assumptions and investigate whether or not there are observations with a large influence on the analysis. A basic set of these is available via the plot method for `lm` objects. Four of them are displayed in a 2×2 layout (Figure 12.6) as follows:

```
> attach(thuesen)
> options(na.action="na.exclude")
> lm.velo <- lm(short.velocity~blood.glucose)
> opar <- par(mfrow=c(2,2), mex=0.6, mar=c(4,4,3,2)+.3)
> plot(lm.velo, which=1:4)
> par(opar)
```

The `par` commands set up for a 2×2 layout with compressed margin texts and go back to normal after plotting.

The top left panel shows residuals versus fitted values. The top right panel is a Q–Q normal distribution plot of standardized residuals. Notice that there are residuals and standardized residuals; the latter have been corrected for differences in the SD of residuals depending on their position in the design. (Residuals corresponding to extreme x-values generally have a lower SD due to overfitting.) The third plot is of the square root of the absolute value of the standardized residuals; this reduces the skewness of the distribution and makes it much easier to detect if there might be a

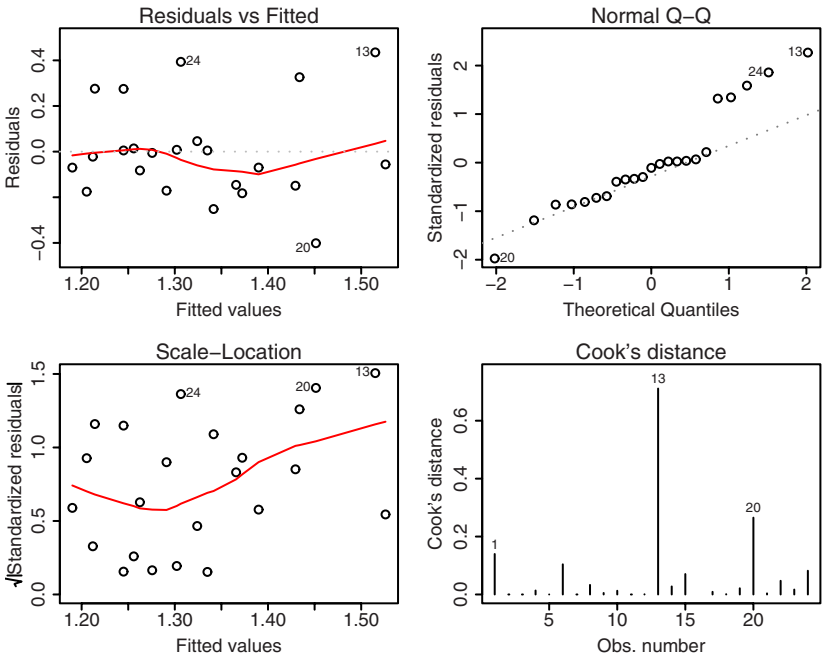

Figure 12.6. Regression diagnostics.

trend in the dispersion. The fourth plot is of "Cook's distance", which is a measure of the influence of each observation on the regression coefficients. We will return to Cook's distance shortly. Actually, this is not the default set of plots; the default replaces the fourth plot by a plot that contains the two components that enter into the calculation of Cook's distance, but this is harder to explain at this level.

The plots for the `thuesen` data show observation no. 13 as extreme in several respects. It has the largest residual as well as a prominent spike in the Cook's distance plot. Observation no. 20 also has a large residual, but not quite as conspicuous a Cook's distance.

```
> opar <- par(mfrow=c(2,2), mex=0.6, mar=c(4,4,3,2)+.3)
> plot(rstandard(lm.velo))
> plot(rstudent(lm.velo))
> plot(dffits(lm.velo),type="l")
> matplot(dfbetas(lm.velo),type="l", col="black")
> lines(sqrt(cooks.distance(lm.velo)), lwd=2)
> par(opar)
```

It is also possible to obtain individual diagnostics; a selection is shown in Figure 12.7. The function `rstandard` gives the standardized residuals

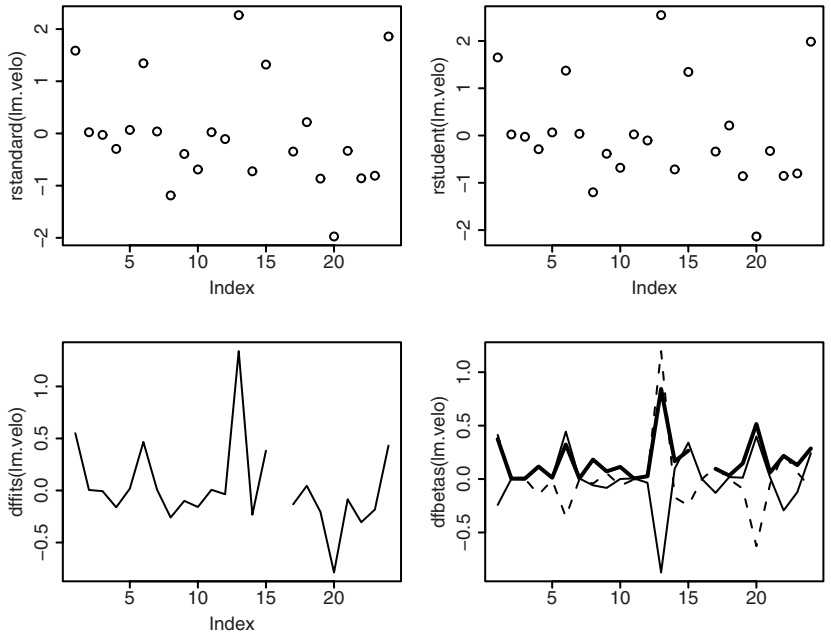

Figure 12.7. Further regression diagnostics.

discussed above. There is also `rstudent`, which gives *leave-out-one residuals*, in which the fitted value is calculated omitting the current point; if the model is correct, then these will follow a (Student's) t distribution. (Unfortunately, some texts use "studentized residuals" for residuals divided by their standard deviation; i.e., what `rstandard` calculates in R.) It takes a keen eye to see the difference between the two types of residuals, but the extreme residuals tend to be a little further out in the case of `rstudent`.

The function `dffits` expresses how much an observation affects the associated fitted value. As with the residuals, observations 13 and maybe 20 seem to stick out. Notice that there is a gap in the line. This is due to the missing observation 16 and the use of `na.exclude`. This looks a little awkward but has the advantage of making the x-axis match the observation number.

The function `dfbetas` gives the change in the estimated parameters if an observation is excluded relative to its standard error. It is a matrix, so `matplot` is useful to plot them all in one plot. Notice that observation 13 affects both α (the solid line) and β by nearly one standard error.

The name `dfbetas` refers to its use in multiple regression analysis, where you write the model as $y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \cdots$. This gets a little con-

fusing in a simple regression analysis, where the intercept is otherwise called α.

Cook's distance D calculated by `cooks.distance` is essentially a joint measure of the components of `dfbetas`. The exact procedure is to take the *unnormalized* change in coefficients and use the norm defined by the estimated covariance matrix for $\hat{\beta}$ and then divide by the number of coefficients. $\sqrt{D}$ is on the same scale as `dfbetas` and was added to that plot as a double-width line. (If you look inside the R functions for some of these quantities, you will find them apparently quite different from the descriptions above, but they are in fact the same, only computationally more efficient.)

Thus, the picture is that observation 13 seems to be influential. Let us look at the analysis without this observation.

We use the `subset` argument to `lm`, which, like other indexing operations, can be used with negative numbers to remove observations.

```
> summary(lm(short.velocity~blood.glucose, subset=-13))

Call:
lm(formula = short.velocity ~ blood.glucose, subset = -13)

Residuals:
     Min       1Q   Median       3Q      Max
-0.31346 -0.11136 -0.01247  0.06043  0.40794

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept)    1.18929    0.11061  10.752 9.22e-10 ***
blood.glucose  0.01082    0.01029   1.052    0.305
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.193 on 20 degrees of freedom
  (1 observation deleted due to missingness)
Multiple R-squared: 0.05241,    Adjusted R-squared: 0.005026
F-statistic: 1.106 on 1 and 20 DF,  p-value: 0.3055
```

The relation practically vanished in thin air! The whole analysis actually hinges on a single observation. If the data and model are valid, then of course the original p-value is correct, and perhaps you could also say that there will always be influential observations in small data sets, but some caution in the interpretation does seem advisable.

The methods for finding influential observations and outliers are even more important in regression analysis with multiple descriptive variables. One of the big problems is how to present the quantities graphically in a sensible way. This might be done using three-dimensional plots (the add-

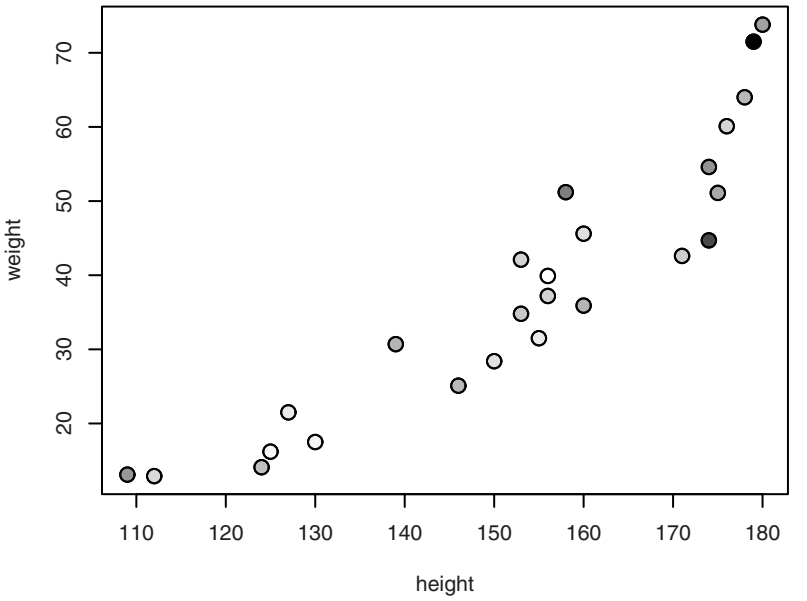

Figure 12.8. Cook's distance (colour coded) in `pemax ~ height + weight`.

on package `scatterplot3d` makes this possible), but you can get quite far using colour coding.

Here, we see how to display the value of Cook's distance (which is always positive) graphically for a model where `pemax` is described using `height` and `weight`, as in Figure 12.8:

```
> cookd <- cooks.distance(lm(pemax~height+weight))
> cookd <- cookd/max(cookd)
> cook.colors <- gray(1-sqrt(cookd))
> plot(height,weight,bg=cook.colors,pch=21,cex=1.5)
> points(height,weight,pch=1,cex=1.5)
```

The first line computes Cook's distance and the second scales it to a value between 0 and 1. Thereafter, a colour coding of the values in `cookd` is made with the function `gray`. The latter interprets its argument as the degree of whiteness, so if you want a large distance represented as black, you need to subtract the value from 1. Furthermore, it is convenient to take the square root of `cookd` because it is a quadratic distance measure (which in practice shows up in the form of too many white or nearly white points). Then a scatterplot of height versus weight is drawn with the cho-

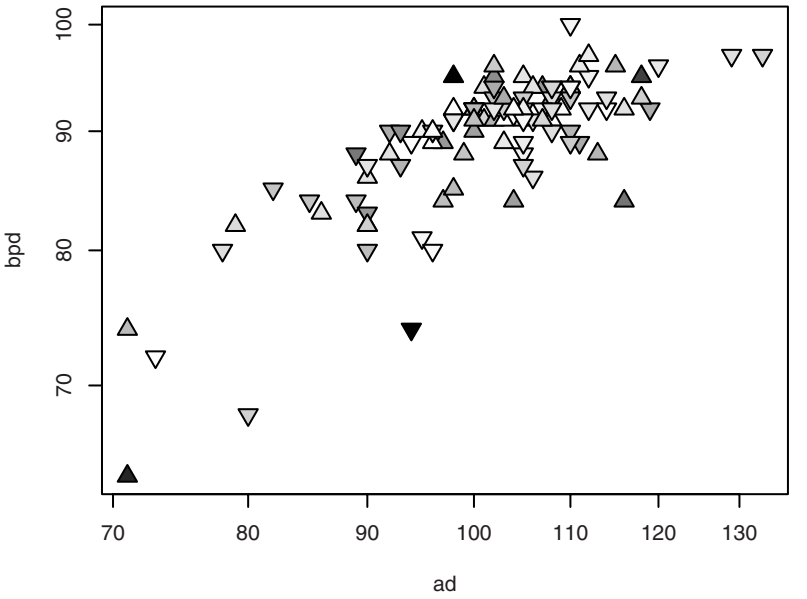

Figure 12.9. Studentized residuals in the Secher data, colour coded. Positive values are marked with upward-pointing triangles; negative ones point down.

sen colours. A filled plotting symbol in enlarged symbol size is used to get the grayscale to stand out more clearly.

You can use similar techniques to describe other influence measures. In the case of signed measures, you might use different symbols for positive and negative values. Here is an example on Studentized residuals in a data set describing birth weight as a function of abdominal and biparietal diameters determined by ultrasonography of the fetus immediately before birth, also used in Exercise 11.1 (Figure 12.9):

```
> attach(secher)
> rst <- rstudent(lm(log10(bwt)~log10(ad)+log10(bpd)))
> range(rst)
[1] -3.707509  3.674050
> rst <- rst/3.71
> plot(ad,bpd,log="xy",bg=gray(1-abs(rst)),
+      pch=ifelse(rst>0,24,25), cex=1.5)
```

12.9 Exercises

12.1 Set up an additive model for the `ashina` data (see Exercise 5.6) containing additive effects of subjects, period, and treatment. Compare the results with those obtained from *t* tests.

12.2 Perform a two-way analysis of variance on the `tb.dilute` data. Modify the model to have a dose effect that is linear in log dose. Compute a confidence interval for the slope. An alternative approach could be to calculate a slope for each animal and perform a test based on them. Compute a confidence interval for the mean slope, and compare it with the preceding result.

12.3 Consider the following definitions:

```
a <- gl(2, 2, 8)
b <- gl(2, 4, 8)
x <- 1:8
y <- c(1:4,8:5)
z <- rnorm(8)
```

Generate the model matrices for models `z ~ a*b`, `z ~ a:b`, etc. Discuss the implications. Carry out the model fits, and notice which models contain singularities.

12.4 (Advanced) In the `secretin` experiment, you may expect to find inter-individual differences not only in the level of glucose but also in the change induced by the injection of secretin. The factor `time.comb` combines time values at 30, 60, and 90 minutes. The factor `time20plus` combines all values from 20 minutes onward. Discuss the differences and relations among the following linear models:

```
attach(secretin)
model1 <- lm(gluc ~ person * time)
model2 <- lm(gluc ~ person + time)
model3 <- lm(gluc ~ person * time20plus + time)
model4 <- lm(gluc ~ person * time20plus + time.comb)
```

12.5 Analyze the blood pressure in the `bp.obese` data set as a function of obesity and gender.

12.6 Analyze the `vitcap2` data set using analysis of covariance. Revisit Exercise 5.2 and compare the conclusions. Try using the `drop1` function with `test="F"` instead of `summary` in this model.

12.7 In the `juul` data set make regression analyses for prepubescent children (Tanner stage 1) of $\sqrt{\texttt{igf1}}$ versus age separately for boys and girls. Compare the two regression lines.

12.8 Try `step` on the `kfm` data and discuss the result. One observation appears to be influential on the diagnostic plot for this model — explain why. What happens if you reduce the model further?

12.9 For the `juul` data, fit a model for `igf1` with interactions between age, sex, and Tanner stage for those under 25 years old. Explain the interpretation of this model. Hint: A plot of the fitted values against age should be helpful. Use diagnostic plots to evaluate possible transformations of the dependent variable: untransformed, log, or square root.