SUMMARY

This chapter rapidly reviews much of classical statistics, discussing the underlying likelihood models for procedures such as ANOVA, linear regression, and generalized linear models. It also gives brief pointers to the built-in procedures in R that implement these standard techniques. This summary connects maximum likelihood approaches with more familiar classical techniques. If you're already familiar with classical techniques, it may help you understood maximum likelihood better. It also provides a starting point for using efficient, "canned" approaches when they are appropriate for your data. It does not, and cannot, provide full coverage of all these topics. For more details, see Dalgaard (2003), Crawley (2005), or Venables and Ripley (2002).

9.1 INTRODUCTION

So far this book has covered maximum likelihood and Bayesian estimation in some detail. In the course of the discussion I have sometimes mentioned that maximum likelihood analyses give answers equivalent to those provided by familiar, "old-fashioned" statistical procedures. For example, the statistical model $Y \sim \text{Normal}(a+bx, \sigma^2)$ — specifying that Y is a normally distributed random variable whose mean depends linearly on x — underlies ordinary least-squares linear regression. This chapter will briefly review special cases where our general recipe for finding MLEs for statistical models reduces to standard procedures that are built into R and other statistics packages.

In the best case, your data will match a classical technique like linear regression exactly, and the answers provided by classical statistical models will agree with the results from your likelihood model. Other models you build may be formally equivalent to a classical model that is parameterized in a different way. Most often, the customized model you build will not be exactly equivalent to any existing classical model, but a similar classical model may be close enough that you wouldn't mind changing your model slightly in order to gain the convenience of using a standard procedure.

For example, in Chapter 6 we used the model

$$Y \sim \text{NegBinom}(\mu = a \cdot \text{DBH}^b, k) \tag{9.1.1}$$

to represent cone production by fir trees as a function of diameter at breast height. If we approximated the discrete distribution of cones by a continuous log-normal distribution instead,

$$Y \sim \text{LogNormal}(\mu = a \cdot \text{DBH}^b, \sigma^2), \tag{9.1.2}$$

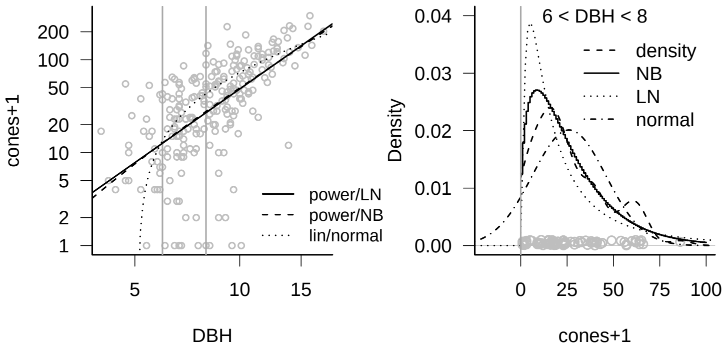

Figure 9.1 Comparing different functional forms for fir fecundity data: power-law with a lognormal (LN) distribution, power-law with a negative binomial (NB) distribution, and linear with a normal distribution. (The linear model appears as a curved line because the data are plotted on a log-log scale.)

we could log-transform both sides and fit the linear regression model

$$\log Y \sim \text{Normal}(\log a + b \cdot \log(\text{DBH}), \sigma^2). \tag{9.1.3}$$

Figure 9.1a shows all three models for the DBH–fecundity relationship — power-law with a negative binomial distribution (power/NB), power-law with a lognormal distribution (power/LN), and linear with a normal distribution — fitted to the fir data; all are plausible. Figure 9.1b shows various models for the distribution of cone production, fitted to the individuals with DBH between 6 and 8 cm: a nonparametric density estimate, the negative binomial, log-normal, and normal. The negative binomial is closest to the nonparametric density estimate of the distribution, while the lognormal is more peaked and the normal distribution has a significant (and unrealistic) negative tail.

Although the power-law/negative binomial is the most realistic and has a plausible mechanistic interpretation (the data are discrete, positive, and overdispersed; we can imagine individual trees producing cones at an approximately constant rate with variation in fecundity among trees), the difference between the fit of negative binomial and lognormal distributions is small enough that the convenience of linear regression may be worthwhile. When the results of different models are similar on both biological and statistical grounds, you choose among them by balancing convenience, mechanistic arguments, and convention.

Why might you want to use standard, special-case procedures rather than the general MLE approach?

- *Computational speed and stability*: the special-case procedures use special-case optimization algorithms that are faster (sometimes much faster) and less likely to encounter numerical problems. Many of these procedures relieve you of the responsibility of choosing starting parameters.
- *Stable definitions*: the definitions of standard models have often been chosen to simplify parameter estimation. For example, to model a relatively sudden change between two states you could choose between a logistic equation or a threshold model. Both might be equally sensible in terms of the biology, but the logistic equation is easier to fit because it involves smoother changes as parameters change. Similarly, generalized linear models such as logistic or Poisson regression fit parameters on scales (logit- or log-transformed, respectively) that allow unconstrained optimization.
- *Convention*: if you use a standard method, you can just say (for example) "we used linear regression" in your Methods section and no-one will think twice. If you use a non-standard method, you need to explain the method carefully and overcome readers' distrust of "fancy" statistics — even if your model is actually simpler and more appropriate than any standard model. Similarly, it may minimize confusion to use the same models, and the same parameterizations, as previous studies of your system.
- *Varying models and comparing hypotheses*: the machinery built into R and other packages makes it easy to compare a variety of models. For example, when analyzing a factorial growth experiment that manipulates nitrogen (`N`) and phosphorus (`P`), you can easily switch between models incorporating the effects of nitrogen only (`growth~N`), phosphorus only (`growth~P`), additive effects of N and P (`growth~N+P`), or the main effects plus interactions between nitrogen and phosphorus (`growth~N*P`). You can carry out all of these comparisons by hand with your own models, and `mle2`'s formula interface is helpful, but R's built-in functions make the process easy for classical models.

This chapter discusses how a variety of different kinds of models fit together, and how they all represent special cases of a general likelihood framework. Figure 9.2 shows how many of these areas are connected. The chapter also gives *brief* descriptions of how to use them in R: if you want more details on any of these approaches, you'll need to check an introductory (Dalgaard, 2003; Crawley, 2005; Verzani, 2005), intermediate (Crawley,

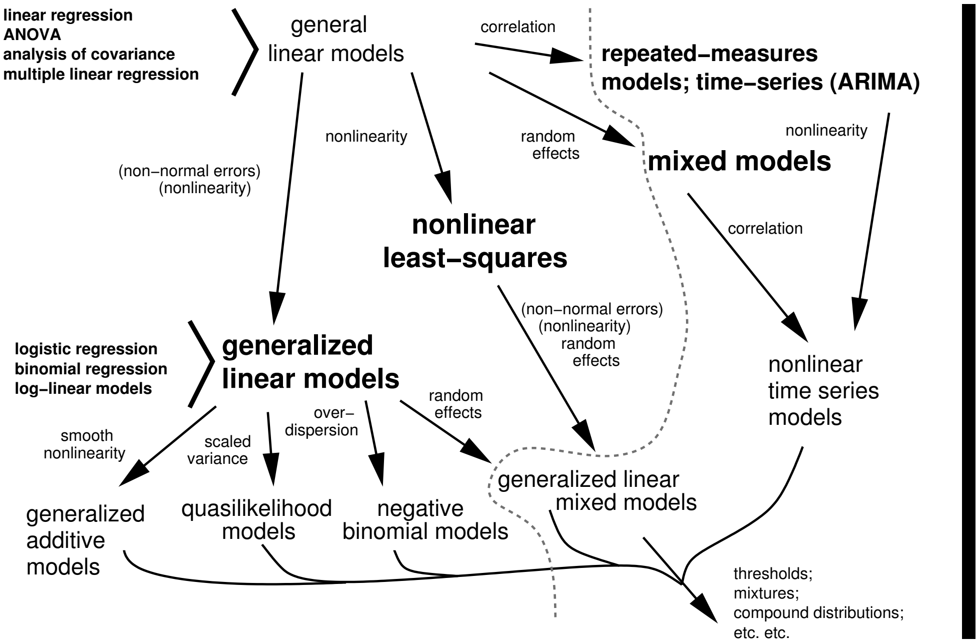

Figure 9.2 All (or most) of statistics. The labels in parentheses (non-normal errors and nonlinearity) imply restricted cases: (non-normal errors) means exponential family (e.g. binomial or Poisson) distributions, while (nonlinearity) means nonlinearities with an invertible linearizing transformation. Models to the right of the gray dashed line involve multiple levels or types of variability; see Chapter 10.

2002), or advanced (Chambers and Hastie, 1992; Venables and Ripley, 2002) reference.

9.2 GENERAL LINEAR MODELS

General linear models include linear regression, one- and multi-way analysis of variance (ANOVA), and analysis of covariance (ANCOVA): R uses the function `lm` for all of these procedures. SAS implements this with PROC GLM*. While regression, ANOVA, and ANCOVA are often handled differently, and they are usually taught differently in introductory statistics classes, they are all variants of the same basic model. The assumptions of the general linear model are that all observed values are independent and normally distributed with a constant variance (*homoscedastic*), and that any continuous predictor variables (covariates) are measured without error. (Remember that the assumption of normality applies to the variation around the expected value — the residuals — not to the whole data set.)

*This terminology is unfortunate since the rest of the world uses "GLM" to mean general*ized* linear models, which correspond to SAS's PROC GENMOD.

The "linear" part of "general linear model" means that the models are linear functions *of the parameters*, not necessarily of the independent variables. For example, quadratic regression

$$Y \sim \text{Normal}(a + bx + cx^2, \sigma^2) \tag{9.2.1}$$

is still linear in the parameters (a, b, c), and thus is a form of multiple linear regression. Another way to think about this is to say that x^2 is just another explanatory variables — if you called it w instead, it would be clear that this model is an example of multivariate linear regression. On the other hand, $Y \sim \text{Normal}(ax^b, \sigma^2)$ is nonlinear: it is linear with respect to a (the second derivative of ax^b with respect to a is zero), but nonlinear with respect to b ($d^2(ax^b)/db^2 = b \cdot (b-1) \cdot ax^{b-2} \neq 0$).

9.2.1 Simple linear regression

Simple, or ordinary, linear regression predicts y as a function of a single continuous covariate x. The model is

$$Y \sim \text{Normal}(a + bx, \sigma^2), \tag{9.2.2}$$

denoting the response variable as Y rather than y since it's a random variable. The equivalent R code is

```
> lm.reg = lm(y ~ x)
```

The intercept term a is implicit in the R model. If you want to force the intercept to be equal to zero, fitting the model $Y \sim \text{Normal}(bx, \sigma^2)$, use `lm(Y~X-1)`.

Typing `lm.reg` by itself prints only the formula and the estimates of the coefficients; `summary(lm.reg)` also gives summary statistics (range and quartiles) of the residuals, standard errors and p-values for the coefficients, and R^2 and F statistics for the full model; `coef(lm.reg)` gives the coefficients alone, and `coef(summary(lm.reg))` pulls out the table of estimates, standard errors, t statistics, and p values. `confint(lm.reg)` calculates confidence intervals. The function `plot(lm.reg)` displays various graphical diagnostics that show how well the assumptions of the model fit and whether particular points have a strong effect on the results: see `?plot.lm` for details. `anova(lm.reg)` prints an ANOVA table for the model*. If you

*`anova` gives so-called *sequential sums of squares*, which SAS calls "type I" sums of squares. If you need SAS-style "type III" sums of squares, you can use the `Anova` function in the `car` package. However, be aware that type III sums of squares are actually problematic, and indeed controversial (Venables, 1998).

need to extract numeric values of, e.g., R^2 values or F statistics for further analysis, wade through the output of `str(summary(lm.reg))` to find the pieces you need (e.g. `summary(lm.reg)$r.squared`).

To do linear regression by brute force with `mle2`, you could write this negative log-likelihood function

```
> linregfun = function(a, b, sigma) {
+     Y.pred = a + b * x
+     -sum(dnorm(Y, mean = Y.pred, sd = sigma, log = TRUE))
+ }
```

or use the formula interface:

```
> mle2(Y ~ dnorm(mean = a + b * x, sd = sigma), start = ...)
```

When using `mle2` you must explicitly fit a standard deviation term σ which is implicit in the `lm` approach.

9.2.2 Multiple linear regression

It's easy to extend the simple linear regression model to multiple continuous predictor variables (covariates). If the extra covariates are powers of the original variable $(x^2, x^3, \ldots)$, the model is called *polynomial* regression (*quadratic* with just the x^2 term added):

$$Y \sim \text{Normal}(a + b_1 x + b_2 x^2, \sigma^2). \tag{9.2.3}$$

Or you can use completely separate variables $(x_1, x_2, \ldots)$:

$$Y \sim \text{Normal}(a + b_1 x_1 + b_2 x_2 + b_3 x_3, \sigma^2) \tag{9.2.4}$$

As with simple regression, the intercept a and the coefficients of the different covariates (b_1, b_2) are implicit in the R formula:

```
> lm.poly = lm(y ~ x + I(x^2))
```

(surround `x^2` and other powers of `x` with `I()`, "as is") or

```
> lm.mreg = lm(y ~ x1 + x2 + x3)
```

You can add interactions among covariates, testing whether the slope with respect to one covariate changes linearly as a function of another covariate — e.g. $Y \sim \text{Normal}(a+b_1x_1+b_2x_2+b_{12}x_1x_2,\sigma^2)$: in R, `lm.intreg = lm(y~x1*x2)`.

Use the `anova` function with `test="Chisq"` to perform likelihood ratio tests on a nested series of multivariate linear regression models (e.g. `anova(lm1,lm2,lm3,test="Chisq")`). If you wonder why `anova` is a test for regression models, remember that regression and analyses of variance are just different subsets of the general linear model.

While multivariate regression is conceptually simple, models with many terms (e.g. models with many covariates or with multi-way interactions) can be difficult to interpret. Blind fitting of models with many covariates can get you in trouble (Whittingham et al., 2006). If you absolutely must go on this kind of fishing expedition, you can use `step`, or `stepAIC` in the `MASS` package to do stepwise modeling, or `regsubsets` in the `leaps` package to search for the best model.

9.2.3 One-way analysis of variance (ANOVA)

If the predictor variables are discrete (factors) rather than continuous (covariate), the general linear model becomes an analysis of variance. The basic model is

$$Y_i \sim \text{Normal}(\alpha_i, \sigma^2); \tag{9.2.5}$$

in R it is

```
> lm.1way = lm(y ~ f)
```

where `f` is a factor. If your original data set has names for the factor levels (e.g. {N,S,E,W} or {high,low}) then R will automatically transform the treatment variable into a factor when it reads in the data. However, if the factor levels look like numbers to R (e.g. you have site designations 101, 227, and 359, or experiments numbered 1 to 5), R will interpret them as continuous rather than discrete predictors, and will fit a linear regression rather than doing an ANOVA — not what you want. Use `v = factor(v)` to turn a numeric variable `v` into a factor, and then fit the linear model.

Executing `anova(lm.1way)` produces a basic ANOVA table; `summary(lm.1way)` gives a different view of the model, testing the significance of each parameter against the null hypothesis that it equals 0; for a factor with only two levels, these tests are statistically identical.

When fitting regression models, the parameters of the model are easy to interpret — they're just the intercept and the slopes with respect to the covariates. When you have factors in the model, however — as in ANOVA — the parameterization becomes trickier. By default, R parameterizes the model in terms of the differences between the first group and subsequent groups (*treatment contrasts*) rather than in terms of the mean of each group, although you can tell it to fit the means of each group by putting a `-1` in the formula (e.g. `lm.1way = lm(y~f-1)`: see pp. 272, 273, and 382).

9.2.4 Multi-way ANOVA

Multi-way ANOVA models Y as a function of two or more different categorical variables (factors). For example, the full model for two-way ANOVA with interactions is

$$Y_{ij} \sim \text{Normal}(\alpha_i + \beta_j + \gamma_{ij}, \sigma^2) \qquad (9.2.6)$$

where i is the level of the first treatment/group, and j is the level of the second. The R code using `lm` is:

```
> lm.2way = lm(Y ~ f1 * f2)
```

(`f1` and `f2` are factors). As before, `summary(lm.2way)` gives more information, testing whether the parameters differ significantly from zero; `confint(lm.2way)` computes confidence intervals; `anova(lm.2way)` generates a standard ANOVA table; `plot(lm.2way)` shows diagnostic plots. If you want to fit just the main effects without the interactions, use `lm(Y~f1+f2)`; use `f1:f2` to specify an interaction between `f1` and `f2`.

A negative log-likelihood function for `mle` could look like this:

```
> aov2fun = function(m11, m12, m21, m22, sigma) {
+     intval = interaction(f1, f2)
+     Y.pred = c(m11, m12, m21, m22)[intval]
+     -sum(dnorm(Y, mean = Y.pred, sd = sigma, log = TRUE))
+ }
```

(`interaction(f1,f2)` defines a factor representing the interaction of `f1` and `f2` with levels in the order (`1.1`, `2.1`, `1.2`, `2.2`)). Using the formula interface:

```
> mle2(Y ~ dnorm(mean = m, sd = sigma), parameters = list(m ~
+     f1 * f2))
```

For a multiway model, R's parameters are again defined in terms of contrasts. If you construct a two-way ANOVA with factors `f1` (with levels `A` and `B`) and `f2` (with levels `I` and `II`), the first ("intercept") parameter will be the mean of individuals in level `A` of the first factor and `I` of the second (`m11`); the second parameter is the difference between `A,II` and `A,I` (`m12-m11`); the third is the difference between `B,I` and `A,I` (`m21-m11`); and the fourth, the interaction term, is the difference between the mean of `B,II` and its expectation if the effects of the two factors were additive (`m22-(m11+(m12-m11)+(m21-m11))` = `m22-m12-m21+m11`).

In its `anova` tables, R One difference between R and other statistical packages to watch

9.2.5 Analysis of covariance (ANCOVA)

Analysis of covariance defines a statistical model that allows for different intercepts and slopes with respect to a covariate x in different groups:

$$Y_i \sim \text{Normal}(\alpha_i + \beta_i x, \sigma^2) \tag{9.2.7}$$

In R:

```
> lm(Y ~ f * x)
```

where f is a factor and x is a covariate (the formula `Y~f+x` would specify parallel slopes, `Y~f` would specify zero slopes but different intercepts, `Y~x` would specify a single slope). Figure 9.3 shows the fit of the model `lm(log(TOTCONES+1) ~ log(DBH)+WAVE_NON)` to the fir data. As suggested by the figure, there is a strong effect of DBH but no significant effect of population (wave vs. non-wave).

As with other general linear models, use `summary`, `confint`, `plot`, and `anova` to analyze the model. The parameters are now the intercept of the first factor level; the slope with respect to `x` for the first factor level; the differences in the intercepts for each factor level other than the first; and the differences in the slopes for each factor level other than the first.

A negative log-likelihood function for ANCOVA:

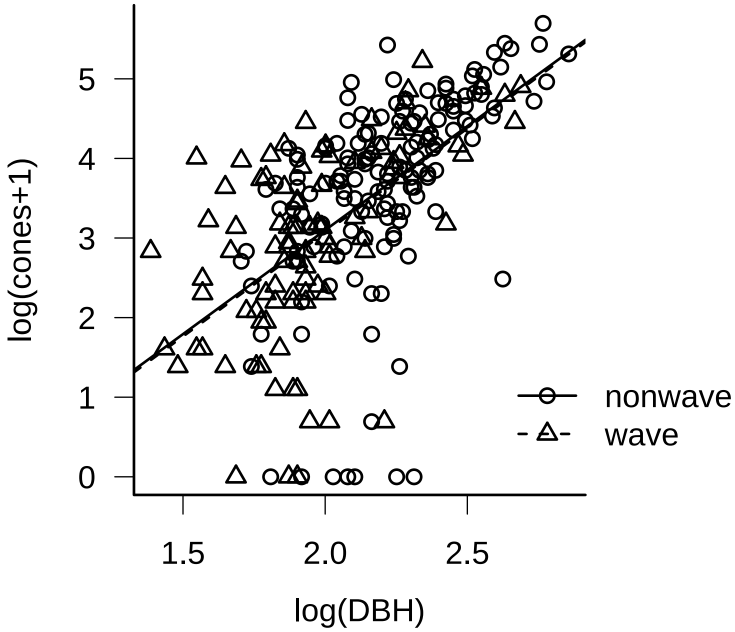

Figure 9.3 General linear model fit to fir fecundity data (analysis of covariance): `lm(log(TOTCONES+1)~log(DBH)+WAVE_NON,data=firdata)`. (Lines are practically indistinguishable between groups.)

```
> ancovafun = function(i1, i2, slope1, slope2, sigma) {
+     int = c(i1, i2)[f]
+     slope = c(slope1, slope2)[f]
+     Y.pred = int + slope * x
+     -sum(dnorm(Y, mean = Y.pred, sd = sigma, log = TRUE))
+ }
```

9.2.6 More complex general linear models

You can add factors (grouping variables) and interactions between factors in different ways to make multi-way ANOVA, covariates (continuous independent variables) to make multiple linear regression, and combinations to make different kinds of analysis of covariance. R will automatically interpret formulas based on whether variables are factors or numeric variables.