16

Proportion Data

An important class of problems involves data on proportions such as:

- studies on percentage mortality,
- infection rates of diseases,
- proportion responding to clinical treatment,
- proportion admitting to particular voting intentions,
- sex ratios, or
- data on proportional response to an experimental treatment.

What all these have in common is that we know how many of the experimental objects are in one category (dead, insolvent, male or infected) and we also know how many are in another (alive, solvent, female or uninfected). This contrasts with Poisson count data, where we knew how many times an event occurred, but *not* how many times it did not occur (p. 527).

We model processes involving proportional response variables in R by specifying a generalized linear model with family=binomial. The only complication is that whereas with Poisson errors we could simply specify family=poisson, with binomial errors we must give the number of failures as well as the numbers of successes in a two-vector response variable. To do this we bind together two vectors using cbind into a single object, *y*, comprising the numbers of successes and the number of failures. The **binomial denominator**, *n*, is the total sample, and

```
number.of.failures = binomial.denominator – number.of.successes
y <- cbind(number.of.successes, number.of.failures)
```

The old fashioned way of modelling this sort of data was to use the percentage mortality as the response variable. There are four problems with this:

- The errors are not normally distributed.
- The variance is not constant.
- The response is bounded (by 1 above and by 0 below).

- By calculating the percentage, we lose information of the size of the sample, n, from which the proportion was estimated.

R carries out weighted regression, using the individual sample sizes as weights, and the logit link function to ensure linearity. There are some kinds of proportion data, such as **percentage cover**, which are best analysed using conventional models (normal errors and constant variance) following **arcsine transformation**. The response variable, y, measured in radians, is $\sin^{-1}\sqrt{0.01 \times p}$, where p is percentage cover. If, however, the response variable takes the form of a **percentage change** in some continuous measurement (such as the percentage change in weight on receiving a particular diet), then rather than arcsine-transforming the data, it is usually better treated by either

- analysis of covariance (see p. 489), using final weight as the response variable and initial weight as a covariate, or
- by specifying the response variable as a relative growth rate, measured as log(final weight/initial weight),

both of which can be analysed with normal errors without further transformation.

Analyses of Data on One and Two Proportions

For comparisons of one binomial proportion with a constant, use binom.test (see p. 300). For comparison of two samples of proportion data, use prop.test (see p. 301). The methods of this chapter are required only for more complex models of proportion data, including regression and contingency tables, where GLMs are used.

Count Data on Proportions

The traditional transformations of proportion data were arcsine and probit. The arcsine transformation took care of the error distribution, while the probit transformation was used to linearize the relationship between percentage mortality and log dose in a bioassay. There is nothing wrong with these transformations, and they are available within R, but a simpler approach is often preferable, and is likely to produce a model that is easier to interpret.

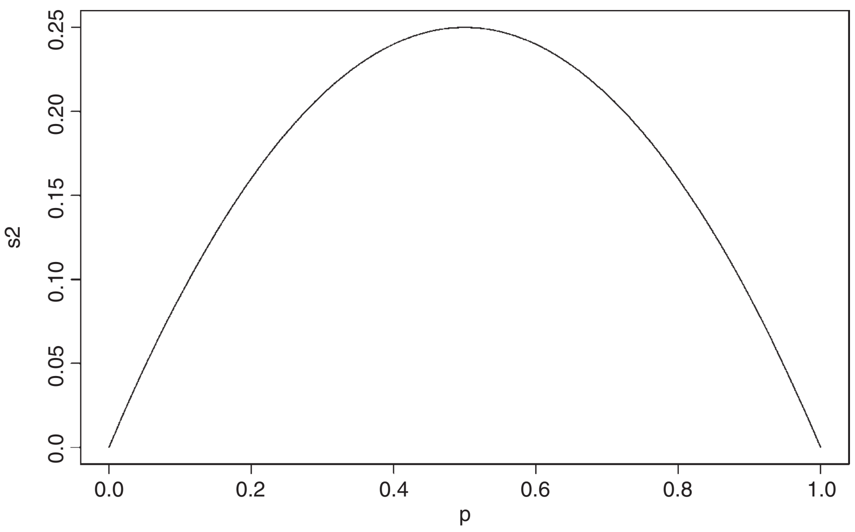
The major difficulty with modelling proportion data is that the responses are *strictly bounded*. There is no way that the percentage dying can be greater than 100% or less than 0%. But if we use simple techniques such as regression or analysis of covariance, then the fitted model could quite easily predict negative values or values greater than 100%, especially if the variance was high and many of the data were close to 0 or close to 100%.

The **logistic** curve is commonly used to describe data on proportions, because, unlike the straight-line model, it asymptotes at 0 and 1 so that negative proportions and responses of more than 100% cannot be predicted. Throughout this discussion we shall use p to describe the proportion of individuals observed to respond in a given way. Because much of their jargon was derived from the theory of gambling, statisticians call these **successes**, although to a demographer measuring death rates this may seem somewhat macabre. The proportion of individuals that respond in other ways (the statistician's **failures**) is therefore $1 - p$, and we shall call this proportion q. The third variable is the size of the sample, n, from which p was estimated (this is the binomial denominator, and the statistician's **number of attempts**).

An important point about the binomial distribution is that the variance is not constant. In fact, the variance of a binomial distribution with mean np is

$$s^2 = npq,$$

so that the variance changes with the mean like this:



The variance is low when p is very high or very low, and the variance is greatest when $p = q = 0.5$. As p gets smaller, so the binomial distribution gets closer and closer to the Poisson distribution. You can see why this is so by considering the formula for the variance of the binomial (above). Remember that for the Poisson, the variance is equal to the mean: $s^2 = np$. Now, as p gets smaller, so q gets closer and closer to 1, so the variance of the binomial converges to the mean:

$$s^2 = npq \approx np \qquad (q \approx 1).$$

Odds

The logistic model for p as a function of x is given by

$$p = \frac{e^{a+bx}}{1 + e^{a+bx}},$$

and there are no prizes for realizing that the model is not linear. But if $x = -\infty$, then $p = 0$, and if $x = +\infty$ then $p = 1$, so the model is strictly bounded. If $x = 0$, then $p = \exp(a)/[1 + \exp(a)]$. The trick of linearizing the logistic model actually involves a very simple transformation. You may have come across the way in which bookmakers specify probabilities by quoting the **odds** against a particular horse winning a race (they might give odds of 2 to 1 on a reasonably good horse or 25 to 1 on an outsider). This is a rather different way of presenting information on probabilities than scientists are used to dealing with. Thus, where the scientist might state a proportion as 0.667 (2 out of 3), the bookmaker would give odds of 2 to 1 (2 successes to 1 failure). In symbols, this is the difference between the scientist stating the probability p, and the bookmaker stating the odds p/q. Now if we take the odds p/q and substitute this into the formula for the logistic, we get

$$\frac{p}{q}=\frac{\mathrm{e}^{a+bx}}{1+\mathrm{e}^{a+bx}}\left[1-\frac{\mathrm{e}^{a+bx}}{1+\mathrm{e}^{a+bx}}\right]^{-1}$$

which looks awful. But a little algebra shows that

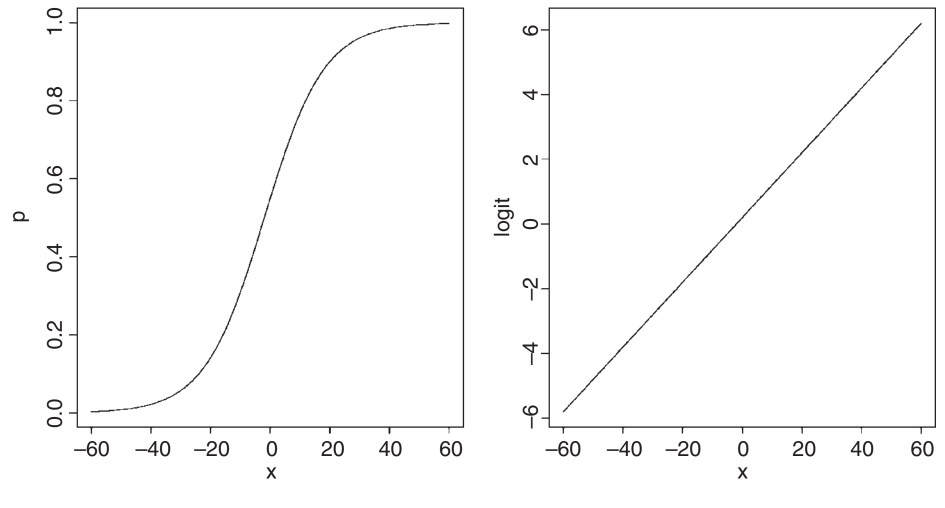
$$\frac{p}{q}=\frac{\mathrm{e}^{a+bx}}{1+\mathrm{e}^{a+bx}}\left[\frac{1}{1+\mathrm{e}^{a+bx}}\right]^{-1}=e^{a+bx}.$$

Now, taking natural logs and recalling that $\ln(e^x)=x$ will simplify matters even further, so that

$$\ln\left(\frac{p}{q}\right)=a+bx.$$

This gives a **linear predictor**, $a+bx$, not for p but for the ***logit*** transformation of p, namely $\ln(p/q)$. In the jargon of R, the logit is the ***link function*** relating the linear predictor to the value of p.

Here are p as a function of x (left panel) and logit(p) as a function of x (right panel) for the logistic model with $a=0.2$ and $b=0.1$:



You might ask at this stage: ‘why not simply do a linear regression of $\ln(p/q)$ against the explanatory x-variable?’ R has three great advantages here:

- It allows for the non-constant binomial variance.
- It deals with the fact that logits for ps near 0 or 1 are infinite.
- It allows for differences between the sample sizes by weighted regression.

Overdispersion and Hypothesis Testing

All the different statistical procedures that we have met in earlier chapters can also be used with data on proportions. Factorial analysis of variance, multiple regression, and a variety of models in which different regression lines are fitted in each of several levels of one or more factors, can be carried out. The only difference is that we assess the significance of terms on the basis of chi-squared – the increase in scaled deviance that results from removal of the term from the current model.

The important point to bear in mind is that hypothesis testing with binomial errors is less clear-cut than with normal errors. While the chi-squared approximation for changes in scaled deviance is reasonable for large samples (i.e. larger than about 30), it is poorer with small samples. Most worrisome is the fact that the degree to which the approximation is satisfactory is itself unknown. This means that considerable care must be exercised in the interpretation of tests of hypotheses on parameters, especially when the parameters are marginally significant or when they explain a very small fraction of the total deviance. With binomial or Poisson errors we cannot hope to provide exact p-values for our tests of hypotheses.

As with Poisson errors, we need to address the question of overdispersion (see p. 522). When we have obtained the minimal adequate model, *the residual scaled deviance should be roughly equal to the residual degrees of freedom*. When the residual deviance is larger than the residual degrees of freedom there are two possibilities: either the model is misspecified, or the probability of success, p, is not constant within a given treatment level. The effect of randomly varying p is to increase the binomial variance from npq to

$$s^2 = npq + n(n-1)\sigma^2,$$

leading to a large residual deviance. This occurs even for models that would fit well if the random variation were correctly specified.

One simple solution is to assume that the variance is not npq but $npq\phi$, where ϕ is an unknown *scale parameter* ($\phi > 1$). We obtain an estimate of the scale parameter by dividing the Pearson chi-squared by the degrees of freedom, and use this estimate of ϕ to compare the resulting scaled deviances. To accomplish this, we use `family = quasibinomial` rather than `family = binomial` when there is overdispersion.

The most important points to emphasize in modelling with binomial errors are as follows:

- Create a two-column object for the response, using `cbind` to join together the two vectors containing the counts of success and failure.
- Check for overdispersion (residual deviance greater than the residual degrees of freedom), and correct for it by using `family=quasibinomial` rather than `binomial` if necessary.
- Remember that you do not obtain exact p-values with binomial errors; the chi-squared approximations are sound for large samples, but small samples may present a problem.
- The fitted values are counts, like the response variable.

- The linear predictor is in logits (the log of the odds $= \ln(p/q)$).
- You can back-transform from logits (z) to proportions (p) by $p = 1/[1 + 1/\exp(z)]$.

Applications

You can do as many kinds of modelling in a GLM as in a linear model. Here we show examples of:

- regression with binomial errors (continuous explanatory variables);
- analysis of deviance with binomial errors (categorical explanatory variables);
- analysis of covariance with binomial errors (both kinds of explanatory variables).

Logistic Regression with Binomial Errors

This example concerns sex ratios in insects (the proportion of all individuals that are males). In the species in question, it has been observed that the sex ratio is highly variable, and an experiment was set up to see whether population density was involved in determining the fraction of males.

```
numbers <-read.table("c:\\temp\\sexratio.txt",header=T)
numbers
```

```
   density  females  males
1        1        1      0
2        4        3      1
3       10        7      3
4       22       18      4
5       55       22     33
6      121       41     80
7      210       52    158
8      444       79    365
```

It certainly looks as if there are proportionally more males at high density, but we should plot the data as proportions to see this more clearly:

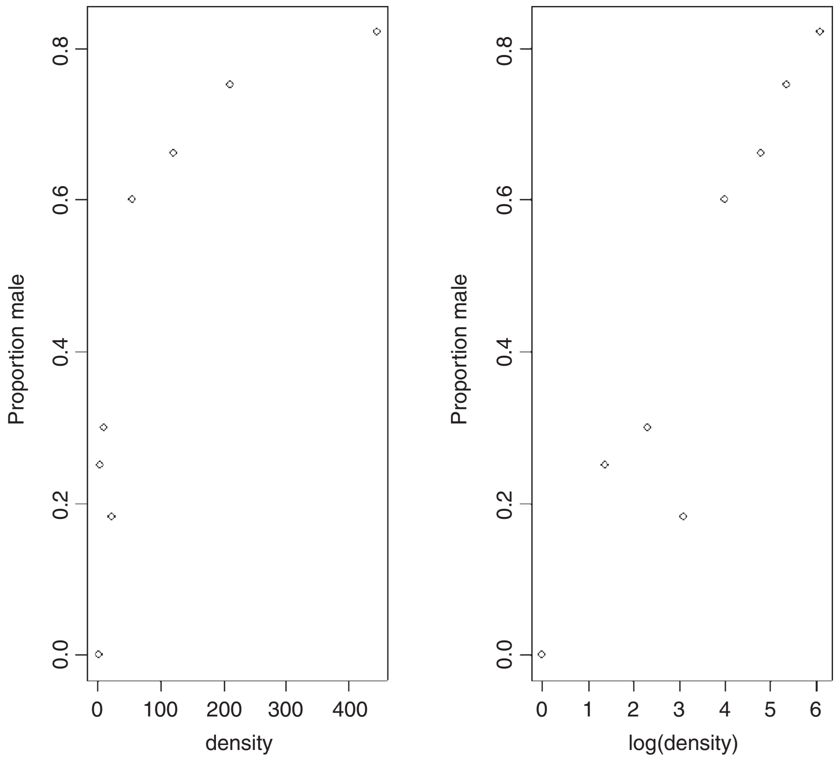
```
attach(numbers)
par(mfrow=c(1,2))
p<-males/(males+females)
plot(density,p,ylab="Proportion male")
plot(log(density),p,ylab="Proportion male")
```

Evidently, a logarithmic transformation of the explanatory variable is likely to improve the model fit. We shall see in a moment.

The question is whether increasing population density leads to a significant increase in the proportion of males in the population – or, more briefly, whether the sex ratio is density-dependent. It certainly looks from the plot as if it is.

The response variable is a matched pair of counts that we wish to analyse as proportion data using a GLM with binomial errors. First, we bind together the vectors of male and female counts into a single object that will be the response in our analysis:

```
y<-cbind(males,females)
```



This means that *y* will be interpreted in the model as the proportion of all individuals that were male. The model is specified like this:

```
model<-glm(y~density,binomial)
```

This says that the object called model gets a generalized linear model in which *y* (the sex ratio) is modelled as a function of a single continuous explanatory variable (called density), using an error distribution from the binomial family. The output looks like this:

```
summary(model)
```

```
Coefficients:
               Estimate  Std. Error  z value  Pr(>| z |)
(Intercept)   0.0807368  0.1550376    0.521      0.603
density       0.0035101  0.0005116    6.862   6.81e-12  ***

    Null deviance: 71.159 on 7 degrees of freedom
Residual deviance: 22.091 on 6 degrees of freedom
AIC: 54.618
```

The model table looks just as it would for a straightforward regression. The first parameter is the intercept and the second is the slope of the graph of sex ratio against population density. The slope is highly significantly steeper than zero (proportionately more males at higher population density; $p = 6.81 \times 10^{-12}$). We can see if log transformation of the explanatory variable reduces the residual deviance below 22.091:

```
model<-glm(y~log(density),binomial)
summary(model)
```

```
Coefficients:
                Estimate  Std. Error  z value  Pr(>|z|)
(Intercept)     -2.65927    0.48758   -5.454   4.92e-08  ***
log(density)     0.69410    0.09056    7.665   1.80e-14  ***

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 71.1593 on 7 degrees of freedom
Residual deviance: 5.6739 on 6 degrees of freedom
AIC: 38.201
```

This is a big improvement, so we shall adopt it. There is a technical point here, too. In a GLM like this, it is assumed that the residual deviance is the same as the residual degrees of freedom. If the residual deviance is larger than the residual degrees of freedom, this is called overdispersion. It means that there is extra, unexplained variation, over and above the binomial variance assumed by the model specification. In the model with log(density) there is no evidence of overdispersion (residual deviance $= 5.67$ on 6 d.f.), whereas the lack of fit introduced by the curvature in our first model caused substantial overdispersion (residual deviance $= 22.09$ on 6 d.f.).

Model checking involves the use of `plot(model)`. As you will see, there is no pattern in the residuals against the fitted values, and the normal plot is reasonably linear. Point no. 4 is highly influential (it has a large Cook's distance), but the model is still significant with this point omitted.

We conclude that the proportion of animals that are males increases significantly with increasing density, and that the logistic model is linearized by logarithmic transformation of the explanatory variable (population density). We finish by drawing the fitted line though the scatterplot:

```
xv<-seq(0,6,0.1)
plot(log(density),p,ylab="Proportion male")
lines(xv,predict(model,list(density=exp(xv)),type="response"))
```

Note the use of `type="response"` to back-transform from the logit scale to the S-shaped proportion scale.

Estimating LD50 and LD90 from bioassay data

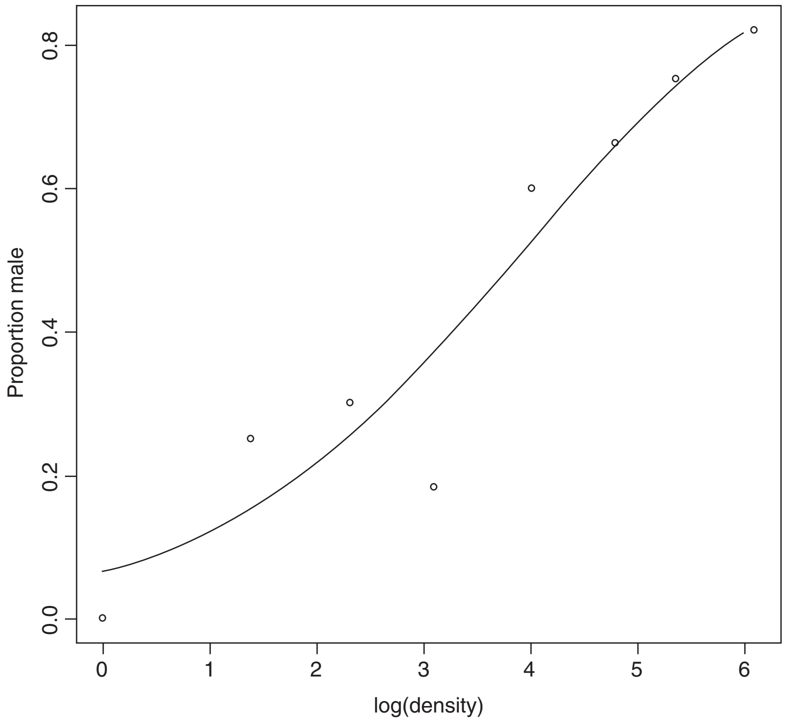
The data consist of numbers dead and initial batch size for five doses of pesticide application, and we wish to know what dose kills 50% of the individuals (or 90% or 95%, as required). The tricky statistical issue is that one is using a value of y (50% dead) to predict a value of x (the relevant dose) and to work out a standard error on the x axis.

```
data<-read.table("c:\\temp\\bioassay.txt",header=T)
attach(data)
names(data)
```

```
[1] "dose"  "dead"  "batch"
```

The logistic regression is carried out in the usual way:

```
y<-cbind(dead,batch-dead)
model<-glm(y~log(dose),binomial)
```



Then the function dose.p from the MASS library is run with the model object, specifying the proportion killed ($p = 0.5$ is the default for LD50):

```
library(MASS)
dose.p(model,p=c(0.5,0.9,0.95))
```

```
              Dose          SE
p = 0.50: 2.306981  0.07772065
p = 0.90: 3.425506  0.12362080
p = 0.95: 3.805885  0.15150043
```

Here the logs of LD50, LD90 and LD95 are printed, along with their standard errors.

Proportion data with categorical explanatory variables

This next example concerns the germination of seeds of two genotypes of the parasitic plant *Orobanche* and two extracts from host plants (bean and cucumber) that were used to stimulate germination. It is a two-way factorial analysis of deviance.

```
germination<-read.table("c:\\temp\\germination.txt",header=T)
attach(germination)
names(germination)
```

```
[1]  "count"  "sample"  "Orobanche"  "extract"
```

Count is the number of seeds that germinated out of a batch of size = sample. So the number that did not germinate is sample – count, and we construct the response vector like this:

```
y<-cbind(count, sample-count)
```

Each of the categorical explanatory variables has two levels:

```
levels(Orobanche)
```

```
[1] "a73" "a75"
```

```
levels(extract)
```

```
[1] "bean" "cucumber"
```

We want to test the hypothesis that there is no interaction between *Orobanche* genotype (a73 or a75) and plant extract (bean or cucumber) on the germination rate of the seeds. This requires a factorial analysis using the asterisk * operator like this:

```
model<-glm(y ~ Orobanche * extract, binomial)
summary(model)
```

```
Coefficients:
                              Estimate Std. Error z value Pr(>|z|)
(Intercept)                    -0.4122     0.1842  -2.238   0.0252 *
Orobanchea75                   -0.1459     0.2232  -0.654   0.5132
extractcucumber                 0.5401     0.2498   2.162   0.0306 *
Orobanchea75:extractcucumber    0.7781     0.3064   2.539   0.0111 *
(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 98.719 on 20 degrees of freedom
 Residual deviance: 33.278 on 17 degrees of freedom
 AIC: 117.87
```

At first glance, it looks as if there is a highly significant interaction ($p = 0.0111$). But we need to check that the model is sound. The first thing is to check for is overdispersion. The residual deviance is 33.278 on 17 d.f. so the model is quite badly overdispersed:

```
33.279 / 17
```

```
[1] 1.957588
```

The overdispersion factor is almost 2. The simplest way to take this into account is to use what is called an 'empirical scale parameter' to reflect the fact that the errors are not binomial as we assumed, but were larger than this (overdispersed) by a factor of 1.9576. We refit the model using quasibinomial to account for the overdispersion:

```
model<-glm(y ~ Orobanche * extract, quasibinomial)
```

Then we use update to remove the interaction term in the normal way:

```
model2<-update(model, ~ . - Orobanche:extract)
```

The only difference is that we use an *F* test instead of a chi-squared test to compare the original and simplified models:

```
anova(model,model2,test="F")
```

```
Analysis of Deviance Table

Model 1: y ~ Orobanche * extract
Model 2: y ~ Orobanche + extract

   Resid. Df   Resid. Dev   Df   Deviance       F    Pr(>F)
1         17       33.278
2         18       39.686   −1     −6.408  3.4418  0.08099  .
```

Now you see that the interaction is not significant ($p = 0.081$). There is no compelling evidence that different genotypes of *Orobanche* respond differently to the two plant extracts. The next step is to see if any further model simplification is possible.

```
anova(model2,test="F")
```

```
Analysis of Deviance Table

Model: quasibinomial, link: logit
Response: y

          Df  Deviance  Resid. Df  Resid. Dev        F     Pr(>F)
NULL                           20      98.719
Orobanche  1     2.544         19      96.175   1.1954     0.2887
extract    1    56.489         18      39.686  26.5412  6.692e-05  ***
```

There is a highly significant difference between the two plant extracts on germination rate, but it is not obvious that we need to keep *Orobanche* genotype in the model. We try removing it:

```
model3<-update(model2, ~ . - Orobanche)
anova(model2,model3,test="F")
```

```
Analysis of Deviance Table

Model 1: y ~ Orobanche + extract
Model 2: y ~ extract
   Resid. Df   Resid. Dev   Df   Deviance       F   Pr(>F)
1         18       39.686
2         19       42.751   −1     −3.065  1.4401  0.2457
```

There is no justification for retaining *Orobanche* in the model. So the minimal adequate model contains just two parameters:

```
coef(model3)
```

```
(Intercept)    extract
−0.5121761  1.0574031
```

What, exactly, do these two numbers mean? Remember that the coefficients are from the linear predictor. They are on the transformed scale, so because we are using binomial errors, they are in logits ($\ln(p/(1-p))$). To turn them into the germination rates for the two plant extracts requires a little calculation. To go from a logit x to a proportion p, you need to do the following sum

$$p = \frac{1}{1 + 1/e^x}.$$

So our first x value is -0.5122 and we calculate

```
1/(1+1/(exp(−0.5122)))
```

```
[1] 0.3746779
```

This says that the mean germination rate of the seeds with the first plant extract was 37%. What about the parameter for extract (1.0574). Remember that with categorical explanatory variables the parameter values are differences between means. So to get the second germination rate we add 1.057 to the intercept before back-transforming:

```
1/(1+1/(exp(−0.5122+1.0574)))
```

```
[1] 0.6330212
```

This says that the germination rate was nearly twice as great (63%) with the second plant extract (cucumber). Obviously we want to generalize this process, and also to speed up the calculations of the estimated mean proportions. We can use predict to help here, because type="response" makes predictions on the back-transformed scale automatically:

```
tapply(predict(model3,type="response"),extract,mean)
```

```
     bean  cucumber
0.3746835 0.6330275
```

It is interesting to compare these figures with the averages of the raw proportions. First we need to calculate the proportion germinating, p, in each sample:

```
p<-count/sample
```

Then we can find the average germination rates for each extract:

```
tapply(p,extract,mean)
```

```
     bean  cucumber
0.3487189 0.6031824
```

You see that this gives different answers. Not too different in this case, it's true, but different none the less. The correct way to average proportion data is to add up the total counts for the different levels of abstract, and only then to turn them into proportions:

```
tapply(count,extract,sum)
```

```
bean  cucumber
 148       276
```

This means that 148 seeds germinated with bean extract and 276 with cucumber. But how many seeds were involved in each case?

```
tapply(sample,extract,sum)
```

```
bean  cucumber
 395       436
```

This means that 395 seeds were treated with bean extract and 436 seeds were treated with cucumber. So the answers we want are 148/395 and 276/436 (i.e. the correct mean proportions). We automate the calculation like this:

```
as.vector(tapply(count,extract,sum))/as.vector(tapply(sample,extract,sum))
```

```
[1] 0.3746835 0.6330275
```

These are the correct mean proportions that were produced by the GLM. The moral here is that you calculate the average of proportions by using total counts and total samples and not by averaging the raw proportions.

To summarize this analysis:

- Make a two-column response vector containing the successes and failures.
- Use glm with family=binomial (you can omit family=).
- Fit the maximal model (in this case it had four parameters).
- Test for overdispersion.
- If, as here, you find overdispersion then use quasibinomial rather than binomial errors.
- Begin model simplification by removing the interaction term.
- This was non-significant once we had adjusted for overdispersion.
- Try removing main effects (we didn't need *Orobanche* genotype in the model).
- Use plot to obtain your model-checking diagnostics.
- Back-transform using predict with the option type="response" to obtain means.

Analysis of covariance with binomial data

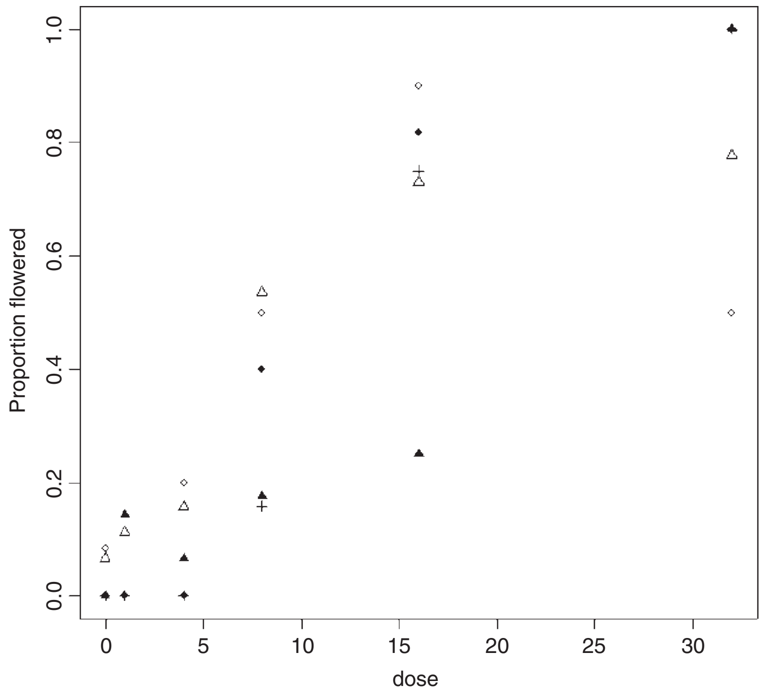
We now turn to an example concerning flowering in five varieties of perennial plants. Replicated individuals in a fully randomized design were sprayed with one of six doses of a controlled mixture of growth promoters. After 6 weeks, plants were scored as flowering or not flowering. The count of flowering individuals forms the response variable. This is an ANCOVA because we have both continuous (dose) and categorical (variety) explanatory variables. We use logistic regression because the response variable is a count (flowered) that can be expressed as a proportion (flowered/number).

```
props<-read.table("c:\\temp\\flowering.txt",header=T)
attach(props)
names(props)
```

```
[1] "flowered" "number" "dose" "variety"
```

```
y<-cbind(flowered,number-flowered)
pf<-flowered/number
pfc<-split(pf,variety)
dc<-split(dose,variety)
```

```
plot(dose,pf,type="n",ylab="Proportion flowered")
points(dc[[1]],pfc[[1]],pch=16)
points(dc[[2]],pfc[[2]],pch=1)
points(dc[[3]],pfc[[3]],pch=17)
points(dc[[4]],pfc[[4]],pch=2)
points(dc[[5]],pfc[[5]],pch=3)
```



There is clearly a substantial difference between the plant varieties in their response to the flowering stimulant. The modelling proceeds in the normal way. We begin by fitting the maximal model with different slopes and intercepts for each variety (estimating ten parameters in all):

```
model1<-glm(y~dose*variety,binomial)
summary(model1)
```

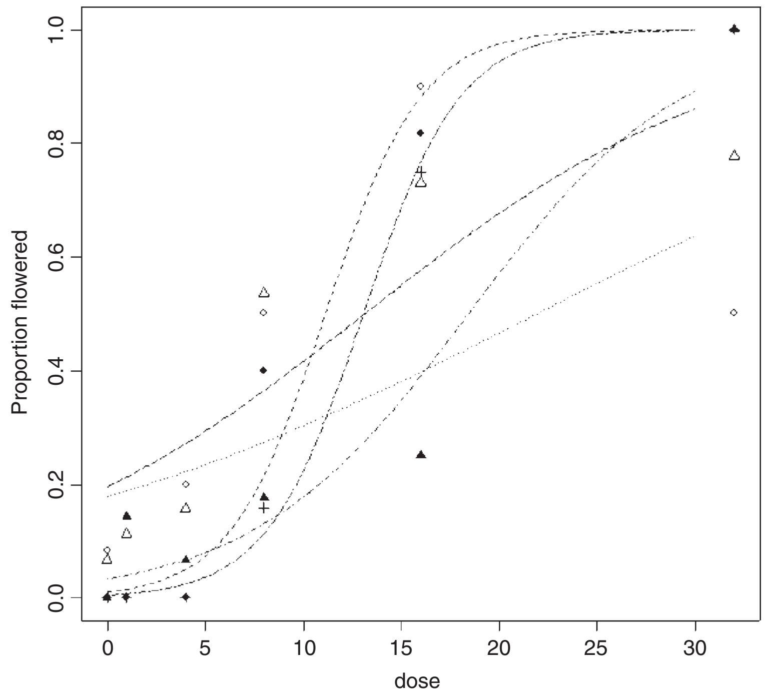
```
Coefficients:
                 Estimate  Std. Error  z value  Pr(>|z|)
(Intercept)     -4.591189    1.021236   -4.496  6.93e-06   ***
dose             0.412564    0.099107    4.163  3.14e-05   ***
varietyB         3.061504    1.082866    2.827  0.004695   **
varietyC         1.232022    1.178527    1.045  0.295842
varietyD         3.174594    1.064689    2.982  0.002866   **
varietyE        -0.715041    1.537320   -0.465  0.641844
dose:varietyB   -0.342767    0.101188   -3.387  0.000706   ***
dose:varietyC   -0.230334    0.105826   -2.177  0.029515   *
dose:varietyD   -0.304762    0.101374   -3.006  0.002644   **
dose:varietyE   -0.006443    0.131786   -0.049  0.961006

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 303.350 on 29 degrees of freedom
Residual deviance: 51.083 on 20 degrees of freedom
AIC: 123.55
```

The model exhibits substantial overdispersion, but this is probably due to poor model selection rather than extra, unmeasured variability. Here is the mean proportion flowered at each dose for each variety:

```
p<-flowered/number
tapply(p,list(dose,variety),mean)
```

```
          A          B          C          D          E
0   0.0000000 0.08333333 0.00000000 0.06666667 0.0000000
1   0.0000000 0.00000000 0.14285714 0.11111111 0.0000000
4   0.0000000 0.20000000 0.06666667 0.15789474 0.0000000
8   0.4000000 0.50000000 0.17647059 0.53571429 0.1578947
16  0.8181818 0.90000000 0.25000000 0.73076923 0.7500000
32  1.0000000 0.50000000 1.00000000 0.77777778 1.0000000
```

There are several ways to plot the five different curves on the scatterplot, but perhaps the simplest is to fit the regression model separately for each variety (see the book's website):



As you can see, the model is reasonable for two of the genotypes (A and E, represented by open and solid diamonds, respectively), moderate for one genotype (C, solid triangles) but poor for two of them, B (open circles) and D (the open triangles). For both of the latter, the model overestimates the proportion flowering at zero dose, and for genotype B there seems to be some inhibition of flowering at the highest dose because the graph falls from 90% flowering at dose 16 to just 50% at dose 32. Variety D appears to be asymptoting at less than 100% flowering. These failures of the model focus attention for future work.

The moral is that the fact that we have proportion data does not mean that the data will necessarily be well described by the logistic model. For instance, in order to describe the response of genotype B, the model would need to have a hump, rather than to asymptote at $p = 1$ for large doses.

Converting Complex Contingency Tables to Proportions

In this section we show how to remove the need for all of the nuisance variables that are involved in complex contingency table modelling (see p. 302) by converting the response variable from a count to a proportion. We work thorough the analysis of Schoener's lizards, which we first encountered in Chapter 15. Instead of analysing the *counts* of the numbers of *Anolis grahamii* and *A. opalinus*, we restructure the data to represent the *proportions* of all lizards that were *A. grahamii* under the various contingencies.

```
lizards<-read.table("c:\\temp\\lizards.txt",header=T)
attach(lizards)
names(lizards)
```

```
[1] "n" "sun" "height" "perch" "time" "species"
```

First, we need to make absolutely sure that all the explanatory variables are in exactly the same order for both species of lizards. The reason for this is that we are going to cbind the counts for one of the lizard species onto the half dataframe containing the other species counts and all of the explanatory variables. Any mistakes here would be disastrous because the count would be lined up with the wrong combination of explanatory variables, and the analysis would be wrong and utterly meaningless.

```
sorted<-lizards[order(species,sun,height,perch,time),]
sorted
```

```
      n    sun  height   perch      time   species
41    4  Shade    High   Broad  Afternoon  grahamii
33    1  Shade    High   Broad    Mid.day  grahamii
25    2  Shade    High   Broad    Morning  grahamii
43    3  Shade    High  Narrow  Afternoon  grahamii
35    1  Shade    High  Narrow    Mid.day  grahamii
27    3  Shade    High  Narrow    Morning  grahamii
42    0  Shade     Low   Broad  Afternoon  grahamii
34    0  Shade     Low   Broad    Mid.day  grahamii
26    0  Shade     Low   Broad    Morning  grahamii
44    1  Shade     Low  Narrow  Afternoon  grahamii
36    0  Shade     Low  Narrow    Mid.day  grahamii
28    0  Shade     Low  Narrow    Morning  grahamii
45   10    Sun    High   Broad  Afternoon  grahamii
37   20    Sun    High   Broad    Mid.day  grahamii
...
24    4    Sun     Low  Narrow  Afternoon  opalinus
16   21    Sun     Low  Narrow    Mid.day  opalinus
8    12    Sun     Low  Narrow    Morning  opalinus
```

Next we need to take the top half of this dataframe (i.e. rows 1–24):

```
short<-sorted[1:24,]
short
```