13

Logistic regression

Sometimes you wish to model *binary outcomes*, variables that can have only two possible values: diseased or nondiseased, and so forth. For instance, you want to describe the risk of getting a disease depending on various kinds of exposures. Chapter 8 discusses some simple techniques based on tabulation, but you might also want to model dose-response relationships (where the predictor is a continuous variable) or model the effect of multiple variables simultaneously. It would be very attractive to be able to use the same modelling techniques as for linear models.

However, it is not really attractive to use additive models for probabilities since they have a limited range and regression models could predict off-scale values below zero or above 1. It makes better sense to model the probabilities on a transformed scale; this is what is done in logistic regression analysis.

A linear model for transformed probabilities can be set up as

$$\text{logit } p = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \ldots \beta_k x_k$$

in which $\text{logit } p = \log[p/(1-p)]$ is the *log odds*. A constant additive effect on the logit scale corresponds to a constant odds ratio. The choice of the logit function is not the only one possible, but it has some mathematically convenient properties. Other choices do exist; the probit function (the quantile function of the normal distribution) or $\log(-\log p)$, which has a connection to survival analysis models.

P. Dalgaard, *Introductory Statistics with R*,
DOI: 10.1007/978-0-387-79054-1_13,

One thing to notice about the logistic model is that there is no error term as in linear models. We are modelling the probability of an event directly, and that in itself will determine the variability of the binary outcome. There is no variance parameter as in the normal distribution.

The parameters of the model can be estimated by the *method of maximum likelihood*. This is a quite general technique, similar to the least-squares method in that it finds a set of parameters that optimizes a goodness-of-fit criterion (in fact, the least-squares method itself is a slightly modified maximum-likelihood procedure). The *likelihood function* $L(\beta)$ is simply the probability of the entire observed data set for varying parameters.

The *deviance* is the difference between the maximized value of $-2 \log L$ and the similar quantity under a "maximal model" that fits data perfectly. Changes in deviance caused by a model reduction will be approximately χ^2-distributed with degrees of freedom equal to the change in the number of parameters.

In this chapter, we see how to perform logistic regression analysis in R. There naturally is quite a large overlap with the material on linear models since the description of models is quite similar, but there are also some special issues concerning deviance tables and the specification of models for pretabulated data.

13.1 Generalized linear models

Logistic regression analysis belongs to the class of *generalized linear models*. These models are characterized by their response distribution (here the binomial distribution) and a *link function*, which transfers the mean value to a scale in which the relation to background variables is described as linear ans additive. In a logistic regression analysis, the link function is logit $p = \log[p/(1-p)]$.

There are several other examples of generalized linear models; for instance, analysis of count data is often handled by the multiplicative Poisson model, where the link function is $\log \lambda$, with λ the mean of the Poisson-distributed observation. All of these models can be handled using the same algorithm, which also allows the user some freedom to define his or her own models by defining suitable link functions.

In R generalized linear models are handled by the `glm` function. This function is very similar to `lm`, which we have used many times for linear normal models. The two functions use essentially the same model formulas and extractor functions (`summary`, etc.), but `glm` also needs to have specified *which* generalized linear model is desired. This is done via

the `family` argument. To specify a binomial model with logit link (i.e., logistic regression analysis), you write `family=binomial("logit")`.

13.2 Logistic regression on tabular data

In this section, we analyze the example concerning hypertension from Altman (1991, p. 353). First, we need to enter data, which is done as follows:

```
> no.yes <- c("No","Yes")
> smoking <- gl(2,1,8,no.yes)
> obesity <- gl(2,2,8,no.yes)
> snoring <- gl(2,4,8,no.yes)
> n.tot <- c(60,17,8,2,187,85,51,23)
> n.hyp <- c(5,2,1,0,35,13,15,8)
> data.frame(smoking,obesity,snoring,n.tot,n.hyp)
  smoking obesity snoring n.tot n.hyp
1      No      No      No    60     5
2     Yes      No      No    17     2
3      No     Yes      No     8     1
4     Yes     Yes      No     2     0
5      No      No     Yes   187    35
6     Yes      No     Yes    85    13
7      No     Yes     Yes    51    15
8     Yes     Yes     Yes    23     8
```

The `gl` function to "generate levels" was briefly introduced in Section 7.3. The first three arguments to `gl` are, respectively, the number of levels, the repeat count of each level, and the total length of the vector. A fourth argument can be used to specify the level names of the resulting factor. The result is apparent from the printout of the generated variables. They were put together in a data frame to get a nicer layout. Another way of generating a regular pattern like this is to use `expand.grid`:

```
> expand.grid(smoking=no.yes, obesity=no.yes, snoring=no.yes)
  smoking obesity snoring
1      No      No      No
2     Yes      No      No
3      No     Yes      No
4     Yes     Yes      No
5      No      No     Yes
6     Yes      No     Yes
7      No     Yes     Yes
8     Yes     Yes     Yes
```

R is able to fit logistic regression analyses for tabular data in two different ways. You have to specify the response as a matrix, where one column is

the number of "diseased" and the other is the number of "healthy" (or "success" and "failure", depending on context).

```
> hyp.tbl <- cbind(n.hyp,n.tot-n.hyp)
> hyp.tbl
     n.hyp
[1,]    5   55
[2,]    2   15
[3,]    1    7
[4,]    0    2
[5,]   35  152
[6,]   13   72
[7,]   15   36
[8,]    8   15
```

The `cbind` function ("c" for "column") is used to bind variables together, columnwise, to form a matrix. Note that it would be a horrible mistake to use the total count for column 2 instead of the number of failures.

Then, you can specify the logistic regression model as

```
> glm(hyp.tbl~smoking+obesity+snoring,family=binomial("logit"))
```

Actually, `"logit"` is the default for `binomial` and the `family` argument is the second argument to `glm`, so it suffices to write

```
> glm(hyp.tbl~smoking+obesity+snoring,binomial)
```

The other way to specify a logistic regression model is to give the *proportion* of diseased in each cell:

```
> prop.hyp <- n.hyp/n.tot
> glm.hyp <- glm(prop.hyp~smoking+obesity+snoring,
+                binomial,weights=n.tot)
```

It is necessary to give `weights` because R cannot see how many observations a proportion is based on.

As output, you get in either case (except for minor details)

```
Call:  glm(formula = hyp.tbl ~ smoking + obesity + snoring, ...

Coefficients:
(Intercept)   smokingYes   obesityYes   snoringYes
   -2.37766     -0.06777      0.69531      0.87194

Degrees of Freedom: 7 Total (i.e. Null);  4 Residual
Null Deviance:      14.13
Residual Deviance: 1.618        AIC: 34.54
```

which is in a minimal style similar to that used for printing `lm` objects. Also in the result of `glm` is some nonvisible information, which may be extracted with particular functions. You can, for instance, save the result of a fit of a generalized linear model in a variable and obtain a table of regression coefficients and so forth using `summary`:

```
> glm.hyp <- glm(hyp.tbl~smoking+obesity+snoring,binomial)
> summary(glm.hyp)

Call:
glm(formula = hyp.tbl ~ smoking + obesity + snoring, family ...

Deviance Residuals:
       1         2         3         4         5         6
-0.04344   0.54145  -0.25476  -0.80051   0.19759  -0.46602
       7         8
-0.21262   0.56231

Coefficients:
            Estimate Std. Error z value Pr(>|z|)
(Intercept) -2.37766    0.38018  -6.254    4e-10 ***
smokingYes  -0.06777    0.27812  -0.244   0.8075
obesityYes   0.69531    0.28509   2.439   0.0147 *
snoringYes   0.87194    0.39757   2.193   0.0283 *
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 14.1259  on 7  degrees of freedom
Residual deviance:  1.6184  on 4  degrees of freedom
AIC: 34.537

Number of Fisher Scoring iterations: 4
```

In the following, we go through the components of `summary` output for generalized linear models:

```
Call:
glm(formula = hyp.tbl ~ smoking + obesity + snoring, family = ...
```

As usual, we start off with a repeat of the model specification. Obviously, more interesting is when the output is not viewed in connection with the function call that generated it.

```
Deviance Residuals:
       1         2         3         4         5         6
-0.04344   0.54145  -0.25476  -0.80051   0.19759  -0.46602
       7         8
-0.21262   0.56231
```

This is the contribution of each cell of the table to the deviance of the model (the deviance corresponds to the sum of squares in linear normal models), with a sign according to whether the observation is larger or smaller than expected. They can be used to pinpoint cells that are particularly poorly fitted, but you have to be wary of the interpretation in sparse tables.

```
Coefficients:
            Estimate Std. Error z value Pr(>|z|)
(Intercept) -2.37766    0.38018  -6.254    4e-10 ***
smokingYes  -0.06777    0.27812  -0.244   0.8075
obesityYes   0.69531    0.28509   2.439   0.0147 *
snoringYes   0.87194    0.39757   2.193   0.0283 *
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)
```

This is the table of primary interest. Here, we get estimates of the regression coefficients, standard errors of same, and tests for whether each regression coefficient can be assumed to be zero. The layout is nearly identical to the corresponding part of the `lm` output.

The note about the dispersion parameter is related to the fact that the binomial variance depends entirely on the mean. There is no scale parameter like the variance in the normal distribution.

```
    Null deviance: 14.1259  on 7  degrees of freedom
Residual deviance:  1.6184  on 4  degrees of freedom
AIC: 34.537
```

"Residual deviance" corresponds to the residual sum of squares in ordinary regression analyses which is used to estimate the standard deviation about the regression line. In binomial models, however, the standard deviation of the observations is known, and you can therefore use the deviance in a test for model specification. The AIC (Akaike information criterion) is a measure of goodness of fit that takes the number of fitted parameters into account.

R is reluctant to associate a p-value with the deviance. This is just as well because no exact p-value can be found, only an approximation that is valid for large expected counts. In the present case, there are actually a couple of places where the expected cell count is rather small.

The asymptotic distribution of the residual deviance is a χ^2 distribution with the stated degrees of freedom, so even though the approximation may be poor, nothing in the data indicates that the model is wrong (the 5% significance limit is at 9.49 and the value found here is 1.62).

The null deviance is the deviance of a model that contains only the intercept (that is, describes a fixed probability, here for hypertension, in all cells). What you would normally be interested in is the difference from the residual deviance, here $14.13 - 1.62 = 12.51$, which can be used for a joint test for whether any effects are present in the model. In the present case, a p-value of approximately 0.6% is obtained.

```
Number of Fisher Scoring iterations: 4
```

This refers to the actual fitting procedure and is a purely technical item. There is no statistical information in it, but you should keep an eye on whether the number of iterations becomes too large because that might be a sign that the model is too complex to fit based on the available data. Normally, `glm` halts the fitting procedure if the number of iterations exceeds 25, but it is possible to configure the limit.

The fitting procedure is *iterative* in that there is no explicit formula that can be used to compute the estimates, only a set of equations that they should satisfy. However, there is an approximate solution of the equations if you supply an initial guess at the solution. This solution is then used as a starting point for an improved solution, and the procedure is repeated until the guesses are sufficiently stable.

A table of correlations between parameter estimates can be obtained via the optional argument `corr=T` to `summary` (this also works for linear models). It looks like this:

```
Correlation of Coefficients:
           (Intercept) smokingYes obesityYes
smokingYes     -0.1520
obesityYes     -0.1361 -9.499e-05
snoringYes     -0.8965 -6.707e-02   -0.07186
```

It is seen that the correlation between the estimates is fairly small, so that it may be expected that removing a variable from the model does not change the coefficients and p-values for other variables much. (The correlations between the regression coefficients and intercept are not very informative; they mostly relate to whether the variable in question has many or few observations in the "Yes" category.)

The z test in the table of regression coefficients immediately shows that the model can be simplified by removing `smoking`. The result then looks as follows (abbreviated):

```
> glm.hyp <- glm(hyp.tbl~obesity+snoring,binomial)
> summary(glm.hyp)

...
```

```
Coefficients:
            Estimate Std. Error z value Pr(>|z|)
(Intercept)  -2.3921     0.3757  -6.366 1.94e-10 ***
obesityYes    0.6954     0.2851   2.440   0.0147 *
snoringYes    0.8655     0.3967   2.182   0.0291 *
```

13.2.1 *The analysis of deviance table*

Deviance tables correspond to ANOVA tables for multiple regression analyses and are generated like these with the `anova` function:

```
> glm.hyp <- glm(hyp.tbl~smoking+obesity+snoring,binomial)
> anova(glm.hyp, test="Chisq")
Analysis of Deviance Table

Model: binomial, link: logit

Response: hyp.tbl

Terms added sequentially (first to last)


        Df Deviance Resid. Df Resid. Dev P(>|Chi|)
NULL                        7    14.1259
smoking  1   0.0022         6    14.1237    0.9627
obesity  1   6.8274         5     7.2963    0.0090
snoring  1   5.6779         4     1.6184    0.0172
```

Notice that the `Deviance` column gives *differences* between models as variables are added to the model in turn. The deviances are approximately χ^2-distributed with the stated degrees of freedom. It is necessary to add the `test="chisq"` argument to get the approximate χ^2 tests.

Since the `snoring` variable on the last line is significant, it may not be removed from the model and we cannot use the table to justify model reductions. If, however, the terms are rearranged so that `smoking` comes last, we get a deviance-based test for removal of that variable:

```
> glm.hyp <- glm(hyp.tbl~snoring+obesity+smoking,binomial)
> anova(glm.hyp, test="Chisq")
...
        Df Deviance Resid. Df Resid. Dev P(>|Chi|)
NULL                        7    14.1259
snoring  1   6.7887         6     7.3372    0.0092
obesity  1   5.6591         5     1.6781    0.0174
smoking  1   0.0597         4     1.6184    0.8069
```

From this you can read that `smoking` is removable, whereas `obesity` is not, after removal of `smoking`.

For good measure, you should also set up the analysis with the two remaining explanatory variables interchanged, so that you get a test of whether `snoring` may be removed from a model that also contains `obesity`:

```
> glm.hyp <- glm(hyp.tbl~obesity+snoring,binomial)
> anova(glm.hyp, test="Chisq")
...
        Df Deviance Resid. Df Resid. Dev P(>|Chi|)
NULL                        7    14.1259
obesity  1   6.8260         6     7.2999    0.0090
snoring  1   5.6218         5     1.6781    0.0177
```

An alternative method is to use `drop1` to try removing one term at a time:

```
> drop1(glm.hyp, test="Chisq")
Single term deletions

Model:
hyp.tbl ~ obesity + snoring
        Df Deviance    AIC    LRT Pr(Chi)
<none>       1.678 32.597
obesity  1    7.337 36.256  5.659 0.01737 *
snoring  1    7.300 36.219  5.622 0.01774 *
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Here LRT is the likelihood ratio test, another name for the deviance change.

The information in the deviance tables is fundamentally the same as that given by the z tests in the table of regression coefficients. The results may differ due to the use of different approximations, though. From theoretical considerations, the deviance test is preferable, but in practice the difference is often small because of the large-sample approximation $\chi^2 \approx z^2$ for tests with a single degree of freedom. However, to test factors with more than two categories, you have to use the deviance table because the z tests only relate to some of the possible group comparisons. Also, the small-sample situation requires special attention; see the next section.

13.2.2 Connection to test for trend

In Chapter 8, we considered tests for comparing relative frequencies using `prop.test` and `prop.trend.test`, in particular the example of caesarean section versus shoe size. This example can also be analyzed as a logistic regression analysis on a "shoe score", which — for want of a better idea — may be chosen as the group number. This gives essentially the same analysis in the sense that the same models are involved.

```
> caesar.shoe
    <4  4 4.5  5 5.5  6+
Yes  5  7   6  7   8  10
No  17 28  36 41  46 140
> shoe.score <- 1:6
> shoe.score
[1] 1 2 3 4 5 6

> summary(glm(t(caesar.shoe)~shoe.score,binomial))
...
Coefficients:
            Estimate Std. Error z value Pr(>|z|)
(Intercept) -0.87058    0.40506  -2.149  0.03161 *
shoe.score  -0.25971    0.09361  -2.774  0.00553 **
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 9.3442  on 5  degrees of freedom
Residual deviance: 1.7845  on 4  degrees of freedom
AIC: 27.616
...
```

Notice that `caesar.shoe` had to be transposed with `t(...)`, so that the matrix was "stood on its end" in order to be used as the response variable by `glm`.

You can also write the results in a deviance table

```
> anova(glm(t(caesar.shoe)~shoe.score,binomial))
...
           Df Deviance Resid. Df Resid. Dev
NULL                           5     9.3442
shoe.score  1   7.5597         4     1.7845
```

from the last line of which you see that there is no significant deviation from linearity (1.78 on 4 degrees of freedom), whereas `shoe.score` has a significant contribution.

For comparison, the previous analyses using standard tests are repeated:

```
> caesar.shoe.yes <- caesar.shoe["Yes",]
> caesar.shoe.no <- caesar.shoe["No",]
> caesar.shoe.total <- caesar.shoe.yes+caesar.shoe.no
> prop.trend.test(caesar.shoe.yes,caesar.shoe.total)
       Chi-squared Test for Trend in Proportions
...
X-squared = 8.0237, df = 1, p-value = 0.004617

> prop.test(caesar.shoe.yes,caesar.shoe.total)
```

```
        6-sample test for equality of proportions without
        continuity correction
...
X-squared = 9.2874, df = 5, p-value = 0.09814
...
Warning message:
In prop.test(caesar.shoe.yes, caesar.shoe.total) :
  Chi-squared approximation may be incorrect
```

The 9.29 from `prop.test` corresponds to the 9.34 in residual deviance from a `NULL` model, whereas the 8.02 in the trend test corresponds to the 7.56 in the test of significance of `shoe.score`. Thus, the tests do not give exactly the same result but generally *almost* the same. Theoretical considerations indicate that the specialized trend test is probably slightly better than the regression-based test. However, testing the linearity by subtracting the two χ^2 tests is definitely not as good as the real test for linearity.

13.3 Likelihood profiling

The `z` tests in the summary output are based on the *Wald approximation*, which calculates what the approximate standard error of the parameter estimate would be if the true values of the parameters were equal to the estimates. In large data sets, this is fine because the result is nearly the same for all parameter values that fit the data reasonably well. In smaller data sets, however, the difference between the Wald tests and the likelihood ratio test can be considerable.

This also affects the calculation of confidence intervals since these are based on inverting the tests, giving a set of parameter values that are not rejected by a statistical test. As an alternative to the Wald-based $\pm 1.96 \times$ s.e. technique, the `MASS` package allows you to compute intervals that are based on inverting the likelihood ratio test. In practice, this works like this

```
> confint(glm.hyp)
Waiting for profiling to be done...
                 2.5 %    97.5 %
(Intercept) -3.2102369 -1.718143
obesityYes   0.1254382  1.246788
snoringYes   0.1410865  1.715860
```

The standard type of result can be obtained using `confint.default`. The difference in this case is not very large, although visible in the lines relating to snoring and the intercept:

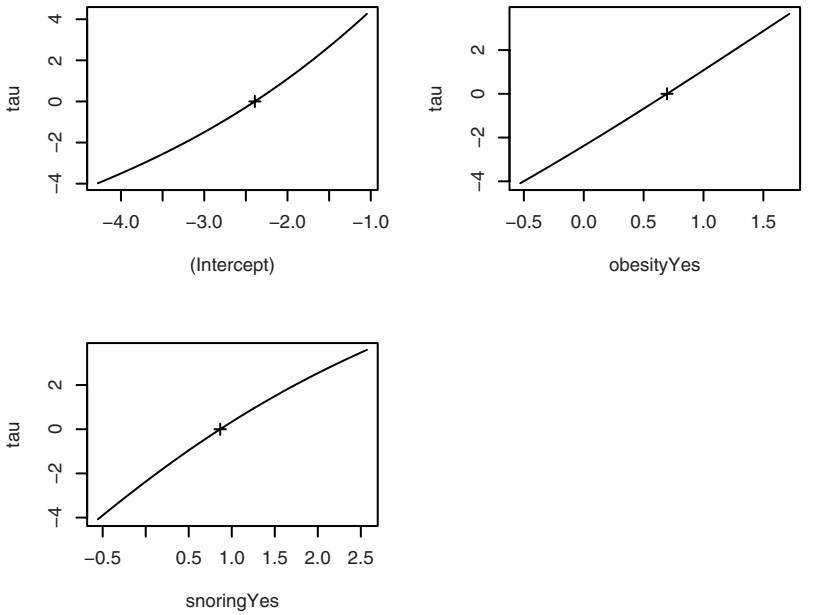

Figure 13.1. Profile plot for hypertension model.

```
> confint.default(glm.hyp)
                  2.5 %    97.5 %
(Intercept) -3.12852108 -1.655631
obesityYes   0.13670388  1.254134
snoringYes   0.08801498  1.642902
```

The way this works is via *likelihood profiling*. For a set of trial values of the parameter, the likelihood is maximized over the other parameters in the model. The result can be displayed in a profile plot as follows:

```
> library(MASS)
> plot(profile(glm.hyp))
```

Notice that we need to load the `MASS` package at this point. (The function was used by `confint` earlier on, but without putting it on the search path.)

The plots require a little explanation. The quantity on the y-axis, labelled `tau`, is the signed square root of the likelihood ratio test.

$$\tau(\beta) = \text{sgn}(\beta - \hat{\beta})\sqrt{-2(\ell(\beta) - \ell(\hat{\beta}))}$$

Here ℓ denotes the profile log-likelihood. The main idea is that when the profile likelihood function is approximately quadratic, $\tau(\beta)$ is approximately linear. Conversely, likelihood functions not well approximated by a quadratic show up as nonlinear profile plots.

One important thing to notice, though, is that although the profiling method will capture nonquadratic behaviour of the likelihood function, confidence intervals based on the likelihood ratio test will always be limited in accuracy by the approximation of the distribution of the test.

13.4 Presentation as odds-ratio estimates

In parts of the epidemiological literature, it has become traditional to present logistic regression analyses in terms of odds ratios. In the case of a quantitative covariate, this means odds ratio per unit change in the covariate. That is, the antilogarithm (exp) of the regression coefficients is given instead of the coefficients themselves. Since standard errors make little sense after the transformation, it is also customary to give confidence intervals instead. This can be obtained quite easily as follows:

```
> exp(cbind(OR=coef(glm.hyp), confint(glm.hyp)))
Waiting for profiling to be done...
                    OR      2.5 %    97.5 %
(Intercept) 0.09143963 0.04034706 0.1793989
obesityYes  2.00454846 1.13364514 3.4791490
snoringYes  2.37609483 1.15152424 5.5614585
```

The `(Intercept)` is really the odds of hypertension (for the not snoring non-obese) and not an odds ratio.

13.5 Logistic regression using raw data

In this section, we again use Anders Juul's data (see p. 85). For easy reference, here is how to read data and convert the variables that describe groupings into factors (this time slightly simplified):

```
> juul$menarche <- factor(juul$menarche, labels=c("No","Yes"))
> juul$tanner <- factor(juul$tanner)
```

In the following, we look at `menarche` as the response variable. This variable indicates for each girl whether or not she has had her first period. It is coded 1 for "no" and 2 for "yes". It is convenient to look at a subset of data consisting of 8–20-year-old girls. This can be extracted as follows:

```
> juul.girl <- subset(juul,age>8 & age<20 &
+                     complete.cases(menarche))
> attach(juul.girl)
```

For obvious reasons, no boys have a nonmissing `menarche`, so it is not necessary to select on gender explicitly.

Then you can analyze `menarche` as a function of `age` like this:

```
> summary(glm(menarche~age,binomial))
Call:
glm(formula = menarche ~ age, family = binomial)

Deviance Residuals:
     Min        1Q    Median        3Q       Max
-2.32759  -0.18998   0.01253   0.12132   2.45922

Coefficients:
            Estimate Std. Error z value Pr(>|z|)
(Intercept) -20.0132     2.0284  -9.867   <2e-16 ***
age           1.5173     0.1544   9.829   <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 719.39  on 518  degrees of freedom
Residual deviance: 200.66  on 517  degrees of freedom
AIC: 204.66

Number of Fisher Scoring iterations: 7
```

The response variable `menarche` is a factor with two levels, where the last level is considered the event. It also works to use a variable that has the values 0 and 1 (but *not*, for instance, 1 and 2!).

Notice that from this model you can estimate the median menarcheal age as the age where logit $p = 0$. A little thought (solve $-20.0132 + 1.5173 \times \text{age} = 0$) reveals that it is $20.0132/1.5173 = 13.19$ years.

You should not pay too much attention to the deviance residuals in this case since they automatically become large in every case where the fitted probability "goes against" the observations (which is bound to happen in some cases). The residual deviance is also difficult to interpret when there is only one observation per cell.

A hint of a more complicated analysis is obtained by including the Tanner stage of puberty in the model. You should be warned that the exact interpretation of such an analysis is quite tricky and *qualitatively* different from the analysis of `menarche` as a function of `age`. It can be used for prediction purposes (although asking the girl whether she has had her first

period would likely be much easier than determining her Tanner stage!), but the interpretation of the terms is not clear-cut.

```
> summary(glm(menarche~age+tanner,binomial))
...
Coefficients:
            Estimate Std. Error z value Pr(>|z|)
(Intercept) -13.7758     2.7630  -4.986 6.17e-07 ***
age           0.8603     0.2311   3.723 0.000197 ***
tanner2      -0.5211     1.4846  -0.351 0.725609
tanner3       0.8264     1.2377   0.668 0.504313
tanner4       2.5645     1.2172   2.107 0.035132 *
tanner5       5.1897     1.4140   3.670 0.000242 ***
...
```

Notice that there is no joint test for the effect of `tanner`. There are a couple of significant z-values, so you would expect that the `tanner` variable has some effect (which, of course, you would probably expect even in the absence of data!). The formal test, however, must be obtained from the deviances:

```
> drop1(glm(menarche~age+tanner,binomial),test="Chisq")
...
       Df Deviance     AIC     LRT   Pr(Chi)
<none>     106.599 118.599
age     1  124.500 134.500  17.901 2.327e-05 ***
tanner  4  161.881 165.881  55.282 2.835e-11 ***
...
```

Clearly, both terms are highly significant.

13.6 Prediction

The `predict` function works for generalized linear models, too. Let us first consider the hypertension example, where data were given in tabular form:

```
> predict(glm.hyp)
         1          2          3          4          5          6
-2.3920763 -2.3920763 -1.6966575 -1.6966575 -1.5266180 -1.5266180
         7          8
-0.8311991 -0.8311991
```

Recall that `smoking` was eliminated from the model, which is why the expected values come in identical pairs.

These numbers are on the logit scale, which reveals the additive structure. Notice that $2.392 - 1.697 = 1.527 - 0.831 = 0.695$ (except for roundoff er-

ror), which is exactly the regression coefficient to `obesity`. Likewise, the regression coefficient to `snoring` is obtained by looking at the differences $2.392 - 1.527 = 1.697 - 0.831 = 0.866$.

To get predicted values on the response scale (i.e., probabilities), use the `type="response"` argument to `predict`:

```
> predict(glm.hyp, type="response")
        1         2         3         4         5         6
0.08377892 0.08377892 0.15490233 0.15490233 0.17848906 0.17848906
        7         8
0.30339158 0.30339158
```

These may also be obtained using `fitted`, although you then cannot use the techniques for predicting on new data, etc.

In the analysis of `menarche`, the primary interest is probably in seeing a plot of the expected probabilities versus age (Figure 13.2). A crude plot could be obtained using something like

```
plot(age, fitted(glm(menarche~age,binomial)))
```

(it will look better if a different plotting symbol in a smaller size, using the `pch` and `cex` arguments, is used) but here is a more ambitious plan:

```
> glm.menarche <- glm(menarche~age, binomial)
> Age <- seq(8,20,.1)
> newages <- data.frame(age=Age)
> predicted.probability <- predict(glm.menarche,
+                                  newages,type="resp")
> plot(predicted.probability ~ Age, type="l")
```

This is Figure 13.2. Recall that `seq` generates equispaced vectors, here ages from 8 to 20 in steps of 0.1, so that connecting the points with lines will give a nearly smooth curve.

13.7 Model checking

For tabular data it is obvious to try to compare observed and fitted proportions. In the hypertension example you get

```
> fitted(glm.hyp)
        1         2         3         4         5         6
0.08377892 0.08377892 0.15490233 0.15490233 0.17848906 0.17848906
        7         8
0.30339158 0.30339158
> prop.hyp
```

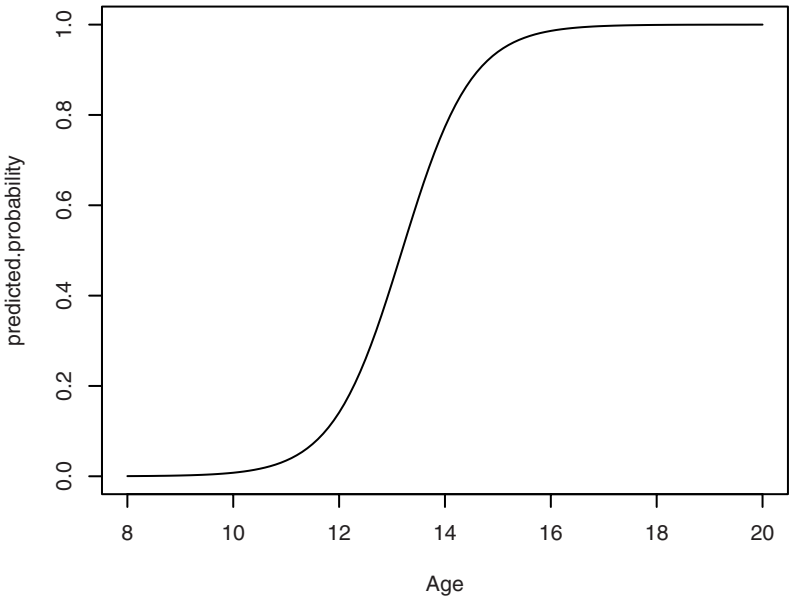

Figure 13.2. Fitted probability of menarche having occurred.

```
[1] 0.08333333 0.11764706 0.12500000 0.00000000 0.18716578
[6] 0.15294118 0.29411765 0.34782609
```

The problem with this is that you get no feeling for how well the relative frequencies are determined. It can be better to look at observed and expected *counts* instead. The former can be computed as

```
> fitted(glm.hyp)*n.tot
         1          2          3          4          5          6
 5.0267351  1.4242416  1.2392186  0.3098047 33.3774535 15.1715698
         7          8
15.4729705  6.9780063
```

and to get a nice print for the comparison, you can use

```
> data.frame(fit=fitted(glm.hyp)*n.tot,n.hyp,n.tot)
         fit n.hyp n.tot
1  5.0267351     5    60
2  1.4242416     2    17
3  1.2392186     1     8
4  0.3098047     0     2
5 33.3774535    35   187
6 15.1715698    13    85
```

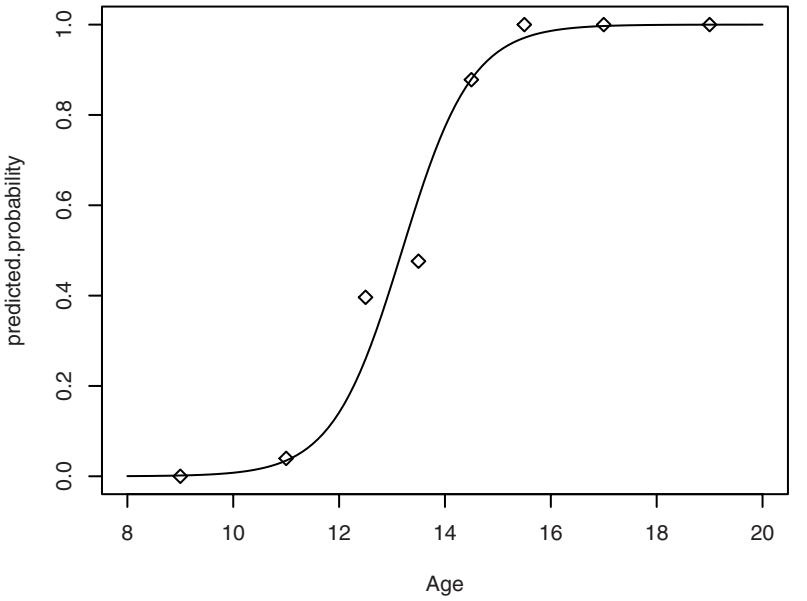

Figure 13.3. Fitted probability for menarche having occurred and observed proportion in age groups.

```
7 15.4729705    15    51
8  6.9780063     8    23
```

Notice that the discrepancy in cell 4 between 15% expected and 0% observed really is that there are 0 hypertensives out of 2 in a cell where the model yields an expectation of 0.3 hypertensives!

For complex models with continuous background variables, it becomes more difficult to perform an adequate model check. It is especially a hindrance that nothing really corresponds to a residual plot when the observations have only two different values.

Consider the example of the probability of `menarche` as a function of `age`. The problem here is whether the relation can really be assumed linear on the logit scale. For this case, you might try subdividing the x-axis in a number of intervals and see how the counts in each interval fit with the expected probabilities. This is presented graphically in Figure 13.3. Notice that the code *adds* points to Figure 13.2, which you are assumed not to have deleted at this point.

```
> age.group <- cut(age,c(8,10,12,13,14,15,16,18,20))
```

```
> tb <- table(age.group,menarche)
> tb
         menarche
age.group  No Yes
  (8,10]  100   0
  (10,12]  97   4
  (12,13]  32  21
  (13,14]  22  20
  (14,15]   5  36
  (15,16]   0  31
  (16,18]   0 105
  (18,20]   0  46
> rel.freq <- prop.table(tb,1)[,2]
> rel.freq
    (8,10]    (10,12]    (12,13]    (13,14]    (14,15]    (15,16]
0.00000000 0.03960396 0.39622642 0.47619048 0.87804878 1.00000000
   (16,18]    (18,20]
1.00000000 1.00000000
> points(rel.freq ~ c(9,11,12.5,13.5,14.5,15.5,17,19),pch=5)
```

The technique used above probably requires some explanation. First, `cut` is used to define the factor `age.group`, which describes a grouping into age intervals. Then a crosstable `tb` is formed from `menarche` and `age.group`. Using `prop.table`, the numbers are expressed relative to the row total, and column 2 of the resulting table is extracted. This contains the relative proportion in each age group of girls for whom menarche has occurred. Finally, a plot of expected probabilities is made, overlaid by the observed proportions.

The plot looks reasonable on the whole, although the observed proportion among 12–13-year-olds appears a bit high and the proportion among 13–14-year-olds is a bit too low.

But how do you evaluate whether the deviation is larger than what can be expected from the statistical variation? One thing to try is to extend the model with a factor that describes a division into intervals. It is not practical to use the full division of `age.group` because there are cells where either none or all of the girls have had their menarche.

We therefore try a division into four groups, with cutpoints at 12, 13, and 14 years, and add this factor to the model containing a linear age effect.

```
> age.gr <- cut(age,c(8,12,13,14,20))
> summary(glm(menarche~age+age.gr,binomial))
...
Coefficients:
             Estimate Std. Error z value Pr(>|z|)
(Intercept)  -21.5683     5.0645  -4.259 2.06e-05 ***
age            1.6250     0.4416   3.680 0.000233 ***
age.gr(12,13]  0.7296     0.7856   0.929 0.353024
age.gr(13,14] -0.5219     1.1184  -0.467 0.640765
```

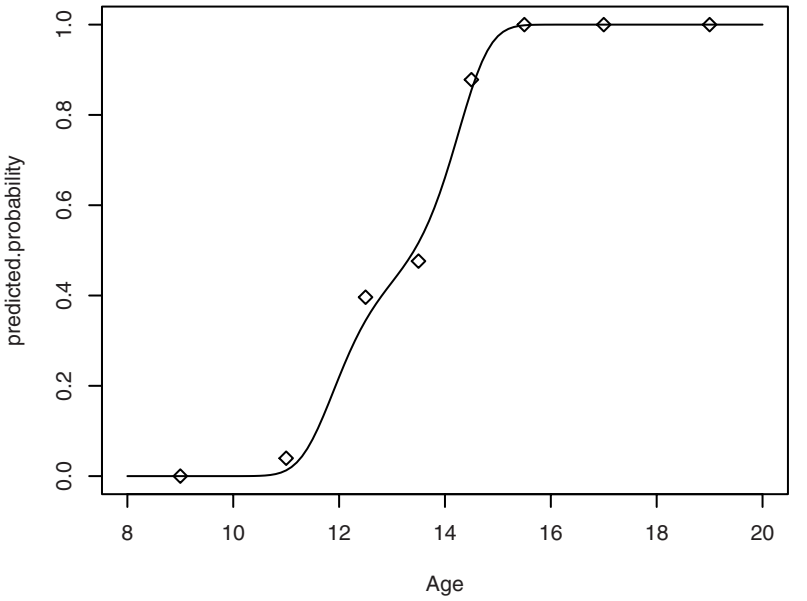

Figure 13.4. Logit-cubical fit of menarche data.

```
age.gr(14,20]   0.2751     1.6065   0.171 0.864053
...

> anova(glm(menarche~age+age.gr,binomial))
...
        Df Deviance Resid. Df Resid. Dev
NULL                     518     719.39
age      1   518.73       517     200.66
age.gr   3     8.06       514     192.61
> 1-pchisq(8.058,3)
[1] 0.04482811
```

That is, the addition of the grouping actually does give a significantly better deviance. The effect is not highly significant, but since the deviation concerns the ages where "much happens", you should probably be cautious about postulating a logit-linear age effect.

Another possibility is to try a polynomial regression model. Here you need at least a third-degree polynomial to describe the apparent stagnation of the curve around 13 years of age. We do not look at this in great detail, but just show part of the output and in Figure 13.4 a graphical presentation of the model.

```
> anova(glm(menarche~age+I(age^2)+I(age^3)+age.gr,binomial))
...
          Df Deviance Resid. Df Resid. Dev
NULL                        518     719.39
age        1   518.73       517     200.66
I(age^2)   1     0.05       516     200.61
I(age^3)   1     8.82       515     191.80
age.gr     3     3.34       512     188.46
Warning messages:
1: In glm.fit(x = X, y = Y, weights = weights, .... :
  fitted probabilities numerically 0 or 1 occurred
2: In method(x = x[, varseq <= i, drop = FALSE], .... :
  fitted probabilities numerically 0 or 1 occurred
> glm.menarche <- glm(menarche~age+I(age^2)+I(age^3), binomial)
Warning message:
In glm.fit(x = X, y = Y, weights = weights, start = start, .... :
  fitted probabilities numerically 0 or 1 occurred
> predicted.probability <-
+     predict(glm.menarche, newages, type="resp")
> plot(predicted.probability ~ Age, type="l")
> points(rel.freq~c(9,11,12.5,13.5,14.5,15.5,17,19), pch=5)
```

The warnings about fitted probabilities of 0 or 1 occur because the cubic term makes the logit tend much faster to $\pm\infty$ than the linear model did. There are two occurrences for the `anova` call because two of the models include the cubic term.

The thing to note in the deviance table is that the cubic term gives a substantial improvement of the deviance, but once that is included, the age grouping gives no additional improvement. The plot should speak for itself.

13.8 Exercises

13.1 In the `malaria` data set, analyze the risk of malaria with age and log-transformed antibody level as explanatory variables.

13.2 Fit a logistic regression model to the `graft.vs.host` data set, predicting the `gvhd` response. Use different transformations of the `index` variable. Reduce the model using backwards elimination.

13.3 In the analyses of the `malaria` and `graft.vs.host` data, try using the `confint` function to find improved confidence intervals for the regression coefficients.

13.4 Following up on Exercise 8.2 about "Rocky Mountain spotted fever", splitting the data by age groups gives the table below. Does this

confirm the earlier analysis?

	Western Type		Eastern Type	
Age Group	Total	Fatal	Total	Fatal
Under 15	108	13	310	40
15–39	264	40	189	21
40 or above	375	157	162	61
	747	210	661	122

13.5 A *probit* regression is just like a logistic regression but uses a different link function. Try the analysis of the `menarche` variable in the `juul` data set with this link. Does the fit improve?