Chapter 2
Modeling in R

2.1 Introduction

One of the many strengths of R is in the diversity and convenience of its modeling functions. In this chapter, we describe several standard statistical models and show how to fit them to data using R.

The key to modeling in R is the *formula object*, which provides a shorthand method to describe the exact model to be fit to the data. Modeling functions in R typically require a formula object as an argument. The modeling functions return a *model object* that contains all the information about the fit. Generic R functions such as `print`, `summary`, `plot`, `anova`, etc. will have methods defined for specific object classes to return information that is appropriate for that kind of object.

We first present the *linear model* (LM) and corresponding modeling functions. We begin with these precisely because the reader is likely familiar with the linear model in some form (multiple regression, analysis of variance, etc.) and it is easier to get used to working with objects and R in a familiar context. Moreover, all the other models considered here are extensions of the linear model. Second, we describe *linear mixed-effects models*. These permit modeling random effects to account appropriately for variation due to factors such as observer differences. Then, we consider some tools for fitting *nonlinear models* in R. Finally, we describe *generalized linear models* (GLM). These models allow us to fit many different kinds of psychophysical data. They will play a central role in succeeding chapters.

There are several excellent books that describe the theory and fitting of the models discussed here in R, notably [58, 178, 198].

2.2 The Linear Model

In the basic linear model, a response vector (dependent variable) is modeled as a weighted linear combination of explanatory variable vectors (independent variables)

K. Knoblauch and L.T. Maloney, *Modeling Psychophysical Data in R*, Use R! 32,
DOI 10.1007/978-1-4614-4475-6_2,
© Springer Science+Business Media New York 2012

and an additive error vector, $\varepsilon = (\varepsilon_1, \ldots, \varepsilon_n)$, where each ε_i is an independent, identically distributed (iid) Gaussian random variable with mean 0 and variance σ^2 (hereafter abbreviated by $\varepsilon \sim N(0, \sigma^2)$).

For example, in a multiple regression model with dependent variable $Y = (Y_1, \ldots, Y_n)$ and independent variables $X_1 = (X_{1,1}, \ldots, X_{1,n})$ and $X_2 = (X_{2,1}, \ldots, X_{2,n})$ the linear model is

$$Y_i = \beta_0 + \beta_1 X_{1,i} + \beta_2 X_{2,i} + \varepsilon_i, \quad i = 1, n \tag{2.1}$$

where n is the number of observations of Y, the β's are coefficients to be estimated when fitting the model and $\varepsilon_i \sim N(0, \sigma^2)$. We use subscripts, $1, 2$, here to index the independent variables, but we will see below that they correspond to columns in a matrix.

The model is simply summarized in vector form as

$$Y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \varepsilon \tag{2.2}$$

and even more simply in matrix form as

$$Y = \boldsymbol{X}\beta + \varepsilon \tag{2.3}$$

where $\boldsymbol{X}$, the *model* or *design matrix*, is composed of a column of ones followed by columns with the values of X_1 and X_2, and $\beta = (\beta_0, \beta_1, \beta_2)$. Fitting the model involves estimating $\beta = (\beta_0, \beta_1, \beta_2)$ and the variance σ^2 of the Gaussian errors. Once we have the estimate $\hat{\beta} = (\hat{\beta}_0, \hat{\beta}_1, \hat{\beta}_2)$, we can also estimate the *residual vector*

$$e = Y - \boldsymbol{X}\hat{\beta} \tag{2.4}$$

Rearranging, and comparing to (2.3), we see that the residual vector e is an estimate of the error vector ε

$$Y = \boldsymbol{X}\hat{\beta} + e \tag{2.5}$$

We use maximum likelihood estimation to estimate β. In the case of the linear model this choice of $\hat{\beta}$ also minimizes the sum of the squared residuals. This solution is often referred to as the *least squares solution*.

In the basic linear model, the independent variables contain numbers that we weight and add to form estimates of the dependent variable. In the case that the explanatory variables are continuous numerical values, we call them *covariates* and we are engaged in *multiple regression*.

Explanatory variables can also have entries that are categorical labels such as "Male," "Female." When the explanatory variables are categorical, they are called *factors* and the linear model is referred to as *analysis of variance* (ANOVA). When the model contains both factor and continuous explanatory variables, it is called *analysis of covariance* (ANCOVA).

To use the modeling functions for the linear model in R, we first need to specify the model with dependent and independent variables. R uses a formula language developed by Wilkinson and Rogers [191]. We first create a data frame that contains the vectors of the response and explanatory values as columns.

If we have such a data frame containing columns named Y, X1, X2, each of class "numeric," then the formula object corresponding to (2.3) would be

```
> Y ~ X1 + X2 + 1
```

The tilde, "~," is an operator that separates the response to be modeled from the explanatory variables. It can be read as "is modeled by." The term corresponding to the intercept β_0 is represented in the formula object by `+ 1`. We can simplify the formula object slightly as `+ 1` is the default case:

```
> Y ~ X1 + X2
```

Either formula indicates to a modeling function that a 3 column model matrix should be constructed with a first column of 1's and the second and third from the column vectors X1 and X2. In general, formula objects for linear models have an intercept unless we explicitly suppress it by writing

```
> Y ~ X1 + X2 - 1
```

or

```
> Y ~ X1 + X2 + 0
```

From here on, we omit the `+ 1`.

The formula object doesn't explicitly mention the coefficients β or the error term. The formula object simply specifies the relations among the dependent and independent variables.

When the explanatory variables are factors, the notation of the linear model is typically different. For example, with factors A and B, where A has levels "Male" and "Female" and B has levels "Old" and "Young," each observation might be modeled by

$$Y_{ijk} = \mu + \alpha_i + \beta_j + \alpha\beta_{ij} + e_{ijk} \tag{2.6}$$

where μ is the grand mean of the Y_{ijk}, α_i is the effect of level i of factor A, β_j is the effect of level j of factor B, $\alpha\beta_{ij}$ is the interaction of levels i and j of the two factors and, as before, $\varepsilon_{ijk} \sim N(0, \sigma^2)$. In other words, we think of each response as the grand mean "adjusted" by adding α_1 if the subject is male or adding α_2 if female, "adjusted" further by β_1 if the subject is "Old," etc. The term $\alpha\beta_{ij}$ captures possible interactions between the factors. Any text on analysis of variance will develop this sort of model at length.

ANOVA is just a form of the linear model and the notation of ANOVA in R makes this clear. The model in (2.6) can be specified in a formula object as

```
> Y ~ A + B + A:B
```

where the term with : indicates the interaction between all of the levels of A and B. A shorthand notation that is equivalent employs the * symbol

```
> Y ~ A * B
```

which means A, B, and the interaction term from above. There are simple ways to include only some interactions and not others, and that allow the user to explore alternative models for a given data set (see Sect. 11.1 of *An Introduction to R*, in the documentation that comes with R) [146].

The only thing special about ANOVA is the use of independent variables that are factors. This difference, however, leads to differences in how the model matrix $\boldsymbol{X}$ is constructed. In the model matrix for an ANOVA, each categorical (or factor) variable is expanded into columns of indicator variables with one column allocated to each level. For example, a model with one factor with 3 levels and 2 replications per level is written out as

$$Y = \begin{pmatrix} 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} \mu \\ \alpha_1 \\ \alpha_2 \\ \alpha_3 \end{pmatrix} \tag{2.7}$$

The first column of 1's corresponds to the grand mean, μ, and each succeeding column to a level of the factor variable. The difficulty here is that the columns are linearly dependent; the sum of columns 2–4 equals column 1. If we attempted to fit a model with this model matrix, we would not succeed as there is no unique solution to the equation. This issue does not arise in typical multiple regression models unless there is a linear dependence among some subset of the explanatory variables including the constant term.

To deal with the inherent dependence in the model matrix of an ANOVA, constraints are placed on the model matrix. Many different choices for the constraints are possible, and we illustrate some alternatives, by way of which we also introduce the useful function `model.matrix` that computes a model matrix for a formula object.

Continuing with the example from (2.7), we use the function `factor` to create a variable of class "factor" with 3 levels and 2 replications per level that we assign to a variable A.

```
> (A <- rep(factor(1:3), each = 2))
[1] 1 1 2 2 3 3
Levels: 1 2 3
```

The function `rep` is useful for generating repeating patterns.

With the responses denoted as Y, the formula object for the linear model in (2.7) is

```
> Y ~ A
```

To view the default model matrix that is generated for this model, we use the function `model.matrix` with a one-sided formula object.

```
> model.matrix( ~  A)
  (Intercept) A2 A3
1           1  0  0
2           1  0  0
3           1  1  0
4           1  1  0
5           1  0  1
6           1  0  1
attr(,"assign")
[1] 0 1 1
attr(,"contrasts")
attr(,"contrasts")$A
[1] "contr.treatment"
```

The resulting matrix contains a column of all 1's corresponding to the intercept term and columns corresponding to just two of the levels, `A2` and `A3`. This model matrix has linearly independent columns and R has achieved this outcome by dropping the first level, `A1`.

The choice of model matrix changes the parameterization of the model (the price of being able to identify uniquely the parameters, a model property referred to as *identifiability*). The default choice in R is called *treatment contrasts*. The first column of 1's will generate an estimate of the mean of the first level of the factor. The succeeding columns generate estimates of the difference of means between each level and the first level. If the first level corresponds to a control condition, then the subsequent levels could be used to contrast each with the control.

We could alternatively remove the intercept term to yield the following model matrix:

```
> model.matrix( ~  A - 1)
  A1 A2 A3
1  1  0  0
2  1  0  0
3  0  1  0
4  0  1  0
5  0  0  1
6  0  0  1
attr(,"assign")
[1] 1 1 1
attr(,"contrasts")
attr(,"contrasts")$A
[1] "contr.treatment"
```

The new model matrix still has three columns but now each column corresponds to one level of the factor. Recall that the syntax `(-1)` tells R to omit the intercept term in a linear model, here the mean of the first level. In this parameterization, the estimates will correspond to the means of each of the factor levels. This is referred to as the *cell means model*.

Several other options are available in R (see, `?contr.treatment`). For example, the `contr.sum` function can be used to generate a model matrix in which the sum of contrasts across levels equals 0. The `contrasts` argument of `model.matrix` requires a list whose components are named by the variables in the formula and are assigned character strings indicating the name of a function to calculate the contrasts for that variable.

```
> model.matrix( ~  A, contrasts = list(A = "contr.sum"))
  (Intercept) A1 A2
1           1  1  0
2           1  1  0
3           1  0  1
4           1  0  1
5           1 -1 -1
6           1 -1 -1
attr(,"assign")
[1] 0 1 1
attr(,"contrasts")
attr(,"contrasts")$A
[1] "contr.sum"
```

Studying the built-in functions will be of use if the reader needs to tailor the contrast functions for specialized situations.

2.2.1 Development of First- and Second-Order Motion

As a linear model example, we consider the data of Thibault et al. [171] who examined the minimum modulation contrast to detect two kinds of motion as a function of age, based on the spatiotemporal modulation of luminance (first order) or of contrast (second order). The data set `Motion`, obtained from the **MPDiR** package by,

```
> data(Motion)
```

contains five components, `Subject`, a 70-level factor identifying individual observers, a covariate `LnAge`, the (natural) logarithm of the age in months, `Mtype`, a 2-level factor with levels “FO” and “SO” coding the type of motion stimulus, `Sex`, a 2-level factor indicating the sex of the observer, and `LnThresh`, the (natural) logarithm of the minimum contrast modulation of the stimulus detected by the observer. We check that all is as expected by

```
> str(Motion)
'data.frame':   112 obs. of  5 variables:
 $ Subject : Factor w/ 70 levels "S01","S02","S03",..:
               1 2..
 $ LnAge   : num  2.2 2.2 2.2 2.2 2.2 ...
 $ Mtype   : Factor w/ 2 levels "FO","SO": 1 1 1 1 1 2
               1 2..
 $ Sex     : Factor w/ 2 levels "f","m": 1 1 2 2 1 1 2
               2 1..
 $ LnThresh: num  3.22 2.19 2.19 2.88 2.1 ...
```

The data were collected using the preferential looking technique [170]. Only the estimated thresholds are reported, so we confine attention to them.

We can ask several questions about these data. Does threshold depend upon age? Are there differences in threshold as a function of sex or type of motion? Does a change in threshold with age depend on either of these factors? A first appreciation of the data is obtained by plotting the age dependence of threshold for each level of the two factors in separate panels, using the following code fragment:

```
> library(lattice)
> xyplot(LnThresh ~ LnAge | Mtype + Sex, data = Motion,
+    xlab = "Log Age (months)",
+    ylab = "Log Threshold (contrast)",
+    panel = function(x, y) {
+       panel.lmline(x, y)
+       panel.loess(x, y, lty = 2, lwd = 2,
         col = "black")
+       panel.xyplot(x, y, col = "black", pch = 16,
         cex = 1.5)
+       })
```

The first argument is a formula object that specifies what we want to plot. The term `LnAge | Mtype + Sex` specifies that we want separate plots for each combination of `Mtype` and `Sex`. The `panel` argument allows us to customize what will be plotted in each graph with a panel function. If it is left out, the default panel function generates scatterplots using the part of the formula to the left of the vertical bar for each combination of the factors indicated to the right of the bar. In the panel function above, we first add a least squares regression line to each plot using `panel.lmline` and then add a local (loess) regression curve, using `panel.loess`. The loess curve follows a smoothed average trend in the data. Adding a smooth curve like the loess to the data provides a quick visual assessment of whether a line is a reasonable description of the data. Finally, we add the points with the `panel.xyplot` function. Each of the plot functions can take additional arguments to control the appearance of what it plots.

Examination of the plots suggests that a linear model would be appropriate for the first-order motion data, but the loess curves hint that there may be a ceiling

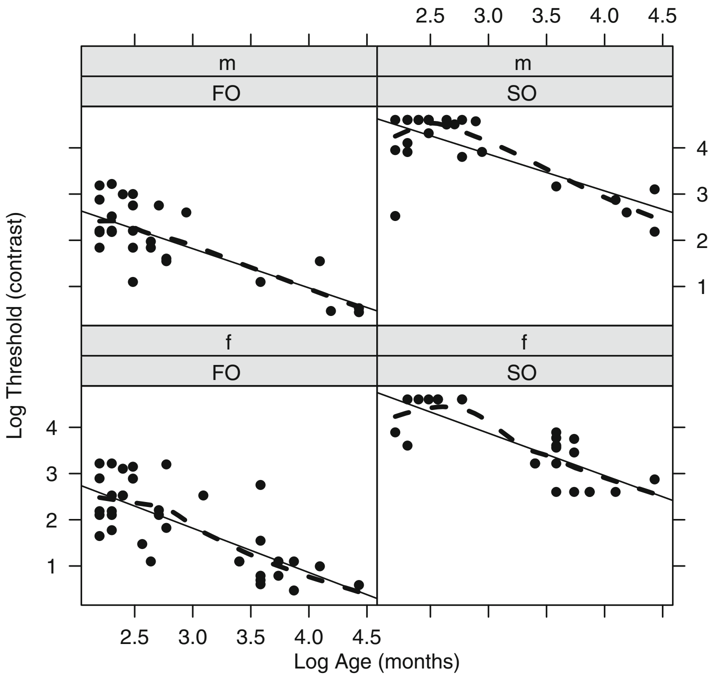

Fig. 2.1 Contrast modulation thresholds as a function of age for male (m) and female (f) observers and for first-order (FO) and second-order (SO) motion stimuli. The *solid lines* are from linear regression and the *dashed line* using local regression

effect present at the youngest ages for the detection of second-order motion. We will ignore this possibility here but note that tools are available in R that would allow us to fit censored data (see, e.g., the function `tobit` in the **AER** package [92]) (Fig. 2.1).

The function `lm` is the principle tool for analyzing linear models. It can handle models in which the explanatory variables are all continuous (*covariates*) or categorical (*factors*). In most applications, `lm` has at least two arguments, a formula object specifying the model we wish to fit and a data frame which provides the environment within which the names in the formula are interpreted. Each variable in the formula object corresponds to a column in the data frame with the same name.

The first model we consider is

$$Y = \beta_0 + \beta_1 \text{LnAge} + \beta_2 \text{Mtype} + \beta_3 \text{Sex} \tag{2.8}$$

It includes only additive effects of (log transformed) Age, Sex and Motion type. The model contains both covariates (`LnAge`) and factors (`Sex` and `Mtype`) and, thus, is an example of an ANCOVA.

It is often recommended to begin with the most complex model, including all explanatory variables and their interactions of all orders. Successive comparisons with simpler models will then allow us to determine the minimum set of terms that adequately explains the data. For didactic purposes, however, we start with a simpler model. The model in (2.8) is fit as follows:

```
> Motion.lm0 <- lm(LnThresh ~ Mtype + Sex +  LnAge,
+   data = Motion)
```

The first argument is the formula object, the second argument, the data frame containing all of the variables.

In applications of the linear model, it is typically recommended to transform all covariates to be centered around their respective means. We can do so by using the `scale` function, replacing the covariate `LnAge` by `scale(LnAge, center = TRUE, scale = FALSE)`. This subtracts the mean from each value. If we set `scale = TRUE`, then the values will also be scaled by the root mean square of the vector. When `center = TRUE`, this is simply the standard deviation. After centering the means, the estimated intercept is then centered with respect to the sampled data, rather than to an extrapolated point and is more easily interpretable. Here, the curves are nearly parallel, and this procedure has little effect on the interpretation (see Exercise 2.3).

The results of the fit are stored in the object `Motion.lm0` which is of class "lm" and can be probed with methods defined for objects of this class. For example, the `summary` method displays the table of estimated coefficients, their standard errors, *t*- and *p*-values among other information. To extract just the table, we use the `coefficient` method which is aliased to the abbreviation `coef`.

```
> coef(summary(Motion.lm0))
            Estimate Std. Error t value Pr(>|t|)
(Intercept)   4.4698     0.2373   18.84 6.54e-36
MtypeSO       2.0404     0.1042   19.57 2.62e-37
Sexm         -0.0137     0.1047   -0.13 8.96e-01
LnAge        -0.8822     0.0749  -11.78 4.34e-21
```

The values for each model coefficient are the estimated differences of successive levels from the first level of the factor `Mtype` (i.e, treatment contrasts). Thus, `MtypeSO` is the effect of second-order motion with respect to first and `Sexm` is the difference in effect between males and females. There appears to be no main effect of `Sex` though we cannot yet exclude an effect of Sex on the slopes of the lines. That is, there is no hint that thresholds for male subjects are consistently higher or lower by a fixed amount.

We test for a possible effect of `Sex` on slope by adding all second-order interactions and performing a likelihood ratio test with the `anova` method. We could do this by rerunning `lm` with a new formula object specifying all the interaction terms but R provides an efficient way to update and rerun a linear model fit after modifying the formula. We want to add all second-order interactions. We could explicitly add interaction terms of the form `LnAge:Mtype`, `LnAge:Sex`, etc. to the model, but the formula language provides a convenient shortcut. Raising the model to the *n*th power generates all interaction terms up to order *n*.

The new model is fit by

```
> Motion.lm1 <- update(Motion.lm0, . ~ .^2)
```

The "." in this context indicates that the previous value should be substituted. The formula says to include the same dependent variable as used in `Motion.lm0` and to the explanatory variables add all of their second-order interactions.

After recomputing the model fit with second-order interactions added, we use the `anova` method to compare the two models, with and without the second-order interactions.

```
> anova(Motion.lm0, Motion.lm1)

Analysis of Variance Table

Model 1: LnThresh ~ Mtype + Sex + LnAge
Model 2: LnThresh ~ Mtype + Sex + LnAge + Mtype:Sex +
    Mtype:LnAge + Sex:LnAge
  Res.Df  RSS Df Sum of Sq    F Pr(>F)
1    108 31.5
2    105 31.2  3     0.207 0.23   0.87
```

The reader can also try `anova(Motion.lm1)` to obtain an anova table for the new fit with second-order interactions. The table shows the reduction in the residual sum of squares as each term from the formula is added in. When the data are unbalanced, this depends on the order of the terms in the formula. The results of the nested hypothesis tests performed by `anova` on a set of models do not depend on the order of the terms in the formulae, however.

Formally, we are testing whether we can reject the first model in favor of the second. We want to compare the models and see if any of the second-order interactions are significant. The comparisons of the models with and without each interaction lead to likelihood ratio tests (see Sect. B.5). It is based on an F-statistic rather than the χ^2 that someone familiar with nested hypothesis testing might expect. Intuitively, we can draw an analogy to the comparison of two means where the t-statistic is used instead of the z when the sample standard deviation is estimated from data. Examining the output of the comparison, we see that the models do not differ significantly: that is, there are no significant interactions.

Next we test whether there is any effect of `Sex` by updating the model and adding the term `- Sex`. The minus sign in the added term specifies that we are removing the term `Sex` from the model:

```
> Motion.lm2 <- update(Motion.lm0, . ~ . - Sex)
> anova(Motion.lm2, Motion.lm0)
Analysis of Variance Table
```

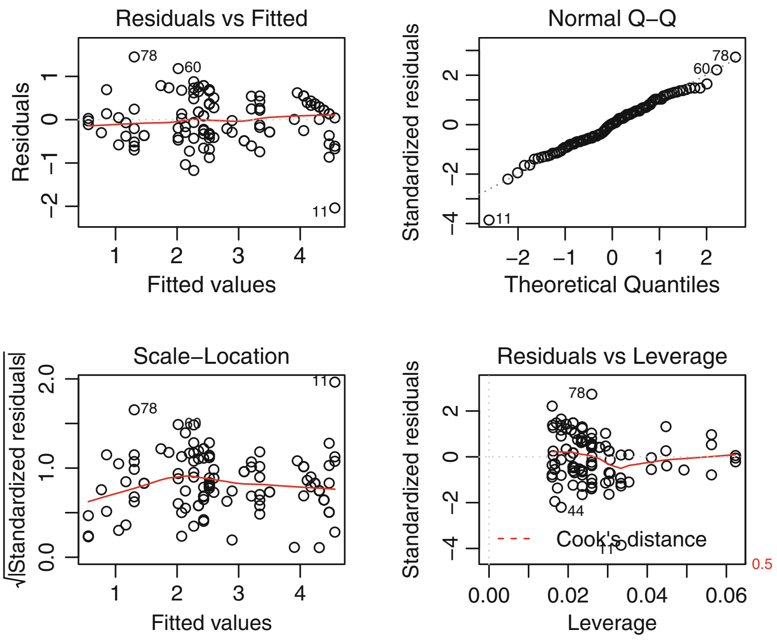

Fig. 2.2 Diagnostic plots for the linear model stored in object `Motion.lm2`

```
Model 1: LnThresh ~ Mtype + LnAge
Model 2: LnThresh ~ Mtype + Sex + LnAge
  Res.Df  RSS Df Sum of Sq    F Pr(>F)
1    109 31.5
2    108 31.5  1   0.00495 0.02    0.9
```

and the call to `anova` tests for a significant effect of including `Sex`. We find no evidence for an effect of `Sex`.

R provides several tools for assessing whether the linear model is appropriate for a particular application. The method `plot` for "lm" objects generates by default a set of four graphs that are useful in diagnosing whether the model assumptions underlying the fit are reasonable.

```
> plot(motion.lm2)
```

The two plots in the first column of Fig. 2.2 display the pattern of residuals with respect to the fitted values. These are useful for detecting inhomogeneity in the distribution of residuals. If the model is appropriate, we expect residuals that are unpatterned with no trends in magnitude.

Patterns or trends in magnitude would indicate that the model is inappropriate and that we might need to consider additional explanatory variables, transform the response nonlinearly, or consider a nonlinear model. All but the last option can be performed with `lm`. The results of the first diagnostic plot raise no warning flags concerning our modeling assumptions. It is also useful to examine the residuals as a function of the values of each explanatory variable, but you will have to program that yourself!

The upper-right plot is a quantile–quantile (Q–Q) plot that permits a visual assessment of the assumption that the error is Gaussian. A Q–Q plot graphs the quantiles of the residuals against the theoretical quantiles of a Gaussian distribution. Marked deviation from a straight line indicates that the actual error distribution is not Gaussian. The plot we see is roughly linear.

On the lower left, we see a plot of the square root of the magnitude of the residuals as a function of the fitted values. This diagnostic plot is used to detect heteroscedasticity of the residuals. Again, there are no indications of failures of the assumptions underlying the linear model.

Finally, the fourth plot (lower right) allows us to detect points that have a large impact (leverage) on the fitted values. These so-called *influence points* are data that have disproportionate impact on fitted model values. (See [40] for a discussion of this and other regression diagnostics.) This plot allows one to check whether the conclusions drawn are due to a small, anomalous subset of the full data set. The size of each residual is plotted as a function of its "leverage" in determining the estimated coefficients. The plot of residuals versus leverage discloses no problems with the fit. Note that certain points have been labelled by row number in the data frame. These have been flagged as potential outliers and the user might want to examine them in more detail.

Based on our analyses and model comparisons, we arrive at a model in which the logarithm of contrast threshold is a linear function of the logarithm of age that does not depend on the sex of the observer. The linear functions for all conditions have the same slope but observers are about 8 times more sensitive to the first order motion stimulus than to the second. We leave as an exercise plotting the data with the fitted lines.

2.3 Linear Mixed-Effects Models

Thus far in this chapter, we have considered only fixed-effect linear models. These are linear models in which the explanatory variables are *not* themselves random and the only random source of variation in the model is the residual variation that is not accounted for by the explanatory variables. In certain circumstances, however, some of the explanatory variables may represent random samples from a larger population and we would like to draw conclusions not just about the data set that we have but also the population from which it was drawn.

A typical example would be data for a group of subjects in an experiment who are drawn from a population of all potential subjects who might have participated in the experiment. Other examples include a set of natural images used as stimuli considered as a sample from the population of all natural images or a set of tone sequences sampled from the population of all melodies. If we model the effects due to these variables (people or stimuli) as fixed effects, then the conclusions that we draw pertain to the specific characteristics of the sample used in the study and cannot be attributed to the characteristics of the population as a whole.

If we wish to generalize the conclusions drawn from the sample to the population, such variables must be treated as *random effects*. If all the explanatory variables are treated as random effects, then our model is a *random effects model*. If only some of them are, then the resulting model is a *mixed-effects model*.

In fitting a random-effects or mixed-effects model, we estimate the variability of the population from which the sample is drawn rather than just the effects attributed to the individual members of the sample. Such models then include multiple sources of variation, those due to the random explanatory variables as well as the residual variation unexplained by the fixed and the random effects. A remarkably clear introduction to mixed-effects models and how to fit them can be found in Pinheiro and Bates [141].

In a mixed-effects model, the response vector is taken conditionally on the random effects and is modeled as the sum of a fixed effects term $\boldsymbol{X}$ and a random effects term, $\boldsymbol{Z}$.

$$(Y|b) = \boldsymbol{X}\beta + \boldsymbol{Z}b + \varepsilon, \tag{2.9}$$

where $(Y|b)$ is the response vector conditional on b, the random effects vector, $\boldsymbol{X}\beta$ is the fixed effects term, the product of a design matrix and a vector of fixed effects coefficients, $\boldsymbol{Z}b$ is the random effects term, the product of a design matrix for the random effects and the vector of random-effects coefficients such that $b \sim N(0, \sigma_b^2 I)$ and $\varepsilon \sim N(0, \sigma^2)$ is the residual variation unaccounted for by the rest of the model. In fitting a mixed-effects model the fixed effects coefficients, β and the variance of the random effects, σ_b^2 are both estimated. Mixed-effects models introduce an important simplification permitted by random effects. Suppose, for example, 1,000 observers participated in a study. If a fixed intercept was associated with each subject, then the model would require 1,000 parameters to be estimated. If instead, the observers were considered a random effect, then only a single extra parameter, the variance due to observers, would be added to the model.

2.3.1 *The ModelFest Data Set*

The `ModelFest` data set originates from a study conducted by a consortium of researchers to produce a data base under fixed conditions from several laboratories that would serve as a reference data set for models of human spatial vision [26, 183]. The data set consists of contrast thresholds for 43 stimuli from each of 16 observers with each condition repeated 4 times, yielding a balanced data set. The stimuli can be viewed at http://vision.arc.nasa.gov/modelfest/stimuli.html and consist of a variety of luminance patterns of varying complexity. The data can be accessed in several fashions from links at the web site http://vision.arc.nasa.gov/modelfest/data.html.[1]

[1]An alternate source for the data is at http://journalofvision.org/5/9/6/modelfestbaselinedata.csv.

We read the data into R directly from the URL in "comma separated value" or csv format, using the function `read.csv`. This is a wrapper function for `read.table` that presets the arguments necessary for reading files in csv format. See Appendix A.

The structure of each record consists of a character string identifying the observer followed by the four thresholds of the first stimulus, then the second, etc. This produces a data frame with one row for each observer, i.e., a wide format, but we would prefer the long format, in which there is one observation per line with the stimulus and observer variables provided in separate columns. The code below retrieves the data from the web site and effects the required modifications.

```
>  site <- file.path(url{"http://vision.arc.
    nasa.gov/modelfest/",
+   "data/modelfestbaselinedata.csv"})
>  ModelFest <- read.csv(site, header = FALSE)
>  Obs <- ModelFest$V1
>  ModelFest.df <- data.frame(LContSens =
     c(t(ModelFest[, -1])),
+   Obs = rep(Obs, each = ncol(ModelFest) - 1),
+   Stim = rep(paste("Stim", 1:43, sep = ""), each = 4)
+   )
> ModelFest.df$Stim <- with(ModelFest.df, factor(Stim,
+       levels = unique(Stim)))
```

The `file.path` function is used here simply as a convenience to construct a path that is valid under any operating system and to permit us to break up the URL address so that it stays within the margins of our page. The thresholds are given as $-\log_{10}$(contrast) or log contrast sensitivity, which explains the variable name that we chose, `LContSens`. The last line of the code fragment above reorders the factor levels so that they are in numerical rather than lexical order.

For simplification, we restrict our analyses to the data for the first ten stimuli. We use the `paste` function to construct the labels of the first ten levels of the factor `Stim` and then the `subset` function to extract the rows of the data frame corresponding to those levels. Then, we redefine the factor so as to drop the unused levels from its definition.

```
> mfGab <- subset(ModelFest.df,
+   Stim %in% paste("Stim", 1:10, sep = ""))
> mfGab$Stim <- mfGab$Stim[, drop = TRUE]
> SpatFreq <- c(1.12, 2^seq(1, 4.5, 0.5), 30)
> mfGab$SpatFreq <- rep(SpatFreq, each = 4)

> with(mfGab, interaction.plot(SpatFreq, Obs, LContSens, mean,
+ type = "b", xlab = "Spatial Frequency (c/deg)",
+ ylab = "Log Contrast Sensitivity", cex.lab = 1.5))
```

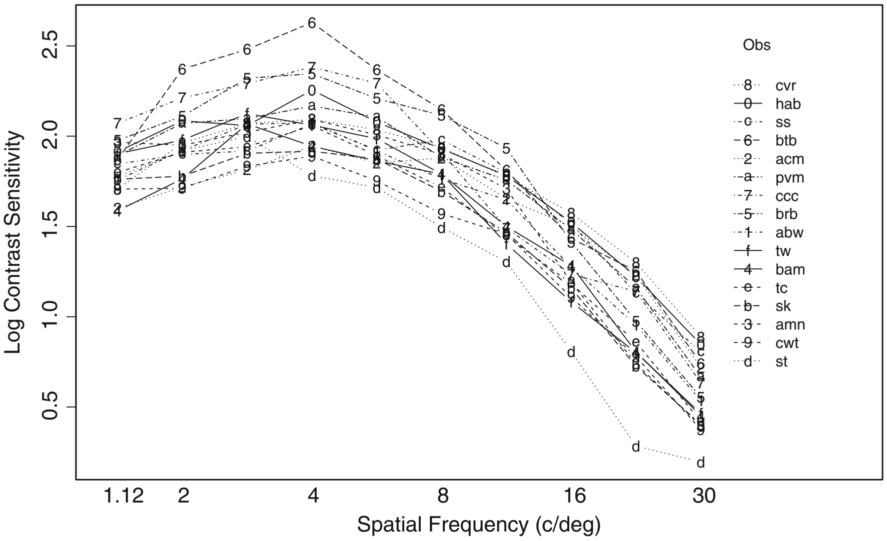

Fig. 2.3 Log contrast sensitivity as a function of spatial frequency for the Gabor patterns of the `ModelFest` data set for each of the 16 observers

The stimuli are Gabor patterns (Gaussian-damped, sine-wave gratings) of fixed size and increasing spatial frequency and their associated responses define a contrast sensitivity function over the spatial frequency of the carrier frequency. We add a column to the data frame including these spatial frequencies. The spatial frequency variable will be convenient for plotting the data and also if we decide to use spatial frequency as a covariate (see Problem 2.10). The spatial frequencies of the Gabor carrier frequencies are separated by half-octave steps except for the first and last ones.

The mean log contrast sensitivity is plotted for each observer as a function of spatial frequency in Fig. 2.3, using the function `interaction.plot`. Each observer's data are conveniently plotted with a different symbol and line type, indicated in the legend at the right. The dependence on spatial frequency is qualitatively similar for each observer, but the average heights of the points vary between observers. Some observers are overall more sensitive than others. The average SD by stimulus and observer is 0.1 log contrast (maximum of 0.23) while SD across observers for each stimulus is nearly twice as large. This increased variability across observers provides additional justification for considering the factor `Obs` as a random effect.

As mentioned above, the data set is balanced with each observer having the same number of replications at each spatial frequency. Balanced data sets possess the nice property that the contributions of the explanatory variables to the total variation in the response are orthogonal. This means that the response can be partitioned into independent and orthogonal sources of variation. If we ignore `SpatFreq` in the `mfGab` data frame, for the moment, then the remaining two explanatory variables, Obs and Stim, are factors. Recall that models with only factor variables are termed ANOVA models. After fitting a linear model to the data, we obtain the variance of each of these terms and their interaction using the anova method.

```
> mfGab.lm <- lm(LContSens ~ Obs * Stim, mfGab)
> anova(mfGab.lm)
Analysis of Variance Table

Response: LContSens
           Df Sum Sq Mean Sq F value Pr(>F)
Obs        15   14.6    0.97    87.8 <2e-16 ***
Stim        9  155.2   17.25  1553.7 <2e-16 ***
Obs:Stim  135    8.2    0.06     5.5 <2e-16 ***
Residuals 480    5.3    0.01
---
Signif. codes:  0 `***' 0.001 `**' 0.01 `*' 0.05 `.'
   0.1 ` ' 1
```

The second column gives the sum of the squared error (SSE) associated with each term. It is simple to verify that this sums to the total variation in log contrast sensitivity.

```
> sum(anova(mfGab.lm)[, 2])
[1] 183
> sd(mfGab$LContSens)^2 * (nrow(mfGab) - 1)
[1] 183
```

The third column indicates the mean squared errors (MSE), obtained by dividing each term of column 2 by its degrees of freedom in column 1. As can also be verified, however, the F-values were all obtained by dividing each MSE by the MSE of the last row, Residuals which corresponds to a test treating both Stim and Obs as fixed effects.

In order to model Obs as a random effect, we would normally calculate an F-value by using the MSE for the interaction term, Obs:Stim in the denominator [88, 195]. Doing so is equivalent to calculating the error with respect to each level of Stim within each level of Obs, stratifying or nesting the error calculation with respect to observers, as one does in a paired t-test to increase the sensitivity of the test. For balanced data sets, as in this case, the aov function provides a convenient interface for specifying error strata and obtaining the appropriate F-test for such circumstances. The format is similar to lm but permits inclusion of a term Error for specifying the nesting factor.[2]

```
> mfGab.aov <- aov(LContSens ~ Stim + Error(Obs/Stim),
   mfGab)
> summary(mfGab.aov)
```

[2]In fact, it works by calling lm, but isolating the correct error terms for the F calculation.

```
Error: Obs
          Df Sum Sq Mean Sq F value Pr(>F)
Residuals 15   14.6   0.975

Error: Obs:Stim
           Df Sum Sq Mean Sq F value Pr(>F)
Stim        9  155.2   17.25     283 <2e-16 ***
Residuals 135    8.2    0.06
---
Signif. codes:  0 `***' 0.001 `**' 0.01 `*' 0.05 `.'
                  0.1 ` ' 1

Error: Within
           Df Sum Sq Mean Sq F value Pr(>F)
Residuals 480   5.33  0.0111
```

Note how we specify the `Error` term as `Stim` nested within `Obs`. Some additional insight may come by noting that a term `Error(Obs + Obs:Stim)` is an equivalent formulation and yields identical results.

The `summary` method outputs a set of anova tables using the same df, SSE and MSE values as obtained using `lm` but reorganized with respect to different error strata, `Obs`, `Obs:Stim` and `Within`. The second table contains the appropriate F-test calculated using the interaction term. More detailed explanation of the `Error` term and further examples of its usage can be found in Chambers and Hastie [28] and in Li and Baron [112] and in the notes by Revelle at http://personality-project.org/r/r.guide.html.

Data sets are often unbalanced, however, in which case the simple methods presented above are not easily applied. The SSE terms will no longer be orthogonal so the decomposition may, for example, depend on the order or presence/absence of other terms in the model. It is still possible to analyze such data, but one must take recourse in methods that calculate maximum likelihoods or variants thereof. R contains several packages that permit such modeling.

Two of the principal packages devoted to analyzing mixed-effects models are the recommended package **nlme** [142] described in detail in the book by its authors [141] and the newer **lme4** package [8, 9, 11]. While the older **nlme** provides a more complete set of methods and options for linear and nonlinear mixed-effects models, the newer **lme4** contains a function for analyzing generalized linear mixed-effects models whose use we will explore in Chap. 9. At the time of writing this book, however, this package is under rapid development with changes and extensions of some of its current functionality. For this reason, we limit our discussion here to the `lmer` function from the **lme4.0** package [10] (currently available from Rforge (https://r-forge.r-project.org/projects/lme4/) but eventually to be released on CRAN), which is a stable version of the current package.

The `lmer` function works similarly to the other modeling functions in R, requiring minimally a formula object. The terms in the formula are obtained from a data frame, if present, specified as the second argument, and otherwise are searched for in the environment in which `lmer` was called. The formula object must include at least one fixed effect and one random effect term. Note that this required fixed effect could be as simple as the implicit intercept term in the formula specification. The fixed effects terms are defined as shown previously. The random effect terms are specified by a two part expression separated by a vertical bar (or pipe symbol), "`|`," and enclosed in parentheses.

```
(1 | Obs)
```

The left-hand term indicates the random component and the right the grouping or nesting factor.

For the contrast sensitivity data, a random intercept associated with each observer would indicate a model in which the shape of the contrast sensitivity function across frequency was independent of observers but the height of the data depended on the observer. In the following code fragment, we load the **lme4.0** package in memory and fit the model, after which we explicitly evoke the `print` method for the object in order to suppress the print-out of the large correlation matrix for the fixed effects.

```
> library(lme4.0)
> mfGab.lmer <- lmer(LContSens ~ Stim + (1 | Obs),
   mfGab)
> print(mfGab.lmer, correlation = FALSE)
Linear mixed model fit by REML
Formula: LContSens ~ Stim + (1 | Obs)
   Data: mfGab
  AIC  BIC logLik deviance REMLdev
 -492 -439    258     -574    -516
Random effects:
 Groups   Name        Variance Std.Dev.
 Obs      (Intercept) 0.0238   0.154
 Residual             0.0221   0.149
Number of obs: 640, groups: Obs, 16
Fixed effects:
            Estimate Std. Error t value
(Intercept)   1.8210     0.0428    42.5
StimStim2     0.1394     0.0263     5.3
StimStim3     0.2422     0.0263     9.2
StimStim4     0.2855     0.0263    10.9
StimStim5     0.1710     0.0263     6.5
StimStim6     0.0227     0.0263     0.9
StimStim7    -0.2001     0.0263    -7.6
StimStim8    -0.5232     0.0263   -19.9
StimStim9    -0.8615     0.0263   -32.8
StimStim10   -1.2535     0.0263   -47.7
```

The model is fit with 11 parameters, the 10 fixed effects associated with the factor `Stim` and the single random effect associated with the factor `Obs`. This contrasts with the model fit to these data using `lm` which requires the estimation of 160 coefficients. Each random component of the model is indicated by its variance and its square-root, or standard deviation. It is not the standard deviation of the variance.

Being in the same units as the response variable, the standard deviation is usually more easily related to the variation visible graphically when the data are plotted. For example, the contrast sensitivities at a given spatial frequency in Fig. 2.3 vary over a bit more than 0.5 log contrast sensitivity which is close to ± 2 SDs the random term `Obs`. The residual variance, however, is about double that of the models fit with `lm` and `aov`. This is because the model that we fit with `lmer` contains only a random effect for `Obs` whereas above we grouped the data by `Stim` in `Obs`. We can fit that model, too, with `lmer` using the `update` method as follows.

```
> mfGab.lmer2 <-update(mfGab.lmer, . ~ . +
   (1 | Obs:Stim))
> print(mfGab.lmer2, correlation = FALSE)
Linear mixed model fit by REML
Formula: LContSens ~ Stim + (1 | Obs) + (1 | Obs:Stim)
   Data: mfGab
  AIC  BIC logLik deviance REMLdev
 -683 -625    354     -757    -709
Random effects:
 Groups   Name        Variance Std.Dev.
 Obs:Stim (Intercept) 0.0125   0.112
 Obs      (Intercept) 0.0228   0.151
 Residual             0.0111   0.105
Number of obs: 640, groups: Obs:Stim, 160; Obs, 16

Fixed effects:
            Estimate Std. Error t value
(Intercept)   1.8210     0.0488    37.3
StimStim2     0.1394     0.0437     3.2
StimStim3     0.2422     0.0437     5.5
StimStim4     0.2855     0.0437     6.5
StimStim5     0.1710     0.0437     3.9
StimStim6     0.0227     0.0437     0.5
StimStim7    -0.2001     0.0437    -4.6
StimStim8    -0.5232     0.0437   -12.0
StimStim9    -0.8615     0.0437   -19.7
StimStim10   -1.2535     0.0437   -28.7
```

Now the residual variance does match that obtained with `aov` and `lm`. The variances of the estimated random components differ, but evaluation of the model with the `anova` method demonstrates that the calculated F values are the same.

```
> anova(mfGab.lmer2)
Analysis of Variance Table
     Df Sum Sq Mean Sq F value
Stim  9   28.2    3.14     283
```

The additional random effect, `Obs:Stim` provides for a random variation of the shape of the contrast sensitivity in addition to the height. The reduction in the AIC value as well as the results of a likelihood ratio test obtained using the `anova` method indicates that the data are indeed better described with this additional random component.

```
> anova(mfGab.lmer, mfGab.lmer2)
Data: mfGab
Models:
mfGab.lmer: LContSens ~ Stim + (1 | Obs)
mfGab.lmer2: LContSens ~ Stim + (1 | Obs) +
  (1 | Obs:Stim)
           Df  AIC  BIC logLik Chisq Chi Df Pr(>Chisq)
mfGab.lmer  12 -550 -497    287
mfGab.lmer2 13 -731 -673    379   183      1   <2e-16
                                                   ***
---
Signif. codes:  0 `***' 0.001 `**' 0.01 `*' 0.05 `.'
                0.1 ` ' 1
```

The AIC, BIC and log likelihood values displayed in the output of the `anova` method differ from those reported by the `print` and `summary` methods. The default method for estimating the parameters of the model is termed REML which stands for restricted or residual maximum likelihood. It is described by Pinheiro and Bates [141] as corresponding "to assuming a locally uniform prior distribution for the fixed effects β and integrating them out of the likelihood." The REML method is often preferred because it leads to an unbiased estimate of the variance [11]. Unlike the maximum likelihood criterion, however, REML is not invariant to monotone transformations of the fixed effects, so that two REML fits with different fixed effects structures cannot be compared as nested models using likelihood ratio tests. When evaluating fixed effects, one is advised to refit the models with the argument `REML = FALSE` set to obtain fits by MLE before comparing them using `anova`. Even doing this, however, the test is only approximate, and the p-values tend to be anti-conservative (too low), so that more stringent cut-offs should be employed for evaluating significance. The `anova` method appears to adjust the likelihoods to the maximum likelihood values before performing the test.

An astute reader will have noticed that the printed summary of the results of `lmer` does not include p-values for the fixed effect coefficients. The reasons for not including these in the output are outlined by one its authors, Douglas Bates, at https://stat.ethz.ch/pipermail/r-help/2006-May/094765.html. In brief, for general applications of these models, i.e., with data that can be unbalanced, nested at multiple levels and/or with crossed random effects, there is a difficulty in knowing how to calculate the degrees of freedom for the t-statistics that are shown (or the denominator degrees of freedom for the F-statistic, which would be the square of the t-statistic). Without these, the p-values cannot be computed. Other methods do exist, however, for assessing the significance of terms or calculating confidence intervals, using simulation. For example, the function `mcmcsamp` uses Markov Chain Monte Carlo methods to sample from the posterior distribution of the parameters of a fitted model. This produces an object that can be passed to the function `HPDinterval` to return intervals based on the posterior density estimates. These are referred to as Highest Posterior Density (or, sometimes, credible) intervals to distinguish them from classical confidence intervals. As a simple example, we evaluate the parameter distributions from the `mfGab.lmer2` model based on 1,000 simulated samples from the posterior distribution and print out their 95% credible intervals.

```
> mfGab.mcmc <- mcmcsamp(mfGab.lmer2, n = 1000)
> HPDinterval(mfGab.mcmc)
$fixef
               lower   upper
(Intercept)  1.7572  1.8844
StimStim2    0.0782  0.1995
StimStim3    0.1839  0.3095
StimStim4    0.2219  0.3496
StimStim5    0.1071  0.2311
StimStim6   -0.0408  0.0834
StimStim7   -0.2619 -0.1434
StimStim8   -0.5888 -0.4624
StimStim9   -0.9202 -0.7991
StimStim10  -1.3155 -1.1908
attr(,"Probability")
[1] 0.95

$ST
     lower upper
[1,] 0.441 0.634
[2,] 0.574 0.960
attr(,"Probability")
[1] 0.95

$sigma
     lower upper
[1,] 0.117 0.134
attr(,"Probability")
[1] 0.95
```

`HPDinterval` returns a list of three components providing the intervals for the fixed-effects coefficients, the random effects and the residual variance.

In a mixed-effects model, there are multiple levels of variation, corresponding to the fixed effects and however many levels of random effects are included in the model. For example, in the second model, there are two levels of random effects: at the level of the observer and at the level of the stimulus within the observer. We obtain the fitted values from each model object using the `fitted` method. Each observation was repeated 4 times so there are 640 fitted values. The fixed effects coefficients estimated in the model are extracted from the model object with the `fixef` method.

```
> ( mfGab.fe2 <- fixef(mfGab.lmer2) )
(Intercept) StimStim2  StimStim3  StimStim4  StimStim5
     1.8210    0.1394     0.2422     0.2855     0.1710
  StimStim6 StimStim7  StimStim8  StimStim9 StimStim10
     0.0227   -0.2001    -0.5232    -0.8615    -1.2535
> mfGab.fe2[-1] <- mfGab.fe2[1] + mfGab.fe2[-1]
```

Since they are coded according to the treatment contrasts, we add the first coefficient to the subsequent coefficients to obtain the mean estimate at each spatial frequency.

The values of the the random coefficients, b in (2.9), evaluated at the parameter estimates are referred to either as the best linear unbiased predictors (BLUPS) or the conditional modes [11]. They are extracted from the model with the `ranef` method, which returns a list of data frames.

```
> str( mfGab.re2 <- ranef(mfGab.lmer2) )
List of 2
 $ Obs:Stim:'data.frame':   160 obs. of  1 variable:
  ..$ (Intercept): num [1:160] -0.02722 -0.00808
     0.02763 -..
 $ Obs     :'data.frame':   16 obs. of  1 variable:
  ..$ (Intercept): num [1:16] -0.0302 -0.0297 -0.0245
     -0.1..
 - attr(*, "class")= chr "ranef.mer"
```

The fixed effects coefficients represent the estimates for the population contrast sensitivity function whereas by combining the conditional modes with the fixed effects, we obtain predictions for the individuals in the sample.

```
> mfGab.pred2 <- mfGab.fe2 +
+   rep(mfGab.re2[[2]][, 1], each = length(mfGab.fe2)) +
+   ranef(mfGab.lmer2)[[1]][, 1]
```

The second term above corresponds to the observer specific effects and so its values are repeated for each spatial frequency, while the third term for the `Obs:Stim` effect is already of the correct length and in the correct order.

We can now compare predictions at different levels in a `lattice` plot with the mean contrast sensitivity values, calculated from the `mfGab` data frame (Fig. 2.4). We leave it as an exercise to make the same graph and comparisons for the simpler model, `mfGab.lmer`.

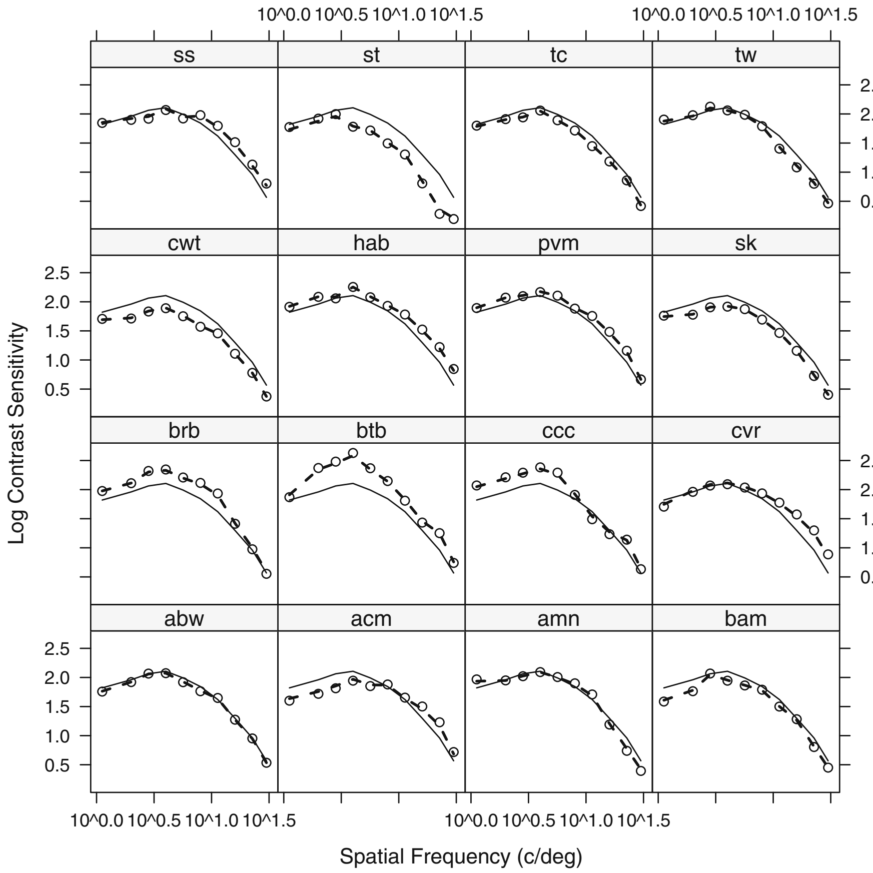

Fig. 2.4 Average contrast sensitivity from the `ModelFest` data set fitted with a mixed-effects model. The *solid lines* indicate the fixed effect prediction and the *dashed lines* have been adjusted for each observer by the estimates of the conditional modes

```
> Thr.mean <- t(with(mfGab, tapply(LContSens, list(Obs, Stim),
    mean)))
> Thr.df <- stack(as.data.frame(Thr.mean))
> names(Thr.df) <- c("Mean", "Obs")
> Thr.df$SpatFreq <- SpatFreq
> Thr.df$pred2 <- mfGab.pred2
> print(
+ xyplot(Mean ~ SpatFreq | Obs, Thr.df, subscripts = TRUE,
+   id = Thr.df$pred2, scales = list(x = list(log = TRUE)),
+   xlab = "Spatial Frequency (c/deg)",
+   ylab = "Log Contrast Sensitivity",
+   panel = function(x, y, subscripts, id, ...){
```

```
+         panel.xyplot(x, y)
+         panel.lines(x, mfGab.fe2)
+         panel.lines(x, id[subscripts], lty = 2, lwd = 2)
+    } )
+ )
```

Additional reduction in the number of parameters to describe these data can be obtained by using the spatial frequency as a covariate rather than as a factor. The dependence of contrast sensitivity on spatial frequency is not linear, however. One could retain the linear approach by using polynomial functions of spatial frequency or spline curves (see the **splines** package [146]), but then one must decide on the degree of complexity (order of the polynomial, number of knots or degrees of freedom for a spline) using a model selection criterion such as AIC, BIC or cross-validation. Other approaches include additive models (see Sect. 2.5) and nonlinear models (see Sect. 2.4). In each of these cases, there exist functions in R or associated packages for extending these to mixed-effects models, though we will not pursue such approaches further here.

2.4 Nonlinear Models

The response cannot always be characterized as a linear function of the explanatory variables. One alternative is to consider *nonlinear models*. In this section we illustrate how to develop and fit one class of nonlinear model using R. These functions have the form

$$Y = f(\boldsymbol{X};\beta) + \varepsilon \tag{2.10}$$

where Y is an n-dimensional vector of responses (dependent variable), $\boldsymbol{X}$ an n-by-p matrix of explanatory variables used in predicting the response, β is a p-dimensional vector of parameters, $f()$, a function and typically $\varepsilon \sim N(0,\sigma^2)$. When $f(\boldsymbol{X};\beta) = \boldsymbol{X}\beta$, the nonlinear model simply reduces to the linear model. The R function `nls` ("*n*onlinear *l*east-*s*quares") determines estimates of parameters $\hat{\beta}$ that minimize the sum of the squared errors, $Y - f(\boldsymbol{X};\hat{\beta})$, using by default a Gauss–Newton algorithm in an iterative search process. When $\varepsilon \sim N(0,\sigma^2)$, then the least-squares solution is also the maximum likelihood solution (Sect. B.4.2). R includes several functions that permit solving this type of problem, for example, the functions `optim` and `nlm`. Each of these requires the user to write a new function to calculate the sum of squared error for each new model considered. The function `nls` which we describe here is a convenient alternative that uses formula objects to specify the model to be fit.

2.4.1 Chromatic Sensitivity Across the Life Span

Knoblauch et al. [99] measured thresholds as a function of age for detecting equiluminant chromatic differences along three axes in the CIE *xy* chromaticity diagram. The ages in their sample ranged from 3 months to 86 years old, covering periods of the life span in which both developmental and aging factors would be expected to be operating. Pre-verbal participants were evaluated with a preferential-looking technique [170]. The data set `Chromatic` in the **MPDiR** package is obtained by

```
> data(Chromatic)
```

and contains four components: `Log2Age`, the base 2 logarithm of the age of each observer, `Age`, the age in years, `Thresh`, the threshold modulation for detection of a chromatic stimulus and `Axis`, a 3-level factor indicating along which of the three axes in the color space the observation was measured. The axes names—Protan, Deutan and Tritan—correspond to the color confusion axes for each of the three types of congenital dichromatic observer and, in theory, permit testing the sensitivity of each of the three classes of cone photoreceptor in the normal eye.

Figure 2.5 shows scatterplots of the thresholds as a function of age for each of the axes tested and was produced using the **ggplot2** package with the following code fragment.

```
> library(ggplot2)
> qplot(Age, Thresh, data = Chroscamatic,
+     facets = .~ Axis, col = "black",
+     geom = c("point"),
+     xlab = "Age (years)", ylab = "Chromatic
              Threshold") +
+     scale_x_continuous(trans = "log2",
+         limits = 2^c(-3.5, 7.5),
+         breaks = 2^seq(-2, 6, 2),
+         labels =   2^seq(-2, 6, 2) ) +
+     scale_y_log10(limits = 10^c(-3.5, -1.3),
+       breaks = c(0.001, 0.002, 0.005, 0.01, 0.02,
                   0.05),
+   labels = c(0.001, 0.002, 0.005, 0.01, 0.02, 0.05))+
+      geom_smooth(colour = "black")
```

Both the ordinates and abscissas are logarithmically spaced as in the original publication (log base 10 on the ordinate and base 2 on the abscissa). A smooth loess curve has been added to each plot in black. This initial view of the data suggests that, in these coordinates, there are two trends in the data, an initial roughly linear decrease in threshold with age (the developmental trend), followed by a loss in sensitivity later in life (the aging trend). The graphs also suggest that the same functional form might adequately describe these trends along each axis, though the thresholds are higher along the tritan axis.

The authors fit these data with the following function,

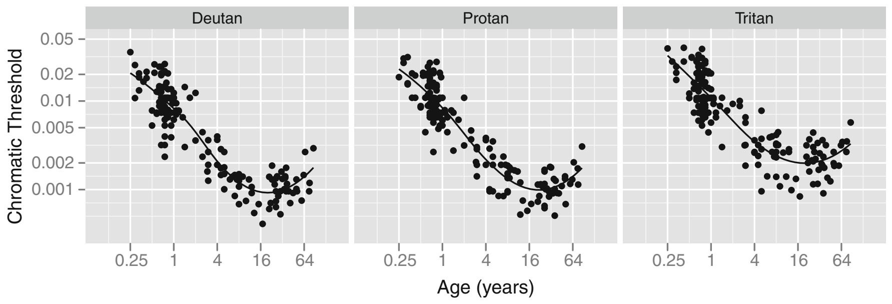

Fig. 2.5 Thresholds for detection of chromatic modulation as a function of age along three axes in color space. The *fitted curve* is obtained by local regression (loess)

$$T = aA^{-\alpha} + bA^{\alpha}, \tag{2.11}$$

where T is the threshold estimate, A the age, and a, b and α are parameters to be estimated. On log-log coordinates, the magnitude of the slopes of the two segments of the curve will approach α.

The `nls` function in the **stats** package [146] in R facilitates estimation of the free parameters in (2.11). The parameters are chosen to minimize the sum of the squared differences between data and predictions, a least squares criterion.

To use the function `nls` we need to specify the model as a formula object, just as we did for `lm`. The model (2.11) can be fit to the data by the following code:

```
> Chrom.nls <- nls(Thresh ~
+        a[Axis] * Age^-alph + b[Axis] * Age^alph,
+        data = Chromatic,
+        start = list(a = c(8, 8, 11) * 10^-3,
+              b = c(3, 3, 8) * 10^-5, alph = 1)
+                )
```

Three arguments are required. First, the model is specified as a formula. Unlike in the `lm` formulae, the asterisk, "`*`," here corresponds to multiplication and the circumflex, "`^`" to exponentiation. Because the terms `a` and `b` are subscripted by the factor `Axis`, different values of each will be estimated for each level of `Axis`. In this case, if an intercept term is desired, it must be included explicitly. In general, the interpretation of a formula object depends on the function that calls it, as we see here and have already seen with `lm` and `xyplot`. The "data" argument, as always, provides the frame for interpreting the terms in the formula object. The argument "start" is a list indicating initial values for the parameters. Because of the subscripting of the linear coefficients in the model, there are seven parameters in all for which to provide initial values and to estimate (three values of `a`, one for each level of `Axis`, three for `b` and one for `alph`).

The function `nls` returns an object of class "nls" containing information about the non-linear, least-squares fit. Over a dozen methods are defined for probing

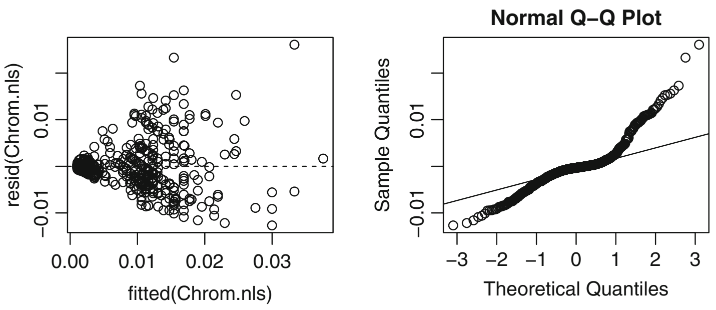

Fig. 2.6 Two diagnostic plots of the residuals obtained from fitting model (2.11) to the `Chromatic` data set with errors on a linear scale

objects of this class. The `summary` method prints out a table of the results in a format similar to that from a linear model.

```
> summary(Chrom.nls)
Formula: Thresh ~ a[Axis] * Age^-alph + b[Axis] *
          Age^alph

Parameters:
     Estimate Std. Error t value Pr(>|t|)
a1   8.36e-03   3.91e-04   21.40   <2e-16 ***
a2   8.88e-03   3.96e-04   22.43   <2e-16 ***
a3   1.21e-02   4.39e-04   27.65   <2e-16 ***
b1   3.42e-05   3.86e-05    0.89    0.376
b2   3.39e-05   3.86e-05    0.88    0.380
b3   8.61e-05   3.90e-05    2.21    0.028 *
alph 8.15e-01   4.14e-02   19.72   <2e-16 ***
---
Signif. codes:  0 `***' 0.001 `**' 0.01 `*' 0.05 `.'
                0.1 ` ' 1

Residual standard error: 0.00486 on 504 degrees of
   freedom

Number of iterations to convergence: 4
Achieved convergence tolerance: 4.44e-07
```

The diagnostic residuals vs. fitted values and Q–Q plots are displayed in Fig. 2.6 and were obtained with the following code fragment:

```
> par(mfrow = c(1, 2))
> plot(fitted(Chrom.nls), resid(Chrom.nls))
> abline(0, 0, lty = 2)
```

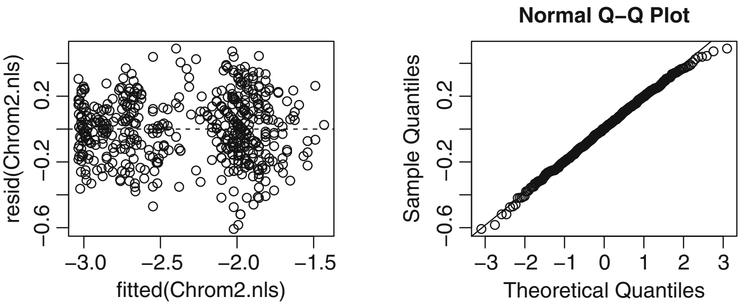

Fig. 2.7 Diagnostic plots of fit to `Chromatic` data set with the errors evaluated on a logarithmic scale

```
> qqnorm(resid(Chrom.nls))
> qqline(resid(Chrom.nls))
```

These plots reveal problems in the assumptions of the model. The left residuals plot displays heterogeneity of the residuals as a function of the fitted values. The Q–Q plot is sigmoidal, not linear. This pattern is consistent with a distribution that has too many extreme values relative to the Gaussian. The pattern of residuals leads us to minimize the logarithm of the errors rather than the errors. This change is easily implemented by taking logarithms of both sides of the formula object.

```
> Chrom2.nls <- update(Chrom.nls, log10(.) ~ log10(.))
```

The diagnostic plots for the new fit in Fig. 2.7 now appear reasonable.

The model includes the assumption that the same exponent, α can be used on each axis. We can consider a more complex model where we assign a different exponent for each axis. To do so, we add the subscript `Axis` to the term `alph`, and a model in which the exponent varies with chromatic axis is obtained. The single exponent model is nested in the multi-exponent model, and so we can readily test whether the added complexity of the multi-exponent model is needed to fit the data. The `anova` method takes a series of "nls" objects as arguments that correspond to nested models and performs a likelihood ratio test between each pair of models in the series. We first fit the more complex model. Notice that we had to update the start parameter list (the third argument) as well, since the new model has two more parameters.

```
> Chrom3.nls <- update(Chrom2.nls, . ~
+   log10(a[Axis] * Age^-alph[Axis] +
+   b[Axis] * Age^alph[Axis]),
+   start = list(a = c(8, 8, 11) * 10^-3,
+   b = c(3, 3, 8) * 10^-5, alph = c(1, 1, 1)))
> anova(Chrom2.nls, Chrom3.nls)
Analysis of Variance Table
```

```
Model 1: log10(Thresh) ~ log10(a[Axis] * Age^-alph +
    b[Axis] * Age^alph)
Model 2: log10(Thresh) ~ log10(a[Axis] *
   Age^-alph[Axis] +
     b[Axis] * Age^alph[Axis])
  Res.Df Res.Sum Sq  Df  Sum Sq F value Pr(>F)
1    504      19.10
2    502      18.97   2   0.13    1.78   0.17
```

The results indicate that the simpler model with one exponent describes the data adequately. This conclusion is in agreement with that of the original article based on evaluation of the confidence intervals of the parameters.

The **MASS** package provides a `confint` method to compute confidence intervals for "nls" objects [178].

```
> library(MASS)
> confint(Chrom2.nls)
         2.5%    97.5%
a1   6.78e-03 7.94e-03
a2   7.26e-03 8.50e-03
a3   9.84e-03 1.16e-02
b1   2.29e-05 3.63e-05
b2   2.31e-05 3.68e-05
b3   5.66e-05 8.71e-05
alph 8.51e-01 9.40e-01
```

The reasonableness of the linear approximation involved can be evaluated by plotting the output of the `profile` method (left as an exercise). If the linear approximation is found to be inappropriate, bootstrap confidence intervals [55] can be obtained instead using the package `boot` [25].

The estimated parameters and fits to the data of the model `Chrom2.nls` shown in Fig. 2.8 match those reported by the authors.

2.5 Additive Models

Nonlinear models are particularly useful when the equation is motivated by an underlying theory and the estimated parameters can be given a meaningful interpretation, either with respect to the data or within the framework of the model. In the absence of theory, finding the best model to describe the data depends in part on the ability of the modeler to conjure up a "good" equation, i.e., one that fits the data well with a minimum number of parameters.

Additive models (AM) provide a flexible, nonparametric approach to describing data that vary nonlinearly as a function of a covariate in a manner that remains close to the framework of linear models [77, 198]. The additive model is of the form

$$Y = f_1(X_1) + f_2(X_2) + \cdots + \varepsilon, \qquad (2.12)$$

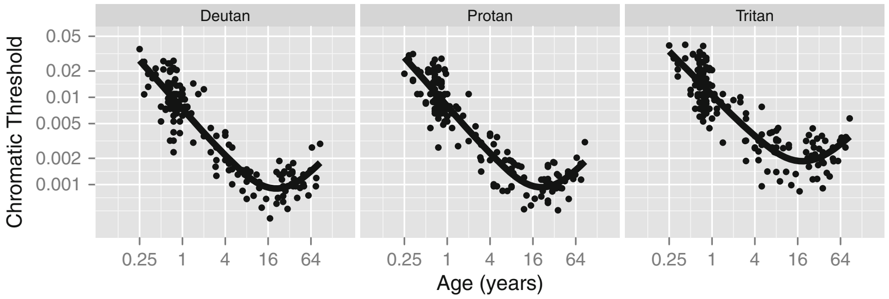

Fig. 2.8 Chromatic modulation thresholds as a function of age along three color axes. The *black curves* were obtained with the fitted values of model `Chrom2.nls` in which the exponent is the same in all color directions

where Y is the response, the f_i are smooth functions of the covariates, X_i and $\varepsilon \sim N(0, \sigma^2)$. The smooth functions are typically represented in terms of linear combinations of smooth basis functions, for example, splines, and, thus, can be added as columns to the model matrix. With a greater number of basis terms, more complex variations of the dependent variable can be described up to the limit of having the predicted curve pass through every point. Such a curve that describes the trend as well as the noise in the data will, in general, be poor at describing new samples from the same population. To control for this, a penalty for undersmoothing is introduced during the optimization. The penalty is usually implemented as a proportion, λ of the integrated square of the second derivative of f (related to its curvature). Larger contributions of the term result in less smooth models. The choice of degree of penalization (or smoothness) is controlled by minimizing a criterion related to prediction error (i.e., fitting some of the data and calculating the error on the remaining portion) called generalized cross validation (GCV).

2.5.1 Chromatic Sensitivity, Again

Additive models can be fit in R using the function `gam` in the **mgcv** package, one of the recommended packages that comes with R by default [198]. It works similarly to the other modeling functions that we have seen, requiring that the model be specified by a formula object. The novelty for `gam` is that smooth terms of a covariate are specified by including them as arguments to a function `s`. For example, the following code loads the package and fits an AM to the `Chromatic` data set and then prints out a summary of the fit.

```
> library(mgcv)
> Chrom.gam <- gam(log10(Thresh) ~ s(Age, by = Axis)
     + Axis,
+   data = Chromatic)
> summary(Chrom.gam)
Family: gaussian
Link function: identity
Formula:
log10(Thresh) ~ s(Age, by = Axis) + Axis

Parametric coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  -2.3830     0.0158 -150.50   <2e-16 ***
AxisProtan    0.0264     0.0224    1.18     0.24
AxisTritan    0.2040     0.0226    9.04   <2e-16 ***
---
Signif. codes:  0 `***' 0.001 `**' 0.01 `*' 0.05 `.'
  0.1 ` ' 1

Approximate significance of smooth terms:
                   edf Ref.df    F p-value
s(Age):AxisDeutan 7.75   8.59 94.5  <2e-16 ***
s(Age):AxisProtan 8.21   8.83 94.4  <2e-16 ***
s(Age):AxisTritan 7.80   8.63 63.9  <2e-16 ***
---
Signif. codes:  0 `***' 0.001 `**' 0.01 `*' 0.05 `.'
  0.1 ` ' 1

R-sq.(adj) =  0.817   Deviance explained = 82.6%
GCV score = 0.045504  Scale est. = 0.043121  n = 511
```

A smooth function of the covariate `Age` has been specified as an explanatory variable. The argument `by = Axis` indicates that a separate smooth function should be fit to the data for each level of the factor. Here, we include an additive contribution of the factor `Axis` for identifiability, since the smooth terms will be centered around 0.

Examining the parametric coefficients, the summary method indeed indicates a difference in the vertical height along the "Tritan" but not the "Protan" axis. The new aspect of the display is the inclusion of the results for the smooth terms. The cross-validation selection of the degree of smoothing produces a non-integer estimate of the degrees of freedom, the estimated (or effective) degrees of freedom (edf) for each smooth term.

We test the above model against the simpler model that the variation in chromatic sensitivity across the life span follows the same course along all three chromatic axes by fitting a single smooth for all three axes. The `update` method facilitates fitting the new model.

```
> Chrom2.gam <- update(Chrom.gam, .  ~ s(Age) + Axis)
> anova(Chrom2.gam, Chrom.gam, test = "F")
```

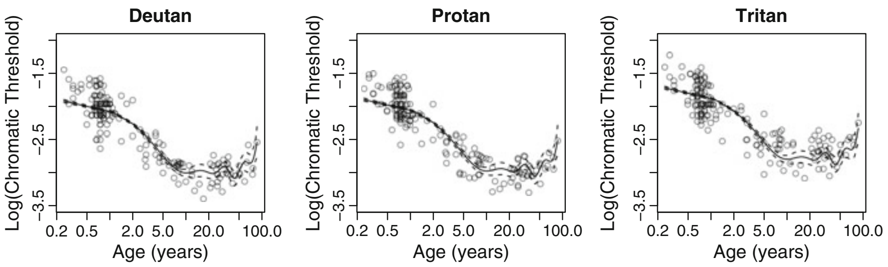

Fig. 2.9 Fits of the smooth function to the `Chromatic` data set for model `Chrom2.gam`, obtained using `gam`

```
Analysis of Deviance Table

Model 1: log10(Thresh) ~ s(Age) + Axis
Model 2: log10(Thresh) ~ s(Age, by = Axis) + Axis
  Resid. Df Resid. Dev   Df Deviance   F Pr(>F)
1       499       21.6
2       484       20.9 15.1    0.714 1.1   0.36
```

An approximate *p*-value from a likelihood test comparing the two models (see `?anova.gam` and [198]) is obtained using the `anova` method and provides supporting evidence for a single curve describing the evolution of sensitivity with age. Unlike in the case of the nonlinear models, this inference does not depend on having chosen the correct equation to describe the data.

A plot of the simpler model with the data is shown in Fig. 2.9 for each of the three axes, using functions from the base **graphics** package. The format of the graphical output is different from that of "lm" objects. The `plot` method for "gam" objects generates a graph of the estimated smooth curve with standard errors of the estimated smooth indicated by dashed lines.[3] The graphs are output with a `for` loop. We use the estimated coefficients for the factor `Axis` to adjust the heights of the single smooth to the data along each axis. The points were plotted using transparency (the fourth parameter of the `rgb` function) to allow the curve to be visualized better behind the crowd of points. The fitted smooth function differs from the parametric curve used with `nls` at the youngest ages, providing evidence against (2.11) in this age range, though the oscillatory structure of the curve at the older ages suggests that additional smoothing might be appropriate. Such smoothing could conceivably lessen the differences between the two fits.

```
> opar <- par(mfrow = c(1, 3))
> cc <- coef(Chrom2.gam)[1:3] + c(0, coef(Chrom2.gam)[1],
```

[3]Use the `gam.check` function on the model object to visualize diagnostic plots that, in this case, raise no flags on the adequacy of the fit. We leave this verification as an exercise for the reader.

```
+          coef(Chrom2.gam)[1])
> names(cc) <- c("Deutan", "Protan", "Tritan")
> for (Ax in names(cc)) {
+   plot(Chrom2.gam, rug = FALSE,
+       shift = cc[Ax],
+       ylim = c(-3.5, -1) - cc[Ax],
+       main = Ax, log = "x",
+       xlab = "Age (years)",
+       ylab = "Log(Chromatic Threshold)")
+   points(log10(Thresh) ~ Age, Chromatic,
+       subset = Axis == Ax,
+       col = rgb(0, 0, 0, 0.5))
+ }
> par(opar)
```

2.6 Generalized Linear Models

GLM [127] extend linear models in two exceedingly useful ways. First, the dependent variable is not necessarily assumed to be distributed as a Gaussian random variable. The choice of possible distributional families now includes the Gaussian but also the Bernoulli, binomial, Poisson and many others that are members of what is called the *exponential family* (Sect. B.3.5). We can use the GLM to model data whose dependent variable has any of these distributions. The Bernoulli distribution (See Sect. B.2.2) is a particularly important case for us since we will often use it to model binary responses ("Yes-No," etc.) in psychophysical experiments. Warning: The term "exponential family" is potentially confusing. The members of the exponential are themselves distributional families. Moreover, the exponential distributional family (Sect. B.3.4) is just another member of the exponential family; it has no special status.

Second, the response is modeled as a nonlinear transformation of a weighted linear combination of the explanatory variables. The nonlinear transformation is referred to as the *link function*. This sort of model is remarkably common in the psychophysical literature.

Familiar linear models (such as ANOVA) can be used to develop analogous GLM models (e.g. "GLM ANOVA") that use a very similar terminology in describing them while accommodating nonlinear transformations of the response variable.

The linear predictor takes the form

$$\eta = \boldsymbol{X}\beta \tag{2.13}$$

where as with the ordinary linear model the design matrix $\boldsymbol{X}$ encapsulates all of the information about the explanatory variables and β is a vector of unknown parameters to be estimated. The relation of the linear predictor to the expected value of the response (or the mean function) is written as

$$g(\mathsf{E}[Y]) = \boldsymbol{X}\beta, \tag{2.14}$$

where g is the link function. The link function is chosen to be invertible.

Thus, in the Bernoulli case, we would have

$$g(\mathsf{E}[P(Y=1)]) = \log\left(\frac{p}{1-p}\right) = \boldsymbol{X}\beta. \tag{2.15}$$

Note that the link function here is the logit transformation but as we will see, other functions can also be used. Inverting g, yields

$$\mathsf{E}[P(Y=1)] = g^{-1}(\boldsymbol{X}\beta), \tag{2.16}$$

where we can think of the function $g^{-1}() = \psi()$ as a *psychometric function* that transforms an intensity variable into a probability of choosing a particular response in a binary task.

To summarize, a GLM is characterized by three components:

1. The responses Y_i are independently and identically distributed and sampled from the same member of the exponential family
2. The model contains a set of explanatory variables represented in a matrix $\boldsymbol{X}$ and coefficients in a vector β that together form a linear predictor
3. The linear predictor is related to the expected value of the μ by an invertible link function, g

2.6.1 Chromatic Sensitivity, Yet Again

The response variable for the previously analyzed `Chromatic` data set can also be modeled using GLM as shown next. Recall that fitting raw thresholds resulted in a distribution of residuals that violated the Gaussian assumption (Fig. 2.6). In Sect 2.4.1, we employed a log transform to stabilize the variance. The log transform requires the data values to be positive and, in fact, the diagnostic plots for the model fit to the transformed data in Fig. 2.7 are consistent with a log normal distribution of the raw threshold values.

It is plausible to consider modeling the raw data with other distributions defined only on positive values and that show a similar dependence of variance on the mean. For example, an interesting candidate within the family of distributions amenable to the GLM is the Gamma distribution. For a Gamma distributed variable, the standard deviation increases linearly with the mean, a pattern that could produce the fanshaped pattern of residuals in Fig. 2.6. An advantage of the GLM approach is that we can analyze the data on their original scale.

In order to apply the GLM, we must be able to specify the model in the form of a linear predictor. We note that the exponent, α, estimated for (2.11) is close to unity. If we set it to this value, the model reduces to a 2-term polynomial that can be represented as a linear predictor.

```
> Chrom.glm <- glm(Thresh ~ Axis:(I(Age^-1) + Age),
+   family = Gamma(link = "identity"), data = Chromatic)
```

Recall that exponents in the formula object language are used to generate interaction terms in linear models. To protect against this interpretation and force the exponent to be interpreted as an arithmetic operation, it must be enclosed by the function `I`. We model the thresholds as an interaction with the sum of the age covariates so that there will be coefficients for the developmental and aging sections of the data and the possibility of different coefficients for each axis. The `Gamma` family is specified with an identity link so that the untransformed thresholds are analyzed.

As before, we can compare this model to one in which the coefficients are forced to be the same along all three axes though the heights of the curves can depend on the axis.

```
> Chrom2.glm <- update(Chrom.glm, . ~ Axis + (I(Age^-1)
   + Age))
> anova(Chrom2.glm, Chrom.glm, test = "F")
Analysis of Deviance Table

Model 1: Thresh ~ Axis + I(Age^-1) + Age
Model 2: Thresh ~ Axis:(I(Age^-1) + Age)
  Resid. Df Resid. Dev Df Deviance    F Pr(>F)
1       506        101
2       504        100  2    0.506 1.23   0.29
```

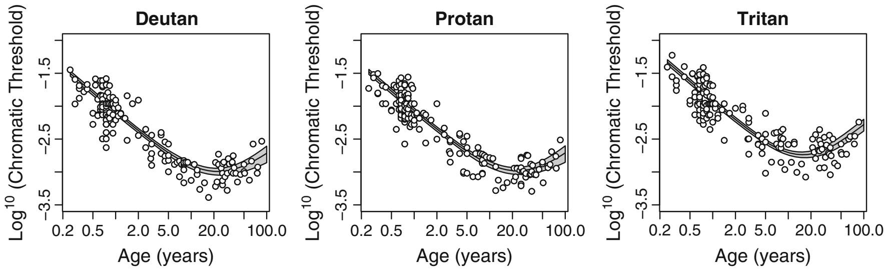

Fig. 2.10 GLM fits to the `Chromatic` data set for model `Chrom2.glm`

```
> opar <- par(mfrow = c(1, 3))
> xx <- 10^seq(-0.6, 2, len = 200)
> for (Ax in levels(Chromatic$Axis)){
+   plot(log10(Thresh) ~ Age, Chromatic,
+       subset = Axis == Ax,
+       xlab = "Age (years)",
+       ylab = expression(paste(plain(Log)[10], " (Chromatic
+            Threshold)")),
+       main = Ax, log = "x",
+       ylim = c(-3.5, -1))
+   Chrom.pred <- predict(Chrom.glm,
+       newdata = data.frame(Age = xx, Axis = Ax),
+       se.fit = TRUE)
+   polygon(c(xx, rev(xx)), with(Chrom.pred, log10(c(fit + 1.96
+       * se.fit, rev(fit - 1.96 * se.fit)))), col = "grey")
+   lines(xx, log10(Chrom.pred$fit))
+   points(log10(Thresh) ~ Age, Chromatic, subset = Axis == Ax,
+       pch = 21, bg = "white")
+ }
> par(opar)
```

Finally, we plot the predicted curves from the model `Chrom2.glm` with the data in Fig. 2.10. Even though the thresholds were fit in raw units, we scale the axes logarithmically to compare the data more easily with the previous plots and because it spreads out the data improving the visual display. By specifying the argument `se.fit = TRUE` in the `predict` method, standard error estimates are returned as well as the predicted values. We use these to plot a 95% confidence interval on the fit as a grey envelope around the predicted curve. The diagnostic plots should be examined systematically during the fitting process. In this case, they look reasonable. We leave verification of this as an exercise for the reader.

2.7 Which Model to Choose?

We analyzed the `Chromatic` data set using three different approaches. It is reasonable to ask which type of model is best or when is one to be preferred over the others. In R, all three were easily applied to the data set thanks to the consistent interface to the modeling functions, and subsequent analyses were facilitated by the method functions. Each permits interesting characteristics of the data to be described and tested.

If theory prescribes a particular nonlinear equation to describe a data set and the Gaussian assumption on the residual errors is reasonable, then the approach using nonlinear least squares with the `nls` function is recommended. If no obvious functional form is available for the data or the exact functional form is unimportant, then the use of `gam` is worth considering. In the event of doubt about the best choice of functional form to describe a data set, the possibility of drawing conclusions about similarities across data sets nonparametrically, as demonstrated above, certainly provides a powerful option. If the model can be expressed as a linear predictor, then `lm` or `glm` should certainly be considered. The choice of link function for the `glm` will allow certain nonlinear transformations of the dependent variable to be handled. In addition, the choice of exponential family member affords some flexibility in accounting for the distribution and variance of the dependent variable.

An advantage of `glm` is that the predicted responses of the mean of the response variable are on the untransformed scale, whereas if we were to fit a linear model to the transformed responses (e.g., to the logit transformation in (2.15)), then we would be estimating the mean logit value rather than the mean probability. We demonstrate this difference with a simple example. We generate five random numbers from a binomial distribution using `rbinom`. We assume that each is the number of successes observed out of 30 trials and we would like to estimate the probability of a success, assuming that it is constant across samples. For the example, we set the underlying value of p to 0.8. We use `glm` with a binomial family and the default logit link to estimate a model with only an intercept term and compare the estimated coefficient on the response scale with the mean proportion of successes of the sample and the mean of the logits.

```
> NumSuccess <- rbinom(5, 30, 0.8)
> pSuccess <- NumSuccess/30
> resp.mat <- cbind(NumSuccess, 30 - NumSuccess)
> pSuccess.glm <- glm(resp.mat ~ 1, binomial)
> # glm estimate
> plogis(coef( pSuccess.glm ))
(Intercept)
      0.787
> # mean proportion Success
> mean(pSuccess)
[1] 0.787
> # mean of logits on response scale
> plogis(mean(qlogis( pSuccess )))
[1] 0.801
```

The model returns the estimated coefficient on the linear predictor scale so we transform it to the response scale with the `plogis` function.

2.8 Exercises

2.1. Create two factors `A` and `B`, `A` having 3 levels as in Sect. 2.2 and `B` with only 2. Use `expand.grid` to make a data frame with two columns representing a factorial crossing of the levels of the two factors. Then, with `model.matrix` generate the design matrices for formula objects representing (a) a model containing only additive effects of the two factors and (b) also, including their interaction. Study the design matrices so that you understand them with respect to the levels of each factor and terms in the model. Repeat this exercise with the intercept removed from the formula object.

2.2. The design matrix indicates the contributions of the model terms to an estimated coefficient but it does not indicate the contrasts in a direct fashion. To visualize the contrasts, one may use the `ginv` function from the **MASS** package on the design matrix. The contrasts will be in the rows of the resulting matrix. Examine the contrasts for a model with (a) just factor `A`, (b) only additive effects of factors `A` and `B`, (c) additive effects of the two factors and their interaction. Using the `fractions` function from the same package will aid in the interpretation of the contrasts. Using the contrasts, interpret the relation of each estimated coefficient to the terms in the model.

2.3. Re-fit the model (2.8) for the `Motion` data set but rescale the values of `LnAge` to center them with respect to their mean, using the function `scale`. How does rescaling affect the estimated coefficients, their precision and their interpretation?

2.4. In the `Motion` data set, how would you test the significance of the 3-way interaction `LnAge:Mtype:Sex`?

2.5. Random effects can enter a model as covariates as well as factors. A random covariate `x` would be specified with respect to a grouping factor `f` as a term `(x + 0 | f)`. Recall that the intercept term is implicit in a model formula so that it must be suppressed by adding a 0 or subtracting a 1 from the formula or else a random effect will be estimated for the intercept as well.

Refit the Gabor functions of the `ModelFest` data using a polynomial series of the covariate `log10(SpatFreq)` instead of the factor `Stim` as explanatory variables. Remember to isolate the terms with powers using the `I()` function. Evaluate how many terms of the polynomial are necessary to describe the average data and which terms need also be included as random effects to describe the individual contrast sensitivity curves.

2.6. Fit the `Chromatic` data set with `nls` using a different exponent on the developmental and aging segments (descending and ascending branches, respectively) and test whether the added parameters yield a significantly better fit.

2.7. In this chapter, the `Chromatic` data set was fit with nonlinear, generalized linear and additive models. Make a graph with the data for one chromatic axis and the predicted curves from each of the three models so as to compare visually the three fits.

2.8. The data set `CorticalCells` in the **MPDiR** package contains the mean responses in action potentials/second of six neurons recorded in visual cortical areas V1 and V2 of the macaque brain in response to increasing levels of stimulus contrast reported in a paper by Peirce [140]. Plot the contrast response data for each cell using either **lattice** or **ggplot2**. A typical model for the contrast response function of cortical cells is given by the Naka-Rushton equation,

$$R(c) = R_m \frac{c^n}{c^n + \sigma^n}, \tag{2.17}$$

a generalization of the Michaelis-Mention function for which $n = 1$. What is the interpretation of the parameters in this model? Fit this model to each data set using `nls`. Then, re-fit the models using the reciprocal of the `SEM` column of the data frame with the `weights` argument. How does this change the fits? Compare the fit to (2.17) and the same model with the exponent, n, constrained to equal 1, using a likelihood ratio test.

2.9. Peirce [140] considers a more general model that allows for the fall-off from saturation shown by some cells in this data set. His model is

$$R(c) = R_m \frac{c^{n_1}}{c^{n_2} + \sigma^{n_2}}. \tag{2.18}$$

Fit this model to the data sets and then for each data set perform a likelihood ratio test between the fits of the models (2.17) and (2.18). Repeat this exercise using the weights. Replot (or update) the plots from the beginning of this exercise to include predicted curves from each model.

2.10. Watson [182] models the average contrast sensitivity data from the `ModelFest` data set (the first ten stimuli) with a difference of hyperbolic secants using four parameters

$$CS(f; f_0, f_1, a, p) = \operatorname{sech}\left((f/f_0)^p\right) - a \operatorname{sech}\left(f/f_1\right), \tag{2.19}$$

where f is spatial frequency and

$$\operatorname{sech}(x) = \frac{2}{e^x + e^{-x}}. \tag{2.20}$$

Calculate the average of the log contrast sensitivities for the 16 observers in this data set and fit this model to the data. Refit the model using the reciprocal of the inter-observer standard deviations as weights (see the "weights" argument of `nls`). Refit the data using an additive model with a smooth function of spatial frequency.

2.11. The GLM can also be used for analyzing count data that would traditionally be modeled with a log-linear model [14]. Typically, one specifies a Poisson family and a log link. This is possible because the multinomial, on which log-linear models are based, and the Poisson distributions have the same sufficient statistics.

The `Grue` data set in the **MPDiR** package is taken from a study by Lindsey and Brown [113] in which they categorized the frequency of languages with respect to how they referred to the words "green" and "blue" (separate words, a combined word (so called grue languages) or the word " dark" for the two) as a function of environmental ultra-violet light exposure, `UV_B`. The data is in the format of a contingency table, wide format. Modify the data set so that it is data frame with three columns indicating the counts, the UV-B exposure and the type of language, the long format (see the `stack` function). Then, fit the data using `glm` and test the hypothesis that UV-B exposure is independent of language type.

Springer

http://www.springer.com/978-1-4614-4474-9

Modeling Psychophysical Data in R
Knoblauch, K.; Maloney, L.T.
2012, XV, 365 p. 103 illus., 4 illus. in color., Softcover
ISBN: 978-1-4614-4474-9