8

Regression Diagnostics for Linear, Generalized Linear, and Mixed-Effects Models

Regression diagnostics are methods for determining whether a fitted regression model adequately represents the data. In this chapter, we present methods for linear, generalized linear, and mixed-effects models, but many of the methods described here are also appropriate for other regression models. Because most of the methods for diagnosing problems in linear models fit by least squares extend naturally to generalized linear and mixed-effects models, we deal at greater length with linear-model diagnostics.

Regression diagnostics address the adequacy of a statistical model *after* it has been fit to the data. Careful thought about the problem at hand along with examination of the data (as in Chapter 3) *prior to* specifying a preliminary model often avoids problems at a later stage. Careful preliminary work doesn't, however, guarantee the adequacy of a regression model, and the practice of statistical modeling is therefore often one of iterative refinement. The methods described in this chapter can help you to reformulate a regression model to represent the data more accurately.

Linear models make strong and sometimes unrealistic assumptions about the structure of the data. When assumptions are violated, estimates and predictions can behave badly and may even completely misrepresent the data. The same is true of other parametric regression models. Regression diagnostics can reveal problems and often point the way toward solutions.

All of the methods discussed in this chapter either are available in standard R functions or are implemented in the **car** and **effects** packages. A few functions that were once in earlier versions of the **car** package are now a standard part of R.

An original and continuing goal of the **car** package is to make regression diagnostics readily available in R. It is our experience that diagnostic methods are much more likely to be used when they are *convenient*. For example, added-variable plots, described in Section 8.3.3, are constructed by regressing a particular regressor and the response on all the other regressors in the model, computing the residuals from these auxiliary regressions, and plotting one set of residuals against the other. This is not hard to do in R, although the steps are somewhat more complicated when there are factors, interactions, or polynomial or regression-spline terms in the model. The `avPlots()` function in the **car** package constructs all the added-variable plots for a linear or generalized linear model and adds such enhancements as a least-squares line and point identification.

- Section 8.1 describes various kinds of residuals in linear models.
- Section 8.2 introduces basic scatterplots of residuals, along with related plots that are used to assess the fit of a model to data.
- Subsequent sections are specialized to particular problems, describing methods for diagnosis and at least touching on possible remedies. Section 8.3 introduces methods for detecting unusual data, including outliers, high-leverage points, and influential cases.
- Section 8.4 returns to the topic of transforming the response and predictors (discussed previously in Section 3.4) to correct problems such as nonnormally distributed errors and nonlinearity.
- Section 8.5 deals with nonconstant error variance.
- Sections 8.6 and 8.7 respectively describe the extension of diagnostic methods to generalized linear models, such as logistic and Poisson regression, and to mixed-effects models.
- Diagnosing collinearity in regression models is the subject of Section 8.8.
- Although extensive, this chapter isn't intended to be encyclopedic. We focus on methods that are most frequently used in practice and that we believe to be most generally useful. There are consequently several diagnostic functions in the **car** package that aren't covered in the chapter. Section 8.9 briefly outlines these functions.

8.1 Residuals

Residuals of one sort or another are the basis of most diagnostic methods. Suppose that we specify and fit a linear model assuming constant error variance σ^2. The *ordinary residuals* are given by the differences between the responses and the fitted values,

$$e_i = y_i - \widehat{y}_i,\ i = 1, \ldots, n \tag{8.1}$$

In ordinary-least-squares (OLS) regression, the residual sum of squares is equal to $\sum e_i^2$. If the regression model includes an intercept, then $\sum e_i = 0$. The ordinary residuals are uncorrelated with the fitted values or indeed any linear combination of the regressors, including the regressors themselves, and so patterns in plots of ordinary residuals versus linear combinations of the regressors can occur only if one or more assumptions of the model are inappropriate.

If the regression model is correct, then the ordinary residuals are random variables with mean zero and with variance given by

$$\text{Var}(e_i) = \sigma^2(1 - h_i) \tag{8.2}$$

The quantity h_i is called a *leverage* or *hat-value*. In linear models with fixed predictors, h_i is a nonrandom value constrained to be between zero and 1, depending on the location of the predictors for a particular case relative to the other cases.[1] Let $\mathbf{x}_i = (1, x_{1i}, \ldots, x_{ki})$ represent the vector of regressors for the ith of n cases.[2] Large values of h_i correspond to cases with relatively unusual $\mathbf{x}_i$-values, whereas small h_i corresponds to cases close to the center of the predictor data (see Section 8.3.2).

Ordinary residuals for cases with large h_i have smaller variance. To correct for the nonconstant variance of the residuals, we can divide them by an estimate of their standard deviation. Letting $\widehat{\sigma}^2 = (\sum e_i^2)/(n - k - 1)$ represent the estimate of σ^2 in a model with an intercept and k other regressors, the *standardized residuals* are

$$e_{Si} = \frac{e_i}{\widehat{\sigma}\sqrt{1 - h_i}} \tag{8.3}$$

While the e_{Si} have constant variance, they are no longer uncorrelated with the fitted values or linear combinations of the predictors, so using standardized residuals in plots is not an obvious improvement.

Studentized residuals are given by

$$e_{Ti} = \frac{e_i}{\widehat{\sigma}_{(-i)}\sqrt{1 - h_i}} \tag{8.4}$$

where $\widehat{\sigma}^2_{(-i)}$ is the estimate of σ^2 computed from the regression without the ith case. Like the standardized residuals, the Studentized residuals have constant variance. In addition, if the original errors are normally distributed, then e_{Ti} follows a t-distribution with $n - k - 2$ degrees of freedom and can be used to test for outliers (see Section 8.3). One can show that in OLS linear regression,

$$\widehat{\sigma}^2_{(-i)} = \frac{\widehat{\sigma}^2(n - k - 1 - e^2_{Si})}{n - k - 2} \tag{8.5}$$

and so computing the Studentized residuals doesn't really require refitting the regression without the ith case.

If the model is fit by weighted-least-squares (WLS) regression with known positive weights w_i, then the ordinary residuals are replaced by the *Pearson residuals*,

$$e_{Pi} = \sqrt{w_i}e_i \tag{8.6}$$

[1] In a model with an intercept, the minimum hat-value is $1/n$.
[2] We assume here the $x_{0i} = 1$ is the constant regressor for a model with an intercept; if there is no intercept, then x_0 is simply omitted.

The residual sum of squares is $\sum e_{Pi}^2$ in WLS estimation. If we construe OLS regression to have implicit weights of $w_i = 1$ for all i, then Equation 8.1 is simply a special case of Equation 8.6, and we will generally use the term *Pearson residuals* to cover both of these cases. The standardized and Studentized residuals are unaffected by weights because the weights cancel in the numerator and denominator of their formulas.

The generic R function `residuals()` can compute various kinds of residuals. The default for a linear model is to return the ordinary residuals even if weights are present. Setting the argument `type="pearson"` (with a lowercase "`p`") returns the Pearson residuals, which produces correctly weighted residuals if weights are present and the ordinary residuals if there are no weights. Pearson residuals are the default when `residuals()` is used with a generalized linear model. The functions `rstandard()` and `rstudent()` return the standardized and Studentized residuals, respectively. The function `hatvalues()` returns the hat-values.

8.2 Basic Diagnostic Plots

The **car** package includes a number of functions that produce plots of residuals and related quantities. The variety of plots reflects the fact that no one diagnostic graph is appropriate for all purposes.

8.2.1 Plotting Residuals

Plots of residuals versus fitted values and versus each of the regressors in turn are the most basic diagnostic graphs. If a linear model is correctly specified, then the Pearson residuals are independent of the fitted values and also of the regressors or the predictors on which they are based, and these graphs should be *null plots*, with no systematic features, in the sense that the conditional distribution of the residuals that are plotted on the vertical axis should not change with the fitted values, a regressor, or a predictor on the horizontal axis. The presence of systematic features generally implies a failure of one or more assumptions of the model. Of interest in these plots are nonlinear patterns, changes in variation across the graph, and isolated points.

Plotting residuals is useful for revealing problems but less useful for determining the exact nature of the problem. Consequently, we will present other diagnostic graphs to suggest improvements to a model.

Consider, for example, a modification of the model used in Section 4.2.2 for the Canadian occupational-prestige data:

```
library("car")
Prestige$type <- factor(Prestige$type,
    levels=c("bc", "wc", "prof"))
prestige.mod.2 <- lm(prestige ~ education + income + type,
    data=Prestige)
brief(prestige.mod.2)
```

```
           (Intercept) education   income typewc typeprof
Estimate        -0.623     3.673 0.001013  -2.74     6.04
Std. Error       5.228     0.641 0.000221   2.51     3.87

 Residual SD = 7.09 on 93 df, R-squared = 0.835
```

In Section 4.2.2, we replaced the predictor `income` by the regressor log(`income`). Here we naïvely use `income` without transformation, in part to demonstrate what happens when a predictor needs transformation. For convenience, we also reorder the levels of the factor `type`.

The standard residual plots for this model are given by the `residualPlots()` function in the **car** package:

```
residualPlots(prestige.mod.2)

           Test stat Pr(>|Test stat|)
education      -0.68           0.4959
income         -2.89           0.0049
type
Tukey test     -2.61           0.0090
```

This command produces the four graphs in Figure 8.1 with the Pearson residuals on the vertical axis. The horizontal axis in the top row is for the numeric regressors `education` and `income`. The first graph in the second row shows boxplots of the residuals for the various levels of the factor `type`. The final graph has the fitted values on the horizontal axis. The broken line in each panel is the horizontal line through $e_P = 0$; as explained below, the solid line is a quadratic fit to the points in the plot.

The most common diagnostic graph in linear regression is the plot of residuals versus the fitted values, shown at the bottom right of Figure 8.1. The plot has a curved general pattern, suggesting that the model we fit is not adequate to describe the data. The plot of residuals versus `education` at the top left, however, resembles a null plot, in which no particular pattern is apparent. A null plot is consistent with an adequate model, but as is the case here, one null plot is insufficient to provide evidence of an adequate model, and indeed one nonnull plot is enough to suggest that the specified model does not match the data. The plot of residuals versus `income` at the top right is also curved, as might have been anticipated in light of our preliminary examination of the data in Section 3.3.2. The residual plot for a factor like `type`, at the bottom left, is a set of boxplots of the residuals at the various levels of the factor. In a null plot, the boxes should all have about the same center and inter-quartile distance, as is more or less the case here.

To help examine these residual plots, a *lack-of-fit test* is computed for each numeric regressor and a curve is added to the corresponding graph. The lack-of-fit test for `education`, for example, is the *t*-test for the regressor $(\texttt{education})^2$ added to the model, for which the corresponding *p*-value rounds to .50, indicating no lack of fit of this type. For `income`, the lack-of-fit test produces the *p*-value .005, clearly confirming the nonlinear pattern visible in the graph. The lines shown on the plot are the fitted quadratic regressions of the Pearson residuals on the numeric regressors.

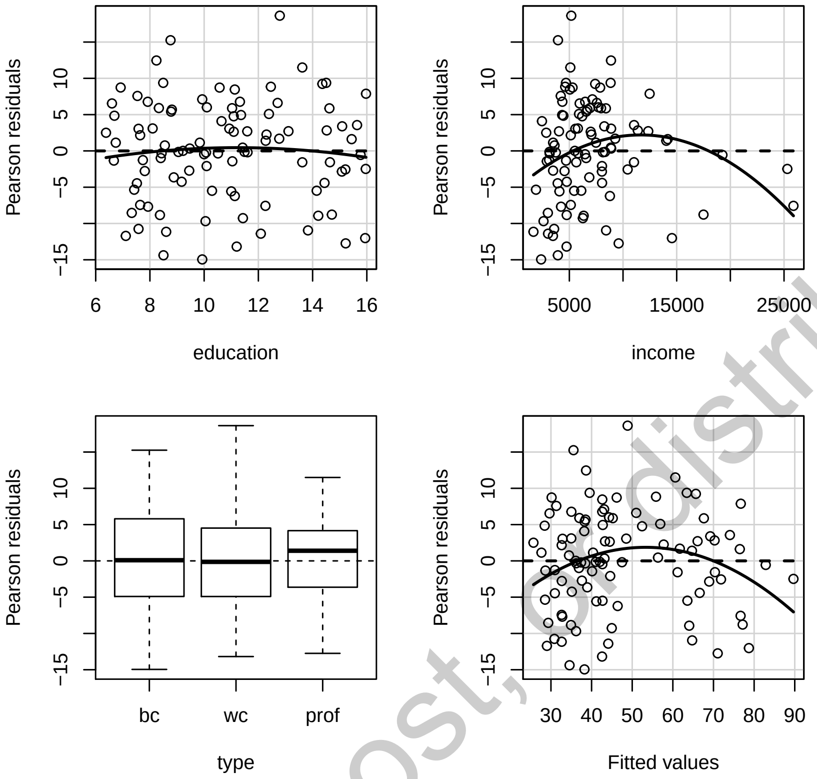

Figure 8.1 Basic residual plots for the regression of `prestige` on `education`, `income`, and `type` in the `Prestige` data set.

For the plot of residuals versus fitted values, the test, called *Tukey's test for nonadditivity* (Tukey, 1949), is obtained by adding the squares of the fitted values to the model and refitting. The *p*-value for Tukey's test is obtained by comparing the test statistic to the standard-normal distribution. The test confirms the visible impression of curvature in the residual plot, further reinforcing the conclusion that the fitted model is not adequate.

The `residualPlots()` function shares many arguments with other graphics functions in the **car** package; see `help("residualPlots")` for details. All arguments beyond the first are optional. The `id` argument controls point identification; for example, setting `id=list(n=3)` would automatically identify the three cases with the largest absolute residuals (see Section 3.5). There are additional arguments to control the layout of the plots and the type of residual plotted; setting `type="rstudent"`, for example, would plot Studentized residuals rather than Pearson residuals. Setting `smooth=TRUE, quadratic=FALSE` would display a loess smooth rather than a quadratic curve on each plot, although the test statistics always correspond to fitting quadratics.

If you want only the plot of residuals against fitted values, you can use

```
residualPlots(prestige.mod.2, ~ 1, fitted=TRUE)
```

whereas the plot against `education` only can be obtained with

```
residualPlots(prestige.mod.2, ~ education, fitted=FALSE)
```

The second argument to `residualPlots()`, and to other functions in the **car** package that can produce graphs with several panels, is a *one-sided formula* that specifies the regressors against which to plot residuals. The formula `~ .` is the default, to plot against *all* the regressors; `~ 1` plots against *none* of the regressors and in the current context produces a plot against fitted values only; `~ . - income` plots against all regressors but `income`. Because the fitted values are not part of the formula that defined the model, there is a separate `fitted` argument, with default `TRUE` to include a plot of residuals against fitted values and set to `FALSE` to exclude it.

8.2.2 Marginal-Model Plots

A variation on the basic residual plot is the *marginal-model plot*, proposed by R. D. Cook and Weisberg (1997) and implemented in the `marginalModelPlots()` function:

```
marginalModelPlots(prestige.mod.2)
```

The plots shown in Figure 8.2 all have the response variable, in this case `prestige`, on the vertical axis, while the horizontal axis is given in turn by each of the numeric regressors in the model and the fitted values. No plot is produced for the factor predictor `type`. The plots of the response versus individual regressors display the conditional distribution of the response given each regressor, ignoring the other regressors; these are *marginal* plots in the sense that they show the marginal relationship between the response and each regressor. The plot versus fitted values is a little different, in that it displays the conditional distribution of the response given the fit of the model.

We can estimate a regression function for each of the marginal plots by fitting a smoother to the points in the plot. The `marginalModelPlots()` function uses a loess smooth, as shown by the solid line on the plot.

Now imagine a second graph that replaces the vertical axis by the fitted values from the model. If the model is appropriate for the data, then, under fairly mild conditions, the smooth fit to this second plot should also estimate the conditional expectation of the response given the regressor on the horizontal axis. The second smooth is also drawn on the marginal-model plot, as a dashed line. If the model fits the data well, then the two smooths should match on each of the marginal-model plots; if any pair of smooths fails to match, then we have evidence that the model does not fit the data well.

An interesting feature of the marginal-model plots in Figure 8.2 is that even though the model that we fit to the `Prestige` data specifies linear *partial* relationships between `prestige` and each of `education` and `income`, it is able to reproduce nonlinear *marginal* relationships for these two regressors. Indeed, the model, as represented by the dashed lines, does a fairly good job of matching the

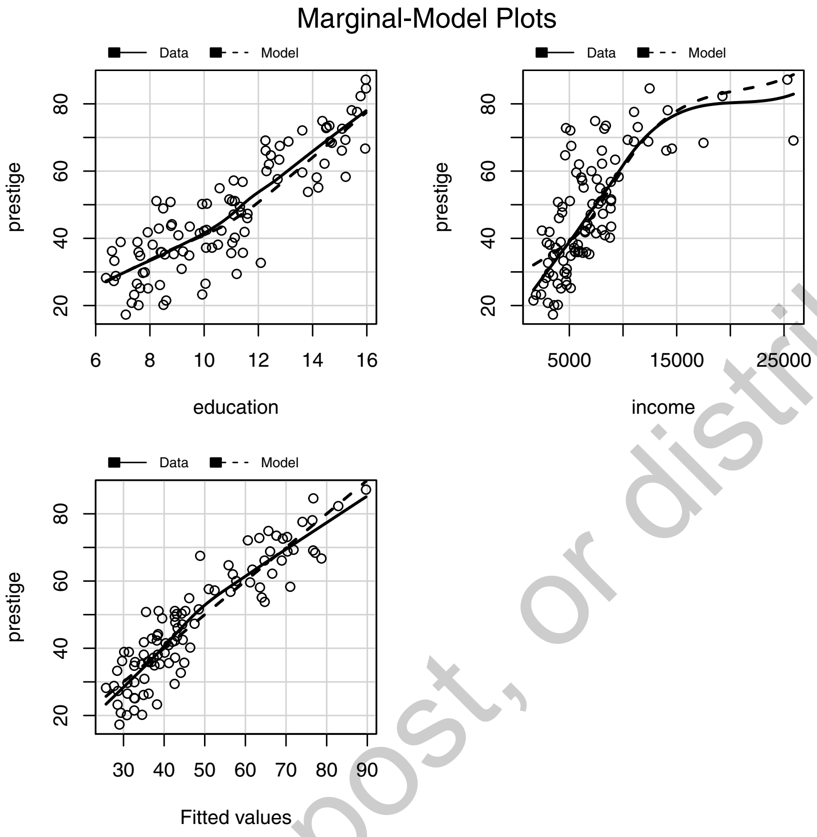

Figure 8.2 Marginal-model plots for the regression of `prestige` on `education`, `income`, and `type` in the `Prestige` data set.

marginal relationships represented by the solid lines, although the systematic failures discovered in the residual plots are discernable here as well.

Marginal-model plots can be used with any fitting or modeling method that produces fitted values, and so they can be applied to some problems where the definition of residuals is unclear. In particular, marginal-model plots work well with generalized linear models.

The `marginalModelPlots()` function has an `SD` argument, which, if set to `TRUE`, adds estimated standard-deviation lines to the graph. The plots can therefore be used to check both the regression function, as illustrated here, and assumptions about variance. Other arguments to the `marginalModelPlots()` function are similar to those for `residualPlots()`.

8.2.3 Added-Variable Plots

The marginal-model plots of the last section display the *marginal* relationships between the response and each regressor, *ignoring* other regressors in the model.

In contrast, *added-variable plots*, also called *partial-regression plots*, display the *partial* relationship between the response and a regressor, *adjusted for* all the other regressors.

Suppose that we have a regression problem with response y and regressors $x_1, \ldots, x_k$.[3] To draw the added-variable plot for one of the regressors, say the first, x_1, conduct the following two auxiliary regressions:

1. Regress y on all the regressors excluding x_1. The residuals from this regression are *the part of y that is not "explained" by all the regressors except for x_1*.
2. Regress x_1 on the other regressors and again obtain residuals. These residuals represent the *part of x_1 that is not explained by the other regressors*; put another way, the residuals are the part of x_1 that remains when we condition on the other regressors.

The added-variable plot for x_1 is simply a scatterplot, with the residuals from Step 1 on the vertical axis and the residuals from Step 2 on the horizontal axis.

The `avPlots()` function in the **car** package works both for linear and generalized linear models. It has arguments for controlling which plots are drawn, point labeling, and plot layout, and these arguments are the same as for the `residualPlots()` function described in Section 8.2.1.

Added-variable plots for the Canadian occupational-prestige regression (in Figure 8.3) are produced by the following command:

```
avPlots(prestige.mod.2, id=list(n=2, cex=0.6))
```

The argument `id=list(n=2, cex=0.6)` identifies up to four points in each graph: the two that are furthest from the mean on the horizontal axis and the two with the largest absolute residuals from the fitted line. Because the case labels in the `Prestige` data set are long, we used `cex=0.6` to reduce the size of the printed labels to 60% of their default size.

The added-variable plot has several interesting and useful properties:

- The least-squares line on the added-variable plot for the regressor x_j has the same slope b_j as x_j in the full regression. Thus, for example, the slope in the added-variable plot for `education` is $b_1 = 3.67$, and the slope in the added-variable plot for `income` is $b_2 = 0.00101$.[4]
- The residuals from the least-squares line in the added-variable plot are the same as the residuals e_i from the regression of the response on *all* the regressors.
- Because positions on the horizontal axis of the added-variable plot show values of x_j conditional on the other regressors, points far to the left or right

[3] Although it is not usually of interest, when there is an intercept in the model, it is also possible to construct an added-variable plot for the constant regressor, x_0, which is equal to 1 for every case.

[4] Income is in dollars per year, so the slope for `income` is in prestige points per dollar. Units are always important in interpreting slope estimates: In the current example, an additional dollar of annual `income` is a very small increment.

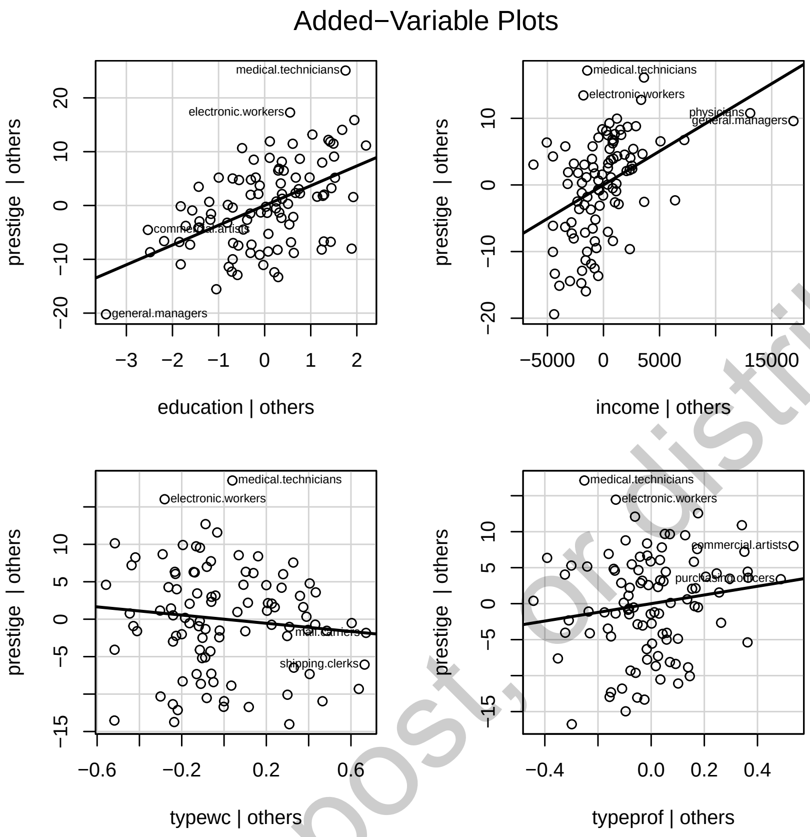

Figure 8.3 Added-variable plots for the regression of `prestige` on `education`, `income`, and `type` in the `Prestige` data set.

represent cases for which the value of x_j is unusual given the values of the other regressors. The variation of the variable on the horizontal axis is the conditional variation of x_j, and the added-variable plot therefore allows us to visualize the precision of estimation of b_j, along with the leverage of each case on the regression coefficient (see Section 8.3.2 for a discussion of leverage in regression).

- For factors, an added-variable plot is produced for each of the contrasts that are used to define the factor, and thus if we change the way that contrasts are coded for a factor, the corresponding added-variable plots will change as well.

The added-variable plot allows us to visualize the effect of each regressor after adjusting for all the other regressors in the model, effectively reducing the $(k + 1)$-dimensional regression problem to a sequence of 2D graphs.

In Figure 8.3, the plot for `income` has a positive slope, but the slope appears to be influenced by two high-income occupations (physicians and general managers) that pull down the regression line at the right. There don't seem to be any particularly noteworthy points in the added-variable plots for the other regressors.

Although added-variable plots are useful for studying the impact of cases on the regression coefficients (see Section 8.3.3), they can prove misleading for diagnosing other sorts of problems, such as nonlinearity. A further disadvantage of the added-variable plot is that the variables on both axes are sets of residuals, and so neither the response nor the regressors is displayed directly.

Sall (1990) and R. D. Cook and Weisberg (1991) generalize added-variable plots to terms with more than one degree of freedom, such as a factor or polynomial regressors. Following Sall, we call these graphs *leverage plots*. For terms with 1 degree of freedom, leverage plots are very similar to added-variable plots, except that the slope in the plot is always equal to 1, not to the corresponding regression coefficient. Although leverage plots can be misleading in certain circumstances,[5] they can be useful for locating groups of cases that are jointly high leverage or influential. Leverage, influence, and related ideas are explored in Section 8.3. There is a `leveragePlots()` function in the **car** package, which works only for linear models.

8.2.4 Marginal-Conditional Plots

Marginal-conditional plots are a pedagogical tool to help understand the distinction between the unconditional and conditional relationships of the response variable to a specific regressor. The unconditional relationship is visualized in a scatterplot of the response versus the regressor. The conditional relationship is visualized by an added-variable plot. The `mcPlot()` and `mcPlots()` functions in the **car** package compare these two graphs either in the same scale or superimposed on the same plot.

Figure 8.4 is the marginal-conditional plot for `education` in the model for the Canadian occcupational-prestige data used in the last few sections, drawn as two separate panels (by setting `overlaid=FALSE`):

```
mcPlots(prestige.mod.2, ~ education, overlaid=FALSE)
```

The scatterplot on the left displays the response `prestige` on the vertical axis and the regressor `education` on the horizontal axis. The variables on both axes are centered by subtracting their respective sample means. Centering changes the labels on the tick marks to make the scatterplot comparable to the added-variable plot, but it does not change the shape of the scatterplot. Marginally, `prestige` increases linearly with `education`. The line shown on the graph has slope given by the OLS regression of `prestige` on `education` alone. As is typical of functions in the **car** package, `mcPlot()` draws a marginal-conditional plot for one regressor, and `mcPlots()` draws one or more plots. See `help("mcPlot")` for descriptions of available options.

[5] For example, if a particular case causes one dummy-regressor coefficient to get larger and another smaller, these changes can cancel each other out in the leverage plot for the corresponding factor, even though a different pattern of results for the factor would be produced by removing the case.

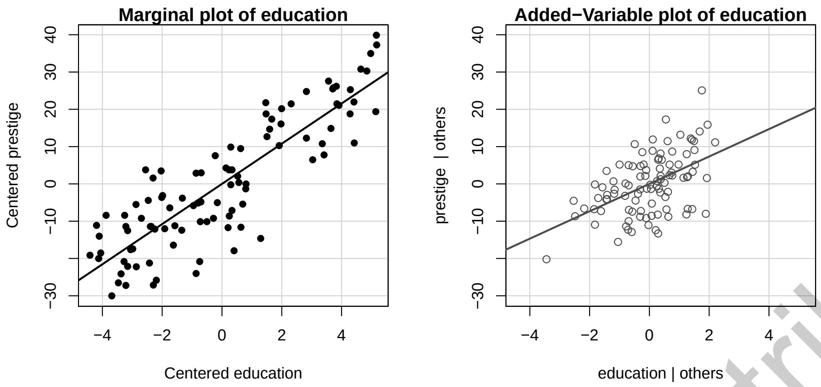

Figure 8.4 Marginal-conditional plots for `education` in the regression of `prestige` on `education`, `income`, and `type` of occupation in the `Prestige` data set. The panel on the left is the centered marginal scatterplot for `prestige` versus education; the panel on the right is the conditional added-variable plot for the two variables in the model.

The graph on the right of Figure 8.4 displays the conditional added-variable plot for `education` after adjusting for the other regressors in the model. The added-variable plot has the same size and scaling as the marginal plot. Both plots display one point for each case in the data. If both the response and the regressor were independent of the other regressors, then these two plots would be identical. If the response were perfectly correlated with the other regressors, then the points in the second plot would all have vertical coordinates of zero, and if the regressor were perfectly correlated with the other regressors, then all the horizontal coordinates in the second plot would equal zero. In this example, we see that conditioning `education` on the other regressors accounts for most of the variation in `education` (the R^2 for the regression of `education` on `income` and `type` is 0.83), as reflected by much smaller variation in horizontal coordinates in the conditional plot than in the marginal plot. Similarly, much of the variation in the response in the marginal plot is accounted for by the other regressors, reflected in the relatively small variation in the response in the conditional plot. The regression line in the conditional plot has slope equal to the `education` regression coefficient from the full model, `prestige.mod.2` (page 388).

8.3 Unusual Data

Unusual data can wreak havoc with least-squares estimates and may prove interesting in their own right. Unusual data in regression include outliers, high-leverage points, and influential cases.

8.3.1 Outliers and Studentized Residuals

Regression outliers are y-values that are unusual *conditional on* the values of the predictors. An illuminating route to search for outliers is via the *mean-shift outlier model*,

$$y = \beta_0 + \beta_1 x_1 + \cdots + \beta_k x_k + \gamma d_i + \varepsilon$$

where d_i is a dummy regressor coded 1 for case i and zero for all others. If $\gamma \neq 0$, then the conditional expectation of the ith case has the same dependence on $x_1, \ldots, x_k$ as the other cases, but its intercept is shifted from β_0 to $\beta_0 + \gamma$. The t-statistic for testing the null hypothesis H_0: $\gamma = 0$ against a two-sided alternative has $n - k - 2$ degrees of freedom if the errors are normally distributed and is the appropriate test for a single mean-shift outlier at case i. Remarkably, this t-statistic turns out to be identical to the ith Studentized residual, e_{Ti} (Equation 8.4, page 387), and so we can get the test statistics for the n different null hypotheses, H_{0i}: case i is not a mean-shift outlier, $i = 1, \ldots, n$, at minimal computational cost.

Our attention is generally drawn to the largest absolute Studentized residual, and this presents a problem: Even if the Studentized residuals were independent, which they are not, there would be an issue of simultaneous inference entailed by picking the largest of n test statistics. The dependence of the Studentized residuals complicates the issue. We can deal with this problem (1) by a *Bonferroni adjustment* of the p-value for the largest absolute Studentized residual, multiplying the usual two-tail p by the sample size n, or (2) by constructing a quantile-comparison plot of the Studentized residuals with a confidence envelope that takes their dependence into account.

To illustrate, we reconsider Duncan's occupational-prestige data (introduced in Section 1.5), regressing `prestige` on occupational `income` and `education` levels:

```
mod.duncan <- lm(prestige ~ income + education, data=Duncan)
```

The generic `qqPlot()` function in the **car** package has a method for linear models, plotting Studentized residuals against the corresponding quantiles of $t(n - k - 2)$. By default, `qqPlot()` generates a 95% pointwise confidence envelope for the Studentized residuals, using a parametric version of the bootstrap, as suggested by Atkinson (1985):[6]

```
qqPlot(mod.duncan, id=list(n=3))

  minister   reporter contractor
         6          9         17
```

The resulting plot is shown in Figure 8.5. Setting the argument `id=list(n=3)` to the `qqPlot()` function returns the names and indices of the three cases with the largest absolute Studentized residuals and identifies these points in the graph (see Section 3.5 on point identification); of these points, only `minister` strays slightly outside of the confidence envelope. If you repeat this command, your plot may

[6] Bootstrap methods in R are described in Section 5.1.3 and in an online appendix to the book.

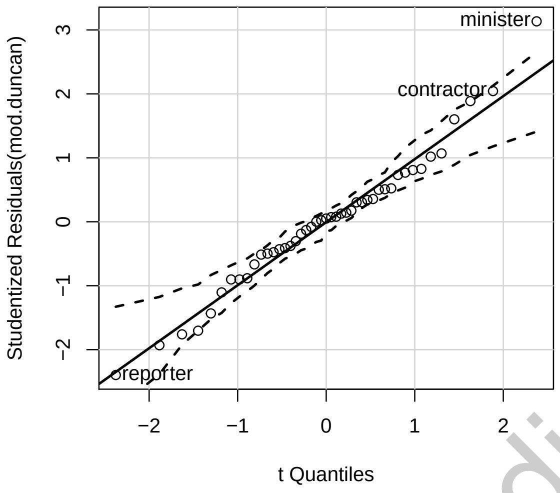

Figure 8.5 Quantile-comparison plot of Studentized residuals from Duncan's occupational-prestige regression, showing the pointwise 95% simulated confidence envelope.

look a little different from ours because the envelope is computed by simulation. The distribution of the Studentized residuals looks heavy-tailed compared to the reference *t*-distribution, and perhaps a method of robust regression would be more appropriate for these data.[7]

The `outlierTest()` function in the **car** package locates the largest Studentized residual in absolute value and computes the Bonferroni-corrected *t*-test:

```
outlierTest(mod.duncan)
```

```
No Studentized residuals with Bonferonni p < 0.05
Largest |rstudent|:
         rstudent unadjusted p-value Bonferonni p
minister   3.1345          0.0031772      0.14297
```

The Bonferroni-adjusted *p*-value is fairly large, .14, and so we conclude that it isn't very surprising that the biggest Studentized residual in a sample of size $n = 45$ would be as great as 3.135. The somewhat different conclusions suggested by the QQ-plot and the Bonferroni outlier test have a simple explanation: The confidence envelope in the QQ-plot is based on *pointwise* 95% confidence intervals while the outlier test adjusts for *simultaneous* inference.

8.3.2 Leverage: Hat-Values

Cases that are relatively far from the center of the predictor space, taking account of the correlational pattern among the predictors, have potentially greater influence

[7] R functions for robust and resistant regression are described in an online appendix to the *R Companion*.

on the least-squares regression coefficients; such points are said to have *high leverage*. The most common measures of leverage are the h_i or *hat-values*.[8] The h_i are bounded between zero and 1 (in models with an intercept, they are bounded between $1/n$ and 1); their sum, $\sum h_i$, is always equal to the number of coefficients in the model, including the intercept, and so in a model with an intercept, the average hat-value is $\bar{h} = (k+1)/n$. Problems in which there are a few very large h_i can be troublesome because large-sample normality of some linear combinations of the predictors is likely to fail, and high-leverage cases may exert undue influence on the results (see below).

The `hatvalues()` function works for both linear and generalized linear models. One way of examining the hat-values and other individual-case diagnostic statistics is to construct *index plots*, graphing the statistics against the corresponding case indices.

For example, the following command uses the **car** function `influenceIndexPlot()` to produce Figure 8.6, which includes index plots of Studentized residuals, the corresponding Bonferroni p-values for outlier testing, the hat-values, and Cook's distances (discussed in the next section) for Duncan's occupational-prestige regression:

```
influenceIndexPlot(mod.duncan, id=list(n=3))
```

The occupations railroad engineer (`RR.engineer`), `conductor`, and `minister` stand out from the rest in the plot of hat-values, indicating that their predictor values are unusual relative to the other occupations. In the plot of p-values for the outlier tests, cases for which the Bonferroni bound is bigger than 1 are set equal to 1, and here only one case (`minister`) has a Bonferroni p-value much less than 1.

8.3.3 Influence Measures

A case that is both outlying and has high leverage exerts *influence* on the regression coefficients, in the sense that if the case is removed, the coefficients change considerably. As usual, let **b** be the estimated value of the coefficient vector $\boldsymbol{\beta}$ and as new notation define $\mathbf{b}_{(-i)}$ to be the estimate of $\boldsymbol{\beta}$ computed without the ith case.[9] Then the difference $\mathbf{b}_{(-i)} - \mathbf{b}$ directly measures the influence of the ith case on the estimate of $\boldsymbol{\beta}$. If this difference is small, then the influence of case i is small, while if the difference is large, then its influence is large.

[8] * The name "hat-values" comes from the relationship between the observed vector of responses (i.e., the ys) and the fitted values (i.e., the $\widehat{y}$s or "y-hats"). The vector of fitted values is given by $\widehat{\mathbf{y}} = \mathbf{Xb} = \mathbf{X}(\mathbf{X}'\mathbf{X})^{-1}\mathbf{X}'\mathbf{y} = \mathbf{Hy}$ where $\mathbf{H} = \{h_{ij}\} = \mathbf{X}(\mathbf{X}'\mathbf{X})^{-1}\mathbf{X}'$, called the *hat-matrix*, projects $\mathbf{y}$ into the subspace spanned by the columns of the model matrix $\mathbf{X}$. Because $\mathbf{H} = \mathbf{H}'\mathbf{H}$, the hat-values h_i are simply the diagonal entries of the hat-matrix.

[9] If vector notation is unfamiliar, simply think of **b** as the collection of estimated regression coefficients, $b_0, b_1, \ldots, b_k$.

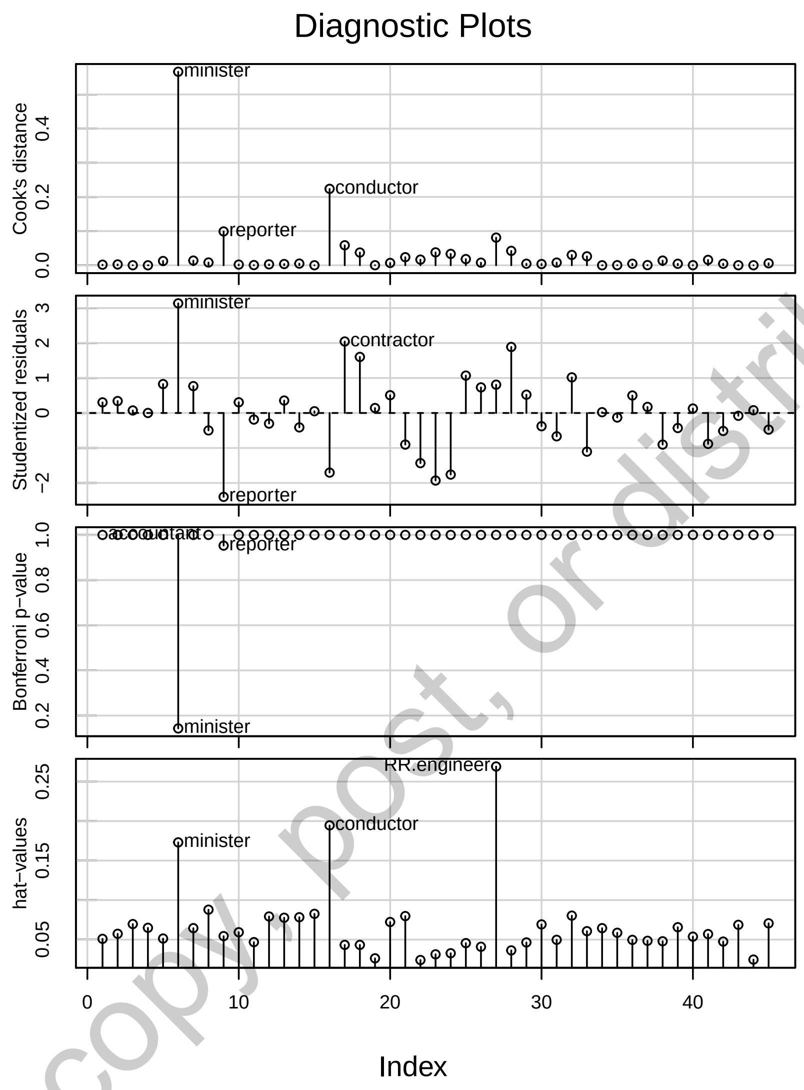

Figure 8.6 Index plots of diagnostic statistics for Duncan's occupational-prestige regression, indentifying the three most extreme cases in each plot.

Cook's Distance

It is convenient to summarize the size of the difference $\mathbf{b}_{(-i)} - \mathbf{b}$ by a single number, and this can be done in several ways. The most common summary measure of influence is *Cook's distance* (R. D. Cook, 1977), D_i, which is just a weighted sum of squares of the differences between the individual elements of the coefficient vectors.[10] Interestingly, Cook's distance can be computed from diagnostic statistics that we have already encountered,

$$D_i = \frac{e_{Si}^2}{k+1} \times \frac{h_i}{1-h_i}$$

where e_{Si}^2 is the squared standardized residual (Equation 8.3 on page 387) and h_i is the hat-value for case i. The first factor may be thought of as a measure of outlyingness and the second as a measure of leverage. Cases for which D_i is largest are potentially influential cases. If any noteworthy D_i are apparent, then a prudent approach is to remove the corresponding cases temporarily from the data, refit the regression, and see how the results change. Because an influential case can affect the fit of the model at other cases, it is best to remove cases one at a time, refitting the model at each step and reexamining the resulting Cook's distances.

The generic function `cooks.distance()` has methods for linear and generalized linear models. Cook's distances are also plotted, along with Studentized residuals and hat-values, by the `influenceIndexPlot()` function, as illustrated for Duncan's regression in Figure 8.6. The occupation `minister` is the most influential according to Cook's distance, and we therefore see what happens when we delete this case and refit the model:

```
mod.duncan.2 <- update(mod.duncan,
    subset= rownames(Duncan) != "minister")
compareCoefs(mod.duncan, mod.duncan.2)
```

```
Calls:
1: lm(formula = prestige ~ income + education, data =
  Duncan)
2: lm(formula = prestige ~ income + education, data =
  Duncan, subset = rownames(Duncan) != "minister")
            Model 1 Model 2
(Intercept)   -6.06   -6.63
SE             4.27    3.89

income        0.599   0.732
SE            0.120   0.117

education    0.5458  0.4330
SE           0.0983  0.0963
```

Removing `minister` increases the coefficient for `income` by about 20% and decreases the coefficient for `education` by about the same amount. Standard errors are much less affected. In other problems, removing a case can change "statistically significant" results to "nonsignificant" ones and vice-versa.

[10] * In matrix notation,

$$D_i = \frac{\left(\mathbf{b}_{(-i)} - \mathbf{b}\right)' \mathbf{X}'\mathbf{X} \left(\mathbf{b}_{(-i)} - \mathbf{b}\right)}{(k+1)\widehat{\sigma}^2}$$

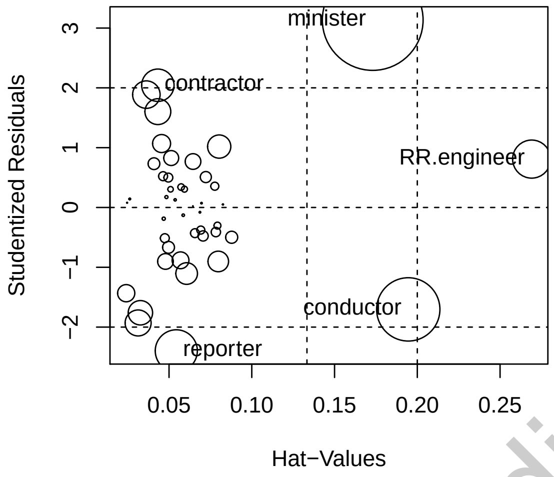

Figure 8.7 Plot of hat-values, Studentized residuals, and Cook's distances for Duncan's occupational-prestige regression. The size of the circles is proportional to Cook's D_i.

The `influencePlot()` function in the **car** package provides an alternative to index plots of diagnostic statistics:

```
influencePlot(mod.duncan, id=list(n=3))

               StudRes      Hat     CookD
minister      3.13452 0.173058 0.566380
reporter     -2.39702 0.054394 0.098985
conductor    -1.70403 0.194542 0.223641
contractor    2.04380 0.043255 0.058523
RR.engineer   0.80892 0.269090 0.080968
```

This command produces a *bubble plot*, shown in Figure 8.7, combining the display of Studentized residuals, hat-values, and Cook's distances, with the areas of the circles proportional to Cook's D_i. As usual, the `id` argument is used to label points. In this case, the `n=3` points with the largest hat-values, Cook's distance, or absolute Studentized residuals will be flagged, so more than three points in all are labeled.

We invite the reader to continue the analysis by examining influence diagnostics for Duncan's regression after the case `minister` has been removed.

Influence Separately for Each Coefficient

Rather than summarizing influence by looking at all coefficients simultaneously, we could create $k + 1$ measures of influence by looking at individual differences

$$\text{dfbeta}_{ij} = b_{(-i)j} - b_j \text{ for } j = 0, \ldots, k$$

where b_j is the coefficient computed using all of the data, and $b_{(-i)j}$ is the same coefficient computed with case i omitted. As with D_i, computation of the dfbeta_{ij} can be accomplished efficiently without having to refit the model. The dfbeta_{ij} are expressed in the metric (units of measurement) of the coefficient b_j. A standardized version, dfbetas$_{ij}$, divides dfbeta$_{ij}$ by an estimate of the standard error of b_j computed with case i removed.

The `dfbeta()` function in R takes a linear-model or generalized-linear-model object as its argument and returns all of the dfbeta$_{ij}$; similarly, `dfbetas()` computes the dfbetas$_{ij}$. For example, for Duncan's regression:

```
dfbs.duncan <- dfbetas(mod.duncan)
head(dfbs.duncan)  # first few rows

            (Intercept)      income   education
accountant -2.2534e-02  6.6621e-04  0.03594387
pilot      -2.5435e-02  5.0877e-02 -0.00811827
architect  -9.1867e-03  6.4837e-03  0.00561927
author     -4.7204e-05 -6.0177e-05  0.00013975
chemist    -6.5817e-02  1.7005e-02  0.08677706
minister    1.4494e-01 -1.2209e+00  1.26301904
```

We could examine each column of the dfbetas matrix separately (e.g., via an index plot), but because we are not really interested here in influence on the regression intercept, and because there are just two slope coefficients, we instead plot influence on the `income` coefficient against influence on the `education` coefficient (Figure 8.8):

```
plot(dfbs.duncan[ , c("income", "education")])  # for b1 and b2
showLabels(dfbs.duncan[ , "income"],
    dfbs.duncan[ , "education"],
    labels=rownames(Duncan), method="identify")
    # remember to exit from point identification mode
```

The negative relationship between the dfbetas$_{ij}$ values for the two predictors reflects the *positive* correlation of the predictors themselves. Two pairs of values stand out: Consistent with our earlier remarks, the cases `minister` and `conductor` make the `income` coefficient smaller and the `education` coefficient larger. We also identify the occupation `RR.engineer` in the plot.

Added-Variable Plots as Influence Diagnostics

The added-variable plots introduced in Section 8.2.3 are a useful diagnostic for finding potentially jointly influential points, which correspond to sets of points that are out of line with the rest of the data and are at the extreme left or right of the horizontal axis. When two or more such points act in concert to affect the fitted regression surface, they may not be revealed by individual-case statistics such as Cook's D_i. Figure 8.9, for example, shows the added-variable plots for `income` and `education` in Duncan's regression:

```
avPlots(mod.duncan, id=list(n=3, method="mahal"))
```

The argument `id=list(n=3, method="mahal")` serves to identify the three points in each panel with the largest Mahalanobis distances from the center of

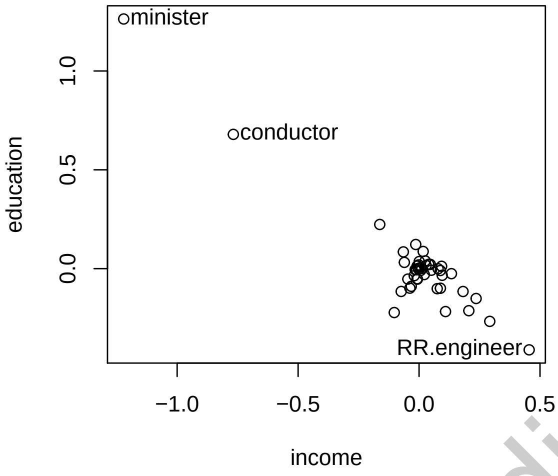

Figure 8.8 dfbetas$_{ij}$ values for the `income` and `education` coefficients in Duncan's occupational-prestige regression. Three points were identified interactively.

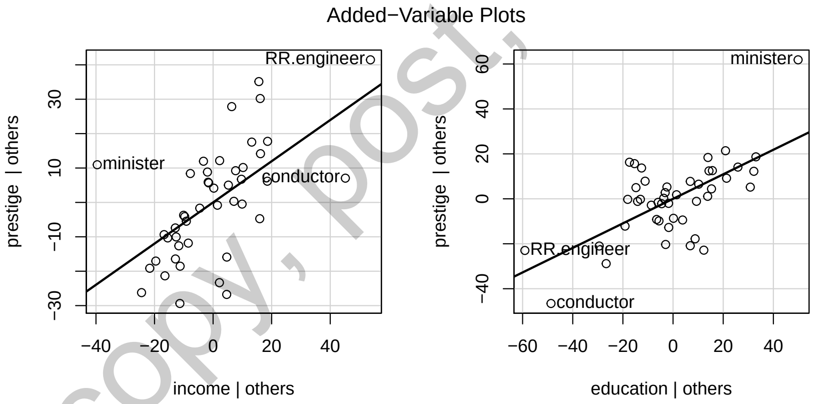

Figure 8.9 Added-variable plots for Duncan's occupational-prestige regression.

all the points.[11] The cases `minister`, `conductor`, and `RR.engineer` (railroad engineer) have high leverage on both coefficients. The cases `minister` and `conductor` also work together to decrease the `income` slope and increase the `education` slope; `RR.engineer`, on the other hand, is more in line with the rest of the data. Removing *both* `minister` and `conductor` changes the regression

[11] The *Mahalanobis* or *generalized distance* takes into account the standard deviation of each variable and their correlation. Although the graphics functions in the **car** package have reasonable general defaults for point identification, we can also identify points interactively in diagnostic plots. Interactive point identification is supported by the argument `id=list(method="identify")` to `avPlots()` and other **car** graphics functions. See `help("showLabels")` and the discussion of point identification in Section 3.5.

coefficients much more so than deleting minister alone:

```
mod.duncan.3 <- update(mod.duncan,
    subset = - whichNames(c("minister", "conductor"), Duncan))
compareCoefs(mod.duncan, mod.duncan.2, mod.duncan.3, se=FALSE)

 Calls:
 1: lm(formula = prestige ~ income + education, data =
   Duncan)
 2: lm(formula = prestige ~ income + education, data =
   Duncan, subset = rownames(Duncan) != "minister")
 3: lm(formula = prestige ~ income + education, data =
   Duncan, subset = -whichNames(c("minister", "conductor"),
   Duncan))

              Model 1 Model 2 Model 3
 (Intercept)   -6.06   -6.63   -6.41
 income        0.599   0.732   0.867
 education     0.546   0.433   0.332
```

We use the whichNames() function in the **car** package to return the indices of the cases "minister" and "conductor" in the Duncan data frame; setting the subset argument to the negative of these indices excludes the two cases from the regression.

8.4 Transformations After Fitting a Regression Model

Suspected outliers, and possibly cases with high leverage, should be studied individually to decide if they should be included in an analysis or not. Influential cases can cause changes in conclusions in an analysis and also require special treatment. Other systematic features in residual plots, such as curvature or apparent nonconstant variance, require action on the part of the analyst to modify the *structure* of the model to match the data more closely. Apparently distinct problems can also interact: For example, if the errors have a skewed distribution, then apparent outliers may be produced in the direction of the skew. Transforming the response to make the errors less skewed can solve this problem. Similarly, properly modeling a nonlinear relationship may bring apparently outlying cases in line with the rest of the data.

Transformations were introduced in Section 3.4 in the context of examining data and with the understanding that regression modeling is often easier and more effective when the predictors behave as if they were normal random variables. Transformations can also be used *after* fitting a model, to improve a model that does not adequately represent the data. The methodology in these two contexts is very similar.

8.4.1 Transforming the Response

Box-Cox Transformations

The goals of fitting a model that exhibits linearity, constant variance, and normality can in principle require three different response transformations, but experience suggests that one transformation is often effective for all of these tasks. The most common method for selecting a transformation of the response in regression is due to Box and Cox (1964). If the response y is a strictly positive variable, then the Box-Cox power transformations, introduced in Section 3.4.2 and implemented in the `bcPower()` function in the **car** package, are often effective. We replace the response y by $T_{\mathrm{BC}}(y, \lambda)$, where

$$T_{\mathrm{BC}}(y,\lambda) = y^{(\lambda)} = \begin{cases} \dfrac{y^{\lambda}-1}{\lambda} & \text{when } \lambda \neq 0 \\ \log y & \text{when } \lambda = 0 \end{cases} \tag{8.7}$$

The power parameter λ determines the transformation. The Box-Cox family essentially replaces y by y^{λ}, with y^0 interpreted as $\log(y)$.[12]

Box and Cox proposed selecting the value of λ by analogy to the method of maximum likelihood, so that the residuals from the linear regression of $T_{\mathrm{BC}}(y, \lambda)$ on the predictors are as close to normally distributed as possible.[13] The **car** package provides three functions for estimating λ:

- The `boxCox()` function, a slight generalization of the `boxcox()` function in the **MASS** package (Venables & Ripley, 2002),[14] was illustrated in a related context in Section 3.4.2.
- We illustrated the `powerTransform()` function in Section 7.2.5 and will do so again in the current section.
- The `inverseResponsePlot()` function provides a visual method for selecting a normalizing response transformation, but we will not present it here (see Section 8.9 for a brief description).

By way of illustration, we introduce an example that is of historical interest, because it was first used by Box and Cox (1964). The data, in the `Wool` data set in the **carData** package, are from an industrial experiment to study the strength of wool yarn under various conditions. Three predictors were varied in the experiment:

[12] The subtraction of 1 and division by λ is inessential in that it doesn't alter the *shape* of the power transformation y^{λ}, with the caveat that dividing by λ does preserve the *order* of the data when λ is negative: For negative λ, such as $\lambda = -1$ (the inverse transformation), the simple power y^{λ} *reverses* the order of the y-values.

[13] * If $T_{\mathrm{BC}}(y, \lambda_0)|\mathbf{x}$ is normally distributed, then $T_{\mathrm{BC}}(y, \lambda_1)|\mathbf{x}$ cannot be normally distributed for $\lambda_1 \neq \lambda_0$, and so the distribution changes for every value of λ. The method Box and Cox proposed ignores this fact to get a maximum-likelihood-like estimate that turns out to have properties similar to those of maximum-likelihood estimates. In the interest of brevity, in the sequel, we refer to Box-Cox and similar estimates of transformation parameters as "maximum-likelihood estimates."

[14] `boxCox()` adds a `family` argument, providing greater flexibility in the choice of a response transformation.

len, the length of each sample of yarn in millimeters; amp, the amplitude of the loading cycle in angular minutes; and load, the amount of weight used in grams. The response, cycles, was the number of cycles until the sample failed. Data were collected using a $3 \times 3 \times 3$ design, with each of the predictors at three equally spaced levels. We fit a linear model treating each of the three predictors as numeric variables with linear effects:

```
brief(wool.mod <- lm(cycles ~ len + amp + load, data=Wool))

           (Intercept)  len  amp  load
Estimate          4521 13.2 -536 -62.2
Std. Error        1622  2.3  115  23.0

 Residual SD = 488 on 23 df, R-squared = 0.729
```

This model fits the data poorly, as a residual plot versus fitted values (not shown) reveals obvious curvature. We can attempt to remedy the situation by transforming the response:

```
summary(p1 <- powerTransform(wool.mod))

bcPower Transformation to Normality
   Est Power Rounded Pwr Wald Lwr Bnd Wald Upr Bnd
Y1   -0.0592           0      -0.1789       0.0606

Likelihood ratio test that transformation parameter is equal to 0
 (log transformation)
                          LRT df  pval
LR test, lambda = (0) 0.92134  1 0.337

Likelihood ratio test that no transformation is needed
                         LRT df   pval
LR test, lambda = (1) 84.076  1 <2e-16
```

The maximum-likelihood estimate of the transformation parameter is $\widehat{\lambda} = -0.059$, with the 95% confidence interval for λ running from -0.179 to 0.061. The p-value for a test of $\lambda = 0$ is large, and the p-value for $\lambda = 1$ is very small. Both the likelihood-ratio tests and the Wald confidence interval suggest $\lambda = 0$, or a log transformation, as a reasonable choice, while leaving the response untransformed, $\lambda = 1$, is not acceptable. Replacing cycles by log(cycles) results in null residual plots, as the reader can verify.

Zero or Negative Responses

The Box-Cox power family requires that the response variable be strictly positive, but in some instances, zero or negative values may occur in the data. A common approach to this problem is to add a "start" to the response to make all values of the response positive and then to apply the Box-Cox power family of transformations. Adding a start is problematic, however, because the transformation selected, and the effectiveness of the transformation, can be greatly affected by the value of the start. If the start is too small, then the transformed zero and negative cases are likely to become overly influential in the fit. If the start is too large, then information about variation in the response is potentially attenuated.

As a partial remedy, we recommend the use of the *Box-Cox with negatives* family of transformations, implemented in the `bcnPower()` function in the **car** package, in place of the standard Box-Cox power family. The Box-Cox with negatives family was introduced by Hawkins and Weisberg (2017) and is defined in Section 3.4.2. Like the Box-Cox family, there is a power parameter λ. In addition, there is a location parameter $\gamma \geq 0$ that takes the place of a start. As γ approaches zero, the Box-Cox with negatives transformation approaches the standard Box-Cox power family.

In Section 6.5, we fit a Poisson regression to Ornstein's data on interlocking directorates among Canadian corporations, regressing the number of interlocks maintained by each firm on the log of the firm's assets, nation of control, and sector of operation. Because number of interlocks is a count, the Poisson model is a natural starting point, but the original source (Ornstein, 1976) used a least-squares regression similar to the following:

```
mod.ornstein <- lm(interlocks  ~ log(assets) + nation + sector,
    data=Ornstein)
```

About 10% of the values of `interlocks` are zero, so we use the `"bcnPower"` family to select a normalizing transformation:

```
summary(p2 <- powerTransform(mod.ornstein, family="bcnPower"))

 bcnPower transformation to Normality

 Estimated power, lambda
    Est Power Rounded Pwr Wald Lwr Bnd Wald Upr Bnd
 Y1    0.3545        0.33       0.2928       0.4162

 Location gamma was fixed at its lower bound
    Est gamma Std Err. Wald Lower Bound Wald Upper Bound
 Y1       0.1       NA               NA               NA

 Likelihood ratio tests about transformation parameters
                           LRT df pval
 LR test, lambda = (0) 138.93  1    0
 LR test, lambda = (1) 306.76  1    0
```

As is often the case when there are many zeros or negative values, the estimate of γ is close to zero, and so it is set to a value somewhat larger than zero (0.1) and treated as fixed. The estimate of λ is then found by the maximum-likelihood method proposed by Box and Cox. The rounded estimate of the power is $\widehat{\lambda} = 0.33$, or cube root, with a fairly narrow Wald confidence interval. No confidence interval is given for γ because its estimate is too close to zero. The likelihood-ratio tests provide evidence against $\lambda = 0$ or $\lambda = 1$.[15] The suggestion is therefore to fit the model

```
Ornstein$interlocks.tran <-
  bcnPower(Ornstein$interlocks, lambda=1/3, gamma=0.1)
mod.ornstein.2 <- update(mod.ornstein, interlocks.tran ~ .)
```

and then proceed with the analysis. In this instance, the cube-root transformation *is* defined for zero values, and because γ is small, we could simply have used the cube-root transformation of `interlocks` to get virtually the same results.

Additional options for power transformations are given by `help("powerTransform")`, `help("bcPower")`, and `help("boxCox")`. The `powerTransform()` function works for linear models, for multivariate linear models, and for linear mixed models. It is not useful, however, for non-Gaussian generalized linear models, where transformation of the expected response is accomplished by selection of a link function.

Understanding Models With a Transformed Response

A common complaint about using a transformed response is that the resulting model is uninterpretable because the results are expressed in strange, transformed units. In Ornstein's interlocking-directorate regression, for example, a model with the response given by the cube root of the number of `interlocks` doesn't produce directly interpretable coefficients. We think that this complaint is misguided:

1. Most regression models are at best approximations, and allowing for a transformed response can help make for a better approximate model. As long as the transformation is monotone, tests of effects have a straightforward interpretation. For example, in the Ornstein regression, we find that the transformed number of interlocks increases with log(`assets`), even though the coefficient estimate is in uninterpretable units.

2. Some common transformations have straightforward interpretations. A few examples: Increasing the log base-2 by 1 implies doubling the response. If the response is the time in seconds required by a runner to traverse 100 meters, then the inverse of the response is the average velocity of the runner in units of 100 meters per second. If the response is the area of an apartment in square meters, than its square root is a linear measure of size in meters.

3. In any event, invariant regression summaries such as effect plots can be "untransformed" using an inverse transformation, as we have illustrated by example in Figure 7.10 (page 369). For the `bcnPower()` transformation family, where the inverse is complicated, we supply the function `bcnPowerInverse()` to reverse the transformation; if, alternatively, $z = y^\lambda$ is a simple power transformation of the response y (taken as $z = \log(y)$ when $\lambda = 0$), then the inverse transformation is just $z^{1/\lambda}$ for $\lambda \neq 0$ and $\exp(z)$ for $\lambda = 0$.

[15] The likelihood-ratio-type tests concern λ only and treat γ as a nuisance parameter by averaging over γ.

8.4.2 Predictor Transformations

As outlined in Section 3.4, predictor transformations can, and typically should, be performed *before* fitting models to the data. Even well-behaved predictors, however, aren't necessarily linearly related to the response, and graphical diagnostic methods are available that can help select a transformation *after* fitting a model. Moreover, some kinds of nonlinearity can't be fixed by transforming a predictor, and other strategies, such as polynomial regression or regression splines, may be entertained.

Component-Plus-Residual and CERES Plots

Component-plus-residual plots, also called *partial-residual plots*, are a simple graphical device that can be effective for detecting the need to transform a predictor, say x_j, to a new variable $T(x_j)$, for some transformation T. The plot has x_{ij} on the horizontal axis and the *partial residuals*, $e_{\text{Partial},ij} = e_i + b_j x_{ij}$, on the vertical axis. R. D. Cook (1993) shows that if the regressions of x_j on the other xs are approximately linear, then the regression function in the component-plus-residual plot provides a visualization of T. Alternatively, if the regressions of x_j on the other xs resemble polynomials, then a modification of the component-plus-residual plot due to Mallows (1986) can be used.

The `crPlots()` function in the **car** package constructs component-plus-residual plots for linear and generalized linear models. By way of example, we return to the Canadian occupational-prestige regression (from Section 8.2.1), this time fitting a regression model for `prestige` in which the predictors are `income`, `education`, and `women`. A scatterplot matrix of the response and the three predictors (Figure 3.14 on page 147) suggests that the predictors are not all linearly related to each other, but no more complicated than quadratic regressions should provide reasonable approximations. Consequently, we draw the component-plus-residual plots specifying `order=2`, permitting quadratic relationships among the predictors:

```
prestige.mod.3 <- lm(prestige ~ income + education + women,
    data=Prestige)
crPlots(prestige.mod.3, order=2)
```

The component-plus-residual plots for the three predictors appear in Figure 8.10. The broken line on each panel is the *partial fit*, $b_j x_j$, assuming linearity in the partial relationship between y and x_j. The solid line is a loess smooth, and it should suggest a transformation if one is appropriate, for example, via the bulging rule (see Section 3.4.3). Alternatively, the smooth might suggest a quadratic or cubic partial regression or, in more complex cases, the use of a regression spline.

For the Canadian occupational-prestige regression, the component-plus-residual plot for `income` is the most clearly curved, and transforming this variable first and refitting the model is therefore appropriate. In contrast, the component-plus-residual plot for `education` is only slightly nonlinear, and the partial relationship is not simple (in the sense of Section 3.4.3). Finally, the component-plus-residual plot for `women` looks mildly quadratic (although the lack-of-fit test computed by

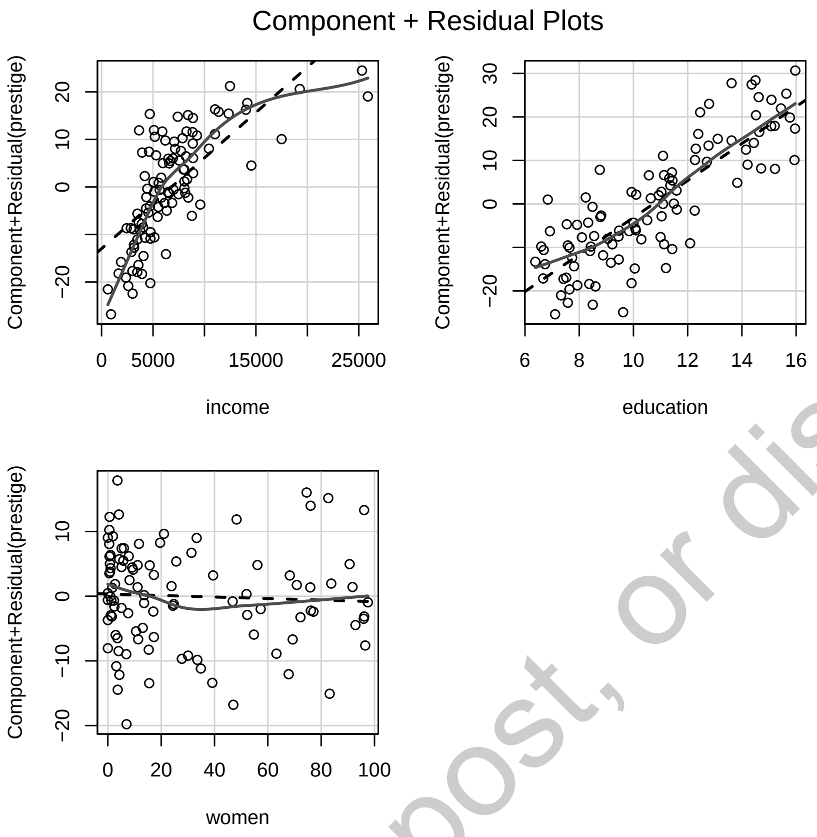

Figure 8.10 Component-plus-residual plots of `order=2` for the Canadian occupational-prestige regression.

the `residualPlots()` function does not suggest a "significant" quadratic effect), with `prestige` first declining and then rising as `women` increases.

Trial-and-error experimentation moving `income` down the ladder of powers and roots suggests that a log transformation of this predictor produces a reasonable fit to the data:

```
prestige.mod.4 <- update(prestige.mod.3,
    . ~ . + log2(income) - income)
```

This is the model that we fit in Section 4.2.2. The component-plus-residual plot for `women` in the revised model (not shown) is broadly similar to the plot for `women` in Figure 8.10, and the lack-of-fit test computed by `residualPlots()` has a *p*-value of 0.025, suggesting a quadratic regression:

```
prestige.mod.5 <- update(prestige.mod.4,
    . ~ . - women + poly(women, 2))
brief(prestige.mod.5, pvalues=TRUE)
```

```
            (Intercept) education log2(income) poly(women, 2)1
Estimate      -1.11e+02  3.77e+00     9.36e+00          15.088
Std. Error     1.40e+01  3.47e-01     1.30e+00           9.336
Pr(>|t|)       4.16e-12  1.98e-18     1.26e-10           0.109
           poly(women, 2)2
Estimate            15.871
Std. Error           6.970
Pr(>|t|)             0.025

 Residual SD = 6.95 on 97 df, R-squared = 0.843
```

The *p*-value for the test of the quadratic term for `women` in the new model is also .025.

If the regressions among the predictors are strongly nonlinear and not well described by polynomials, then component-plus-residual plots may not be effective in recovering nonlinear partial relationships between the response and the predictors. For this situation, R. D. Cook (1993) provides another generalization of component-plus-residual plots called *CERES plots* (for *C*ombining conditional *E*xpectations and *RES*iduals). CERES plots use nonparametric-regression smoothers rather than polynomial regressions to adjust for nonlinear relationships among the predictors. The `ceresPlots()` function in the **car** package implements Cook's approach.

Experience suggests that nonlinear relationships among the predictors induce problems for component-plus-residual plots only when these relationships are strong. In such cases, a component-plus-residual plot can appear nonlinear even when the true partial regression is linear—a phenomenon termed *leakage*. For the Canadian occupational-prestige regression, quadratic component-plus-residual plots (in Figure 8.10) and CERES plots are nearly identical to the standard component-plus-residual plots, as the reader may verify.

Adding Partial Residuals to Effect Plots

Traditional component-plus-residual plots, as implemented in the `crPlots()` function, are drawn only for numeric predictors that enter a regression model additively. Fox and Weisberg (2018, in press) show how partial residuals can be added to effect plots for linear and generalized linear models of arbitrary complexity, including models with interactions between numeric predictors and factors and between numeric predictors. These methods are implemented in the **effects** package.

To illustrate, let's return to the regression model `prestige.mod.2` for the Canadian occupational-prestige data, in which `prestige` is regressed on `income`, `education`, and `type`. An effect plot with partial residuals for `income` in this model is a traditional component-plus-residual plot and is shown in Figure 8.11:[16]

[16] A subtle, and unimportant, distinction is that the partial residuals in the effect plot add back in the intercept from the regression model, together with constant terms involving the coefficients and means of the other predictors, and so the scaling of the vertical axis of the plot is different from that of a traditional component-plus-residual plot. The *shape* of the plot—that is, the configuration of points—is the same, however.

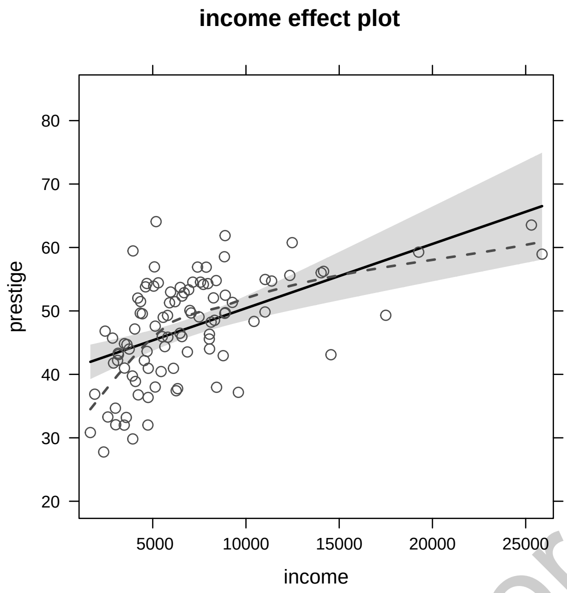

Figure 8.11 Effect plot with partial residuals for `income` in the regression of `prestige` on `income`, `education`, and `type` for the Canadian occupational-prestige data.

```
library("effects")
plot(Effect("income", prestige.mod.2, residuals=TRUE),
     partial.residuals=list(lty="dashed"))
```

The straight line represents the fitted effect, and the curved broken line is a loess smooth of the partial residuals.[17] The plot shows obvious unmodeled nonlinearity in the partial relationship of `prestige` to `income` (and is very similar to the quadratic component-plus-residual plot for `income` in the model regressing `prestige` on `income`, `education`, and `women`, shown at the top left of Figure 8.10).

The graph at the top of Figure 8.12 shows the effect plot with partial residuals for the predictors `income` and `type` simultaneously:

```
plot(Effect(c("income", "type"), prestige.mod.2,
        residuals=TRUE),
    partial.residuals=list(span=0.9, lty="dashed"),
    lattice=list(layout=c(3, 1)))
```

This effect plot is for the interaction of `income` with `type` in a model in which these predictors enter additively, as reflected in the parallel straight lines in the graph representing the fitted model, `prestige.mod.2`. We use a large span,

[17] We specify a broken line (`lty="dashed"`) for the smooth because the default is to plot lines of different colors for the model fit and the smooth, and color isn't available to us in this book.

`partial.residuals=list(span=0.9, lty="dashed")`, for the loess smoother because of the small numbers of cases in the various levels of `type`; we also arrange the panels horizontally via the argument `lattice=list(layout=c(3, 1))` (see `help("plot.eff")` for details). The loess smooths are nearly linear but with different slopes—greatest for blue-collar (`"bc"`) occupations and smallest for professional and managerial (`"prof"`) occupations—suggesting an unmodeled interaction between the two predictors.

Refitting the model reveals that the `income` × `type` interaction has a very small *p*-value, and the effect plot for `income` and `type` in this model, shown at the bottom of Figure 8.12, supports the model:[18]

```
prestige.mod.6 <- update(prestige.mod.2,
    . ~  income*type + education, data=Prestige)
Anova(prestige.mod.6)
```

```
Anova Table (Type II tests)

Response: prestige
            Sum Sq Df F value  Pr(>F)
income        1059  1   25.41 2.3e-06
type           591  2    7.09  0.0014
education     1068  1   25.63 2.1e-06
income:type    890  2   10.68 6.8e-05
Residuals     3791 91
```

```
plot(Effect(c("income", "type"), prestige.mod.6,
       residuals=TRUE),
    partial.residuals=list(span=0.9, lty="dashed"),
    lattice=list(layout=c(3, 1)))
```

This example nicely illustrates how unmodeled interaction can be reflected in apparent nonlinearity in a component-plus-residual plot.

8.5 Nonconstant Error Variance

One of the assumptions of the standard linear model is that the error variance is fully known apart from an unknown constant, σ^2. It is, however, possible that the error variance depends on one or more of the predictors, on the magnitude of the response, or systematically on some other variable.

To detect nonconstant variance as a function of a variable *z*, we can plot the Pearson residuals versus *z*. Nonconstant variance would be diagnosed if the variability of the residuals in the graph either increased from left to right, decreased from left to right, or displayed another systematic pattern, such as large variation in the middle of the range of *z* and smaller variation at the edges.

[18] Using the log of `income` in place of `income` in the respecified model straightens the slight curvature barely discernable in the panels for blue-collar and white-collar occupations at the bottom of Figure 8.12.

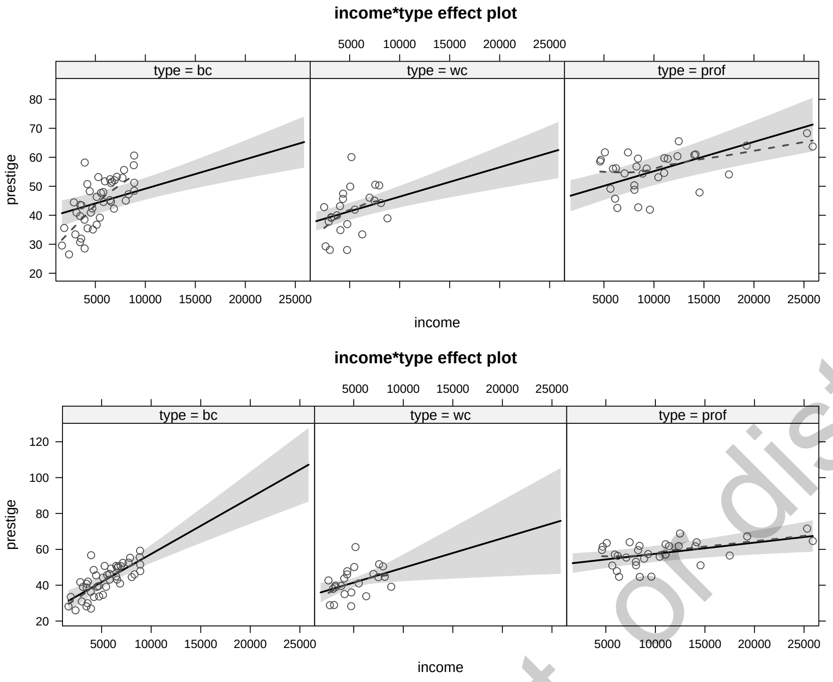

Figure 8.12 Effect plots with partial residuals for `income` and `type` in the regression of `prestige` on `income`, `education`, and `type`: additive model (top), model with interaction between `income` and `type` (bottom).

In multiple regression, there are many potential plotting directions. Because obtaining a two-dimensional graph entails projecting the predictors from many dimensions onto one horizontal axis, however, we can never be sure if a 2D plot showing nonconstant variability really reflects nonconstant error variance or some other problem, such as unmodeled nonlinearity (R. D. Cook, 1998, Section 1.2.1).

For an example, we return to the bank transactions data (introduced in Section 5.1.1), relating minutes of labor `time` in branches of a large bank to the number of transactions of two types, `t1` and `t2`; the data are in the `Transact` data set in the **carData** package:

```
mod.transact <- lm(time ~ t1 + t2, data=Transact)
brief(mod.transact)
```

```
             (Intercept)    t1     t2
 Estimate            144 5.462 2.0345
 Std. Error          171 0.433 0.0943

  Residual SD = 1143 on 258 df, R-squared = 0.909
```

```
residualPlots(mod.transact, tests=FALSE, layout=c(1, 3))
```

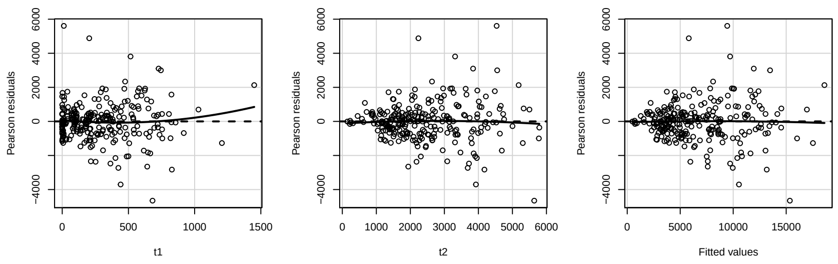

Figure 8.13 Residual plots for the transactions data.

Two of the residual plots shown in Figure 8.13 exhibit a characteristic fan shape, with increasing variability moving from left to right for `t2` and for the plot against fitted values, but not as obviously for `t1`. This pattern suggests nonconstant residual variance, possibly as a function of `t2` only.

8.5.1 Testing for Nonconstant Error Variance

Breusch and Pagan (1979) and R. D. Cook and Weisberg (1983) suggest a score test for nonconstant error variance in a linear model. The assumption underlying the test is that the variance is constant, or it depends on the conditional mean of the response,

$$\text{Var}(\varepsilon_i) = \sigma^2 g[\text{E}(y|\mathbf{x})]$$

or it depends on some linear combination of regressors $z_1, \ldots, z_p$,

$$\text{Var}(\varepsilon_i) = \sigma^2 g(\gamma_1 z_{i1} + \cdots + \gamma_p z_{ip}) \tag{8.8}$$

In typical applications, the *z*s are selected from the *x*s and often include all of the *x*s, but other choices of *z*s are possible. For example, in an educational study, variability of test scores might differ among the schools in the study, and a set of dummy regressors for schools would be candidates for the *z*s.

The `ncvTest()` function in the **car** package implements the score test. We compute four score tests for the bank transactions regression:

```
(test.t1 <- ncvTest(mod.transact, ~ t1))

 Non-constant Variance Score Test
 Variance formula: ~ t1
 Chisquare = 26.525, Df = 1, p = 2.6e-07

(test.t2 <- ncvTest(mod.transact, ~ t2))

 Non-constant Variance Score Test
 Variance formula: ~ t2
 Chisquare = 76.589, Df = 1, p = <2e-16

(test.t1t2 <- ncvTest(mod.transact, ~ t1 + t2))
```

```
 Non-constant Variance Score Test
 Variance formula: ~ t1 + t2
 Chisquare = 82.932, Df = 2, p = <2e-16
```

```
ncvTest(mod.transact)
```

```
 Non-constant Variance Score Test
 Variance formula: ~ fitted.values
 Chisquare = 61.659, Df = 1, p = 4.08e-15
```

All four tests are for the null hypothesis that the γs are equal to zero in Equation 8.8 against the alternatives that nonconstant variance is a function of `t1` alone, a function of `t2` alone, a function of both `t1` and `t2` in some linear combination, and, finally, a function of the conditional mean of the response, as captured by the fitted values. In this instance, all four tests have very small *p*-values, suggesting likely nonconstant variance.

An approximate test of the null hypothesis that residual variance depends on `t2` alone versus the alternative that it depends on both `t1` and `t2` is obtained by computing the difference between the chi-square test statistics in `test.t1t2` and `test.t2` above:

```
(stat <- test.t1t2$ChiSquare - test.t2$ChiSquare)
```

```
 [1] 6.3429
```

```
(df <- test.t1t2$Df - test.t2$Df)
```

```
 [1] 1
```

```
pchisq(stat, df, lower.tail=FALSE)
```

```
 [1] 0.011785
```

Even though this test suggests that the error variance should be modeled as a function of both `t1` and `t2`, residual variation is much more strongly related to `t2` than to `t1`, and so we might, as a reasonable approximation, refit the regression by weighted least squares, with weights given by 1/`t2`. Alternatively, and more naturally, the model could be fit as a generalized linear model with the identity link and gamma errors, as suggested for this example by Cunningham and Heathcote (1989). Finally, because OLS estimates are unbiased even if the variance function is incorrectly specified, the estimates could be obtained by OLS, but with standard errors and tests computed using either the bootstrap (Section 5.1.3) or a sandwich coefficient-variance estimator (Section 5.1.2). These corrections may be used by many functions in the **car** package, including `linearHypothesis()`, `deltaMethod()`, `Anova()`, `Confint()`, `S()`, `brief()`, and `Predict()`.

8.6 Diagnostics for Generalized Linear Models

Most of the diagnostics of the preceding sections extend straightforwardly to generalized linear models. These extensions typically take advantage of the computation of maximum-likelihood estimates for generalized linear models by *iterated weighted least squares* (*IWLS*: see Section 6.12), which in effect approximates the true log-likelihood for a GLM by a weighted-least-squares problem. At convergence of the IWLS algorithm, diagnostics are formed as if the weighted-least-squares problem were the problem of interest, and so the exact diagnostics for the weighted-least-squares fit are approximate diagnostics for the original GLM. Seminal work on the extension of linear-least-squares diagnostics to generalized linear models was done by Pregibon (1981), Landwehr, Pregibon, and Shoemaker (1980), Wang (1985, 1987), and Williams (1987). We focus here on methods that differ from their application in linear models.

8.6.1 Residuals and Residual Plots

One of the major philosophical, though not necessarily practical, differences between linear-model diagnostics and GLM diagnostics is in the definition of residuals. In linear models, the ordinary residual is the difference $y - \widehat{y}$, which is meant to mimic the statistical error $\varepsilon = y - \mathrm{E}(y|\mathbf{x})$. Apart from Gaussian generalized linear models, there is no additive error in the definition of a GLM, and so the idea of a residual has a much less firm footing.

Residuals for GLMs are generally defined in analogy to linear models. Here are the various types of GLM residuals that are available in R:

- *Response residuals* are simply the differences between the observed response and its estimated expected value: $y_i - \widehat{\mu}_i$. These correspond to the ordinary residuals in the linear model. Apart from the Gaussian case, the response residuals are not used in diagnostics, however, because they ignore the non-constant variance that is intrinsic to non-Gaussian GLMs.

- *Working residuals* are the residuals from the final IWLS fit. The working residuals may be used to define partial residuals for component-plus-residual plots and effect plots (see below) but are not usually accessed directly by the user.

- *Pearson residuals* are casewise components of the Pearson goodness-of-fit statistic for the model,

$$e_{Pi} = \frac{y_i - \widehat{\mu}_i}{\sqrt{\widehat{\mathrm{Var}}(y_i|\mathbf{x})/\widehat{\phi}}}$$

  where $\widehat{\phi}$ is the estimated dispersion parameter in the GLM. Formulas for $\mathrm{Var}(y|\mathbf{x})$ are given in the last column of Table 6.2 (page 274). This definition of e_{Pi} corresponds exactly to the Pearson residuals defined in Equation 8.6 (page 387) for WLS regression. These are a basic set of residuals for use with a GLM because of their direct analogy to linear models. For a model named `m`, the command `residuals(m, type="pearson")` returns the Pearson residuals.

- *Standardized Pearson residuals* correct for conditional response variation and for the leverage of the cases:

$$e_{PSi} = \frac{y_i - \widehat{\mu}_i}{\sqrt{\widehat{\text{Var}}(y_i|\mathbf{x})(1 - h_i)}}$$

  To compute the e_{PSi}, we need to define the hat-values h_i for GLMs. The h_i are taken from the final iteration of the IWLS procedure for fitting the model, and have the usual interpretation, except that, unlike in a linear model, the hat-values in a generalized linear model depend on y as well as on the configuration of the xs.

- *Deviance residuals*, e_{Di}, are the square roots of the casewise components of the residual deviance, attaching the sign of $y_i - \widehat{\mu}_i$. In the linear model, the deviance residuals reduce to the Pearson residuals. The deviance residuals are often the preferred form of residual for generalized linear models and are returned by the command `residuals(m, type="deviance")`.

- *Standardized deviance residuals* are

$$e_{DSi} = \frac{e_{Di}}{\sqrt{\widehat{\phi}(1 - h_i)}}$$

- The ith *Studentized residual* in a linear model is the scaled difference between the response and the fitted value computed without case i. Because of the special structure of the linear model, these differences can be computed without actually refitting the model removing case i, but this is not true for generalized linear models. While computing n regressions to get the Studentized residuals is not impossible, it is not a desirable option when the sample size is large. An approximation due to Williams (1987) is therefore used instead:

$$e_{Ti} = \text{sign}(y_i - \widehat{\mu}_i)\sqrt{(1 - h_i)e_{DSi}^2 + h_i e_{PSi}^2}$$

  The approximate Studentized residuals are computed when the function `rstudent()` is applied to a GLM. A Bonferroni outlier test using the standard-normal distribution may be based on the largest absolute Studentized residual and is implemented in the `outlierTest()` function.

As an example, we return to the Canadian women's labor force participation data described in Section 6.7. We define a binary rather than a polytomous response, with categories working or not working outside the home, and fit a logistic-regression model to the data:

```
mod.working <- glm(partic != "not.work" ~ hincome + children,
     family=binomial, data=Womenlf)
brief(mod.working)
```

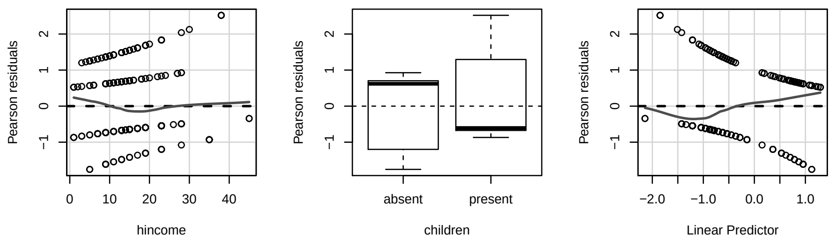

Figure 8.14 Residual plots for the binary logistic regression fit to the Canadian women's labor force participation data.

```
               (Intercept) hincome childrenpresent
Estimate             1.336 -0.0423          -1.576
Std. Error           0.384  0.0198           0.292
exp(Estimate)        3.803  0.9586           0.207

 Residual deviance = 320 on 260 df
```

The expression `partic != "not.work"` creates a logical vector, which serves as the binary response variable in the model.

The `residualPlots()` function provides basic plots of residuals versus the predictors and versus the linear predictor (Figure 8.14):

```
residualPlots(mod.working, layout=c(1, 3))

          Test stat Pr(>|Test stat|)
hincome        1.23             0.27
children
```

The function plots Pearson residuals versus each of the predictors in turn. Instead of plotting residuals against fitted values, however, `residualPlots()` plots residuals against the estimated linear predictor, $\widehat{\eta}(\mathbf{x})$. Each panel in the graph by default includes a smooth fit rather than a quadratic fit; a lack of fit test is provided only for the numeric predictor `hincome` and not for the factor `children` or for the estimated linear predictor.

In binary regression models, plots of Pearson residuals or of deviance residuals are strongly patterned. In a plot against the linear predictor, the residuals can only assume two values, depending on whether the response is equal to zero or 1. Because the factor `children` only has two levels, the residuals when plotted against `hin-come` can only take on four distinct values, correspondng to the combinations of the binary response and the two levels of `children`. A correct model requires that the conditional mean function in any residual plot be constant as we move across the plot, and a fitted smooth helps us to learn about the conditional mean function even in these highly discrete cases. Neither of the smooths, against `hincome` or the linear predictor, shown in Figure 8.14, is especially curved. The lack-of-fit test for `hincome` has a large *p*-value, confirming our view that this plot does not indicate lack of fit.

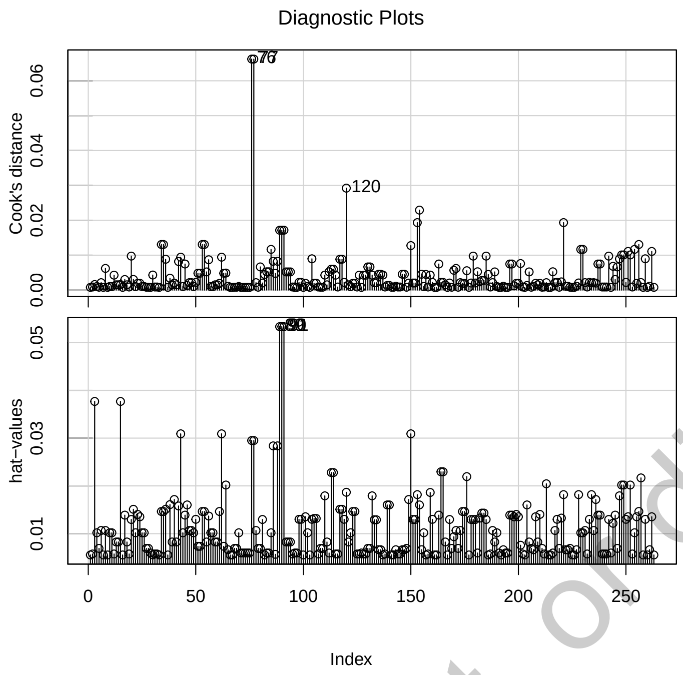

Figure 8.15 Index plots of diagnostic statistics for the logistic regression fit to the Canadian women's labor force participation data.

The residuals for `children` are shown as boxplots because `children` is a factor. Recalling that the heavy line in each boxplot represents the median, most of the residuals for `children "absent"` are positive but with a strong negative skew, while most of the residuals for `children "present"` are negative with a positive skew, implying that the model used may be inadequate. This residual plot suggests that the interaction of `children` and `hincome` should be explored, an exercise we leave to the interested reader.

8.6.2 Influence Measures

An approximation to Cook's distance for GLMs is

$$D_i = \frac{e_{PSi}^2}{(k+1)} \times \frac{h_i}{1-h_i}$$

These values are returned by the `cooks.distance()` function.

Figure 8.15 shows index plots of Cook's distances and hat-values, produced by the command:

```
influenceIndexPlot(mod.working, vars=c("Cook", "hat"),
    id=list(n=3))
```

Setting `vars=c("Cook", "hat")` limits the graphs to these two diagnostics. Cases 76 and 77 have the largest Cook's distances, although even these are quite small. We remove both Cases 76 and 77 as a check:

```
compareCoefs(mod.working,
    update(mod.working, subset=-c(76, 77)))
```

```
Calls:
1: glm(formula = partic != "not.work" ~ hincome + children,
  family = binomial, data = Womenlf)
2: glm(formula = partic != "not.work" ~ hincome + children,
  family = binomial, data = Womenlf, subset = -c(76, 77))

                Model 1 Model 2
(Intercept)       1.336   1.609
SE                0.384   0.405

hincome         -0.0423 -0.0603
SE               0.0198  0.0212

childrenpresent  -1.576  -1.648
SE                0.292   0.298
```

The reader can verify that removing just one of the two cases does not alter the results much, but removing *both* cases changes the coefficient of husband's income by more than 40%, about one standard error. Apparently, the two cases are an influential pair that partially mask each other, and removing them both is required to produce a meaningful change in the coefficient for `hincome`. Cases 76 and 77 are women working outside the home even though both have children and high-income husbands.

8.6.3 Graphical Methods: Added-Variable Plots, Component-Plus-Residual Plots, and Effect Plots With Partial Residuals

Added-variable plots are extended to generalized linear models by approximating the two fitted regressions required to generate the plot. By default, the `avPlots()` function uses an approximation suggested by Wang (1985). Added-variable plots for binary-regression models can be uninformative, however, because of the extreme discreteness of the response variable.

Component-plus-residual and CERES plots also extend straightforwardly to generalized linear models. Nonparametric smoothing of the resulting scatterplots can be important to interpretation, especially in models for binary responses, where the discreteness of the response makes the plots difficult to decode. Similar, if less striking, effects due to discreteness can also occur for binomial and count data.

For an illustrative component-plus-residual plot, we reconsider Ornstein's interlocking-directorate quasi-Poisson regression from Section 6.5, but now we fit a model that uses `assets` as a predictor rather than the log of `assets`:

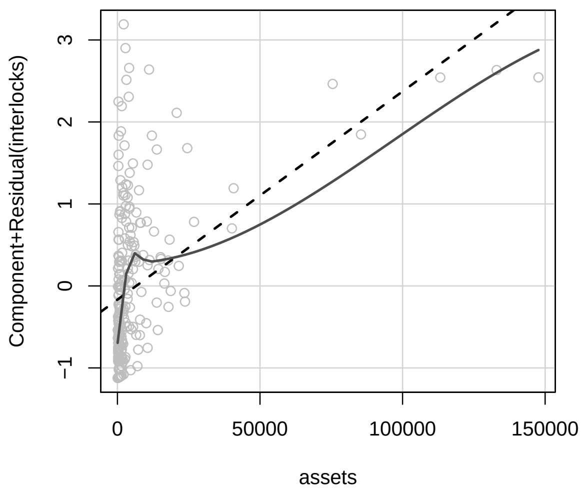

Figure 8.16 Component-plus-residual plot for `assets` in the quasi-Poisson regression fit to Ornstein's interlocking-directorate data.

```
mod.ornstein.qp <- glm(interlocks ~ assets + nation + sector,
    family=quasipoisson, data=Ornstein)
crPlots(mod.ornstein.qp, ~ assets, col="gray")
```

The component-plus-residual plot for `assets` is shown in Figure 8.16. This plot is hard to interpret because of the extreme positive skew in `assets`, but it appears as if the `assets` slope is a good deal steeper at the left than at the right. The bulging rule, therefore, points toward transforming `assets` down the ladder of powers, and indeed the log rule in Section 3.4.1 suggests replacing `assets` by its logarithm *before* fitting the regression (which, of course, is what we did originally):

```
mod.ornstein.qp.2 <- update(mod.ornstein.qp,
    . ~ log2(assets) + nation + sector)
crPlots(mod.ornstein.qp.2, ~ log2(assets))
```

The linearity of the follow-up component-plus-residual plot in Figure 8.17 confirms that the log-transform is a much better scale for `assets`.

We continue with a reexamination of the binary logistic-regression model fit to Mroz's women's labor force participation data in Section 6.3. One of the predictors in this model, the log of the woman's expected wage rate (`lwg`), has an unusual definition: For women in the labor force, for whom the response `lfp` = `"yes"`, `lwg` is the log of the women's *actual* wage rate, while for women not in the labor force, for whom `lfp` = `"no"`, `lwg` is the log of the *predicted* wage rate from the regression of wages on the other predictors.

To obtain a component-plus-residual plot for `lwg` (Figure 8.18):

```
mod.mroz <- glm(lfp ~ k5 + k618 + age + wc + hc + lwg + inc,
    family=binomial, data=Mroz)
```

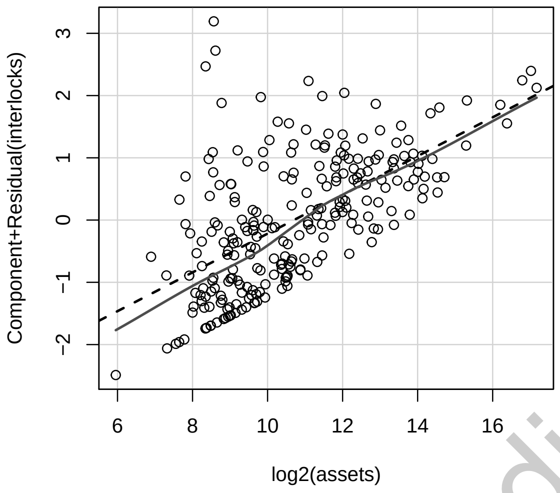

Figure 8.17 Component-plus-residual plot for the log of `assets` in the respecified quasi-Poisson regression for Ornstein's data.

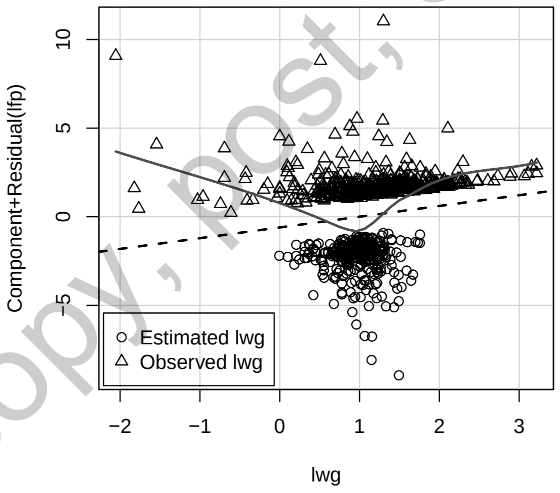

Figure 8.18 Component-plus-residual plot for `lwg` in the binary logistic regression for Mroz's women's labor force participation data.

```
crPlots(mod.mroz, "lwg", pch=as.numeric(Mroz$lfp))
legend("bottomleft",c("Estimated lwg", "Observed lwg"),
    pch=1:2, inset=0.01)
```

We specify the `pch` argument to `crPlots()` to use different plotting symbols for the two values of `lfp` and add a legend to the graph.[19] The peculiar split in the plot

[19] See Chapter 9 on customized graphics in R.

reflects the binary response variable, with the lower cluster of points corresponding to `lfp = "no"` and the upper cluster to `lfp = "yes"`. It is apparent that `lwg` is much less variable when `lfp = "no"`, inducing an artifactually curvilinear relationship between `lwg` and `lfp`: We expect fitted values (such as the values of `lwg` when `lfp = "no"`) to be more homogeneous than observed values, because fitted values lack a residual component of variation.

We leave it to the reader to construct component-plus-residual or CERES plots for the other predictors in the model.

For a final example, we return to a binary logistic regression fit to the Cowles and Davis volunteering data from Section 6.3.2:

```
cowles.mod <- glm(volunteer ~ sex + neuroticism*extraversion,
    data=Cowles, family=binomial)
```

We showed effect plots for Cowles and Davis's model in Figure 6.3 (page 285). To determine whether the linear-by-linear interaction between `neuroticism` and `extraversion` is supported by the data, we can add partial residuals to the effect plots of the interaction (Figure 8.19):

```
plot(predictorEffects(cowles.mod, ~ neuroticism + extraversion,
        residuals=TRUE),
    partial.residuals=list(span=3/4, lty="dashed"),
    lattice=list(layout=c(1, 4)))
```

The predictor effect plot for `neuroticsm` is shown at the left, for `extraversion` at the right. In each case, the other predictor in the interaction increases across its range from the bottom panel to the top panel, as indicated by the black vertical line in the strip at the top of each panel. By default, the conditioning predictor in each plot is set to four values equally spaced across its range. We use the argument `partial.residuals=list(span=3/4, lty="dashed")` to `plot()` to increase the span of the loess smoother (shown as a broken line in each panel) slightly to 3/4 from the default of 2/3 and the argument `lattice=list(layout=c(1, 4))` to orient the panels in each effect plot vertically. The data appear to support the linear-by-linear form of the interaction.

8.7 Diagnostics for Mixed-Effects Models

Regression diagnostics for mixed-effects models are relatively less developed than for linear and generalized linear models. We focus in this section on component-plus-residual plots, which are implemented for mixed-effects models in the **effects** package, and on deletion diagnostics for influential data, which are implemented in the **car** package.

8.7.1 Mixed-Model Component-Plus-Residual Plots

Computing partial residuals for the fixed effects in mixed models is a straightforward extension of the computations for linear and generalized linear models. The

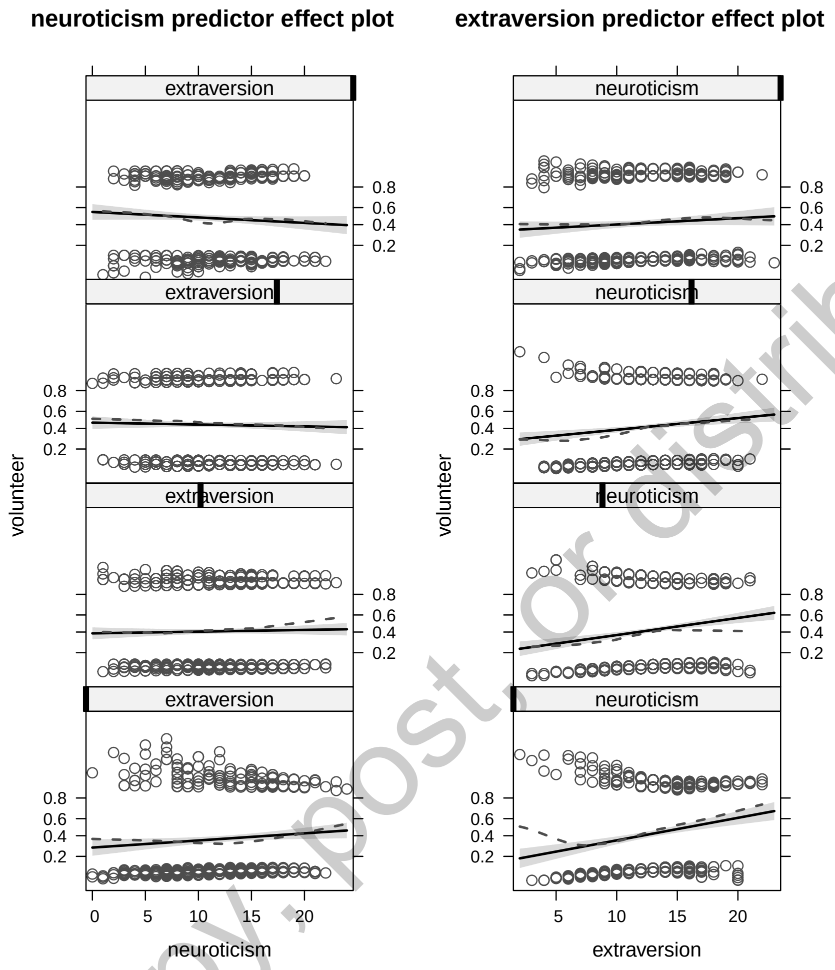

Figure 8.19 Predictor effect plots with partial residuals for `neuroticism` (left) and `extraversion` (right) in Cowles and Davis's logistic regression for volunteering, in which these two predictors are modeled with a linear-by-linear interaction.

`crPlots()` function in the **car** package doesn't handle mixed-effects models, but the `predictorEffects()` function and the more basic and general `Effect()` function in the **effects** package do, for models fit by the `lmer()` and `glmer()` functions in the **lme4** package and for models fit by `lme()` in the **nlme** package.

To illustrate these capabilities, we return to a linear mixed-effects model that we fit with `lme()` in Section 7.2.6 to the `Blackmore` longitudinal data on exercise and eating disorders:

```
library("nlme")
Blackmore$tran.exercise <- bcnPower(Blackmore$exercise,
            lambda=0.25, gamma=0.1)
blackmore.mod.6.lme<- lme(tran.exercise ~ I(age - 8)*group,
    random = ~ 1 | subject,
    correlation = corCAR1(form = ~ I(age - 8) | subject),
    data=Blackmore)
S(blackmore.mod.6.lme)
```

```
 Linear mixed model fit by REML,  Data: Blackmore

 Fixed Effects:
  Formula: tran.exercise ~ I(age - 8) * group

                        Estimate Std.Error  df t value Pr(>|t|)
 (Intercept)             -0.1962    0.1427 712   -1.38    0.170
 I(age - 8)               0.0668    0.0231 712    2.89    0.004
 grouppatient            -0.1569    0.1843 229   -0.85    0.396
 I(age - 8):grouppatient  0.1894    0.0290 712    6.52  1.3e-10

 Random effects:
  Formula: ~1 | subject
        (Intercept) Residual
 StdDev:       0.884     1.12

 Correlation Structure: Continuous AR(1)
  Formula: ~I(age - 8) | subject
  Parameter estimate(s):
   Phi
 0.664

 Number of Observations: 945
 Number of Groups: 231

  logLik      df     AIC     BIC
 -1493.8       7  3001.6  3035.5
```

The fit is in the fourth-root transformed scale for the response, computed by `bcnPower()`, with fixed effects for `age` (with origin set to `age` 8, the start of the study), `group` (`"patient"` or `"control"`), and their interaction; a random intercept by `subject`; and continuous first-order autoregressive errors.

We obtain a component-plus-residual plot for the `age` × `group` interaction just as we would for a linear or generalized linear model, with the result appearing in Figure 8.20:

```
plot(Effect(c("age", "group"), blackmore.mod.6.lme,
    residuals=TRUE), partial.residual=list(lty="dashed"))
```

To display partial residuals, the plot must be in the default linear-predictor scale, so we do not untransform the response as we did in Figure 7.10 (page 369). The earlier values of `age` in the graph are discrete because observations were taken at 2-year intervals up to the point of hospitalization for patients or the date of the interview for controls. Nonlinearity in the component-plus-residual plot is slight, supporting the specification of the model.

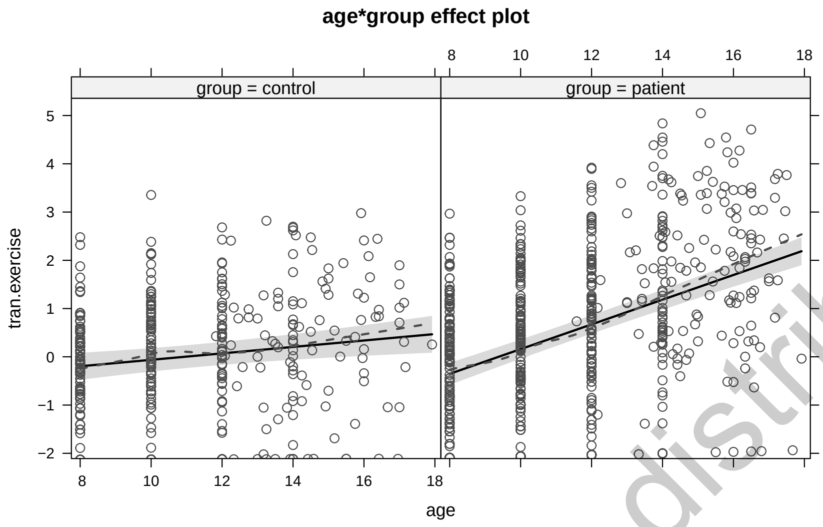

Figure 8.20 Component-plus-residual plot for the `age` × `group` interaction in the linear mixed-effects model fit to the `Blackmore` data.

8.7.2 Influence Diagnostics for Mixed Models

In mixed-effects models, it is natural to focus deletion diagnostics on clusters of cases as well as potentially on individual cases. In linear models (as discussed in Section 8.3), we can efficiently compute the effect of deleting individual cases on statistics such as estimated regression coefficients without having to refit the model. Approximate influence diagnostics for generalized linear models (in Section 8.6.2) start with the maximum-likelihood estimates of the parameters of a model for the full data set and take one step toward the new maximum-likelihood estimates when a case is deleted, rather than the computationally more expensive fully iterated solution omitting each case.

A similar approximation has been suggested for mixed-effects models by several authors (R. Christensen, Pearson, & Johnson, 1992; Demidenko & Stukel, 2005; Shi & Chen, 2008; Pan, Fie, & Foster, 2014), but not, to our knowledge, for models as general as those fit by the `lme()`, `lmer()`, and `glmer()` functions. In the **car** package, we therefore take a more computationally demanding approach, actually deleting subsets of cases corresponding to clusters and refitting the model.[20]

Deletion diagnostics are computed with a method provided by the **car** package for the standard-R `influence()` generic function; for example, for the linear mixed-effects model fit to the `Blackmore` data:

[20] The **influence.ME** package (Nieuwenhuis, te Grotenhuis, & Pelzer, 2012) takes a similar approach, but our implementation is a bit more general and offers some computational efficiencies: We accomodate models fit by `lme()` as well as those fit by `lmer()` and `glmer()`. We also start at the parameter estimates for the full data set, and, for models fit by `lmer()` or `glmer()`, we optionally allow the user to abbreviate the computation. Finally, we plan to provide parallel computations to take advantage of computers with multiple processors or cores (but have not yet implemented this feature).

```
system.time(inf.blackmore <-
    influence(blackmore.mod.6.lme, groups="subject"))

    user  system elapsed
   39.42    0.00   39.46
```

The `influence()` function refits the model deleting each of the 231 subjects (i.e., "clusters") in turn, saving information in the returned object `inf.blackmore`, of class `"influence.lme"`. On our computer, the computation takes about a minute.

The `influenceIndexPlot()` function in the **car** package has a method for objects of class `"influence.lme"` and can create plots for both the fixed effects, displaying influence on the coefficients and Cook's distances, and the random-effect parameters, shown respectively in Figures 8.21 and 8.22:

```
influenceIndexPlot(inf.blackmore)
influenceIndexPlot(inf.blackmore, var="var.cov.comps")
```

Comparing the extreme values plotted in the panels of Figure 8.21 to the magnitudes of the corresponding estimated fixed-effect parameters suggests that none of the subjects have substantial influence on the estimated fixed effects. In the `Blackmore` data set, the cases for patients appear before those for controls. Because the dummy regressor for `group` is coded 1 for patients and zero for controls, the `(Intercept)` parameter in the model is the intercept for controls, and the `I(age - 8)` parameter is the slope for controls. As a consequence, patients have no influence on these two fixed-effect coefficients, explaining the initial strings of zero values in the first two panels of Figure 8.21.

The designation of the random-effect parameters in Figure 8.22 requires a bit more explanation: The panel with vertical axis labeled `reStruct.subject` displays influence on the estimated standard deviation of the random intercepts, the panel with axis labeled `corStruct` displays influence on the estimated autoregression parameter for the errors, and the panel with axis labeled `lSigma` shows influence on the estimated *log* of the residual standard deviation. Noting that the latter is $\log(\widehat{\sigma}) = \log(1.123) = 0.116$ for the full data, we can see that none of the subjects have much influence on the random-effect parameter estimates.

For more information on the influence diagnostics provided by the **car** package for mixed models, see `help("influence.mixed.models")` and `help("influenceIndexPlot")`.

8.8 Collinearity and Variance Inflation Factors

When there are strong linear relationships among the predictors in a linear model, the precision of the estimated regression coefficients declines compared to what it would have been were the predictors uncorrelated with each other. Other important aspects of regression analysis beyond coefficients, such as prediction, may be much

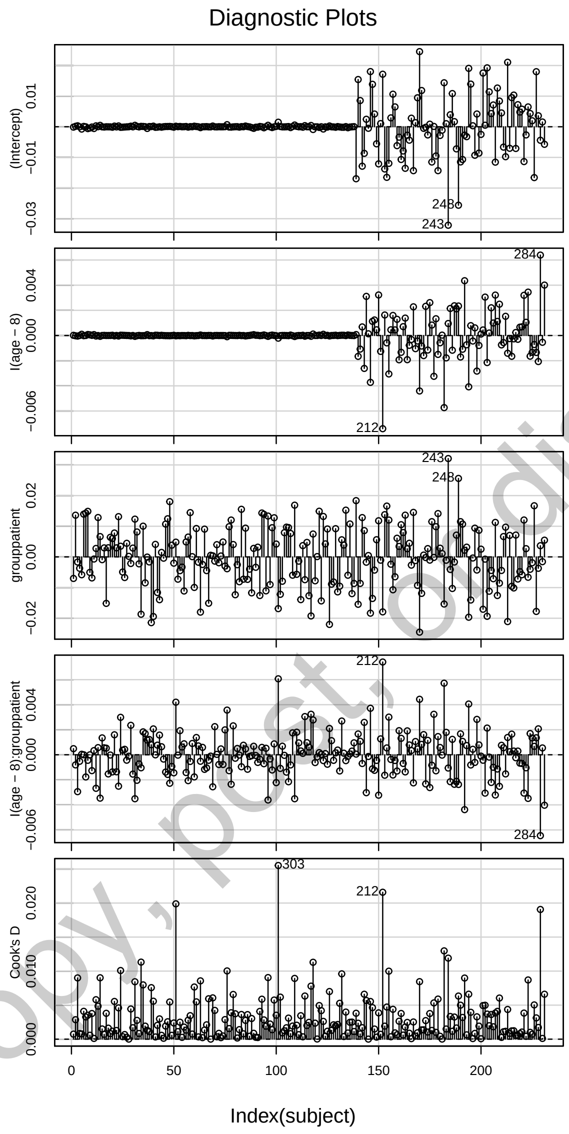

Figure 8.21 Cluster deletion diagnostics for the subjects in the linear mixed-effect model fit to the `Blackmore` data: influence on the fixed effects coefficients, including Cook's distances.

less affected by collinearity (as discussed in Weisberg, 2014, Sections 4.1.5 and 10.1, and Fox, 2016, chap. 13).

The estimated sampling variance of the jth regression coefficient may be written as

$$\widehat{\text{Var}}(b_j) = \frac{\widehat{\sigma}^2}{(n-1)s_j^2} \times \frac{1}{1-R_j^2}$$

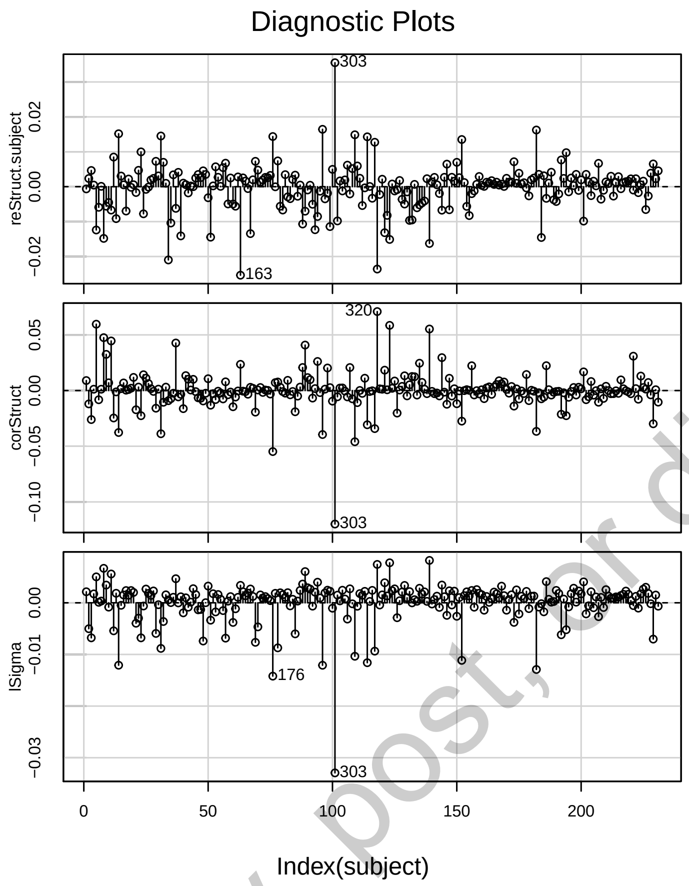

Figure 8.22 Cluster deletion diagnostics for the subjects in the linear mixed-effect model fit to the `Blackmore` data: influence on the estimated random-effects parameters. The top panel is for the standard deviation of the subject-specific intercepts, the middle panel is for the estimated autoregression parameter for the errors, and the bottom panel is for the log of the residual standard deviation.

where $\widehat{\sigma}^2$ is the estimated error variance, s_j^2 is the sample variance of x_j, and $1/(1 - R_j^2)$, called the *variance inflation factor* (VIF$_j$) for b_j, is a function of the multiple correlation R_j from the regression of x_j on the other *x*s. The variance inflation factor is a simple measure of the harm produced by collinearity: The square root of the VIF indicates how much the confidence interval for β_j is expanded relative to similar uncorrelated data, were it possible for such data to exist, as would be the case, for example, in a designed experiment. If we wish to explicate the collinear relationships among the predictors, then we can examine the coefficients from the regression of each predictor with a large VIF on the other predictors.

The variance inflation factor is not straightfowardly applicable to sets of related regressors for multiple-degree-of-freedom effects, such as polynomial regressors, spline regressors, or contrasts constructed to represent a factor. Fox and Monette (1992) generalize the notion of variance inflation by considering the relative size of the joint confidence region for the coefficients associated with a related set of regressors. The resulting measure is called a *generalized variance inflation factor* or GVIF.[21] If there are p regressors in a term, then $\text{GVIF}^{1/2p}$ is a one-dimensional expression of the decrease in precision of estimation due to collinearity, analogous to taking the square root of the usual variance inflation factor. When $p = 1$, the GVIF reduces to the VIF.

The `vif()` function in the **car** package calculates variance inflation factors for the terms in a linear model. When each term has one degree of freedom, the VIFs are returned; otherwise the GVIFs are calculated.

As a first example, consider the data on the 1980 U.S. Census undercount in the data frame `Ericksen` (Ericksen, Kadane, & Tukey, 1989):

```
summary(Ericksen)
```

```
   minority         crime           poverty
 Min.   : 0.70   Min.   : 25.0   Min.   : 6.80
 1st Qu.: 5.03   1st Qu.: 48.0   1st Qu.: 9.95
 Median :15.80   Median : 55.0   Median :12.50
 Mean   :19.44   Mean   : 63.1   Mean   :13.47
 3rd Qu.:28.20   3rd Qu.: 73.0   3rd Qu.:16.60
 Max.   :72.60   Max.   :143.0   Max.   :23.90
   language        highschool      housing         city
 Min.   : 0.20   Min.   :17.5   Min.   : 7.0   city :16
 1st Qu.: 0.50   1st Qu.:27.4   1st Qu.: 9.4   state:50
 Median : 0.85   Median :31.6   Median :11.5
 Mean   : 1.93   Mean   :33.6   Mean   :15.7
 3rd Qu.: 2.35   3rd Qu.:41.7   3rd Qu.:20.3
 Max.   :12.70   Max.   :51.8   Max.   :52.1
  conventional      undercount
 Min.   :  0.00   Min.   :-2.31
 1st Qu.:  0.00   1st Qu.: 0.32
 Median :  0.00   Median : 1.45
 Mean   : 11.73   Mean   : 1.92
 3rd Qu.:  9.75   3rd Qu.: 3.31
 Max.   :100.00   Max.   : 8.18
```

[21] * Let $\mathbf{R}_{11}$ represent the correlation matrix among the regressors in the set in question, $\mathbf{R}_{22}$ the correlation matrix among the other regressors in the model, and $\mathbf{R}$ the correlation matrix among all of the regressors in the model. Fox and Monette show that the squared area, volume, or hypervolume of the ellipsoidal joint confidence region for the coefficients in either set is expanded by the generalized variance inflation factor

$$\text{GVIF} = \frac{\det \mathbf{R}_{11} \det \mathbf{R}_{22}}{\det \mathbf{R}}$$

relative to similar data in which the two sets of regressors are uncorrelated with each other. This measure is independent of the bases selected to span the subspaces of the two sets of regressors and so, for example, is independent of the contrast-coding scheme employed for a factor.

These variables describe 66 areas of the United States, including 16 major cities, the 38 states without major cities, and the "remainders" of the 12 states that contain the 16 major cities. The following variables are included:

- `minority`: percentage of residents who are black or Hispanic
- `crime`: serious crimes per 1000 residents
- `poverty`: percentage of residents who are poor
- `language`: percentage having difficulty speaking or writing English
- `highschool`: percentage of those 25 years of age or older who have *not* finished high school
- `housing`: percentage of dwellings in small, multiunit buildings
- `city`: a factor with levels `"state"` and `"city"`
- `conventional`: percentage of households counted by personal enumeration (rather than by mail-back questionnaire with follow-ups)
- `undercount`: the estimated percentage undercount (with negative values indicating an estimated *over*count)

The `Ericksen` data set figured in a U.S. Supreme Court case concerning correction of the Census undercount.

We regress the Census `undercount` on the other variables in the data set:

```
mod.census <- lm(undercount ~ ., data=Ericksen)
brief(mod.census, pvalues=TRUE)
```

```
            (Intercept) minority  crime poverty language
Estimate         -0.611 0.079834 0.0301 -0.1784   0.2151
Std. Error        1.721 0.022609 0.0130  0.0849   0.0922
Pr(>|t|)          0.724 0.000827 0.0241  0.0401   0.0232
           highschool housing citystate conventional
Estimate       0.0613 -0.0350    -1.160     0.036989
Std. Error     0.0448  0.0246     0.771     0.009253
Pr(>|t|)       0.1764  0.1613     0.138     0.000186

 Residual SD = 1.43 on 57 df, R-squared = 0.708
```

The dot (`.`) on the right-hand side of the model formula represents all of the variables in the `Ericksen` data frame with the exception of the response, `undercount`.

Checking for collinearity, we see that the three coefficients for `minority`, `poverty`, and `highschool` have variance inflation factors exceeding 4, indicating that confidence intervals for these coefficients are more than twice as wide as they would be for uncorrelated predictors:

```
vif(mod.census)
```

```
    minority        crime      poverty     language
      5.0091       3.3436       4.6252       1.6356
  highschool      housing         city conventional
      4.6192       1.8717       3.5378       1.6913
```

To illustrate the computation of generalized variance inflation factors, we return to Ornstein's least-squares interlocking-directorate regression (fit on page 408), where it turns out that collinearity is relatively slight:

```
vif(mod.ornstein)

                GVIF Df GVIF^(1/(2*Df))
log(assets) 1.9087  1          1.3816
nation      1.4434  3          1.0631
sector      2.5968  9          1.0544
```

The `vif()` function can also be applied to generalized linear models,[22] such as the quasi-Poisson regression model fit to Ornstein's data (page 423):

```
vif(mod.ornstein.qp.2)

                GVIF Df GVIF^(1/(2*Df))
log2(assets) 2.6171  1          1.6178
nation       1.6196  3          1.0837
sector       3.7178  9          1.0757
```

The same approach works for the fixed-effect coefficients in linear and generalized linear mixed models.

Other, more complex, approaches to collinearity include principal-components analysis of the predictors or standardized predictors and singular-value decomposition of the model matrix or the mean-centered model matrix. These, too, are simple to implement in R: See the `princomp()`, `prcomp()`, `svd()`, and `eigen()` functions (the last two of which are discussed in Section 10.3).

8.9 Additional Regression Diagnostics

There are several regression-diagnostics functions in the **car** package that we don't describe in this chapter:

`boxCoxVariable()`: Computes a *constructed variable* (Atkinson, 1985) for the Box-Cox transformation of the response variable in a linear model. The constructed variable is added to the regression equation, and an added-variable plot for the constructed variable in the augmented regression then provides a visualization of leverage and influence on the determination of the normalizing power transformation of the response.

[22] Thanks to a contribution from Henric Nilsson.

`boxTidwell()`: Fits the model $y = \beta_0 + \beta_1 x_1^{\lambda_1} + \cdots + \beta_k x_k^{\lambda_k} + \varepsilon$, for positive *x*s, estimating the linearizing power transformation parameters $\lambda_1, \ldots, \lambda_k$ by maximum likelihood using an algorithm suggested by Box and Tidwell (1962). Not all of the regressors (*x*s) in the model need be candidates for transformation. Constructed variables for the Box-Tidwell power transformations are simply $x_j \log(x_j)$; added-variable plots for the constructed variables, added to the linear regression of *y* on the *x*s, visualize leverage and influence on the determination of the linearizing transformations $\widehat{\lambda}_j$.

`durbinWatsonTest()`: Computes autocorrelations and generalized Durbin-Watson statistics (Durbin & Watson, 1950, 1951), along with their bootstrapped *p*-values, for the residuals from a linear model fit to time-series data. The `durbinWatsonTest()` function is discussed in the online appendix to the *R Companion* on time-series regression.

`inverseResponsePlot()`: For a linear model, this function provides a visual method for selecting a normalizing response transformation. The function draws an *inverse response plot* (Weisberg, 2014), with the response *y* on the vertical axis and the fitted values $\widehat{y}$ on the horizontal axis, and uses nonlinear least squares to estimate the power λ in the equation $\widehat{y} = b_0 + b_1 y^{\lambda}$, adding the fitted curve to the graph.

`leveneTest()`: Computes Levene's test (see Conover, Johnson, & Johnson, 1981) for homogeneity of variance across groups defined by one or more factors in an analysis of variance.

`spreadLevelPlot()`: Applied to a linear model, `spreadLevelPlot()` generalizes Tukey's *spread-level plot* for exploratory data analysis (Tukey, 1977), plotting log absolute studentized residuals versus log fitted values (see Fox, 2016, Section 12.2). A positive relationship in the spread-level plot indicates a tendency of residual variation to increase with the magnitude of the response, and the slope of a line fit to the plot (say *b*) can be used to select a variance-stabilizing power transformation (the $1 - b$ power). The spread-level plot requires a positive response and positive fitted values; negative fitted values are ignored.

`yjPower()`: This function, which implements a method due to Yeo and Johnson (2000), provides an alternative to the `bcnPower()` family of modified power transformations when there are zero or negative values in the variable to be transformed.

8.10 Complementary Reading and References

- Residuals and residual plots for linear models are discussed in Weisberg (2014, Sections 9.1–9.2). Added-variable plots are discussed in Weisberg

(2014, Section 3.1). Outliers and influence are taken up in Weisberg (2014, chap. 9).

- Diagnostics for unusual and influential data are described in Fox (2016, chap. 11); for nonnormality, nonconstant error variance, and nonlinearity in Fox (2016, chap. 12); and for collinearity in Fox (2016, chap. 13). Diagnostics for generalized linear models are taken up in Fox (2016, Section 15.4).
- A general treatment of residuals in models without additive errors, which expands on the discussion in Section 8.6.1, is given by Cox and Snell (1968).
- For further information on various aspects of regression diagnostics, see R. D. Cook and Weisberg (1982, 1994, 1997, 1999), Fox (1991), R. D. Cook (1998), and Atkinson (1985).