Intermediate Analytics

Fatemeh Ahmadi

ALY 6015

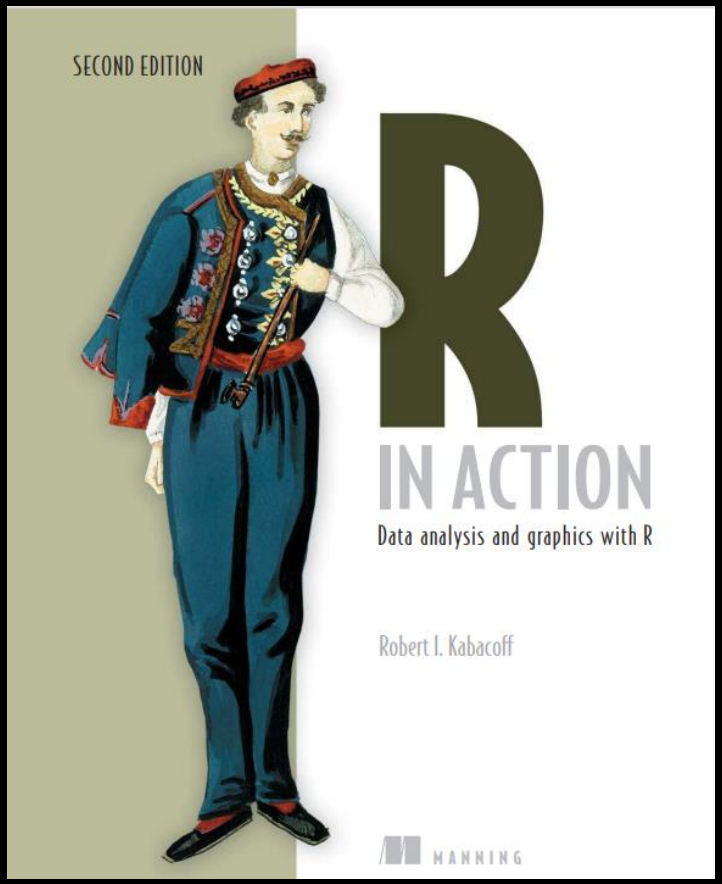
Regression

Slides are mainly borrowed from the textbook:

- ***R in Action, 2nd Edition, R. Kabacoff, Manning***

You will learn in this course:

- Fitting and interpreting linear models
- Evaluating model assumptions
- Selecting among competing models



Introduction

In general, regression analysis can be used to identify **the explanatory variables that are related to a response variable**, to describe the form of the relationships involved, and to provide an equation for predicting the response variable from the explanatory variables.

For example, an exercise physiologist might use regression analysis to develop an equation for predicting the expected number of calories a person will burn while exercising on a treadmill. The response variable is the number of calories burned (calculated from the amount of oxygen consumed), and the predictor variables might include duration of exercise (minutes), percentage of time spent at their target heart rate, average speed (mph), age (years), gender, and body mass index (BMI).

Introduction

From a theoretical point of view, the analysis will help answer such questions as these:

- What’s the relationship between exercise duration and calories burned? Is it linear or curvilinear? For example, does exercise have less impact on the number of calories burned after a certain point?
- How does effort (the percentage of time at the target heart rate, the average walking speed) factor in?
- Are these relationships the same for young and old, male and female, heavy and slim?

Introduction

From a practical point of view, the analysis will help answer such questions as the following:

- How many calories can a 30-year-old man with a BMI of 28.7 expect to burn if he walks for 45 minutes at an average speed of 4 miles per hour and stays within his target heart rate 80% of the time?
- What's the minimum number of variables you need to collect in order to accurately predict the number of calories a person will burn when walking?
- How accurate will your prediction tend to be?

Many Faces of Regression

Table 8.1 Varieties of regression analysis

Type of regression	Typical use
Simple linear	Predicting a quantitative response variable from a quantitative explanatory variable.
Polynomial	Predicting a quantitative response variable from a quantitative explanatory variable, where the relationship is modeled as an nth order polynomial.
Multiple linear	Predicting a quantitative response variable from two or more explanatory variables.
Multilevel	Predicting a response variable from data that have a hierarchical structure (for example, students within classrooms within schools). Also called *hierarchical*, *nested*, or *mixed* models.
Multivariate	Predicting more than one response variable from one or more explanatory variables.
Logistic	Predicting a categorical response variable from one or more explanatory variables.
Poisson	Predicting a response variable representing counts from one or more explanatory variables.
Cox proportional hazards	Predicting time to an event (death, failure, relapse) from one or more explanatory variables.
Time-series	Modeling time-series data with correlated errors.
Nonlinear	Predicting a quantitative response variable from one or more explanatory variables, where the form of the model is nonlinear.
Nonparametric	Predicting a quantitative response variable from one or more explanatory variables, where the form of the model is derived from the data and not specified a priori.
Robust	Predicting a quantitative response variable from one or more explanatory variables using an approach that's resistant to the effect of influential observations.

OLS Regression

In this chapter, we'll focus on regression methods that fall under the rubric of **ordinary least squares (OLS) regression, including simple linear regression, polynomial regression, and multiple linear regression**. OLS regression is the most common variety of statistical analysis today. Other types of regression models (including logistic regression and Poisson regression) will be covered in chapter 13.

In OLS regression, a quantitative dependent variable is predicted from a weighted sum of predictor variables, where the weights are parameters estimated from the data. Let's take a look at a concrete example:

OLS Regression

For most of this chapter, we'll be predicting the response variable from a set of predictor variables (also called regressing the response variable on the predictor variables) using OLS. OLS regression fits models of the form:

$$\hat{Y}_i = \hat{\beta}_0 + \hat{\beta}_1 X_{1i} + \ldots + \hat{\beta}_k X_{ki} \quad i = 1 \ldots n$$

where *n* is the number of observations and *k* is the number of predictor variables. In this equation:

OLS Regression

$\hat{Y}_i$ is the predicted value of the dependent variable for observation i (specifically, it's the estimated mean of the Y distribution, conditional on the set of predictor values).

X_{ji} is the jth predictor value for the ith observation.

$\hat{\beta}_0$ is the intercept (the predicted value of Y when all the predictor variables equal zero).

$\hat{\beta}_j$ is the regression coefficient for the jth predictor (slope representing the change in Y for a unit change in X_j).

OLS Regression

$$\sum_{i=1}^{n}(Y_i - \hat{Y}_i)^2 = \sum_{i=1}^{n}(Y_i - \hat{\beta}_0 + \hat{\beta}_1 X_{1i} + \ldots + \hat{\beta}_k X_{ki})^2 = \sum_{i=1}^{n} \varepsilon_i^2$$

To properly interpret the coefficients of the OLS model, you must satisfy a number of statistical assumptions:

- *Normality*: For fixed values of the independent variables, the dependent variable is normally distributed.
- *Independence*: The *Yi* values are independent of each other.
- *Linearity*: The dependent variable is linearly related to the independent variables.
- *Homoscedasticity*: The variance of the dependent variable doesn't vary with the levels of the independent variables.

Fitting regression models with *lm()*

In *R*, the basic function for fitting a linear model is *lm()*. The format is

```
myfit <- lm(formula, data)
```

where the formula describes the model to be fit and data is the data frame containing the data to be used in fitting the model. The resulting object (*myfit*, in this case) is a list that contains extensive information about the fitted model. The formula is typically written as

```
Y ~ X1 + X2 + ... + Xk
```

where the ~ separates the response variable on the left from the predictor variables on the right, and the predictor variables are separated by + signs. Other symbols can be used to modify the formula in various ways (see table 8.2).

Fitting regression models with *lm()*

Table 8.2 Symbols commonly used in R formulas

Symbol	Usage
~	Separates response variables on the left from the explanatory variables on the right. For example, a prediction of y from x, z, and w would be coded `y ~ x + z + w`.
+	Separates predictor variables.
:	Denotes an interaction between predictor variables. A prediction of y from x, z, and the interaction between x and z would be coded `y ~ x + z + x:z`.
*	A shortcut for denoting all possible interactions. The code `y ~ x * z * w` expands to `y ~ x + z + w + x:z + x:w + z:w + x:z:w`.
^	Denotes interactions up to a specified degree. The code `y ~ (x + z + w)^2` expands to `y ~ x + z + w + x:z + x:w + z:w`.
.	A placeholder for all other variables in the data frame except the dependent variable. For example, if a data frame contained the variables x, y, z, and w, then the code `y ~ .` would expand to `y ~ x + z + w`.
-	A minus sign removes a variable from the equation. For example, `y ~ (x + z + w)^2 - x:w` expands to `y ~ x + z + w + x:z + z:w`.
-1	Suppresses the intercept. For example, the formula `y ~ x -1` fits a regression of y on x, and forces the line through the origin at x=0.
I()	Elements within the parentheses are interpreted arithmetically. For example, `y ~ x + (z + w)^2` would expand to `y ~ x + z + w + z:w`. In contrast, the code `y ~ x + I((z + w)^2)` would expand to `y ~ x + h`, where h is a new variable created by squaring the sum of z and w.
function	Mathematical functions can be used in formulas. For example, `log(y) ~ x + z + w` would predict `log(y)` from x, z, and w.

Fitting regression models with *lm()*

In addition to *lm()*, table 8.3 lists several functions that are useful when generating a simple or multiple regression analysis. Each of these functions is applied to the object returned by *lm()* in order to generate additional information based on that fitted model.

Table 8.3 Other functions that are useful when fitting linear models

Function	Action
`summary()`	Displays detailed results for the fitted model
`coefficients()`	Lists the model parameters (intercept and slopes) for the fitted model
`confint()`	Provides confidence intervals for the model parameters (95% by default)
`fitted()`	Lists the predicted values in a fitted model
`residuals()`	Lists the residual values in a fitted model
`anova()`	Generates an ANOVA table for a fitted model, or an ANOVA table comparing two or more fitted models
`vcov()`	Lists the covariance matrix for model parameters
`AIC()`	Prints Akaike's Information Criterion
`plot()`	Generates diagnostic plots for evaluating the fit of a model
`predict()`	Uses a fitted model to predict response values for a new dataset

Simple Linear Regression

Let’s look at the functions in table 8.3 through a simple regression example.

The dataset women in the base installation provide the height and weight for a set of 15 women ages 30 to 39. Suppose you want to predict weight from height. Having an equation for predicting weight from height can help you to identify overweight or underweight individuals.

The analysis is provided in the following listing, and the resulting graph is shown in figure 8.1.

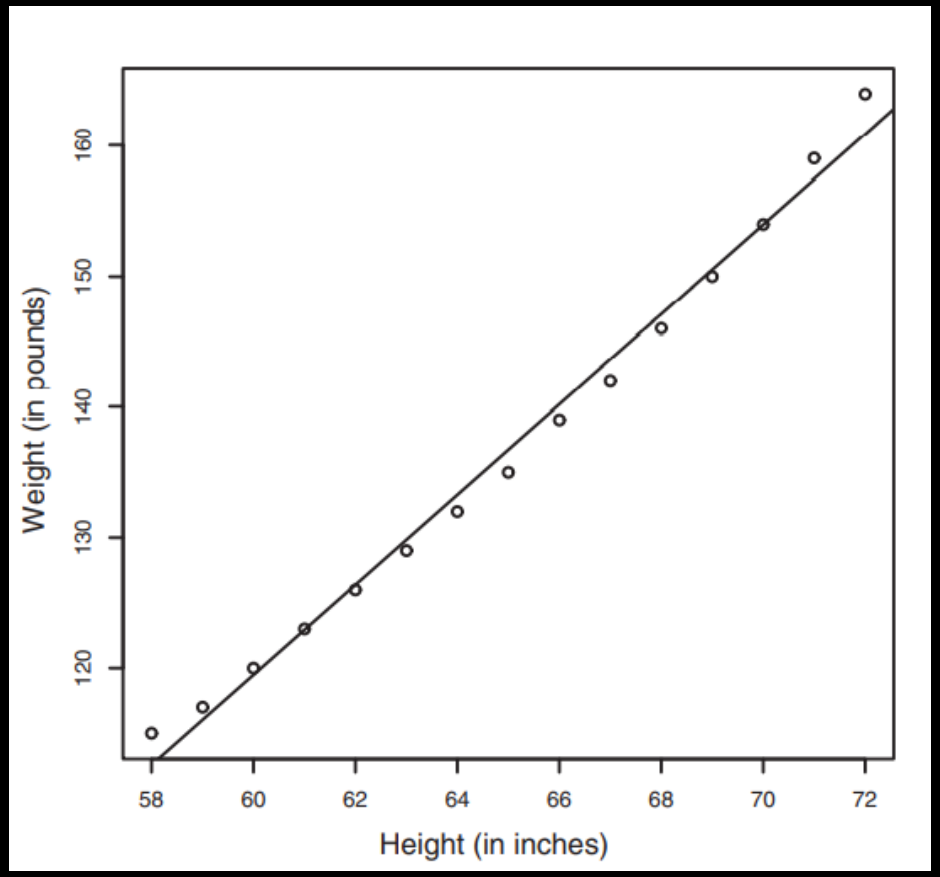


Figure 8.1 Scatter plot with regression line for weight predicted from height

```
> fit <- lm(weight~height, data=women)
> summary(fit)

Call:
lm(formula = weight ~ height, data = women)

Residuals:
    Min      1Q  Median      3Q     Max
-1.7333 -1.1333 -0.3833  0.7417  3.1167

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept) -87.51667    5.93694  -14.74 1.71e-09 ***
height        3.45000    0.09114   37.85 1.09e-14 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 1.525 on 13 degrees of freedom
Multiple R-squared:  0.991,     Adjusted R-squared:  0.9903
F-statistic:  1433 on 1 and 13 DF,  p-value: 1.091e-14

> women$weight
 [1] 115 117 120 123 126 129 132 135 139 142 146 150 154 159 164
> fitted(fit)
       1        2        3        4        5        6        7        8        9
112.5833 116.0333 119.4833 122.9333 126.3833 129.8333 133.2833 136.7333 140.1833
      10       11       12       13       14       15
143.6333 147.0833 150.5333 153.9833 157.4333 160.8833
> residuals(fit)
          1           2           3           4           5           6           7
 2.41666667  0.96666667  0.51666667  0.06666667 -0.38333333 -0.83333333 -1.28333333
          8           9          10          11          12          13          14
-1.73333333 -1.18333333 -1.63333333 -1.08333333 -0.53333333  0.01666667  1.56666667
         15
 3.11666667
> plot(women$height,women$weight)
> abline(fit)
>
```

Simple Linear Regression

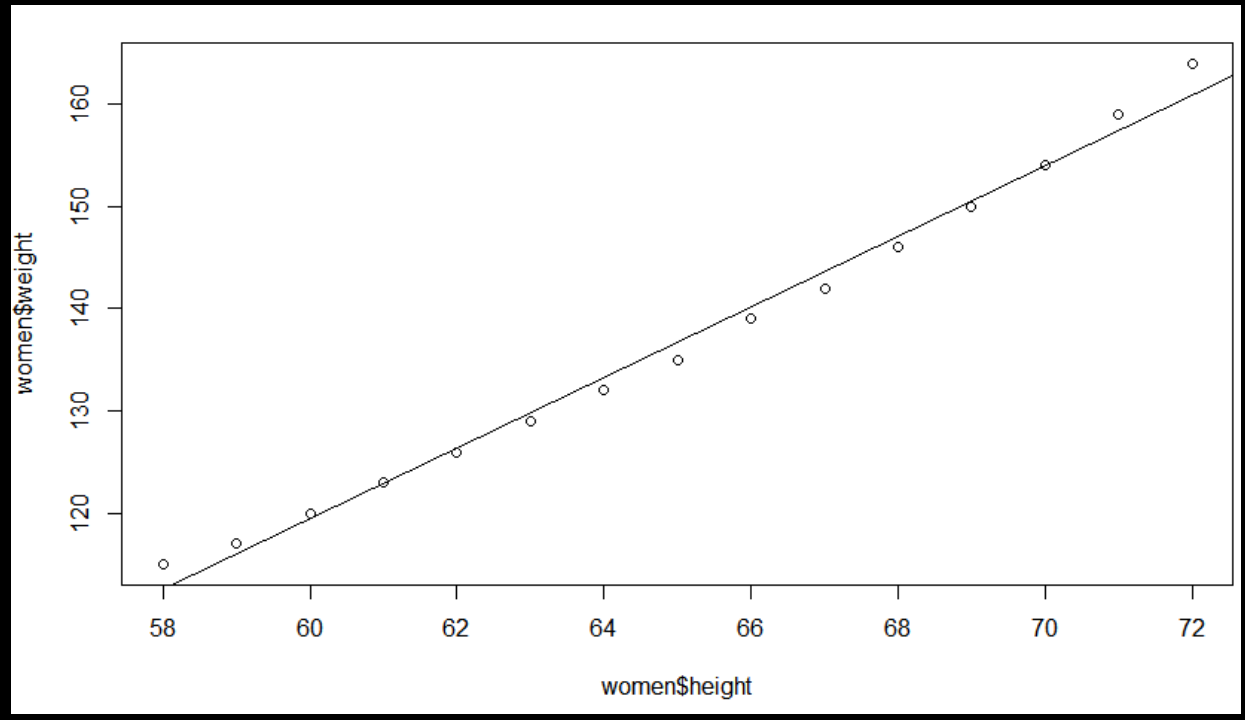
$$\widehat{\text{Weight}} = -87.52 + 3.45 \times \text{Height}$$

```
Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept) -87.51667    5.93694  -14.74 1.71e-09 ***
height        3.45000    0.09114   37.85 1.09e-14 ***
```

The plot suggests that you might be able to improve on the prediction by using a line with one bend. For example, a model of the form:

$$\hat{Y}_i = \hat{\beta}_0 + \hat{\beta}_1 X + \hat{\beta}_2 X^2$$

May provide a better fit to the data. Polynomial regression allows you to predict a response variable from an explanatory variable, where the form of the relationship is an *n*th degree polynomial.



Polynomial Regression

The plot in figure 8.1 suggests that you might be able to improve your prediction using a regression with a quadratic term (that is, X^2). You can fit a quadratic equation using the statement

```
fit2 <- lm(weight ~ height + I(height^2), data=women)
```

The new term *I(height^2)* requires an explanation. *height^2* adds a height-squared term to the prediction equation. The *I* function treats the contents within the parentheses as an *R* regular expression. You need this because the ^ operator has a special meaning in formulas that you don't want to invoke here (see table 8.2).

Polynomial Regression

Listing 8.2 Polynomial regression

```
> fit2 <- lm(weight ~ height + I(height^2), data=women
> summary(fit2)

Call:
lm(formula=weight ~ height + I(height^2), data=women)

Residuals:
    Min      1Q  Median      3Q     Max
-0.5094 -0.2961 -0.0094  0.2862  0.5971

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept) 261.87818   25.19677   10.39  2.4e-07 ***
height       -7.34832    0.77769   -9.45  6.6e-07 ***
I(height^2)   0.08306    0.00598   13.89  9.3e-09 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0

Residual standard error: 0.384 on 12 degrees of freedo
Multiple R-squared: 0.999,      Adjusted R-squared: 0.
F-statistic: 1.14e+04 on 2 and 12 DF,  p-value: <2e-16

> plot(women$height,women$weight,
       xlab="Height (in inches)",
       ylab="Weight (in lbs)")
> lines(women$height,fitted(fit2))
```

```
> fit2<-lm(weight~height+I(height^2), data=women)
> summary(fit2)

Call:
lm(formula = weight ~ height + I(height^2), data = women)

Residuals:
     Min       1Q   Median       3Q      Max
-0.50941 -0.29611 -0.00941  0.28615  0.59706

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept) 261.87818   25.19677  10.393 2.36e-07 ***
height       -7.34832    0.77769  -9.449 6.58e-07 ***
I(height^2)   0.08306    0.00598  13.891 9.32e-09 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 0.3841 on 12 degrees of freedom
Multiple R-squared:  0.9995,    Adjusted R-squared:  0.9994
F-statistic: 1.139e+04 on 2 and 12 DF,  p-value: < 2.2e-16

> plot(women$height,women$weight)
> lines(women$height,fitted(fit2))
>
```

Polynomial Regression

$$\widehat{\text{Weight}} = 261.88 - 7.35 \times \text{Height} + 0.083 \times \text{Height}^2$$

Both regression coefficients are significant at the $p < 0.0001$ level. The amount of variance accounted for has increased to 99.9%.

The significance of the squared term ($t = 13.89$, $p < .001$) suggests that the inclusion of the quadratic term improves the model fit. If you look at the plot of fit2 (figure 8.2) you can see that the curve does indeed provide a better fit.

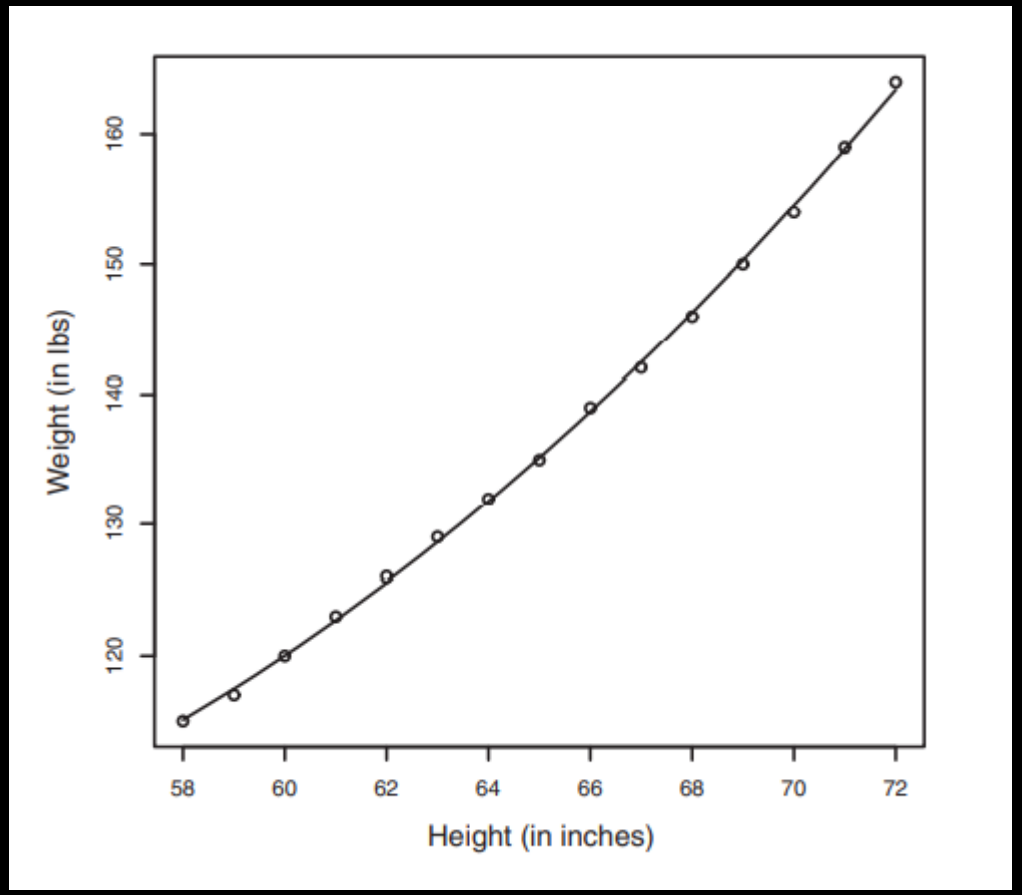


Figure 8.2 Quadratic regression for weight predicted by height

Linear Vs. Nonlinear Regression

Linear vs. nonlinear models

Note that this polynomial equation still fits under the rubric of linear regression. It's linear because the equation involves a weighted sum of predictor variables (height and height-squared in this case). Even a model such as

$$\hat{Y}_i = \hat{\beta}_0 \times \log X_1 + \hat{\beta}_2 \times \sin X_2$$

would be considered a linear model (linear in terms of the parameters) and fit with the formula

```
Y ~ log(X1) + sin(X2)
```

In contrast, here's an example of a truly nonlinear model:

$$\hat{Y}_i = \hat{\beta}_0 + \hat{\beta}_1 e^{x/\beta_2}$$

Nonlinear models of this form can be fit with the `nls()` function.

Multiple Linear Regression

When there's more than one predictor variable, simple linear regression becomes multiple linear regression, and the analysis grows more involved. Technically, polynomial regression is a special case of multiple regression. Quadratic regression has two predictors (X and X^2), and cubic regression has three predictors (X, X^2, and X^3). Let's look at a more general example.

We'll use the *state.x77* dataset in the base package for this example. Suppose you want to explore the relationship between a state's murder rate and other characteristics of the state, including population, illiteracy rate, average income, and frost levels (mean number of days below freezing):

```
> head(state.x77)
           Population Income Illiteracy Life Exp Murder HS Grad Frost   Area
Alabama          3615   3624        2.1    69.05   15.1    41.3    20  50708
Alaska            365   6315        1.5    69.31   11.3    66.7   152 566432
Arizona          2212   4530        1.8    70.55    7.8    58.1    15 113417
Arkansas         2110   3378        1.9    70.66   10.1    39.9    65  51945
California      21198   5114        1.1    71.71   10.3    62.6    20 156361
Colorado         2541   4884        0.7    72.06    6.8    63.9   166 103766
> |
```

Multiple Linear Regression

Because the *lm*() function requires a data frame (and the *state.x77* dataset is contained in a matrix), you can simplify the process with the following code:

```
states <- as.data.frame(state.x77[,c("Murder", "Population",
                                     "Illiteracy", "Income", "Frost")])
```

This code creates a data frame called *states*, containing the variables you're interested in. You'll use this new data frame for the remainder of the chapter. A good first step in multiple regression is to examine the relationships among the variables two at a time. The *bivariate correlations* are provided by the *cor()* function, and scatter plots are generated from the *scatterplotMatrix*() function in the *car* package (see the following listing and figure 8.4).

Multiple Linear Regression

Listing 8.3 Examining bivariate relationships

```
> states <- as.data.frame(state.x77[,c("Murder", "Population",
                                       "Illiteracy", "Income", "Frost")])
```

```
> cor(states)
            Murder Population Illiteracy Income Frost
Murder        1.00       0.34       0.70  -0.23 -0.54
Population    0.34       1.00       0.11   0.21 -0.33
Illiteracy    0.70       0.11       1.00  -0.44 -0.67
Income       -0.23       0.21      -0.44   1.00  0.23
Frost        -0.54      -0.33      -0.67   0.23  1.00
```

```
> library(car)
> scatterplotMatrix(states, spread=FALSE, smoother.args=list(lty=2),
    main="Scatter Plot Matrix")
```

Multiple Linear Regression

By default, the *scatterplotMatrix*() function provides scatter plots of the variables with each other in the off-diagonals and superimposes smoothed (loess) and linear fit lines on these plots.

The principal diagonal contains density and rug plots for each variable.

Now let's fit the multiple regression model with the *lm*() function (see the following listing)

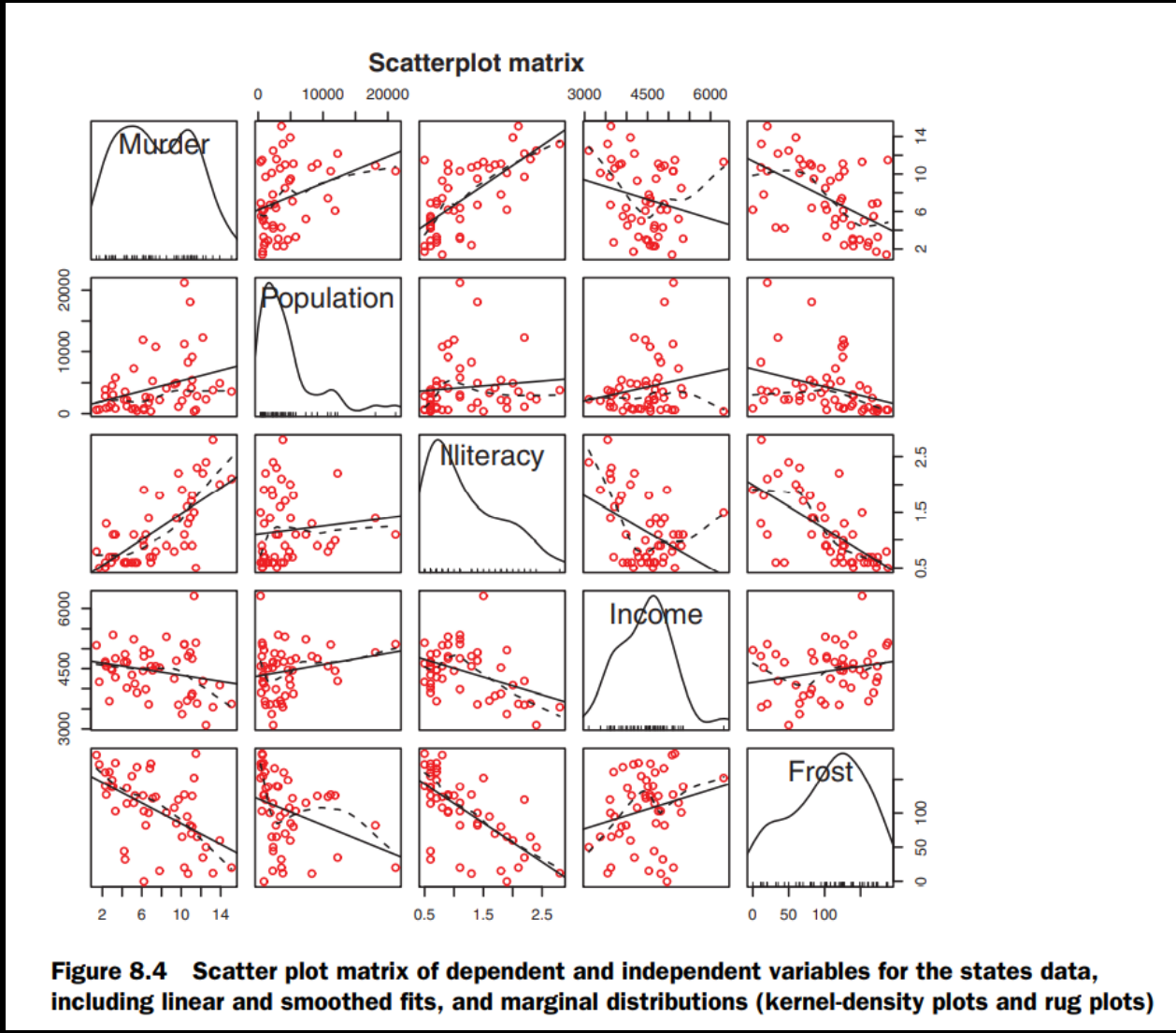


Figure 8.4 Scatter plot matrix of dependent and independent variables for the states data, including linear and smoothed fits, and marginal distributions (kernel-density plots and rug plots)

Multiple Linear Regression

```
> formula2<- lm(Murder~Population+Illiteracy+Income+Frost, data=states)
> summary(formula2)

Call:
lm(formula = Murder ~ Population + Illiteracy + Income + Frost,
    data = states)

Residuals:
    Min      1Q  Median      3Q     Max
-4.7960 -1.6495 -0.0811  1.4815  7.6210

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept) 1.235e+00  3.866e+00   0.319   0.7510
Population  2.237e-04  9.052e-05   2.471   0.0173 *
Illiteracy  4.143e+00  8.744e-01   4.738 2.19e-05 ***
Income      6.442e-05  6.837e-04   0.094   0.9253
Frost       5.813e-04  1.005e-02   0.058   0.9541
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 2.535 on 45 degrees of freedom
Multiple R-squared:  0.567,     Adjusted R-squared:  0.5285
F-statistic: 14.73 on 4 and 45 DF,  p-value: 9.133e-08
```

Multiple Linear Regression

When there's more than one predictor variable, the regression coefficients indicate the increase in the dependent variable for a unit change in a predictor variable, holding all other predictor variables constant.

For example, the regression coefficient for *Illiteracy* is 4.14, suggesting that an increase of 1% in illiteracy is associated with a 4.14% increase in the murder rate, controlling for population, income, and temperature.

The coefficient is significantly different from zero at the $p < .0001$ level. On the other hand, the coefficient for *Frost* isn't significantly different from zero ($p = 0.954$) suggesting that *Frost* and *Murder* aren't linearly related when controlling for the other predictor variables.

Taken together, the predictor variables account for 57% of the variance in murder rates across states. Up to this point, we've assumed that the predictor variables don't interact. In the next section, we'll consider a case in which they do.

Multiple Linear Regression with Interactions

Some of the most interesting research findings are those involving interactions among predictor variables. Consider the automobile data in the *mtcars* data frame. Let's say that you're interested in the impact of automobile weight and horsepower on mileage. You could fit a regression model that includes both predictors, along with their interaction, as shown in the next listing.

You can see from the Pr(>|t|) column that the interaction between *horsepower* and *car weight* is significant. What does this mean? A significant interaction between two predictor variables tells you that the relationship between one predictor and the response variable depends on the level of the other predictor. Here it means the relationship between miles per gallon and horsepower varies by car weight.

Multiple Linear Regression with Interactions

Listing 8.5 Multiple linear regression with a significant interaction term

```
> fit <- lm(mpg ~ hp + wt + hp:wt, data=mtcars)
> summary(fit)

Call:
lm(formula=mpg ~ hp + wt + hp:wt, data=mtcars)

Residuals:
   Min     1Q Median     3Q    Max
-3.063 -1.649 -0.736  1.421  4.551

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept) 49.80842    3.60516   13.82  5.0e-14 ***
hp          -0.12010    0.02470   -4.86  4.0e-05 ***
wt          -8.21662    1.26971   -6.47  5.2e-07 ***
hp:wt        0.02785    0.00742    3.75  0.00081 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 2.1 on 28 degrees of freedom
Multiple R-squared: 0.885,      Adjusted R-squared: 0.872
F-statistic: 71.7 on 3 and 28 DF,  p-value: 2.98e-13
```

Multiple Linear Regression with Interactions

The model for predicting mpg is $\widehat{mpg} = 49.81 - 0.12 \times hp - 8.22 \times wt + 0.03 \times hp \times wt$. To interpret the interaction, you can plug in various values of wt and simplify the equation. For example, you can try the mean of wt (3.2) and one standard deviation below and above the mean (2.2 and 4.2, respectively). For wt=2.2, the equation simplifies to $\widehat{mpg} = 49.81 - 0.12 \times hp - 8.22 \times (2.2) + 0.03 \times hp \times (2.2) = 31.41 - 0.06 \times hp$. For wt=3.2, this becomes $\widehat{mpg} = 23.37 - 0.03 \times hp$. Finally, for wt=4.2 the equation becomes $\widehat{mpg} = 15.33 - 0.003 \times hp$. You see that as weight increases (2.2, 3.2, 4.2), the expected change in mpg from a unit increase in hp decreases (0.06, 0.03, 0.003).

You can visualize interactions using the effect() function in the effects package. The format is

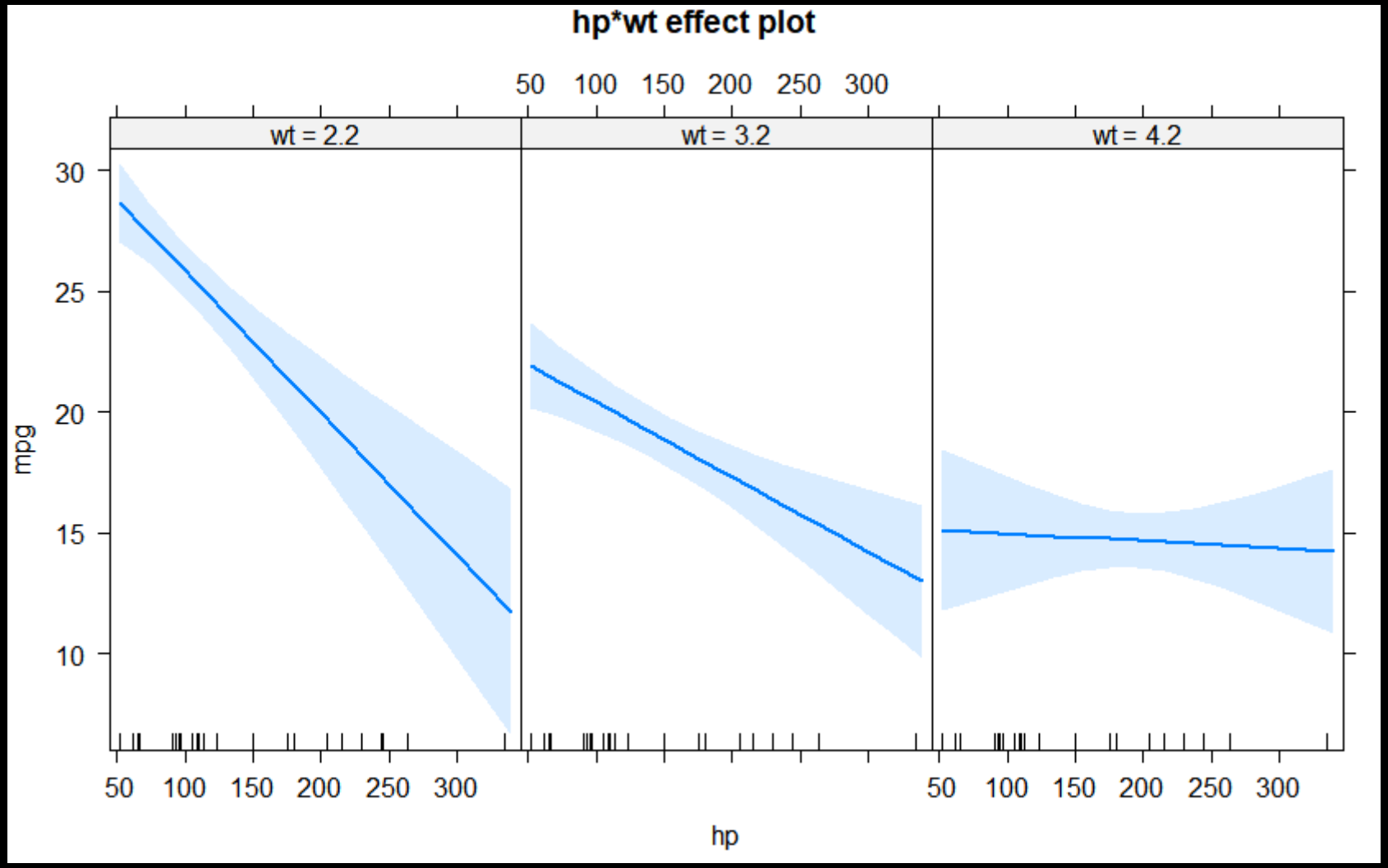
```
plot(effect(term, mod,, xlevels), multiline=TRUE)
```

where *term* is the quoted model term to plot, *mod* is the fitted model returned by lm(), and *xlevels* is a list specifying the variables to be set to constant values and the values to employ. The multiline=TRUE option superimposes the lines being plotted. For the previous model, this becomes

```
library(effects)
plot(effect("hp:wt", fit,, list(wt=c(2.2,3.2,4.2))), multiline=TRUE)
```

The resulting graph is displayed in figure 8.5.

Multiple Linear Regression with Interactions



Multiple Linear Regression with Interactions

You can see from this graph that as the weight of the car increases, the relationship between horsepower and miles per gallon weakens. For *wt*=4.2, the line is almost horizontal, indicating that as *hp* increases, *mpg* doesn't change.

Unfortunately, fitting the model is only the first step in the analysis. Once you fit a regression model, you need to evaluate whether you've met the statistical assumptions underlying your approach before you can have confidence in the inferences you draw. This is the topic of the next section.

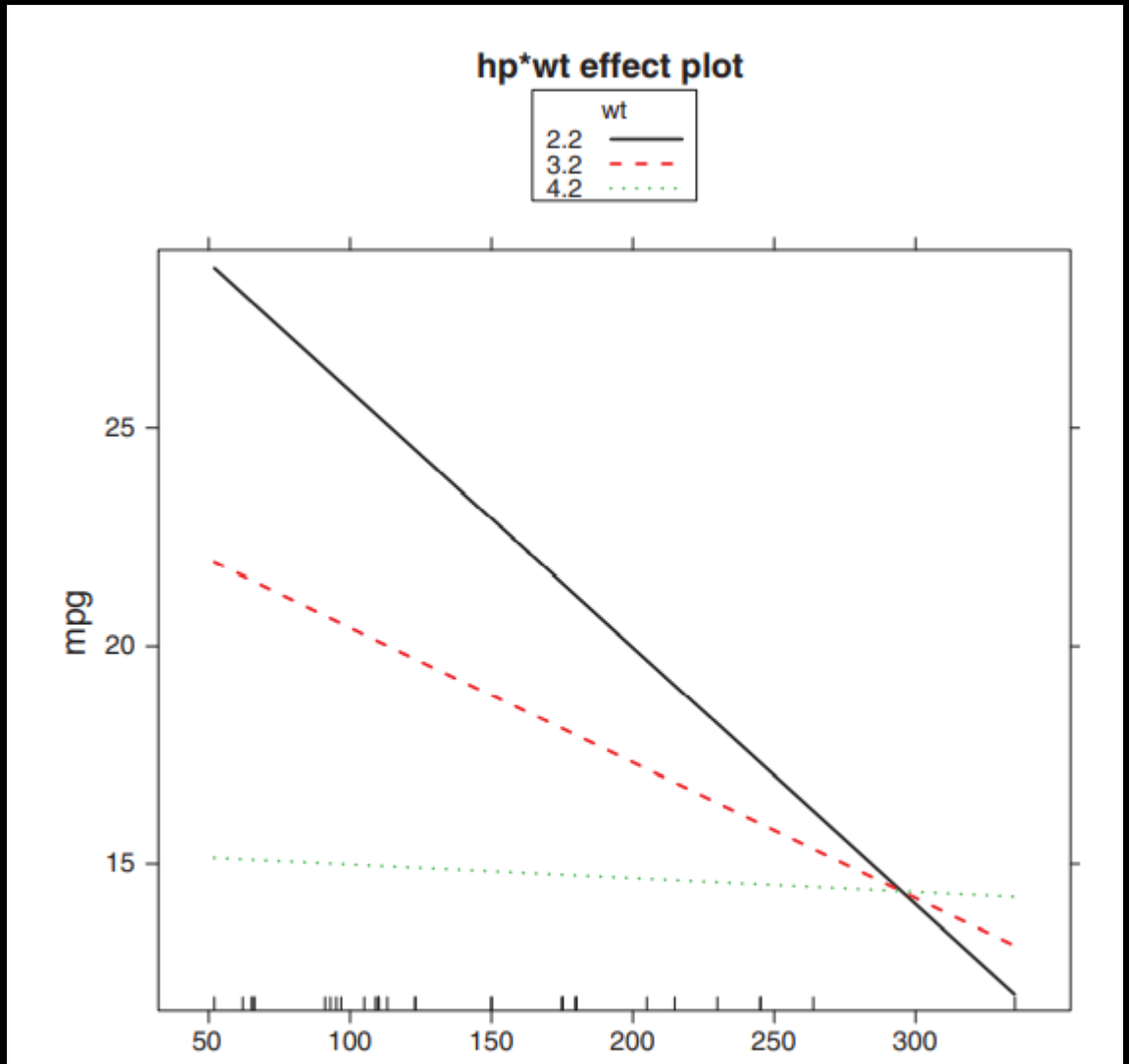


Figure 8.5 Interaction plot for `hp*wt`. This plot displays the relationship between `mpg` and `hp` at three values of `wt`.

Regression Diagnostics

In the previous section, you used the *lm*() function to fit an OLS regression model and the *summary*() function to obtain the model parameters and summary statistics. Unfortunately, nothing in this printout tells you whether the model you've fit is appropriate. Your confidence in inferences about regression parameters depends on the degree to which you've met the statistical assumptions of the OLS model. Although the *summary*() function in Listing 8.4 describes the model, it provides no information concerning the degree to which you've satisfied the statistical assumptions underlying the model.

Irregularities in the data or misspecifications of the relationships between the predictors and the response variable can lead you to settle on a model that's wildly inaccurate. On the one hand, you may conclude that a predictor and a response variable are unrelated when, in fact, they are. On the other hand, you may conclude that a predictor and a response variable are related when, in fact, they aren't! You may also end up with a model that makes poor predictions when applied in real-world settings, with significant and unnecessary errors.

Regression Diagnostics

Let's look at the output from the *confint*() function applied to the *states* multiple regression problem in section 8.2.4:

```
> states <- as.data.frame(state.x77[,c("Murder", "Population",
                                       "Illiteracy", "Income", "Frost")])
> fit <- lm(Murder ~ Population + Illiteracy + Income + Frost, data=states)
> confint(fit)
                  2.5 %   97.5 %
(Intercept) -6.55e+00 9.021318

 Population   4.14e-05 0.000406
 Illiteracy   2.38e+00 5.903874
 Income      -1.31e-03 0.001441
 Frost       -1.97e-02 0.020830
```

The results suggest that you can be 95% confident that the interval [2.38, 5.90] contains the true change in the murder rate for a 1% change in the *Illiteracy* rate. Additionally, because the confidence interval for *Frost* contains 0, you can conclude that a change in temperature is unrelated to the murder rate, holding the other variables constant. But your faith in these results is only as strong as the evidence you have that your data satisfies the statistical assumptions underlying the model.

Regression Diagnostics

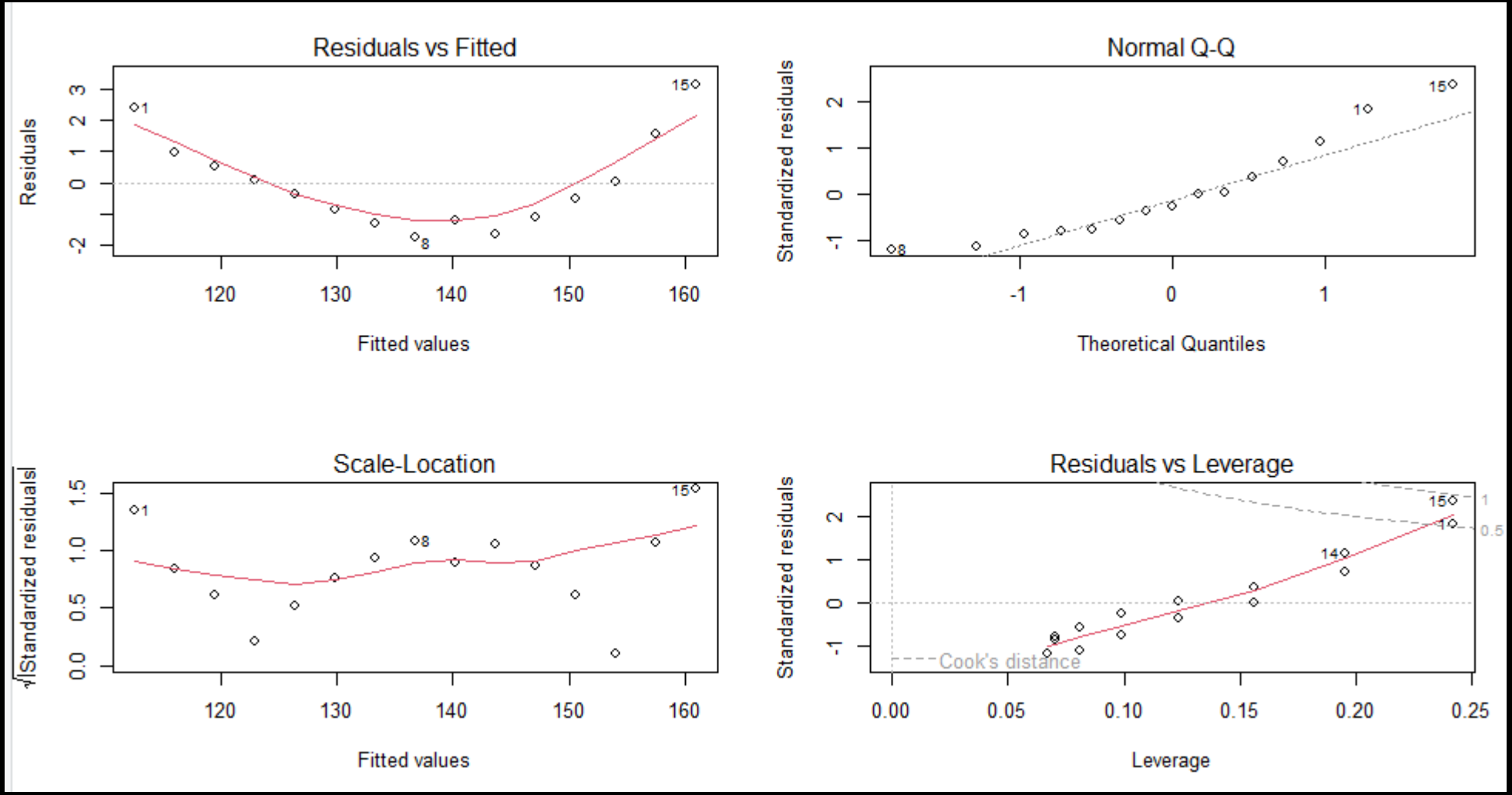
R's base installation provides numerous methods for evaluating the statistical assumptions in a regression analysis. The most common approach is to apply the *plot*() function to the object returned by the *lm*(). Doing so produces four graphs that are useful for evaluating the model fit. Applying this approach to the simple linear regression example

```
fit <- lm(weight ~ height, data=women)
par(mfrow=c(2,2))
plot(fit)
```

Produces the graphs shown in figure 8.6. The *par(mfrow=c(2,2))* statement is used to combine the four plots produced by the *plot*() function into one large 2 × 2 graph. The *par*() function is described in chapter 3.

To understand these graphs, consider the assumptions of OLS regression:

Regression Diagnostics



Regression Diagnostics

- *Normality*: If the dependent variable is normally distributed for a fixed set of predictor values, then the residual values should be normally distributed with a mean of 0. The Normal *Q-Q* plot (upper right) is a probability plot of the standardized residuals against the values that would be expected under normality. If you've met the normality assumption, the points on this graph should fall on the straight 45-degree line. Because they don't, you've clearly violated the normality assumption.

- *Independence*: You can't tell if the dependent variable values are independent of these plots. You have to use your understanding of how the data was collected. There's no a priori reason to believe that one woman's weight influences another woman's weight. If you found out that the data were sampled from families, you might have to adjust your assumption of independence.

Regression Diagnostics

- *Linearity*: If the dependent variable is linearly related to the independent variables, there should be no systematic relationship between the residuals and the predicted (that is, fitted) values. In other words, the model should capture all the systematic variance present in the data, leaving nothing but random noise. In the *Residuals vs. Fitted* graph (upper left), you see clear evidence of a curved relationship, which suggests that you may want to add a quadratic term to the regression.

- *Homoscedasticity:* If you've met the constant variance assumption, the points in the Scale-Location graph (bottom left) should be a random band around a horizontal line. You seem to meet this assumption.

Regression Diagnostics

Finally, the *Residuals vs. Leverage* graph (bottom right) provides information about individual observations that you may wish to attend to. The graph identifies outliers, high-leverage points, and influential observations. Specifically:

- An **outlier** is an observation that isn't predicted well by the fitted regression model (that is, has a large positive or negative residual).
- An observation with a high **leverage** value has an unusual combination of predictor values. That is, it's an outlier in the predictor space. The dependent variable value isn't used to calculate an observation's leverage.
- An **influential observation** is an observation that has a disproportionate impact on the determination of the model parameters. Influential observations are identified using a statistic called *Cook's distance, or Cook's D.*

Regression Diagnostics

```
fit2 <- lm(weight ~ height + I(height^2), data=women)
par(mfrow=c(2,2))
plot(fit2)
```

This second set of plots suggests that the **polynomial regression** provides a better fit with regard to the linearity assumption, normality of residuals (except for observation 13), and homoscedasticity (constant residual variance). Observation 15 appears to be influential (based on a large Cook's D value), and deleting it has an impact on the parameter estimates. In fact, dropping both observations 13 and 15 produces a better model fit

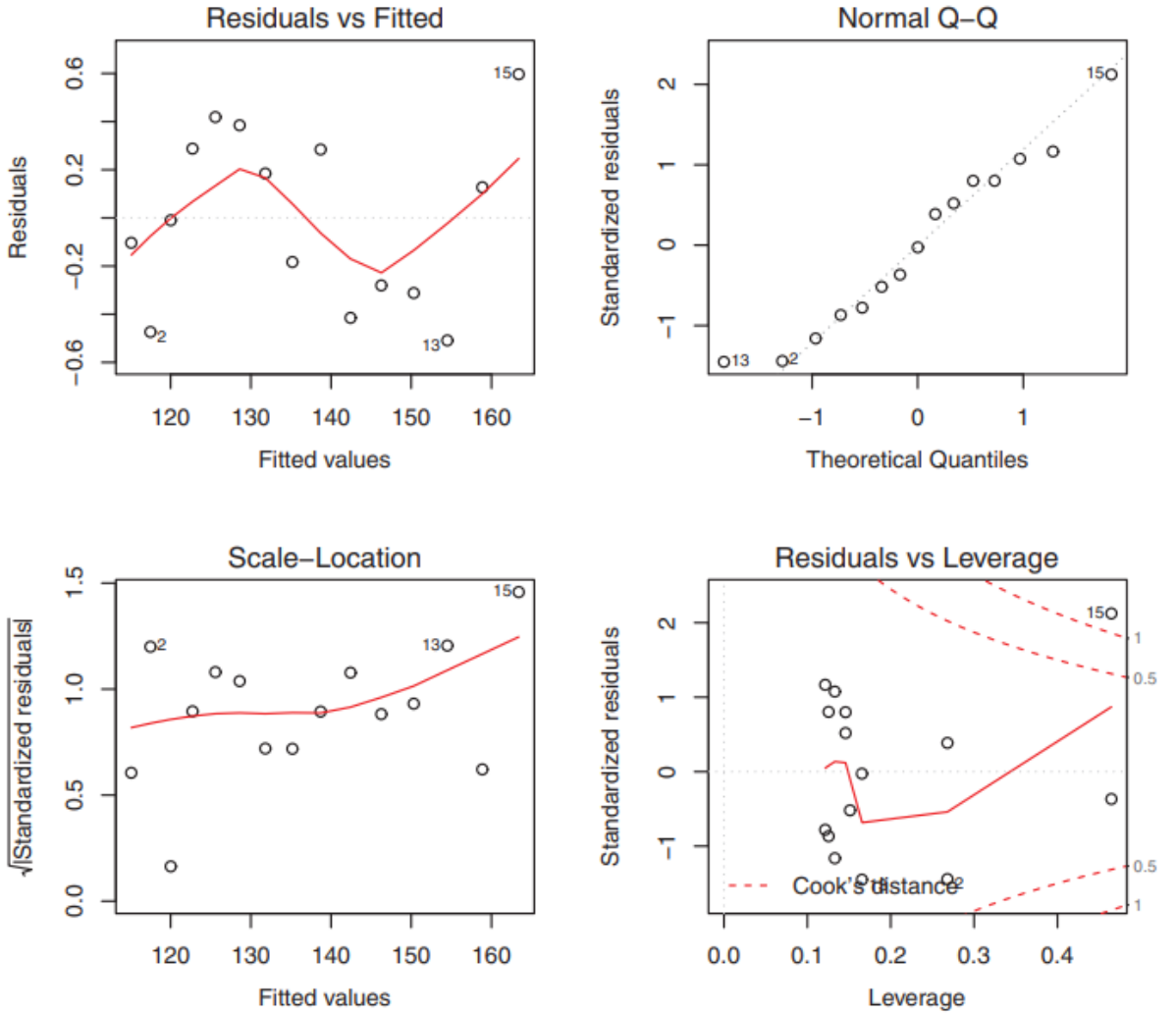


Figure 8.7 Diagnostic plots for the regression of weight on height and height-squared

Regression Diagnostics

```
newfit <- lm(weight~ height + I(height^2), data=women[-c(13,15),])
```

The results are displayed in figure 8.8.

As you can see from the graph, the model assumptions appear to be well satisfied, with the exception that Nevada is an outlier

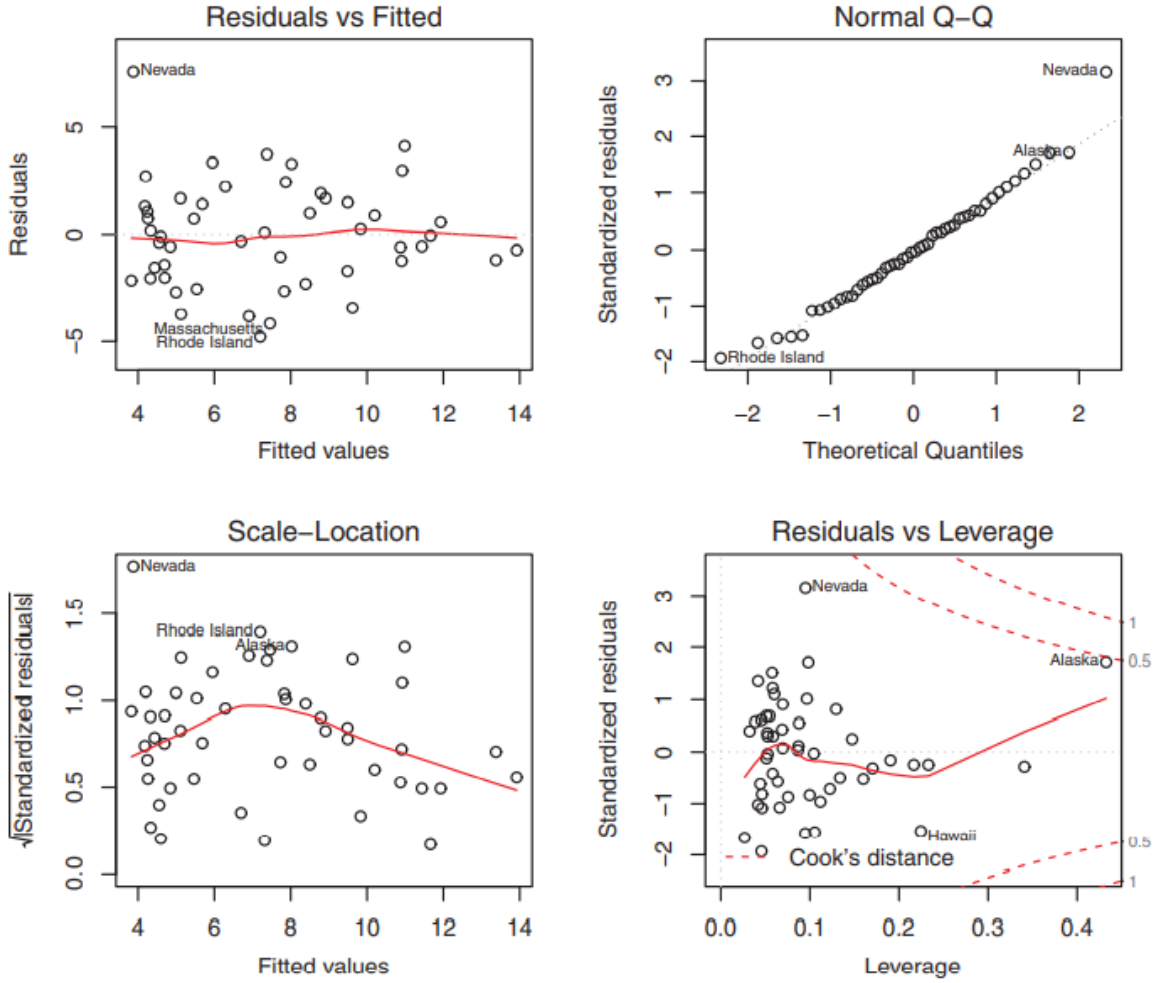


Figure 8.8 Diagnostic plots for the regression of murder rate on state characteristics

Regression Diagnostics

Table 8.4 Useful functions for regression diagnostics (`car` package)

Function	Purpose
`qqPlot()`	Quantile comparisons plot
`durbinWatsonTest()`	Durbin–Watson test for autocorrelated errors
`crPlots()`	Component plus residual plots
`ncvTest()`	Score test for nonconstant error variance
`spreadLevelPlot()`	Spread-level plots
`outlierTest()`	Bonferroni outlier test
`avPlots()`	Added variable plots
`influencePlot()`	Regression influence plots
`scatterplot()`	Enhanced scatter plots
`scatterplotMatrix()`	Enhanced scatter plot matrixes
`vif()`	Variance inflation factors

Normality

The *qqPlot*() function provides a more accurate method of assessing the normality assumption than that provided by the *plot*() function in the base package. It plots the studentized residuals (also called *studentized deleted residuals* or *jackknifed residuals*) against a *t* distribution with $n - p - 1$ degree of freedom, where n is the sample size and p is the number of regression parameters (including the intercept). The code follows:

```
library(car)
states <- as.data.frame(state.x77[,c("Murder", "Population",
                        "Illiteracy", "Income", "Frost")])
fit <- lm(Murder ~ Population + Illiteracy + Income + Frost, data=states)
qqPlot(fit, labels=row.names(states), id.method="identify",
       simulate=TRUE, main="Q-Q Plot")
```

Normality

The *qqPlot*() function generates the probability plot displayed in figure 8.9. The option *id.method ="identify*" makes the plot interactive—after the graph is drawn, the mouse clicks on points in the graph and will label them with values specified in the labels option of the function.

Pressing the *Esc* key, selecting *Stop* from the graph's drop-down menu, or right-clicking the graph turns off this interactive mode.

Here, I identified Nevada. When *simulate=TRUE*, a 95% confidence envelope is produced using a parametric bootstrap. (Bootstrap methods are considered in chapter 12.)

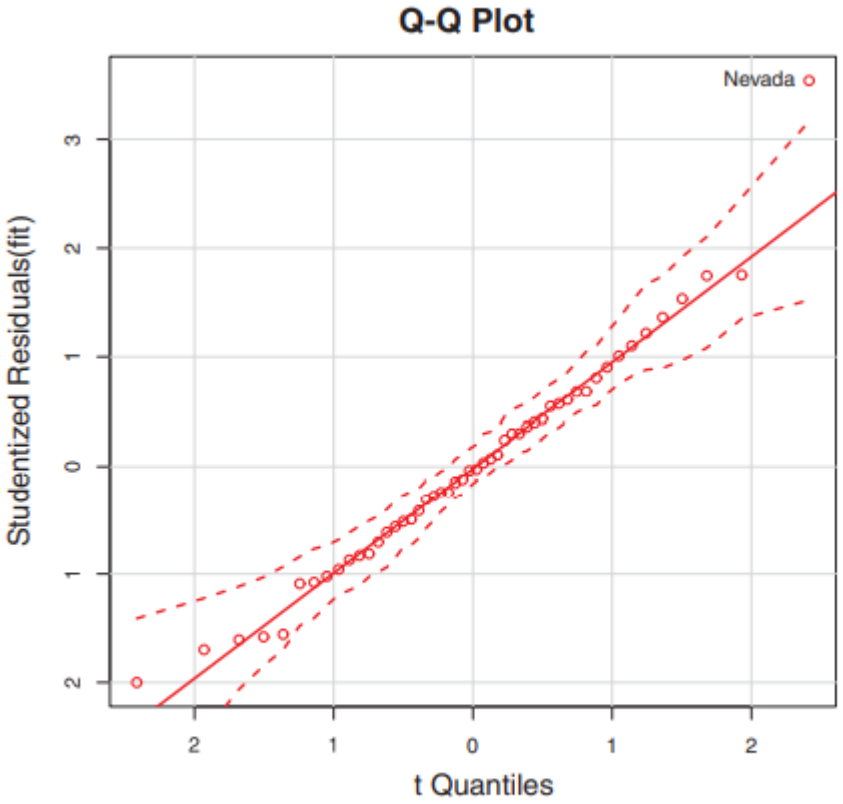


Figure 8.9 Q-Q plot for studentized residuals

Normality

```
> states["Nevada",]

       Murder Population Illiteracy Income Frost
Nevada   11.5        590        0.5   5149   188

> fitted(fit)["Nevada"]

  Nevada
3.878958

> residuals(fit)["Nevada"]

  Nevada
7.621042

> rstudent(fit)["Nevada"]

  Nevada
3.542929
```

Listing 8.6 Function for plotting studentized residuals

```
residplot <- function(fit, nbreaks=10) {
               z <- rstudent(fit)
               hist(z, breaks=nbreaks, freq=FALSE,
                    xlab="Studentized Residual",
                    main="Distribution of Errors")
               rug(jitter(z), col="brown")
               curve(dnorm(x, mean=mean(z), sd=sd(z)),
                     add=TRUE, col="blue", lwd=2)
               lines(density(z)$x, density(z)$y,
                     col="red", lwd=2, lty=2)
               legend("topright",
                      legend = c( "Normal Curve", "Kernel Density Curve"),
                      lty=1:2, col=c("blue","red"), cex=.7)
             }

residplot(fit)
```

Normality

As you can see, the errors follow a normal distribution quite well, with the exception of a large outlier. Although the *Q-Q plot* is probably more informative, I've always found it easier to gauge the skew of a distribution from a *histogram* or *density plot* than from a probability plot. Why not use both?

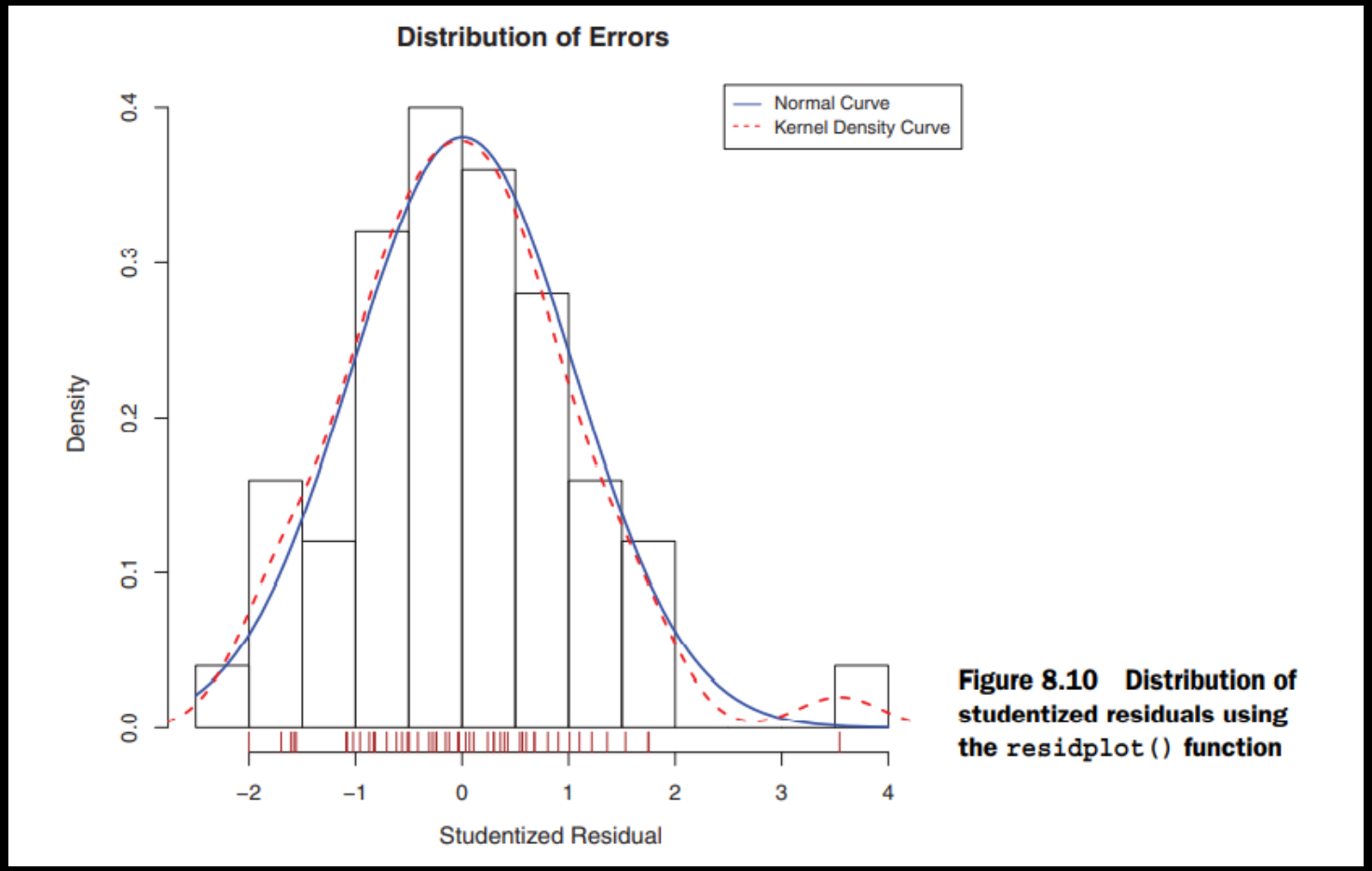


Figure 8.10 Distribution of studentized residuals using the `residplot()` function

Independence of Errors

As indicated earlier, the best way to assess whether the dependent variable values (and thus the residuals) are independent is from your knowledge of how the data were collected. For example, time series data often display autocorrelation—observations collected closer in time are more correlated with each other than with observations distant in time. The car package provides a function for the *Durbin–Watson* test to detect such serially correlated errors. You can apply the *Durbin–Watson* test to the multiple-regression problem with the following code:

```
> durbinWatsonTest(fit)
 lag Autocorrelation D-W Statistic p-value
   1          -0.201           2.32   0.282
 Alternative hypothesis: rho != 0
```

The nonsignificant ***p-value*** (p=0.282) suggests a lack of autocorrelation and, conversely, independence of errors. The ***lag*** value (1 in this case) indicates that each observation is being compared with the one next to it in the dataset. Although appropriate for time-dependent data, the test is less applicable for data that isn't clustered in this fashion. Note that the *durbinWatsonTest*() function uses bootstrapping (see chapter 12) to derive *p-values*. Unless you add the option *simulate=FALSE*, you'll get a slightly different value each time you run the test.

Linearity

You can look for evidence of nonlinearity in the relationship between the dependent variable and the independent variables by using ***component plus residual plots*** **(also known as *partial residual plots*)**. The plot is produced by the *crPlots*() function in the *car* package. You're looking for any systematic departure from the linear model that you've specified. To create a component plus residual plot for variable *X*, you plot the points.

$$\varepsilon_i + (\hat{\beta}_0 + \hat{\beta}_1 \times X_{1i} + \ldots + \hat{\beta}_k \times X_{ki}) \text{ vs. } X_i$$

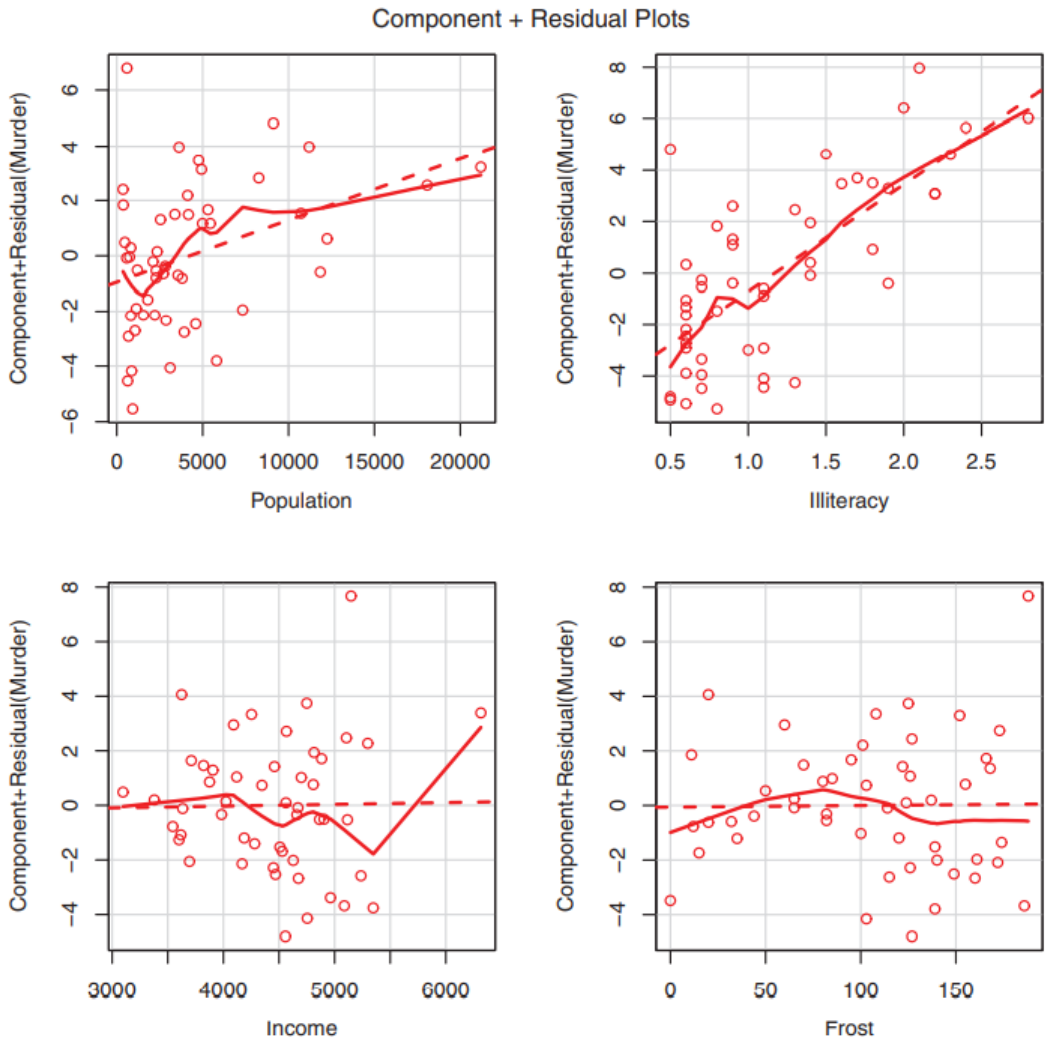


Figure 8.11 Component plus residual plots for the regression of murder rate on state characteristics

Linearity

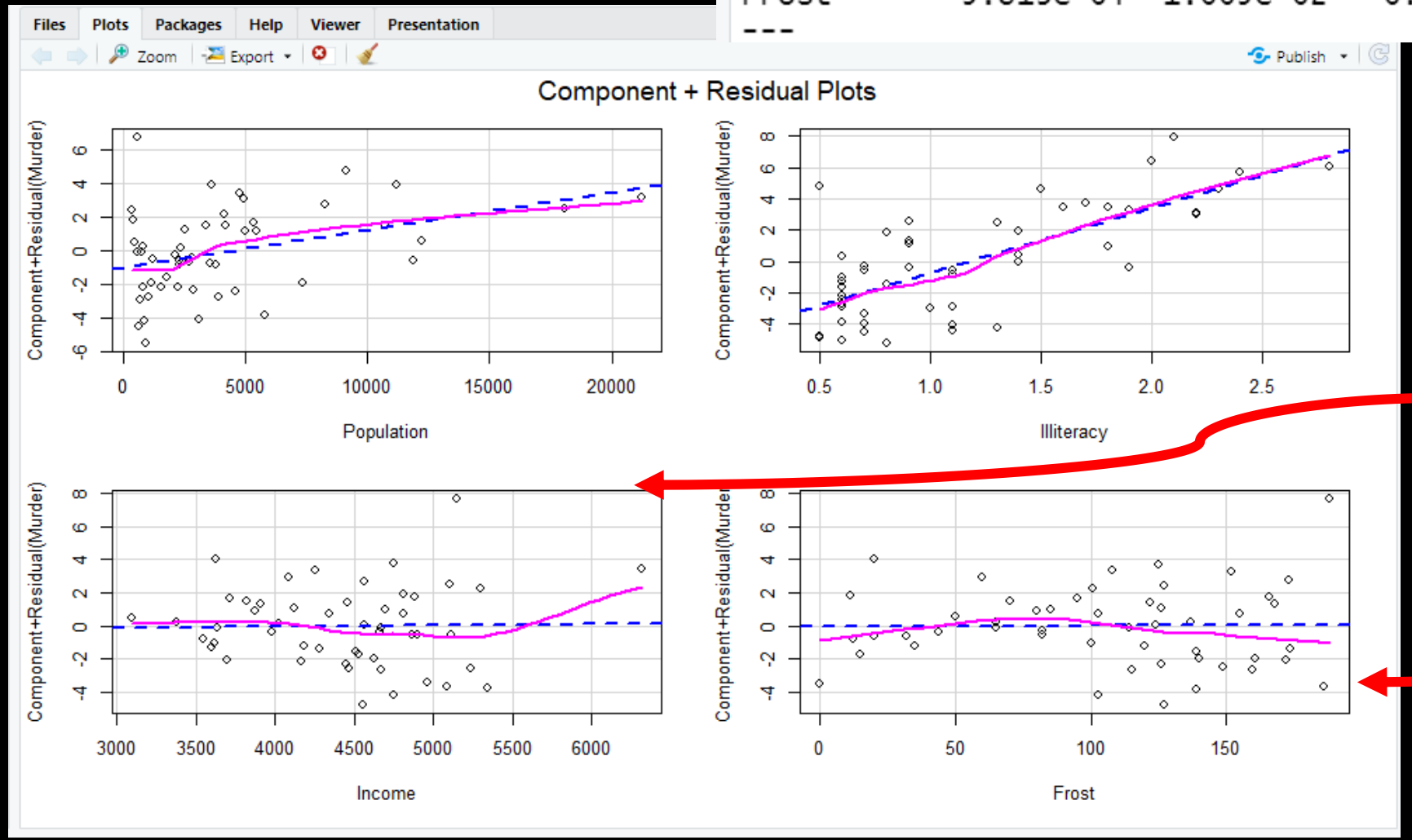
where the residuals are based on the full model, and i =1 … *n*. The straight line in each graph is given by $(\hat{\beta}_0 + \hat{\beta}_1 \times X_{1i} + ... + \hat{\beta}_k \times X_{ki})$ vs. X_i. Loess fit lines are described in chapter 11. The code to produce these plots is as follows:

```
> library(car)
> crPlots(fit)
```

The resulting plots are provided in figure 8.11. Nonlinearity in any of these plots suggests that you may not have adequately modeled the functional form of that predictor in the regression. If so, you may need to add curvilinear components such as polynomial terms, transform one or more variables (for example, use *log(X)* instead of *X*), or abandon linear regression in favor of some other regression variant. Transformations are discussed later in this chapter.

Linearity

```
Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept) 1.235e+00  3.866e+00   0.319   0.7510
Population  2.237e-04  9.052e-05   2.471   0.0173 *
Illiteracy  4.143e+00  8.744e-01   4.738 2.19e-05 ***
Income      6.442e-05  6.837e-04   0.094   0.9253
Frost       5.813e-04  1.005e-02   0.058   0.9541
---
```



Any Increase in *Income* and *Frost* variables will not change the *Murder* rate (in this linear modeling).

Homoscedasticity

The *car* package also provides two useful functions for identifying non-constant error variance. The *ncvTest*() function produces a score test of the hypothesis of constant error variance against the alternative that the error variance changes with the level of the fitted values. A significant result suggests heteroscedasticity (nonconstant error variance). The *spreadLevelPlot*() function creates a scatter plot of the absolute standardized residuals versus the fitted values and superimposes a line of best fit. Both functions are demonstrated in the next listing:

Listing 8.7 Assessing homoscedasticity

```
> library(car)
> ncvTest(fit)

Non-constant Variance Score Test
Variance formula: ~ fitted.values
Chisquare=1.7    Df=1     p=0.19

> spreadLevelPlot(fit)

Suggested power transformation:  1.2
```

Homoscedasticity

The score test is nonsignificant ($p = 0.19$), suggesting that you've met the constant variance assumption. You can also see this in the spread-level plot (figure 8.12). The points form a random horizontal band around a horizontal line of best fit. If you'd violated the assumption, you'd expect to see a nonhorizontal line. The suggested power transformation in listing 8.7 is the suggested power p (Y^p) that would stabilize the nonconstant error variance. For example, if the plot showed a nonhorizontal trend and the suggested power transformation was 0.5, then using rather than Y in the regression equation might lead to a model that satisfied homoscedasticity. If the suggested power was 0, you'd use a log transformation. In the current example, there's no evidence of heteroscedasticity, and the suggested power is close to 1 (no transformation required).

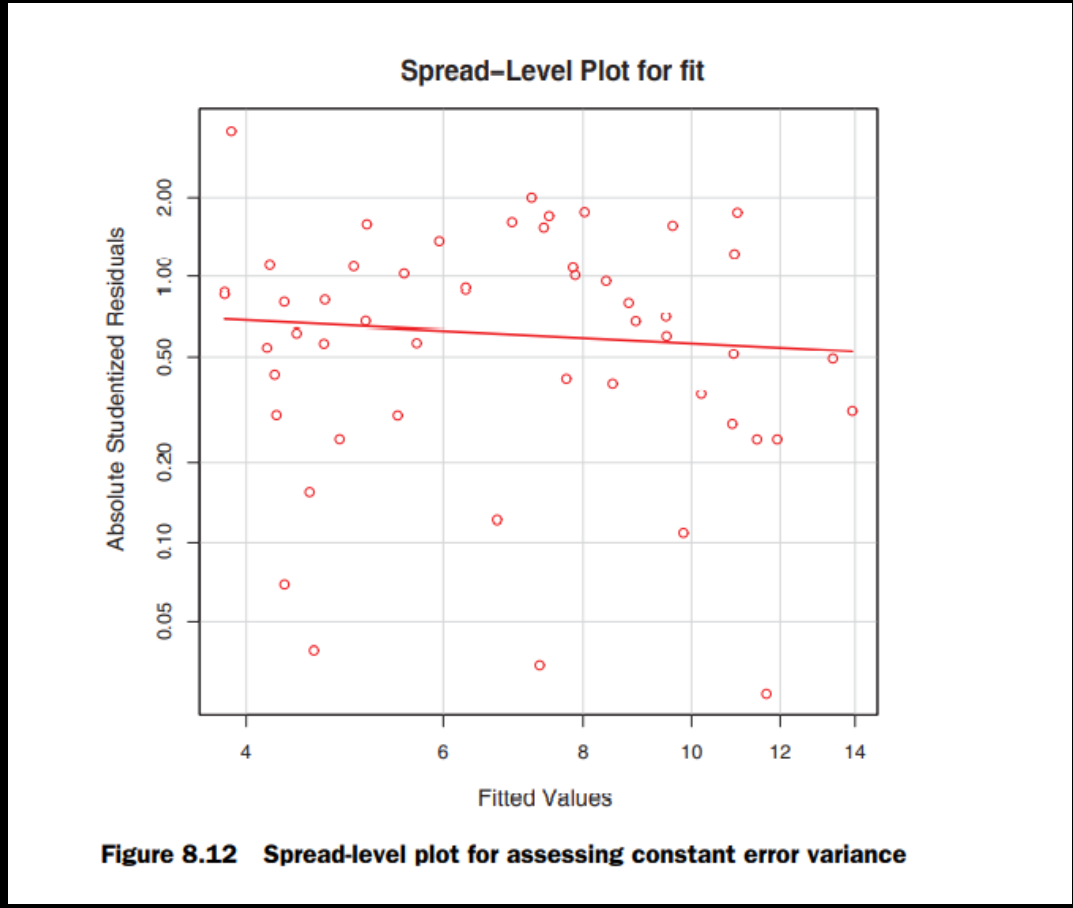


Figure 8.12 Spread-level plot for assessing constant error variance

Global Validation of Linear Model Assumption

Finally, let's examine the *gvlma*() function in the *gvlma* package. Written by Pena and Slate (2006), the *gvlma*() function performs a global validation of linear model assumptions as well as separate evaluations of skewness, kurtosis, and heteroscedasticity. In other words, it provides a single omnibus (go/no go) test of model assumptions. The following listing applies the test to the state's data.

Listing 8.8 Global test of linear model assumptions

```
> library(gvlma)
> gvmodel <- gvlma(fit)
> summary(gvmodel)

ASSESSMENT OF THE LINEAR MODEL ASSUMPTIONS
USING THE GLOBAL TEST ON 4 DEGREES-OF-FREEDOM:
Level of Significance= 0.05

Call:
 gvlma(x=fit)

                    Value p-value                Decision
Global Stat         2.773   0.597 Assumptions acceptable.
Skewness            1.537   0.215 Assumptions acceptable.
Kurtosis            0.638   0.425 Assumptions acceptable.
Link Function       0.115   0.734 Assumptions acceptable.
Heteroscedasticity  0.482   0.487 Assumptions acceptable.
```

Multicollinearity

The problem is that DOB and age are perfectly correlated with rounding errors. A regression coefficient measures the impact of one predictor variable on the response variable, holding all other predictor variables constant. This amounts to looking at the relationship between grip strength and age, holding age constant. The problem is called **multicollinearity**. It leads to large confidence intervals for model parameters and makes the interpretation of individual coefficients difficult.

Multicollinearity can be detected using a statistic called the **variance inflation factor (VIF).** For any predictor variable, the square root of the VIF indicates the degree to which the confidence interval for that variable's regression parameter is expanded relative to a model with uncorrelated predictors. VIF values are provided by the *vif*() function in the *car* package. As a general rule, indicates a multicollinearity problem. The code is provided in the following listing. The results indicate that multicollinearity isn't a problem with these predictor variables.

Multicollinearity

Multicollinearity refers to the statistical phenomenon where two or more independent variables are strongly correlated. It marks the almost perfect or exact relationship between the predictors. This strong correlation between the exploratory variables is one of the major problems in linear regression analysis.

Listing 8.9 Evaluating multicollinearity

```
> library(car)
> vif(fit)

Population Illiteracy     Income      Frost
       1.2        2.2        1.3        2.1

> sqrt(vif(fit)) > 2 # problem?

Population Illiteracy     Income      Frost
     FALSE      FALSE      FALSE      FALSE
```

Outliers

Outliers are observations that aren't predicted well by the model. They have unusually large positive or negative residuals ($Y_i - \hat{Y}_i$). Positive residuals indicate that the model is underestimating the response value, whereas negative residuals indicate an overestimation.

You've already seen one way to identify outliers. Points in the Q-Q plot of figure 8.9 that lie outside the confidence band are considered outliers. A rough rule of thumb is that standardized residuals that are larger than 2 or less than –2 are worth attention.

The `car` package also provides a statistical test for outliers. The `outlierTest()` function reports the Bonferroni adjusted p-value for the largest absolute studentized residual:

```
> library(car)
> outlierTest(fit)

       rstudent unadjusted p-value Bonferonni p
Nevada      3.5             0.00095         0.048
```

Here, you see that Nevada is identified as an outlier (p = 0.048). Note that this function tests the single largest (positive or negative) residual for significance as an outlier.

If it isn't significant, there are no outliers in the dataset. If it's significant, you must delete it and rerun the test to see if others are present.

High-Leverage Points

Observations with high leverage are identified through the *hat statistic*. For a given dataset, the average hat value is p/n, where p is the number of parameters estimated in the model (including the intercept) and n is the sample size. Roughly speaking, an observation with a hat value greater than 2 or 3 times the average hat value should be examined. The code that follows plots the hat values:

```
hat.plot <- function(fit) {
              p <- length(coefficients(fit))
              n <- length(fitted(fit))
              plot(hatvalues(fit), main="Index Plot of Hat Values")
              abline(h=c(2,3)*p/n, col="red", lty=2)
              identify(1:n, hatvalues(fit), names(hatvalues(fit)))
            }
hat.plot(fit)
```

High-Leverage Points

Here you see that Alaska and California are particularly unusual when it comes to their predictor values. Alaska has a much higher income than other states while having a lower population and temperature. California has a much higher population than other states while having a higher income and higher temperature.

These states are atypical compared with the other 48 observations.

High-leverage observations may or may not be influential observations. That will depend on whether they're also outliers.

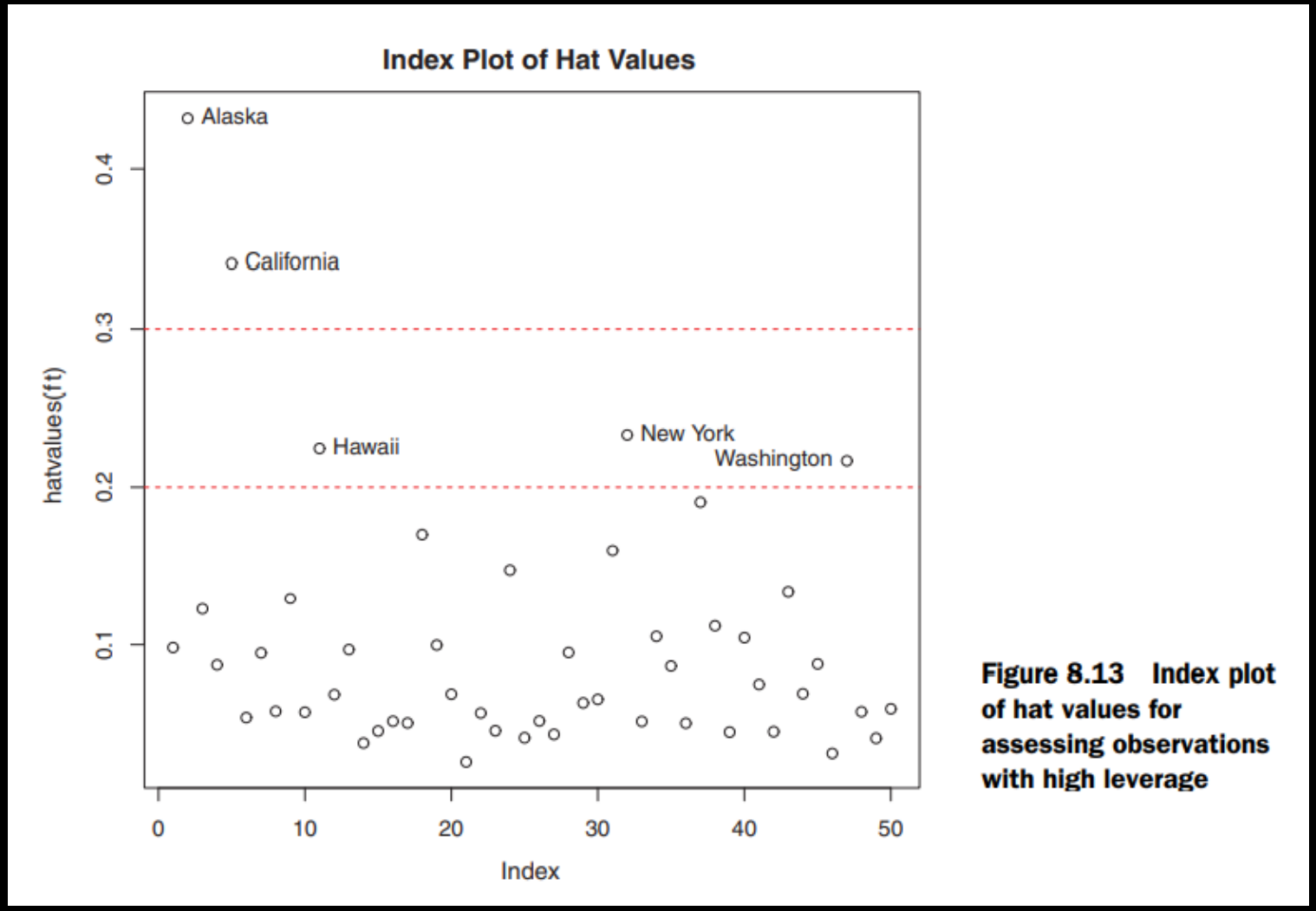


Figure 8.13 Index plot of hat values for assessing observations with high leverage

Influential Observations

Influential observations have a disproportionate impact on the values of the model parameters. **Imagine finding that your model changes dramatically with the removal of a single observation**. It's this concern that leads you to examine your data for influential points. There are two methods for identifying influential observations: ***Cook's distance (or D statistic)*** and ***added variable*** plots. Roughly speaking, *Cook's D* values greater than $4/(n - k - 1)$, where n is the sample size and k is the number of predictor variables, indicate influential observations. You can create a *Cook's D plot* (figure 8.14) with the following code:

```
cutoff <- 4/(nrow(states)-length(fit$coefficients)-2)
plot(fit, which=4, cook.levels=cutoff)
abline(h=cutoff, lty=2, col="red")
```

The graph identifies Alaska, Hawaii, and Nevada as influential observations. Deleting these states will have a notable impact on the values of the intercept and slopes in the regression model. Note that although it's useful to cast a wide net when searching for influential observations, I tend to find a cutoff of 1 more generally useful than $4/(n - k - 1)$. Given a criterion of D=1, none of the observations in the dataset would appear to be influential.

Influential Observations

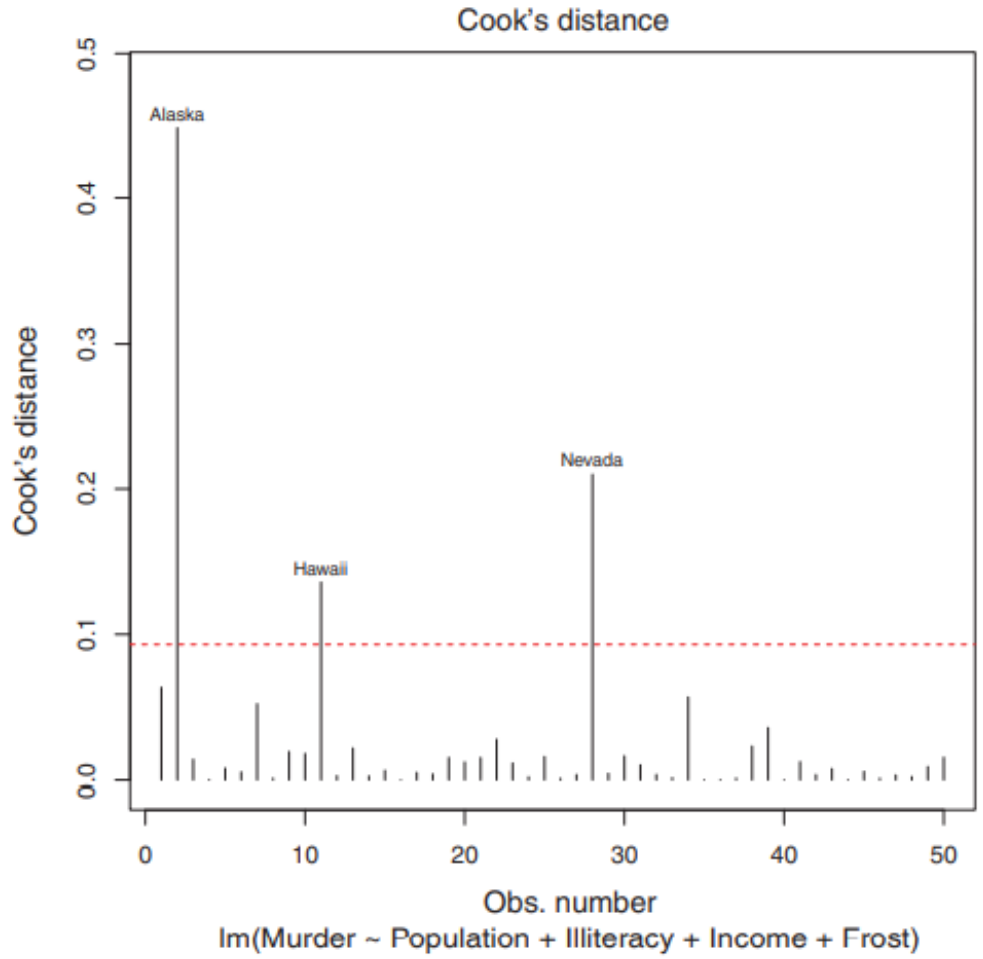


Figure 8.14 Cook's D plot for identifying influential observations

Influential Observations

The straight line in each plot is the actual regression coefficient for that predictor variable. You can see the impact of influential observations by imagining how the line would change if the point representing that observation was deleted. For example, look at the graph of *Murder | Others versus Income | Others* in the lower-left corner. You can see that eliminating the point labeled Alaska would move the line in a negative direction. In fact, deleting Alaska changes the regression coefficient for *Income* from positive (.00006) to negative (–.00085).

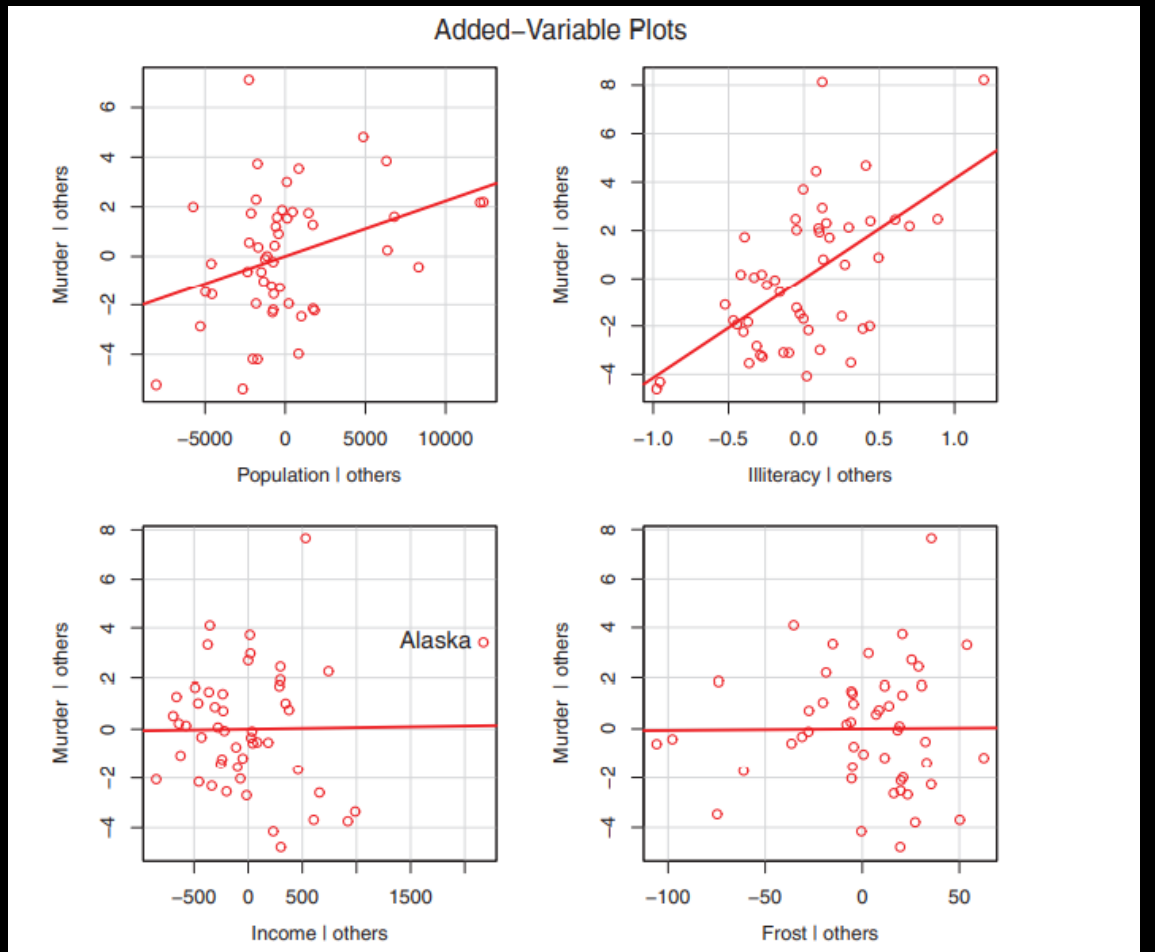
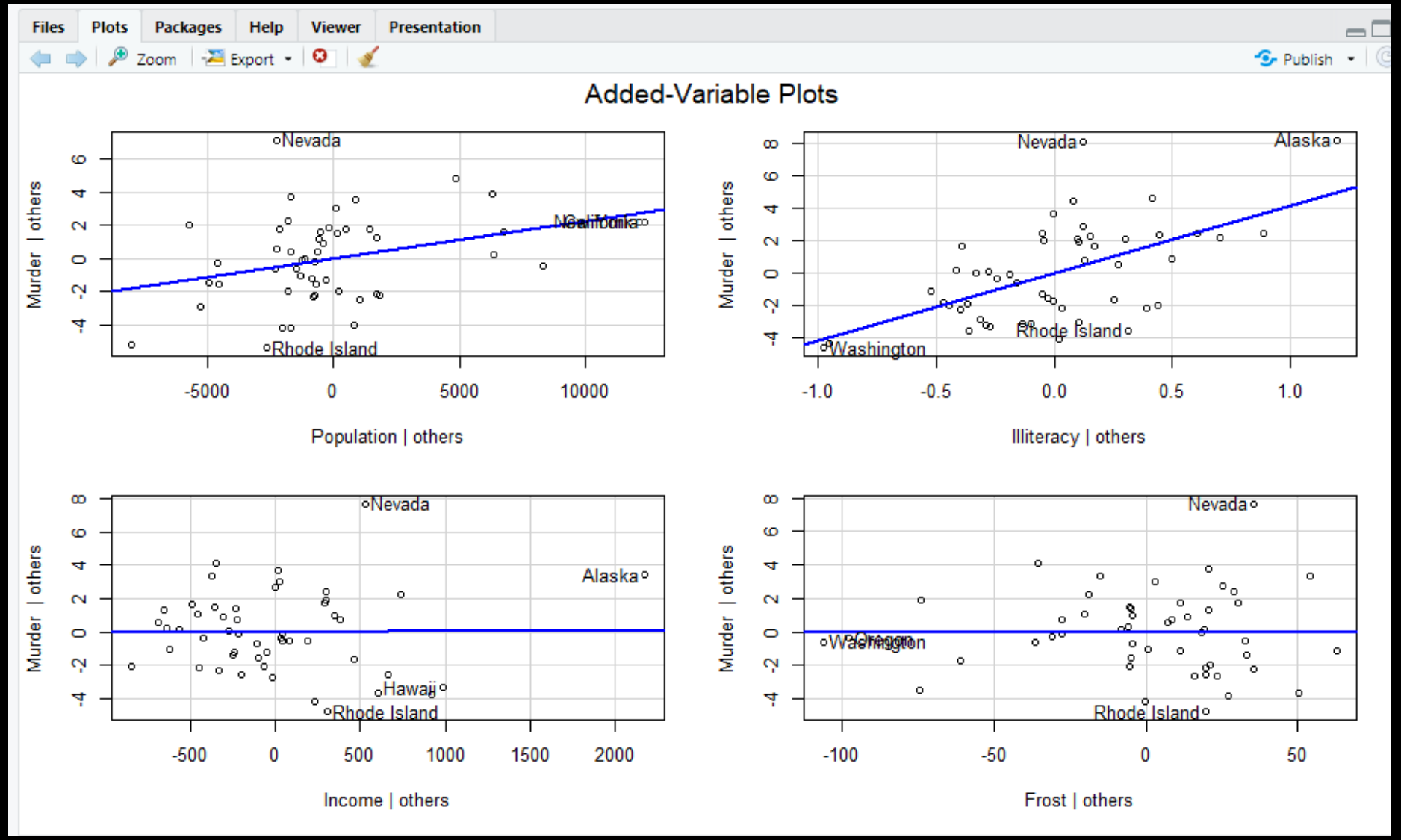


Figure 8.15 Added-variable plots for assessing the impact of influential observations

For each predictor X*k*, plot the residuals from regressing the response variable on the other *k* – 1 predictors versus the residuals from regressing X*k* on the other *k* – 1 predictors. Added-variable plots can be created using the `avPlots()` function in the `car` package:

```
library(car)
avPlots(fit, ask=FALSE, id.method="identify")
```

Influential Observations



Influential Observations

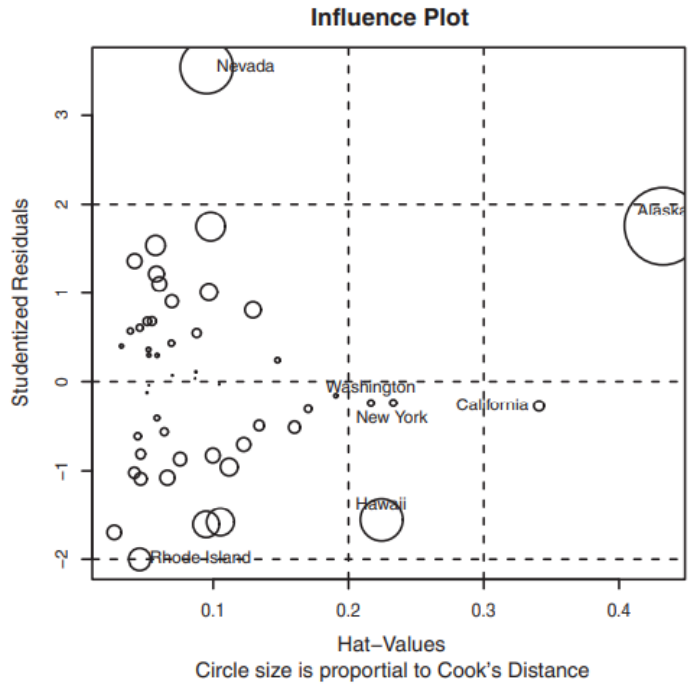


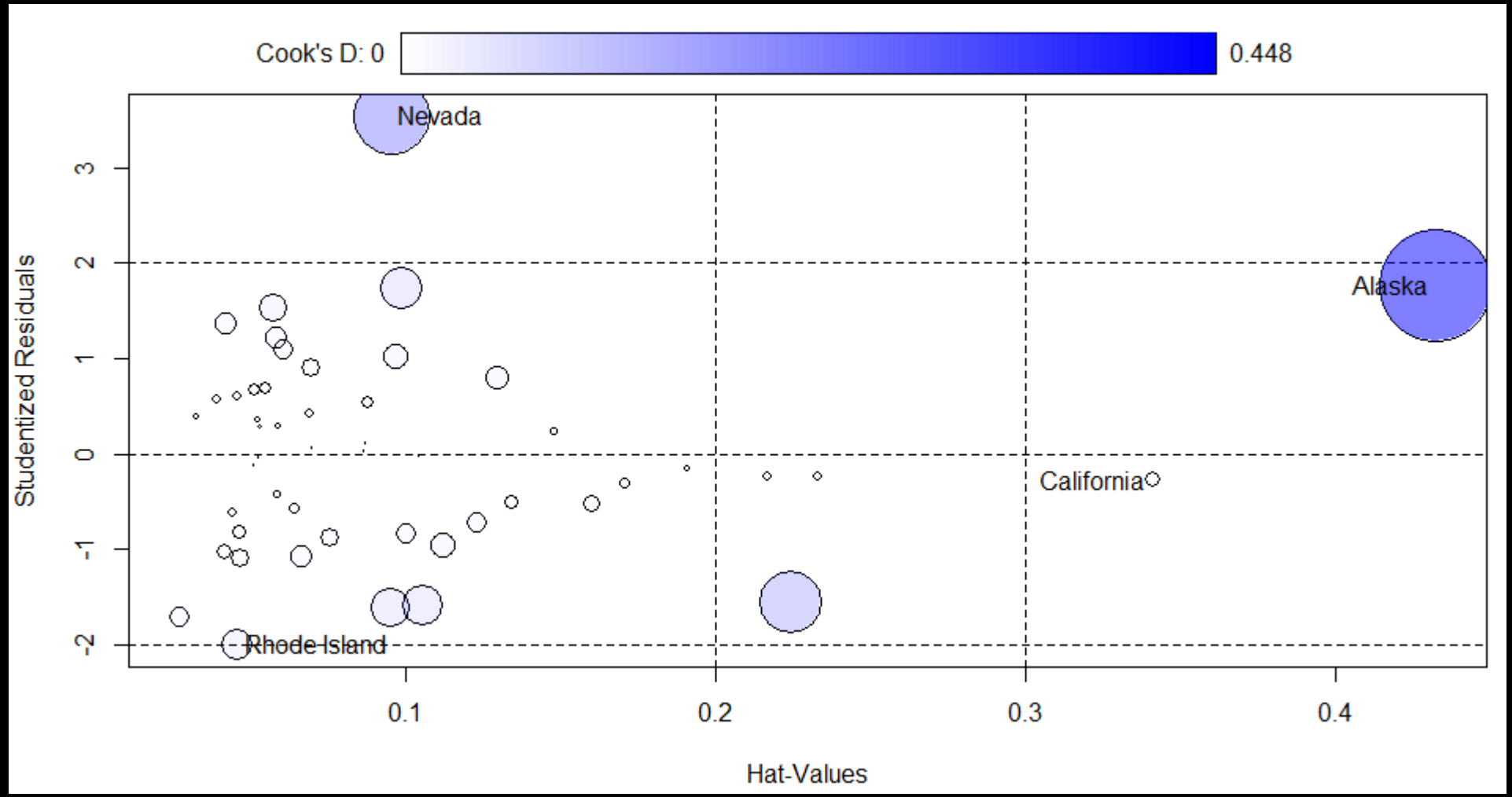
Figure 8.16 Influence plot. States above +2 or below –2 on the vertical axis are considered outliers. States above 0.2 or 0.3 on the horizontal axis have high leverage (unusual combinations of predictor values). Circle size is proportional to influence. Observations depicted by large circles may have disproportionate influence on the parameter estimates of the model.

You can combine the information from outlier, leverage, and influence plots into one highly informative plot using the `influencePlot()` function from the `car` package:

```
library(car)
influencePlot(fit, id.method="identify", main="Influence Plot",
              sub="Circle size is proportional to Cook's distance")
```

The resulting plot (figure 8.16) shows that Nevada and Rhode Island are outliers; New York, California, Hawaii, and Washington have high leverage; and Nevada, Alaska, and Hawaii are influential observations.

Influential Observations



Corrective Measures

Now you may ask, “What do you do if you identify problems?” There are four approaches to dealing with violations of regression assumptions:

- Deleting observations
- Transforming variables
- Adding or deleting variables
- Using another regression approach

Selecting the "best" Regression Model

Comparing models

You can compare the fit of two nested models using the `anova()` function in the base installation. A *nested model* is one whose terms are completely included in the other model. In the `states` multiple-regression model, you found that the regression coefficients for Income and Frost were nonsignificant. You can test whether a model without these two variables predicts as well as one that includes them (see the following listing).

Listing 8.11 Comparing nested models using the `anova()` function

```
> states <- as.data.frame(state.x77[,c("Murder", "Population",
                                       "Illiteracy", "Income", "Frost")])
> fit1 <- lm(Murder ~ Population + Illiteracy + Income + Frost,
             data=states)
> fit2 <- lm(Murder ~ Population + Illiteracy, data=states)
> anova(fit2, fit1)

Analysis of Variance Table

Model 1: Murder ~ Population + Illiteracy
Model 2: Murder ~ Population + Illiteracy + Income + Frost
  Res.Df     RSS  Df      Sum of Sq      F Pr(>F)
1     47 289.246
2     45 289.167   2     0.079 0.0061     0.994
```

Here, model 1 is nested within model 2. The `anova()` function provides a simultaneous test that Income and Frost add to linear prediction above and beyond Population and Illiteracy. Because the test is nonsignificant ($p = .994$), you conclude that they don't add to the linear prediction and you're justified in dropping them from your model.

Selecting the "best" Regression Model

The Akaike Information Criterion (AIC) provides another method for comparing models. The index takes into account a model's statistical fit and the number of parameters needed to achieve this fit. Models with *smaller* AIC values—indicating adequate fit with fewer parameters—are preferred. The criterion is provided by the `AIC()` function (see the following listing).

Listing 8.12 Comparing models with the AIC

```
> fit1 <- lm(Murder ~ Population + Illiteracy + Income + Frost,
            data=states)
> fit2 <- lm(Murder ~ Population + Illiteracy, data=states)
> AIC(fit1,fit2)

      df      AIC
fit1   6 241.6429
fit2   4 237.6565
```

The AIC values suggest that the model without Income and Frost is the better model. Note that although the ANOVA approach requires nested models, the AIC approach doesn't.

Selecting the “best” Regression Model

Variable Selection

Two popular approaches to selecting a final set of predictor variables from a larger pool of candidate variables are stepwise methods and all-subsets regression.

In **stepwise selection**, variables are added to or deleted from a model one at a time, until some stopping criterion is reached.

For example, in **forward stepwise regression**, you add predictor variables to the model one at a time, stopping when the addition of variables would no longer improve the model.

In **backward stepwise regression**, you start with a model that includes all predictor variables, and then you delete them one at a time until removing variables would degrade the quality of the model.

In stepwise regression, you combine the forward and backward stepwise approaches. Variables are entered one at a time, but at each step, the variables in the model are reevaluated, and those that don’t contribute to the model are deleted. A predictor variable may be added to and deleted from, a model several times before a final solution is reached.

Selecting the "best" Regression Model

Listing 8.13 Backward stepwise selection

```
> library(MASS)
> states <- as.data.frame(state.x77[,c("Murder", "Population",
                          "Illiteracy", "Income", "Frost")])

> fit <- lm(Murder ~ Population + Illiteracy + Income + Frost,
          data=states)
> stepAIC(fit, direction="backward")

Start:  AIC=97.75
Murder ~ Population + Illiteracy + Income + Frost

             Df Sum of Sq    RSS    AIC
- Frost       1      0.02 289.19  95.75
- Income      1      0.06 289.22  95.76
<none>                    289.17  97.75
- Population  1     39.24 328.41 102.11
- Illiteracy  1    144.26 433.43 115.99

Step:  AIC=95.75
Murder ~ Population + Illiteracy + Income

             Df Sum of Sq    RSS    AIC
- Income      1      0.06 289.25  93.76
<none>                    289.19  95.75
- Population  1     43.66 332.85 100.78
- Illiteracy  1    236.20 525.38 123.61

Step:  AIC=93.76
Murder ~ Population + Illiteracy

             Df Sum of Sq    RSS    AIC
<none>                    289.25  93.76
- Population  1     48.52 337.76  99.52
- Illiteracy  1    299.65 588.89 127.31

Call:
lm(formula=Murder ~ Population + Illiteracy, data=states)

Coefficients:
(Intercept)   Population   Illiteracy
  1.6515497    0.0002242    4.0807366
```

Selecting the “best” Regression Model

ALL SUBSETS REGRESSION

In all subsets regression, every possible model is inspected. The analyst can choose to have all possible results displayed or ask for the *nbest* models of each subset size (one predictor, two predictors, and so on). For example, if *nbest*=2, the two best one-predictor models are displayed, followed by the two best two-predictor models, followed by the two best three-predictor models, up to a model with all predictors. All subsets regression is performed using the *regsubsets*() function from the leaps package.

You can choose the **R-squared, Adjusted R-squared, or Mallows Cp statistic** as your criterion for reporting “best” models.

As you’ve seen, R-squared is the amount of variance accounted for in the response variable by the predictor variables. Adjusted R-squared is similar but takes into account the number of parameters in the model. R-squared always increases with the addition of predictors. When the number of predictors is large compared to the sample size, this can lead to significant overfitting. The Adjusted R-squared is an attempt to provide a more honest estimate of the population R-squared—one that’s less likely to take advantage of chance variation in the data.

The Mallows Cp statistic is also used as a stopping rule in stepwise regression. It has been widely suggested that a good model is one in which the *Cp* statistic is close to the number of model parameters (including the intercept).

Selecting the "best" Regression Model

In listing 8.14, we'll apply all subsets regression to the state's data. The results can be plotted with either the *plot*() function in the *leaps* package or the *subsets*() function in the car package. An example of the former is provided in figure 8.17, and an example of the latter is given in figure 8.18.

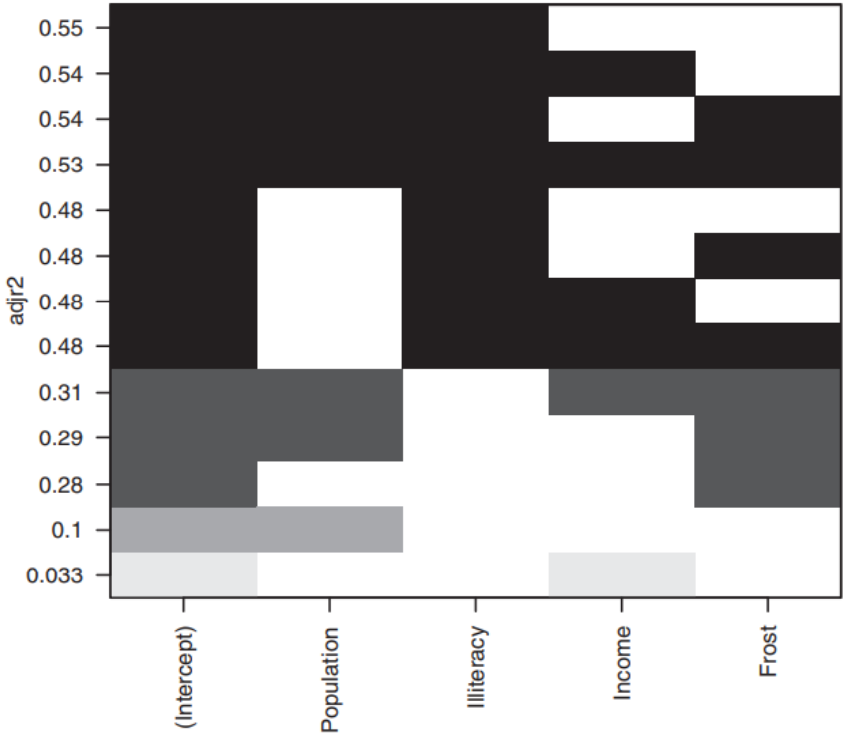


Figure 8.17 Best four models for each subset size based on Adjusted R-square

Listing 8.14 All subsets regression

```
library(leaps)
states <- as.data.frame(state.x77[,c("Murder", "Population",
          "Illiteracy", "Income", "Frost")])

leaps <-regsubsets(Murder ~ Population + Illiteracy + Income +
                   Frost, data=states, nbest=4)
plot(leaps, scale="adjr2")

library(car)
subsets(leaps, statistic="cp",
        main="Cp Plot for All Subsets Regression")
abline(1,1,lty=2,col="red")
```

Selecting the “best” Regression Model

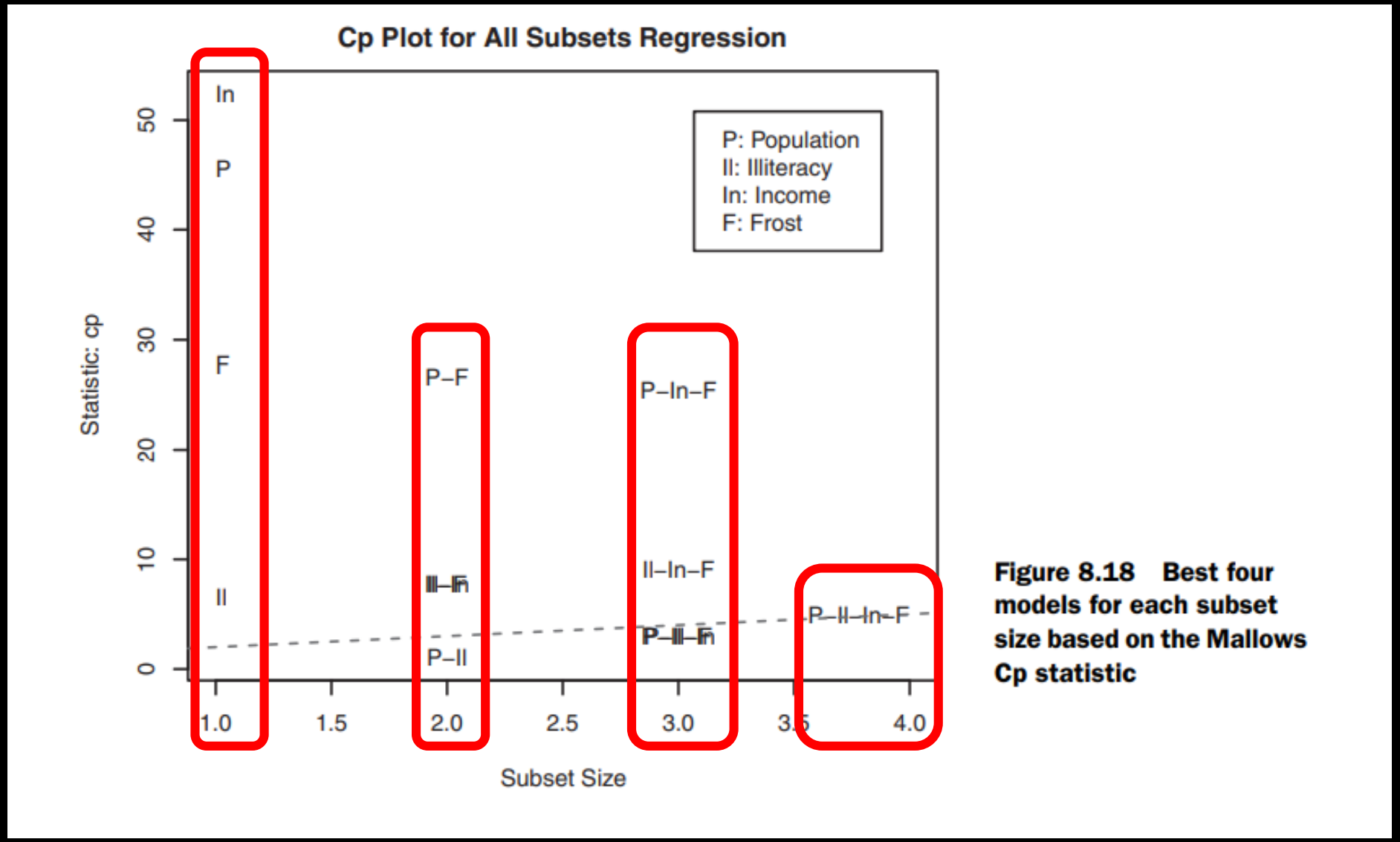


Figure 8.18 Best four models for each subset size based on the Mallows Cp statistic

Cross-Validation

In the previous section, we examined methods for selecting the variables to include in a regression equation. When description is your primary goal, the selection, and interpretation of a regression model signal the end of your labor. But when your goal is prediction, you can justifiably ask, “How well will this equation perform in the real world?

By definition, regression techniques obtain model parameters that are optimal for a given set of data. In OLS regression, the model parameters are selected to minimize the sum of squared errors of prediction (residuals) and, conversely, maximize the amount of variance accounted for in the response variable (R-squared). Because the equation has been optimized for the given set of data, it won’t perform as well with a new set of data.

In cross-validation, a portion of the data is selected as the training sample, and a portion is selected as the hold-out sample. A regression equation is developed on the training sample and then applied to the hold-out sample. Because the hold-out sample wasn’t involved in the selection of the model parameters, the performance of this sample is a more accurate estimate of the operating characteristics of the model with new data. In **k-fold cross-validation**, the sample is divided into *k* subsamples. Each of the *k* subsamples serves as a hold-out group, and the combined observations from the remaining *k – 1* subsample serve as the training group. The performance for the *k* prediction equations applied to the *k* hold-out samples is recorded and then averaged. (When *k* equals *n*, the total number of observations, this approach is called *jackknifing*.)

Cross-Validation

You can perform k-fold cross-validation using the `crossval()` function in the `bootstrap` package. The following listing provides a function (called `shrinkage()`) for cross-validating a model's R-square statistic using k-fold cross-validation.

Listing 8.15 Function for k-fold cross-validated R-square

```
shrinkage <- function(fit, k=10){
  require(bootstrap)

  theta.fit <- function(x,y){lsfit(x,y)}
  theta.predict <- function(fit,x){cbind(1,x)%*%fit$coef}

  x <- fit$model[,2:ncol(fit$model)]
  y <- fit$model[,1]

  results <- crossval(x, y, theta.fit, theta.predict, ngroup=k)
  r2 <- cor(y, fit$fitted.values)^2
  r2cv <- cor(y, results$cv.fit)^2
  cat("Original R-square =", r2, "\n")
  cat(k, "Fold Cross-Validated R-square =", r2cv, "\n")
  cat("Change =", r2-r2cv, "\n")
  }
```

Cross-Validation

Using this listing, you define your functions, create a matrix of predictor and predicted values, get the raw R-squared, and get the cross-validated R-squared. (Chapter 12 covers bootstrapping in detail.)

The `shrinkage()` function is then used to perform a 10-fold cross-validation with the `states` data, using a model with all four predictor variables:

```
> states <- as.data.frame(state.x77[,c("Murder", "Population",
          "Illiteracy", "Income", "Frost")])
> fit <- lm(Murder ~ Population + Income + Illiteracy + Frost, data=states)
> shrinkage(fit)

Original R-square=0.567
10 Fold Cross-Validated R-square=0.4481
Change=0.1188
```

You can see that the R-square based on the sample (0.567) is overly optimistic. A better estimate of the amount of variance in murder rates that this model will account for with new data is the cross-validated R-square (0.448). (Note that observations are assigned to the *k* groups randomly, so you'll get a slightly different result each time you execute the `shrinkage()` function.)

You could use cross-validation in variable selection by choosing a model that demonstrates better generalizability. For example, a model with two predictors (`Population` and `Illiteracy`) shows less R-square shrinkage (.03 versus .12) than the full model:

```
> fit2 <- lm(Murder ~ Population + Illiteracy,data=states)
> shrinkage(fit2)

Original R-square=0.5668327
10 Fold Cross-Validated R-square=0.5346871
Change=0.03214554
```

Relative Importance

Up to this point in the chapter, we've been asking, "Which variables are useful for predicting the outcome?" But often your real interest is in the question, "Which variables are most important in predicting the outcome?" You implicitly want to rank-order the predictors in terms of relative importance. There may be practical grounds for asking the second question. For example, if you could rank-order leadership practices by their relative importance for organizational success, you could help managers focus on the behaviors they most need to develop.

If predictor variables were uncorrelated, this would be a simple task. You would rank-order the predictor variables by their correlation with the response variable. In most cases, though, the predictors are correlated with each other, and this complicates the task significantly.

Relative Importance

There have been many attempts to develop a means for assessing the relative importance of predictors. The simplest has been to compare standardized regression coefficients. Standardized regression coefficients describe the expected change in the response variable (expressed in standard deviation units) for a standard deviation change in a predictor variable, holding the other predictor variables constant. You can obtain the standardized regression coefficients in R by standardizing each of the variables in your dataset to a mean of 0 and standard deviation of 1 using the `scale()` function, before submitting the dataset to a regression analysis. (Note that because the `scale()` function returns a matrix and the `lm()` function requires a data frame, you convert between the two in an intermediate step.) The code and results for the multiple regression problem are shown here:

```
> states <- as.data.frame(state.x77[,c("Murder", "Population",
                          "Illiteracy", "Income", "Frost")])
> zstates <- as.data.frame(scale(states))
> zfit <- lm(Murder~Population + Income + Illiteracy + Frost, data=zstates)
> coef(zfit)
```

```
(Intercept)  Population      Income  Illiteracy       Frost
 -9.406e-17   2.705e-01   1.072e-02   6.840e-01   8.185e-03
```

Relative Importance

A new method called **relative weights** shows significant promise. The method closely approximates the average increase in *R-square* obtained by adding a predictor variable across all possible sub models. A function for generating relative weights is provided in the next listing

In listing 8.17, the *relweights*() function is applied to the *state's* data with the murder rate predicted by the *population*, *illiteracy*, *income*, and *temperature*.
You can see from figure 8.19 that the total amount of variance accounted for by the model (*R-square=0.567*) has been divided among the predictor variables. *Illiteracy* accounts for 59% of the *R-square*, *Frost* accounts for 20.79%, and so forth. Based on the method of relative weights, *Illiteracy* has the greatest relative importance, followed by *Frost*, *Population*, and *Income*, in that order.

Relative Importance

Listing 8.16 `relweights()` for calculating relative importance of predictors

```
relweights <- function(fit,...){
  R <- cor(fit$model)
  nvar <- ncol(R)
  rxx <- R[2:nvar, 2:nvar]
  rxy <- R[2:nvar, 1]
  svd <- eigen(rxx)
  evec <- svd$vectors
  ev <- svd$values
  delta <- diag(sqrt(ev))
  lambda <- evec %*% delta %*% t(evec)


   lambdasq <- lambda ^ 2
   beta <- solve(lambda) %*% rxy
   rsquare <- colSums(beta ^ 2)
   rawwgt <- lambdasq %*% beta ^ 2
   import <- (rawwgt / rsquare) * 100
   import <- as.data.frame(import)
   row.names(import) <- names(fit$model[2:nvar])
   names(import) <- "Weights"
   import <- import[order(import),1, drop=FALSE]
   dotchart(import$Weights, labels=row.names(import),
      xlab="% of R-Square", pch=19,
      main="Relative Importance of Predictor Variables",
      sub=paste("Total R-Square=", round(rsquare, digits=3)),
      ...)
 return(import)
 }
```

Relative Importance

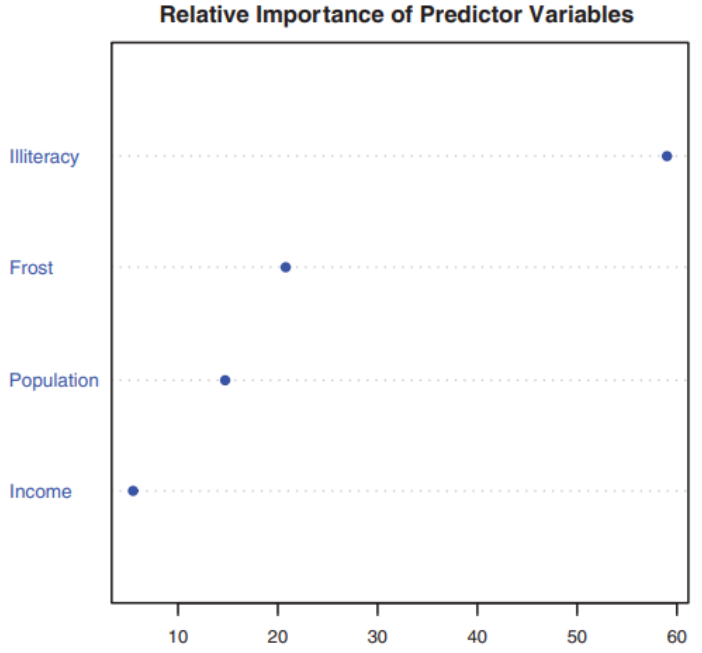


Figure 8.19 Dot chart of relative weights for the states multiple regression problem. Larger weights indicate relatively more important predictors. For example, Illiteracy accounts for 59% of the total explained variance (0.567), whereas Income only accounts for 5.49%. Thus Illiteracy has greater relative importance than Income in this model.

Listing 8.17 Applying the `relweights()` function

```
> states <- as.data.frame(state.x77[,c("Murder", "Population",
          "Illiteracy", "Income", "Frost")])
> fit <- lm(Murder ~ Population + Illiteracy + Income + Frost, data=states)
> relweights(fit, col="blue")
           Weights
Income        5.49
Population   14.72
Frost        20.79
Illiteracy   59.00
```

References

- **R in Action, R. Kabacoff, 2nd edition, Manning, ISBN 978-1-617-29138-8, Chapter 8.**