Answers to Odd-Numbered Exercises for Fox, *Applied Regression Analysis and Generalized Linear Models, Third Edition* (Sage, 2016)

John Fox

Answers last modified: 2021–06–20

This document provides worked-out answers to the odd-numbered exercises in the text (excluding the data-analysis exercises on the website for the text). The answers to both odd- and even-numbered questions, are available from Sage Publications in a separate document for instructors using the text in a college or university class.

About the starred—more difficult–exercises: "More difficult" is interpreted in context, and so there is a higher bar for difficulty in starred sections and chapters. Many of the starred exercises are derivations that I considered too tedious to include in the text, and some of them are at a considerably higher level of difficulty than I assumed for even the starred parts of the text.

In many instances, I simply elided from the text intermediate steps in what would have been lengthy but relatively unenlightening proofs, and in some cases, I didn't bother to complete these proofs but thought that I understood how to do so. I'm embarrassed to say that occasionally proved not to be the case, and the project of working the exercises revealed weaknesses and even errors in some of the questions, and more rarely in the text itself (as enumerated in the errata for the text). A few of the starred exercises are at a higher level of difficulty than I intended, and in answering these I've tried to provide solutions that are as simple as possible.

The answers are nearly, but not quite complete: I've yet to devise suitable answers to Exercise 20.5 and part of Exercise 20.11, and will update this document if and when I do so. Missing answers are marked, "To be completed."

Although the text is written to be software-neutral, some of the exercises require the use of statistical software. I employed the R statistical computing environment (R Core Team, 2021) for these exercises, but you should feel free to substitute other appropriate statistical software for R. See Fox and Weisberg (2019) for an introduction to R in the context of regression analysis that largely overlaps with the text.

Exercises for Chapter 1

Exercise 1.1

(a) There isn't a strong basis for making a causal claim here, because students self-select whether or not to complete homework assignments. It seems reasonable to suppose that the more diligent students are likely to complete more homework assignments, and that, by, e.g., studying more, they likely would have done better in the course even in the absence of homework assignments. That is, this is an observational study in which it's likely that there are important omitted common prior causes of homework completion and grades.

(b) Yes, it should be possible to design a randomized comparative experiment to study the effect of homework completion on grades, but as a practical matter it wouldn't be easy to do so. It would be difficult to randomize homework-assignment policies *within* an individual class and so it would probably be necessary to assign a sufficiently large number of different classes at random to one of two or more conditions with varying homework policies. For example, one condition might dispense with assigned homework while another might require it.

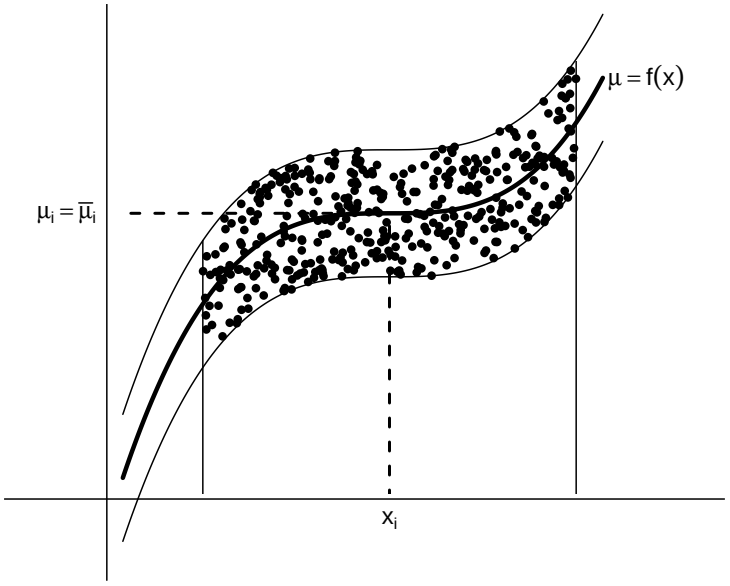
(c) Yes, it should be possible to make a more convincing case on the basis of observational data by trying to control statistically for known potential common prior causes of both homework completion and grades in the course. We could, for example, control statistically for students' prior grade-point averages.

Exercises for Chapter 2

Exercise 2.1*

(a) If X is evenly (or even just symmetrically) distributed in the interval around x_i, and if $E(Y|x)$ is a linear function of X in the interval, then the average value of $E(Y|x)$ in the interval $\bar{\mu}_i$ is the same as the value $\mu_i = E(Y|x_i)$ at the center of the interval.

(b) If X is evenly distributed in the interval, then it's not generally the case that $\bar{\mu}_i = \mu_i$, but it *is* possible, depending on the shape of the regression function in the interval. Consider the following graph, for example, where the regression function in the interval satisfies the equation $f(x_i) - f(x_i - \delta) = f(x_i + \delta) - f(x_i)$, thus "balancing" departures from $f(x_i)$ to the left and right of x_i:



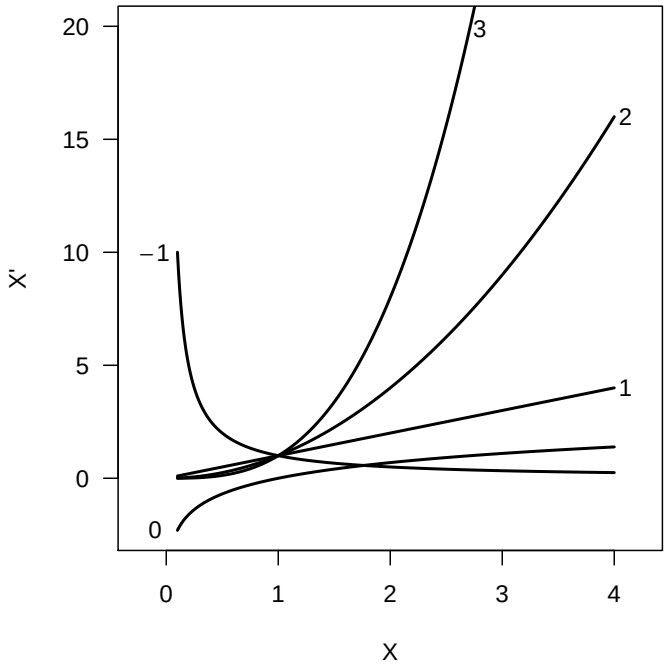
(c) This really isn't different in principle from (b): If the relationship between $E(Y|x)$ and X is nonlinear in the interval around x_i, it could happen by chance that $\bar{\mu}_i = \mu_i$ if the uneven distribution of X values in the interval offsets the nonlinear relationship when the conditional values $E(Y|x)$ in the interval are averaged with weights proportional to their frequency—for example, if the nonlinear relationship is as in the figure above, and if the distribution of X in the interval is uneven but symmetric about x_i.

If, however, the relationship is *linear* in the interval, and if the distribution of X in the interval is both uneven and asymmetric, then $\bar{\mu}_i$ will be pulled away from μ_i.

Exercises for Chapter 4

Exercise 4.1

Here's a graph similar to Figure 4.1 but for the ordinary power transformations:



So that the scale of the transformed scores isn't too compressed, I stopped the vertical axis at 20, which excludes the largest values of $X' = X^3$.

The general effect of the raw power transformations is similar to the corresponding Box-Cox transformations: That is, transformations down the ladder of powers ($\log_e X, X^{-1}$)increasingly spread out the small values of X relative to the large values, while those up the ladder (X^2, X^3) spread out the large values relative to the small ones. In addition, however, the raw inverse transformation $X' = X^{-1}$ is monotone *decreasing* and so reverses the order of the X values. The graph does a much poorer job of revealing the essential properties and unity of the power-transformation family than does Figure 4.1 for the Box-Cox family of transformations.

Exercise 4.3*

(a) I didn't develop the "MLE" for the one-parameter Box-Cox family in any detail in the text (with "MLE" in quotes because, as I explained, there isn't a likelihood in the strict sense of the term), so I'll start with that: Here, the transformation is $Y \equiv X^{(\lambda)} \equiv (X^\lambda - 1)/\lambda$. For simplicity, I'm ignoring the possibility that λ is exactly 0; recall that in this case, $Y \equiv \log_e X$.

At the true value of λ, the transformed variable Y is normally distributed with (say) mean μ and variance σ^2. So

$$
\begin{aligned}
p(y) &= \frac{1}{\sigma\sqrt{2\pi}} \exp\left[-\frac{(y-\mu)^2}{2\sigma^2}\right] \\
&= \frac{1}{\sigma\sqrt{2\pi}} \exp\left[-\frac{\left(\frac{x^\lambda - 1}{\lambda} - \mu\right)^2}{2\sigma^2}\right]
\end{aligned}
$$

A perhaps subtle point is that we can't work with $p(y)$ directly because its scale changes with λ and so we want to find $p(x)$, which depends on the Jacobian of the transformation from X to Y (see on-line Appendix D, Section D.1.3, on transformations of random variables); adapting the result in the preceding problem,

$$\frac{dY}{dX} = X^{\lambda-1}$$

and so

$$\begin{aligned} p(x) &= x^{\lambda-1}p(y) \\ &= x^{\lambda-1}\frac{1}{\sigma\sqrt{2\pi}}\exp\left[-\frac{\left(\frac{x^\lambda-1}{\lambda}-\mu\right)^2}{2\sigma^2}\right] \end{aligned}$$

For a fixed value of λ, the MLEs of μ and σ^2 are

$$\begin{aligned} \widehat{\mu} &= \frac{\sum \frac{x_i^\lambda-1}{\lambda}}{n} \\ \widehat{\sigma}^2 &= \frac{\sum\left(\frac{x_i^\lambda-1}{\lambda}-\widehat{\mu}\right)^2}{n} \end{aligned}$$

The log-likelihood as a function of λ is then

$$\begin{aligned} \log_e L(\lambda) &= (\lambda-1)\sum \log_e X_i - \frac{n}{2}\log_e \widehat{\sigma}^2 - \frac{n}{2}\log_e 2\pi - \frac{\sum\left(\frac{X_i^\lambda-1}{\lambda}-\widehat{\mu}\right)^2}{2\widehat{\sigma}^2} \\ &= (\lambda-1)\sum \log_e X_i - \frac{n}{2}\left(\log_e \widehat{\sigma}^2 + \log_e 2\pi + 1\right) \end{aligned}$$

The "MLE" for the two-parameter Box-Cox transformation $Y \equiv X^{(\lambda,\alpha)} \equiv (X^\lambda - \alpha)/\lambda$ is only slightly more complicated.[1] In this case, Y is normally distributed at the true values of λ and α,

$$p(y) = \frac{1}{\sigma\sqrt{2\pi}}\exp\left\{-\frac{\left[\frac{(x-\alpha)^\lambda}{\lambda}-\mu\right]^2}{2\sigma^2}\right\}$$

The Jacobian of the transformation from X to Y is

$$\frac{dY}{dX} = (X-\alpha)^{\lambda-1}$$

The probability-density at $X = x$ is then

$$p(x) = (x-\alpha)^{\lambda-1}\frac{1}{\sigma\sqrt{2\pi}}\exp\left\{-\frac{\left[\frac{(x-\alpha)^\lambda}{\lambda}-\mu\right]^2}{2\sigma^2}\right\}$$

and the log-likelihood as a function of λ and α is

$$\log_e L(\lambda,\alpha) = (\lambda-1)\sum \log_e(X_i-\alpha) - \frac{n}{2}\left(\log_e \widehat{\sigma}^2 + \log_e 2\pi + 1\right)$$

where, now

$$\begin{aligned} \widehat{\mu} &= \frac{\sum \frac{(x_i-\alpha)^\lambda}{\lambda}}{n} \\ \widehat{\sigma}^2 &= \frac{\sum\left[\frac{(x_i-\alpha)^\lambda}{\lambda}-\widehat{\mu}\right]^2}{n}. \end{aligned}$$

[1] It would possibly have been less confusing if I had *added* α to X rather than subtracted it, but because α can either be negative or positive (or 0), the two definitions of $X^{(\lambda,\alpha)}$ are equivalent.

(b) Programming MLE for the two-parameter Box-Cox transformation in a numerically stable manner isn't straightforward. I experimented both with programs that I wrote myself and with the `boxcoxfit()` function in the **geoR** package (Ribeiro Jr et al., 2020) for the R statistical computing environment.

I was unable to get a satisfactory solution to transforming interlocks in the Ornstein interlocking-directorate data, in that the negative Hessian matrix of second-order partial derivatives of the log-likelihood at the putative MLE wasn't positive-definite, making it impossible to compute the covariance matrix of $\widehat{\lambda}$ and $\widehat{\alpha}$. In contrast, I had no trouble estimating λ for the *one-parameter* Box-Cox transformation of interlocks, using an arbitrary fixed start of 1, added to the data, obtaining $\widehat{\lambda} = 0.125$ with standard error $\text{SE}(\widehat{\lambda}) = 0.053$.

I also tried a variety of artificial problems without much luck, for example, randomly generating $n = 1000$ observations with $\lambda = 3$ and $\alpha = -2$, so that when the resulting X values are properly transformed, $X^{(\lambda,\alpha)}$ would be normally distributed with $\mu = 100$ and $\sigma^2 = 15^2$. I obtained the estimates $\widehat{\lambda} = 4.78$ and $\widehat{\alpha} = -11.8$, with very large standard errors $\text{SE}(\widehat{\lambda}) = 17.51$ and $\text{SE}(\widehat{\alpha}) = 75.9$, and sampling correlation $r(\widehat{\lambda}, \widehat{\alpha}) = -.997$!

Exercises for Chapter 5

Exercise 5.1*

(a) We already know that $\sum E_i = 0$ and $\sum X_i E_i = 0$. Thus,

$$\sum \widehat{Y}_i E_i = \sum (A + BX_i)E_i = A\sum E_i + B\sum X_i E_i = 0 + 0 = 0$$

(b) This follows almost immediately from the preceding result:

$$\sum (Y_i - \widehat{Y}_i)(\widehat{Y}_i - \bar{Y}) = \sum E_i(\widehat{Y}_i - \bar{Y}) = \sum E_i \widehat{Y}_i - \bar{Y}\sum E_i = 0 + 0 = 0$$

Remark: Results like these are obvious from the vector geometry of linear least-squares regression, which is developed in Chapter 10.

Exercise 5.3*

This is just a simpler version of the derivation of the least-squares coefficients in simple regression (on page 85). We have $S(A') = \sum (Y_i - A')^2$ and $dS(A')/dA' = -1 \times 2\sum (Y_i - A')$. Setting the derivative to zero and solving for A' produces

$$\begin{aligned} -2\sum (Y_i - A') &= 0 \\ \sum Y_i - nA' &= 0 \\ A' &= \frac{\sum Y}{n} = \bar{Y} \end{aligned}$$

Exercise 5.5*

This is just a generalization of the derivation of the least-squares coefficients in simple regression. The sum of squares function in multiple regression is

$$S \equiv S(A, B_1, B_2, \ldots, B_k) = \sum [Y_i - (A + B_1 X_{i1} + B_2 X_{i2} + \cdots + B_k X_{ik})]^2$$

The partial derivatives of S with respect to the regression coefficients are

$$\begin{aligned} \frac{\partial S}{\partial A} &= \sum \{1 \times 2 \times [Y_i - (A + B_1 X_{i1} + B_2 X_{i2} + \cdots + B_k X_{ik})]\} \\ \frac{\partial S}{\partial B_1} &= \sum \{-X_{i1} \times 2 \times [Y_i - (A + B_1 X_{i1} + B_2 X_{i2} + \cdots + B_k X_{ik})]\} \\ \frac{\partial S}{\partial B_2} &= \sum \{-X_{i2} \times 2 \times [Y_i - (A + B_1 X_{i1} + B_2 X_{i2} + \cdots + B_k X_{ik})]\} \\ &\vdots \\ \frac{\partial S}{\partial B_k} &= \sum \{-X_{ik} \times 2 \times [Y_i - (A + B_1 X_{i1} + B_2 X_{i2} + \cdots + B_k X_{ik})]\} \end{aligned}$$

Setting the partial derivatives to 0, dividing each equation on the left-hand side by 2, multiplying through by the Xs, summing the terms in each equation separately, bringing the coefficients outside of the sums, and isolating the term for Y in each equation on the right-hand side produces the normal equations as shown in Equations 5.7.

Exercise 5.7

(a) I used R to perform the necessary computations:

```
> library("carData") # for Prestige data set

> summary(lm(prestige ~ income + education + women, data=Prestige))

Call:
lm(formula = prestige ~ income + education + women, data = Prestige)

Residuals:
     Min       1Q   Median       3Q      Max
-19.8246  -5.3332  -0.1364   5.1587  17.5045

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept) -6.7943342  3.2390886  -2.098   0.0385 *
income       0.0013136  0.0002778   4.729 7.58e-06 ***
education    4.1866373  0.3887013  10.771  < 2e-16 ***
women       -0.0089052  0.0304071  -0.293   0.7702
---
Signif. codes:  0 '***'0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 7.846 on 98 degrees of freedom
Multiple R-squared:  0.7982,    Adjusted R-squared:  0.792
F-statistic: 129.2 on 3 and 98 DF,  p-value: < 2.2e-16

> my <- lm(prestige ~ income + women, data=Prestige)
> mx <- lm(education ~ income + women, data=Prestige)
> ey <- residuals(my)
> ex <- residuals(mx)
> round(coef(lm(ey ~ ex)), 6)
(Intercept)          ex
   0.000000    4.186637
```

(b) Both sets of residuals have means of 0 and so the regression necessarily goes through the origin.

(c) Yes, that's a reasonable way to think about the two sets of residuals—at least the *linear* dependence of Y and X_1 on the other Xs is removed.

(d) The procedure has the effect of reducing a multiple regression—that is, a $k + 1$-dimensional problem—to a sequence of simple regressions—that is, a sequence of 2-dimensional problems. As the exercise suggests, that leads to added-variable plots, discussed in Section 11.6.1.

Exercises for Chapter 6

Exercise 6.1*

(a) From the text, $B = \sum m_i Y_i$, where

$$m_i = \frac{x_i - \bar{x}}{\sum_{j=1}^{n}(x_j - \bar{x})^2}$$

Then

$$\begin{aligned} E(B) &= \sum m_i E(Y_i) \\ &= \sum_{i=1}^{n} \frac{x_i - \bar{x}}{\sum_{j=1}^{n}(x_j - \bar{x})^2}(\alpha + \beta x_i) \\ &= \frac{\alpha}{\sum(x_j - \bar{x})^2}\sum(x_i - \bar{x}) + \frac{\beta}{\sum(x_j - \bar{x})^2}\sum(x_i - \bar{x})x_i \end{aligned}$$

The first term in the last line is 0 because $\sum(x_j - \bar{x}) = 0$. To show that the second term is β, and hence that $E(B) = \beta$, I need to prove that $\sum(x_i - \bar{x})x_i = \sum(x_j - \bar{x})^2$, which isn't difficult:

$$\begin{aligned} \sum(x_j - \bar{x})^2 &= \sum(x_j - \bar{x})(x_j - \bar{x}) \\ &= \sum(x_j - \bar{x})x_j - \bar{x}\sum(x_j - \bar{x}) \\ &= \sum(x_j - \bar{x})x_j - \bar{x} \times 0 \\ &= \sum(x_j - \bar{x})x_j \end{aligned}$$

I hope that it's clear that it doesn't matter if we sum over the subscript i or j.

(b) It's straightforward to show the A is a linear function of the Ys:

$$\begin{aligned} A &= \bar{Y} - B\bar{x} \\ &= \frac{1}{n}\sum Y_i - \frac{\sum(x_i - \bar{x})(Y_i - \bar{Y})}{\sum(x_i - \bar{x})^2}\bar{x} \\ &= \frac{1}{n}\sum Y_i - \frac{\sum(x_i - \bar{x})Y_i}{\sum(x_i - \bar{x})^2}\bar{x} \\ &= \sum_i \left[\frac{1}{n} - \frac{\bar{x}(x_i - \bar{x})}{\sum_j(x_j - \bar{x})^2}\right] Y_i \end{aligned}$$

Demonstrating that $E(A) = \alpha$ is then also straightforward, if tedious. Using $E(Y_i) = \alpha + \beta x_i$,

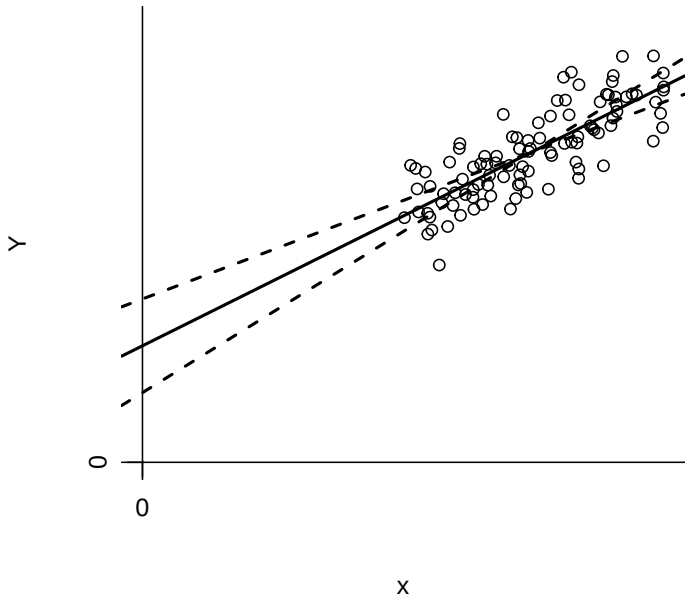
$$\begin{aligned} E(A) &= \sum_i \left[\frac{1}{n} - \frac{\bar{x}(x_i - \bar{x})}{\sum_j(x_j - \bar{x})^2}\right](\alpha + \beta x_i) \\ &= \sum \frac{\alpha}{n} + \sum \beta\frac{x_i}{n} - \alpha\frac{\bar{x}\sum(x_i - \bar{x})}{\sum(x_j - \bar{x})^2} - \beta\frac{\bar{x}\sum(x_i - \bar{x})x_i}{\sum(x_j - \bar{x})^2} \\ &= \alpha - \beta\bar{x} - \alpha\bar{x} \times 0 - \beta\bar{x} \times 1 \\ &= \alpha \end{aligned}$$

because in the second line of this equation, $\sum(x_i - \bar{x}) = 0$ in the numerator of the third term, and $\sum(x_i - \bar{x})x_i = \sum(x_j - \bar{x})^2$ in the numerator and denominator of the fourth term.

Comment: It's perhaps surprisingly much easier to establish the unbias of the least-squares estimators, and other properties of the least-squares estimators, for the more general case of multiple regression using the matrix representation of the linear regression model (see Section 9.3).

Exercise 6.3

When the mean of the xs is far from 0, the sum of squares in the numerator $\sum x_i^2$ is much bigger than the sum of squares in the denominator $\sum(x_i - \bar{x})^2$. That makes intuitive sense because when the x-values are far from 0 the intercept can vary a great deal for a small change in the slope of the regression line, as illustrated in the following graph:



The solid line in the graph is the least-squares line; the broken lines also go through the means of the two variables but have slopes approximately 25% smaller and 25% larger than the least-squares slope.

Exercise 6.5*

The log-likelihood is

$$\log_e L(\alpha, \beta, \sigma_\varepsilon^2) = -\frac{n}{2}\log_e(2\pi\sigma_\varepsilon^2) - \frac{1}{2\sigma_\varepsilon^2}\sum(y_i - \alpha - \beta x_i)^2$$

The partial derivatives of the log-likelihood with respect to the parameters are

$$\begin{aligned}
\frac{\partial \log_e L}{\partial \alpha} &= \frac{1}{\sigma_\varepsilon^2}\sum(y_i - \alpha - \beta x_i) \\
\frac{\partial \log_e L}{\partial \beta} &= \frac{1}{\sigma_\varepsilon^2}\sum x_i(y_i - \alpha - \beta x_i) \\
\frac{\partial \log_e L}{\partial \sigma_\varepsilon^2} &= -\frac{n}{2\sigma_\varepsilon^2} + \frac{\sum(y_i - \alpha - \beta x_i)^2}{2(\sigma_\varepsilon^2)^2}
\end{aligned}$$

Setting the partial derivatives to 0, it's apparent that the first two equations are zero when the sums are zero (i.e., multiply both of these equations by σ_ε^2). The two resulting equations (substituting estimates for the parameters and rearranging),

$$\begin{aligned}
n\widehat{\alpha} + \widehat{\beta}\sum x_i &= \sum y_i \\
\widehat{\alpha}\sum x_i + \widehat{\beta}\sum x_i^2 &= \sum x_i y_i
\end{aligned}$$

are just the least-squares normal equations for A and B in simple regression. Then $\sum(y_i - \widehat{\alpha} - \widehat{\beta}x_i)^2$ in the third equation is the sum of squared least-squares residuals, $\sum E_i^2$, and so

$$\frac{n}{2\widehat{\sigma}_\varepsilon^2} = \frac{\sum E_i^2}{2(\widehat{\sigma}_\varepsilon^2)^2}$$

Multiplying both sides of this equation by $2\widehat{\sigma}_\varepsilon^2$ and solving for $\widehat{\sigma}_\varepsilon^2$ produces

$$\widehat{\sigma}_\varepsilon^2 = \frac{\sum E_i^2}{n}$$

Exercise 6.7

The "hints" effectively answer the question: We compare the regression of Y on x_1 and x_2 with the regression of Y on the sum of x_1 and x_2. The units of x_1 and x_2 must be the same for it to makes sense to compare their coefficients directly.

Using, R for Duncan's regression:

```
> library("car") # for data and compareCoefs()
Loading required package: carData

> m1 <- lm(prestige ~ income + education, data=Duncan)
> m2 <- lm(prestige ~ I(income + education), data=Duncan)

> anova(m2, m1)
Analysis of Variance Table

Model 1: prestige ~ I(income + education)
Model 2: prestige ~ income + education
  Res.Df    RSS Df Sum of Sq      F Pr(>F)
1     43 7518.9
2     42 7506.7  1    12.195 0.0682 0.7952

> compareCoefs(m1, m2)
Calls:
1: lm(formula = prestige ~ income + education, data = Duncan)
2: lm(formula = prestige ~ I(income + education), data = Duncan)

                      Model 1 Model 2
(Intercept)             -6.06   -6.06
SE                       4.27    4.23

income                  0.599
SE                      0.120

education              0.5458
SE                     0.0983

I(income + education)          0.5693
SE                             0.0396
```

Thus the hypothesis that $\beta_1 = \beta_2$ is consistent with the data. The test is arguably sensible, because both income and education are percentages, as long as we're willing to equate a one percent increment in relatively high-income earners with a one-percent increase in high-school graduates (which perhaps doesn't survive close scrutiny).

Exercise 6.9

(a) The model isn't wrong: β_2 is just 0, and B_1 is still an unbiased estimator of β_1.

(b) The second term is the same as the variance of B_1 in simple regression, while the first term inflates the variance of B_1 in the multiple regression as long as $r_{12} \neq 0$, with the inflationary effect growing as r_{12}^2 gets larger. The cost of including an irrelevant explanatory variable X_2 is therefore to decrease the precision of estimation of β_1. The cost of failing to include a relevant (causally prior) explanatory variable is to bias the estimate of β_1. Which cost is more serious depends on their magnitude—the mean-squared error of estimation is the sum of squared bias and variance—but if we want to make a causal claim, we generally are more concerned with bias.

Exercise 6.11*

Multiplying $Y = \beta_1' X_1 + \beta_2' X_2 + \varepsilon$ by X_1 and X_2 and taking expectations produces

$$\sigma_{Y1} = \beta_1' \sigma_1^2 + \beta_2' \sigma_{12}$$
$$\sigma_{Y1} = \beta_1' \sigma_{12} + \beta_2' \sigma_2^2$$

where I put primes on the βs because I ignored the measurement error δ. Then solving for the β's:

$$\beta_1' = \frac{\sigma_{Y1}\sigma_2^2 - \sigma_{12}\sigma_{Y2}}{\sigma_1^2\sigma_2^2 - \sigma_{12}^2}$$
$$\beta_2' = \frac{\sigma_{Y2}\sigma_1^2 - \sigma_{12}\sigma_{Y1}}{\sigma_1^2\sigma_2^2 - \sigma_{12}^2}$$

which are the population analogs of the least-squares regression coefficients.

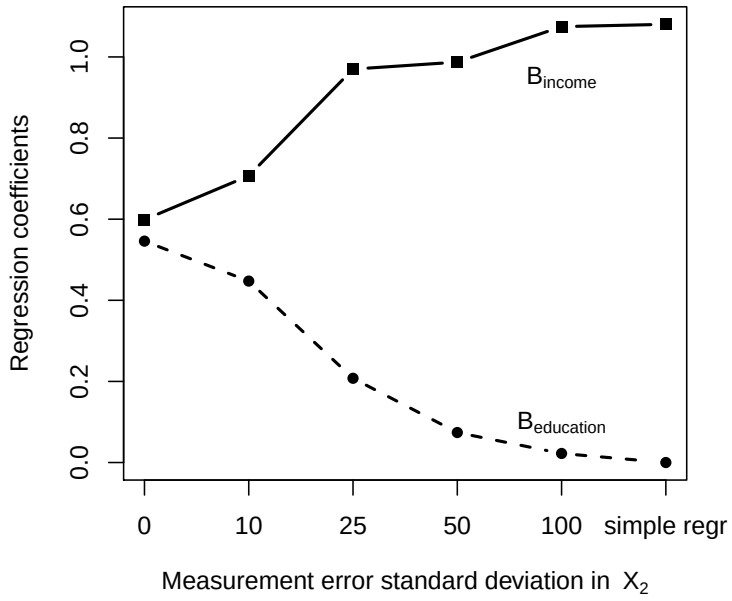
Exercise 6.13

I used R for the necessary computations:

```
> library("car") # for data and compareCoefs()
Loading required package: carData

> m1 <- lm(prestige ~ income + education, data=Duncan)
> m2 <- lm(prestige ~ income, data=Duncan)
> compareCoefs(m1, m2)
Calls:
1: lm(formula = prestige ~ income + education, data = Duncan)
2: lm(formula = prestige ~ income, data = Duncan)

            Model 1 Model 2
(Intercept)   -6.06    2.46
SE             4.27    5.19

income         0.599   1.080
SE             0.120   0.107

education     0.5458
SE            0.0983

> sigma.delta <- c(10, 25, 50, 100)
> set.seed(4875938) # for reproducibility
> b <- matrix(0, 4, 2)
> for (i in 1:4){
+     Duncan$educ <- Duncan$education + rnorm(45, 0, sigma.delta[i])
+     m <- lm(prestige ~ income + educ, data=Duncan)
+     b[i, ] <- coef(m)[2:3]
+ }
> b <- rbind(coef(m1)[2:3], b, c(coef(m2)[2], 0))
> rownames(b) <- c(0, sigma.delta, "simple regr")
> b
               income  education
```



```
0           0.5987328 0.54583391
10          0.7069395 0.44736446
25          0.9700231 0.20770187
50          0.9872692 0.07398901
100         1.0747356 0.02230014
simple regr 1.0803897 0.00000000

> plot(c(1, 6), range(b), type="n", axes=FALSE, frame=TRUE,
+      xlab=expression("Measurement error standard deviation in"~~X[2]),
+      ylab="Regression coefficients")
> axis(1, at=1:6, labels=rownames(b))
> axis(2)
> lines(b[, 1], type="b", pch=15, lwd=2)
> lines(b[, 2], type="b", pch=16, lwd=2, lty=2)
> text(c(5, 5), c(0.95, 0.1), expression(B["income"], B["education"]))
```



As the measurement error in education grows, the income coefficient is driven towards the slope coefficient in the simple regression of prestige on income alone, while the education coefficient is driven towards 0.

Exercises for Chapter 7

Exercise 7.1

If we use the coding $D_i = -1$ for women and 1 for men, the two regression lines are:

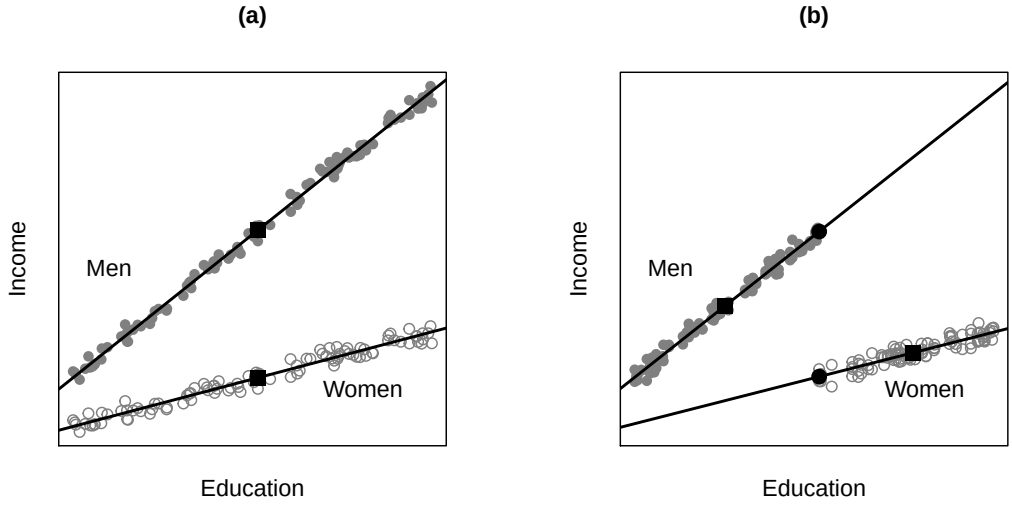
$$\begin{aligned} \text{women}: \quad Y_i &= \alpha + \beta X_i + \gamma \times -1 + \varepsilon_i \\ &= (\alpha - \gamma) + \beta X_i + \varepsilon_i \\ \text{men}: \quad Y_i &= \alpha + \beta X_i + \gamma \times 1 + \varepsilon_i \\ &= (\alpha + \gamma) + \beta X_i + \varepsilon_i \end{aligned}$$

The parameter γ is therefore *half* the difference in intercepts between women and men, and, because the regression lines are parallel, half the difference in average income between women and men of the same education. Yes, this coding captures the gender effect on income, holding education constant. And yes, any coding of a dummy regressor that assigns two distinct values to women and men will fit the same two regression lines, albeit with different parametrizations. We should, however, prefer a coding that leads to easily interpreted coefficients, which is certainly the case for a 0/1 dummy regressor.

Exercise 7.3

Yes, we can compute adjusted means for a model that includes interactions, finding the fitted level of the response within each group holding Xs constant at their overall means. The resulting adjusted means therefore have a simple interpretation as the average value of the response in each group when the Xs are fixed at their means, but are much less compelling than in an additive model, because the differences between or among the groups varies with the values at the Xs are fixed.

Here's a graph derived from Figure 7.7 showing unadjusted and adjusted means:



The graph is similar to that constructed for Exercise 7.1. In panel (a), where men and women have the same average level of education, the adjusted (circles) and unadjusted (squares) means coincide (with the squares obscuring the circles); in panel (b), where women on average have higher education than men and a smaller education slope, the adjusted means differ much more than the unadjusted means.

Exercises for Chapter 8

Exercise 8.1*

We can by simple rearrangement write the square of t_0 as

$$t_0^2 = \frac{\dfrac{(\bar{Y}_1 - \bar{Y}_2)^2}{\frac{1}{n_1} + \frac{1}{n_2}}}{\dfrac{\sum(Y_{i1} - \bar{Y}_1)^2 + \sum(Y_{i2} - \bar{Y}_2)^2}{n_1 + n_2 - 2}}$$

Because $\widehat{Y}_{ij} = \bar{Y}_j$ for $j = 1, 2$, and because the residual degrees of freedom are $n_1 + n_2 - 2$, the denominator of the equation for t_0^2 is the residual mean square.

There is $2 - 1 = 1$ degree of freedom for the numerator of F_0, and so we must show that the numerator of t_0^2 is the regression sum of squares. The numerator t_0^2 can be rewritten as

$$\begin{aligned}\frac{(\bar{Y}_1 - \bar{Y}_2)^2}{\frac{1}{n_1} + \frac{1}{n_2}} &= \frac{(\bar{Y}_1 - \bar{Y}_2)^2}{\frac{n_1 + n_2}{n_1 n_2}} \\ &= \frac{n_1 n_2 (\bar{Y}_1 - \bar{Y}_2)^2}{n_1 + n_2}\end{aligned}$$

Now, the regression sum of squares written directly is

$$\text{RegSS} = n_1(\bar{Y}_1 - \bar{Y})^2 + n_2(\bar{Y}_2 - \bar{Y})^2$$

where $\bar{Y} = (n_1\bar{Y}_1 + n_2\bar{Y}_2)/(n_1 + n_2)$ is the overall mean of Y. Expanding the first term,

$$\begin{aligned}n_1(\bar{Y}_1 - \bar{Y})^2 &= n_1\left(\bar{Y}_1 - \frac{n_1\bar{Y}_1 + n_2\bar{Y}_2}{n_1 + n_2}\right)^2 \\ &= n_1\left[\frac{(n_1 + n_2)\bar{Y}_1 - n_1\bar{Y}_1 - n_2\bar{Y}_2}{n_1 + n_2}\right]^2 \\ &= n_1\left[\frac{n_2(\bar{Y}_1 - \bar{Y}_2)}{n_1 + n_2}\right]^2\end{aligned}$$

The second term is similarly

$$n_2(\bar{Y}_2 - \bar{Y})^2 = n_2\left[\frac{n_1(\bar{Y}_2 - \bar{Y}_1)}{n_1 + n_2}\right]^2$$

Adding the two terms, and noting that $(\bar{Y}_1 - \bar{Y}_2)^2 = (\bar{Y}_2 - \bar{Y}_1)^2$, we have

$$\begin{aligned}\text{RegSS} &= \frac{n_1 n_2^2(\bar{Y}_1 - \bar{Y}_2)^2 + n_2 n_1^2(\bar{Y}_1 - \bar{Y}_2)^2}{(n_1 + n_2)^2} \\ &= \frac{n_1 n_2(n_1 + n_2)(\bar{Y}_1 - \bar{Y}_2)^2}{(n_1 + n_2)^2} \\ &= \frac{n_1 n_2(\bar{Y}_1 - \bar{Y}_2)^2}{n_1 + n_2}\end{aligned}$$

which completes the proof.

Exercise 8.3*

$$\begin{aligned}\gamma_{jk} - \gamma_{j'k} &= \gamma_{jk'} - \gamma_{j'k'}\\ (\mu_{jk} - \mu_{j\cdot} - \mu_{\cdot k} + \mu_{\cdot\cdot}) - (\mu_{j'k} - \mu_{j'\cdot} - \mu_{\cdot k} + \mu_{\cdot\cdot}) &\\ = (\mu_{jk'} - \mu_{j\cdot} - \mu_{\cdot k'} + \mu_{\cdot\cdot}) - (\mu_{j'k'} - \mu_{j'\cdot} - \mu_{\cdot k'} + \mu_{\cdot\cdot})&\end{aligned}$$

The result then follows immediately by canceling the "dotted" terms (i.e., marginal means):

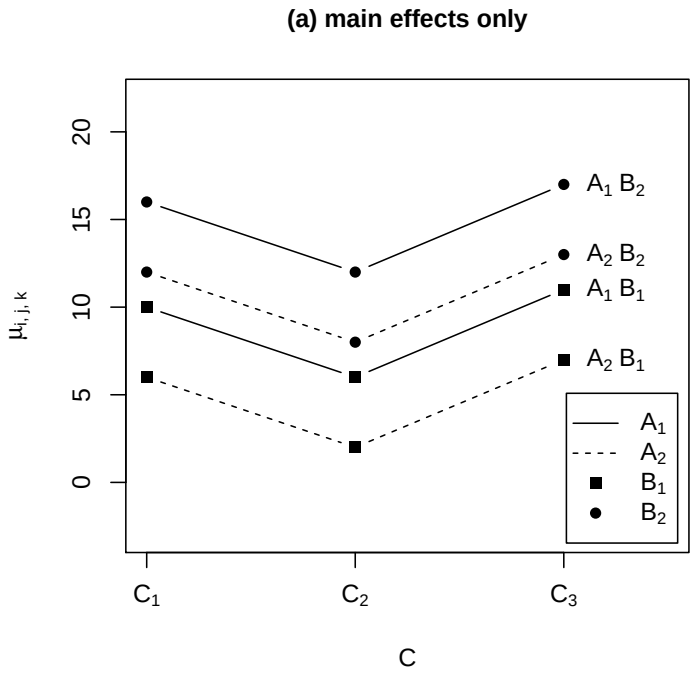
$$\mu_{jk} - \mu_{j'k} = \mu_{jk'} - \mu_{j'k'}$$

Exercise 8.5

I wrote a simple R script (not shown) to compute the cell means from the parameters of the several models. The question doesn't give the value of the general mean μ, which doesn't affect the patterns of cell means. I arbitrarily took $\mu = 10$, but any value, including 0, will do. I chose to plot all of the graphs of cell means on the same scale.

(a) The main-effects-only model. The cell means are

```
          C_1 C_2 C_3
A_1 B_1    10   6  11
    B_2    16  12  17
A_2 B_1     6   2   7
    B_2    12   8  13
```



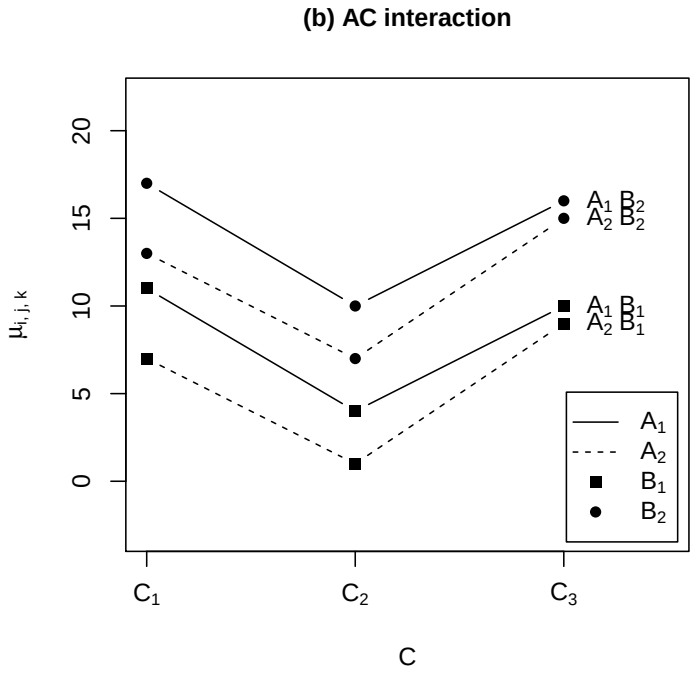
In the absence of interactions, all four profiles of means are parallel.

(b) The model with AC interaction. The cell means are

```
          C_1 C_2 C_3
A_1 B_1    11   4  10
```



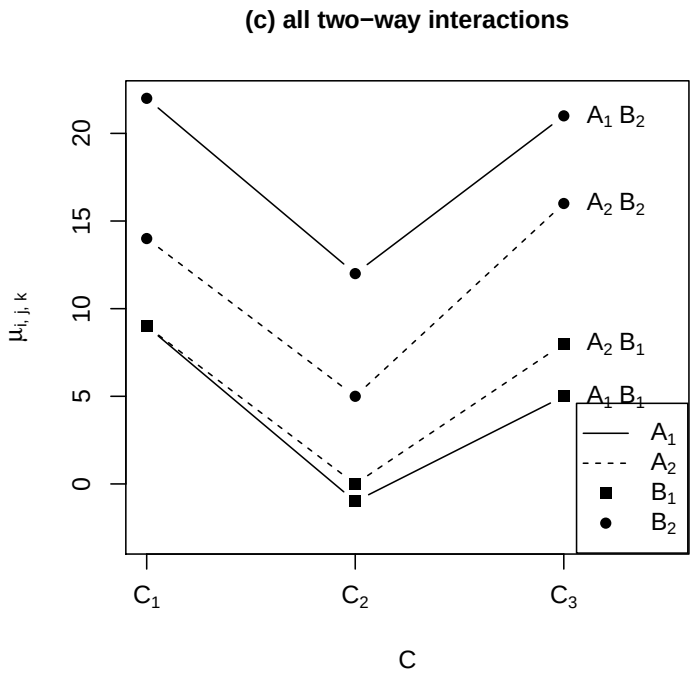
```
     B_2   17  10  16
A_2  B_1    7   1   9
     B_2   13   7  15
```



The pairs of profiles for B_1 and B_2 fixing the level of A are parallel, reflecting the absence of BC interaction, but the profiles for A_1 and A_2 fixing the level of B are not parallel, reflecting the AC interaction. Given the layout of the graph, it's harder to see that the AB interaction is also absent.

(c) The model with all two-way interactions. The cell means are

```
          C_1 C_2 C_3
A_1  B_1    9  -1   5
     B_2   22  12  21
A_2  B_1    9   0   8
     B_2   14   5  16
```

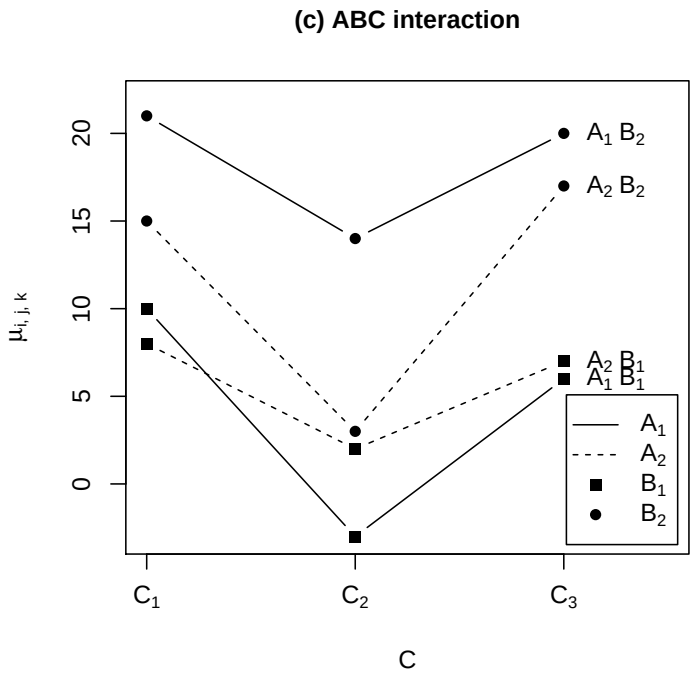



Without explicitly taking differences in cell means (see below) it's very hard to distinguish visually between all two-way interactions and three-way interaction, in part (d). In both cases, the profiles of means are not parallel.

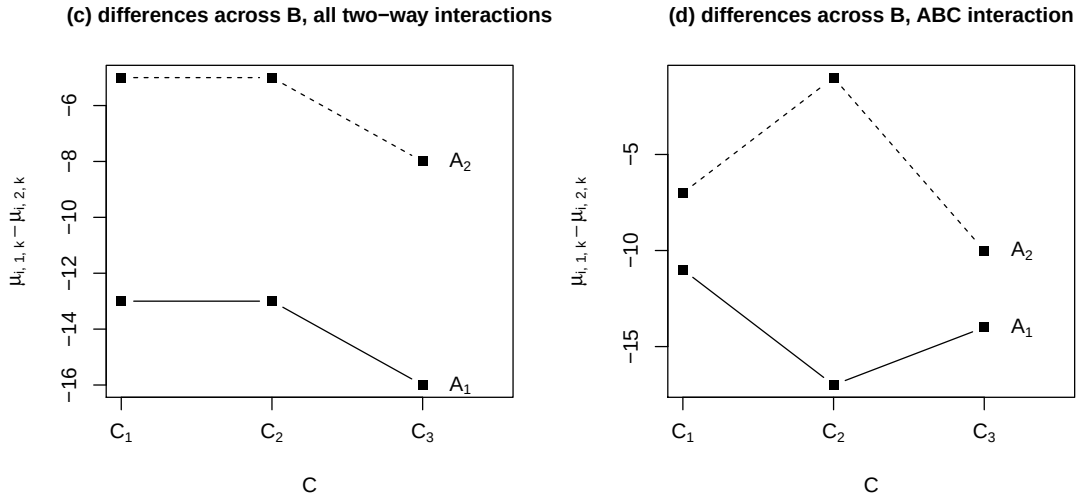
(d) The model with ABC interaction. The cell means are

```
          C_1 C_2 C_3
A_1 B_1    10  -3   6
    B_2    21  14  20
A_2 B_1     8   2   7
    B_2    15   3  17
```

/



(ii) Here are graphs of differences in cell means across the two levels of factor B for cases (c) all two-interactions and (d) ABC interaction:



In the first case, the profiles of differences are parallel, while in the second case they aren't.

Exercise 8.7

$$
\begin{aligned}
\text{SS}(\alpha_A) &= n' \sum_{j=1}^{a} \sum_{k=1}^{b} \sum_{m=1}^{c} A^2_{A(j)} \\
&= n'bc \sum_{j=1}^{a} A^2_{A(j)} \\
&= n'bc \sum_{j=1}^{a} (\bar{Y}_{j\cdot\cdot} - \bar{Y}_{\cdots})^2 \\
\text{SS}(\alpha_{ABC}) &= n' \sum_{j=1}^{a} \sum_{k=1}^{b} \sum_{m=1}^{c} A^2_{ABC(jkm)} \\
&= n' \sum_{j=1}^{a} \sum_{k=1}^{b} \sum_{m=1}^{c} (\bar{Y}_{jkm} - \bar{Y}_{jk\cdot} - \bar{Y}_{j\cdot m} - \bar{Y}_{\cdot km} + \bar{Y}_{j\cdot\cdot} + \bar{Y}_{\cdot k\cdot} + \bar{Y}_{\cdot\cdot m} - \bar{Y}_{\cdots})^2
\end{aligned}
$$

The residuals are $E_{ijkm} = Y_{ijkm} - \bar{Y}_{jkm}$ and so

$$
\begin{aligned}
\text{RSS} &= \sum_{i=1}^{n'} \sum_{j=1}^{a} \sum_{k=1}^{b} \sum_{m=1}^{c} (Y_{ijkm} - \bar{Y}_{jkm})^2 \\
&= \sum_{j=1}^{a} \sum_{k=1}^{b} \sum_{m=1}^{c} (n' - 1) S^2_{jkm} \\
&= (n' - 1) \sum_{j=1}^{a} \sum_{k=1}^{b} \sum_{m=1}^{c} S^2_{jkm}
\end{aligned}
$$

Exercise 8.9

(a) Setting each of X_1 and X_2 to its mean, the two terms involving the covariates drop out, and we can then compute adjusted means as

$$\widetilde{Y}_{jk} = M + A_j + B_k + C_{jk}$$

(b) As mentioned in part (a), the terms involving the covariates drop out because the covariates are expressed as deviations from their means, so we're spared having to compute an adjustment explicitly for each covariate.

(c) For a model that's additive in the two factors, the adjusted means are simply

$$
\begin{aligned}
\text{for rows: } \widetilde{Y}_{(A)j} &= M + A_j \\
\text{for columns: } \widetilde{Y}_{(B)k} &= M + B_k
\end{aligned}
$$

Exercise 8.11*

It's apparent that, as claimed, the two columns for the interactions in the table of contrast "coefficients" meet the criteria for contrasts: The columns sum to 0 and the sum of products for any pair of columns is 0.

We can therefore treat the parameter corresponding to each contrast as a simple linear combination of the six cells means, with weights proportional to the contrast coefficients. To make the meaning of each

parameter clearer, I'll label the means with the subscripts C, A, or B, for the control group and the two experimental groups, and either M or F for gender.

With some rearrangement to clarify the patterns:

(a)

$$\zeta_1 \propto 2(\mu_{CM} - \mu_{CF}) - [(\mu_{AM} + \mu_{BM}) - (\mu_{AF} + \mu_{BF})]$$

Thus, the null hypothesis for the interaction parameter H_0: $\zeta_1 = 0$ specifies that the mean difference between males and females in the control group is the same as the mean difference between males and females in the average if the two experimental groups. All we need is that the parameter ζ_1 be *proportional* to the difference in averages. That is $2(\mu_{CM} - \mu_{CF}) - [(\mu_{AM} + \mu_{BM}) - (\mu_{AF} + \mu_{BF})]$ is 0 if and only if $(\mu_{CM} - \mu_{CF}) - \frac{1}{2}[(\mu_{AM} + \mu_{BM}) - (\mu_{AF} + \mu_{BF})]$ is 0. This is true for the following parts of the exercise as well.

(b)

$$\zeta_2 \propto (\mu_{AM} - \mu_{AF}) - (\mu_{BM} - \mu_{BF})]$$

The null hypothesis for the interaction parameter H_0: $\zeta_2 = 0$ specifies that the mean difference between males and females in the first experimental group is the same as the mean difference between males and females in the second experimental group.

(c)

$$\delta_1 \propto 2(\mu_{CM} + \mu_{CF}) - (\mu_{AM} + \mu_{BM} + \mu_{AF} + \mu_{BF})$$

The null hypothesis for the condition main-effect parameter H_0: $\delta_1 = 0$ specifies that the mean in the control condition averaged over gender is the same as the mean averaged over the two experimental conditions and gender.

(d)

$$\delta_2 \propto (\mu_{AM} + \mu_{BM}) - (\mu_{AF} + \mu_{BF})$$

The null hypothesis for the condition main-effect parameter H_0: $\delta_2 = 0$ specifies that the mean in the first experimental condition averaged over gender is the same as the mean in the second experimental condition averaged over gender.

(e)

$$\beta \propto (\mu_{AM} + \mu_{BM} + \mu_{CM}) - (\mu_{AF} + \mu_{BF} + \mu_{CF})$$

The null hypothesis for the gender main-effect parameter H_0: $\beta = 0$ specifies that the mean for males averaged over condition is the same as the mean for females averaged over condition.

Exercise 8.13*

Here are Equations 8.6 copied from the text:

$$\begin{aligned}
\alpha &= \mu_{23} \\
\beta_1 &= \mu_{13} - \mu_{23} \\
\gamma_1 &= \mu_{21} - \mu_{23} \\
\gamma_2 &= \mu_{22} - \mu_{23} \\
\delta_{11} &= \mu_{11} - \mu_{13} - \mu_{21} + \mu_{23} \\
\delta_{12} &= \mu_{12} - \mu_{13} - \mu_{22} + \mu_{23}
\end{aligned}$$

(a) The hypothesis H_0: $\beta_1 = 0$ is equivalent to H_0: $\mu_{13} - \mu_{23}$ or H_0: $\mu_{13} = \mu_{23}$, that is, that the population cell means for rows R_1 and R_2 are the same at level C_3 of the column factor.

The difference in means $\mu_{13} - \mu_{23}$ is sometimes termed a "simple effect" (the effect of one factor at a *particular* level, C_3, of the other factor), as opposed to a "main effect" (the general effect of one

factor at *any* fixed level of the other factor or possibly *averaged over* the levels of the other factor). When there is interaction, the simple effects for the row factor vary by the level of the column factor, and so the simple effect at any *particular* level isn't reasonably construed as a main effect.

Similarly, the null hypothesis H_0: $\gamma_1 = \gamma_2 = 0$ is equivalent to the null hypothesis H_0: $\mu_{21} - \mu_{23} = 0, \mu_{22} - \mu_{23} = 0$ or H_0: $\mu_{21} = \mu_{22} = \mu_{23}$, testing for the simple effect of the column factor within level R_2 of the row factor. If there is interaction, the simple effects of the column factor will vary across the two rows.

(b) If there is no interaction, however, all of the row simple effects are equal to each other, and so testing that any of them, for example, within the third column, is 0 tests the main effect of the row factor. Likewise, in the absence of interaction, all of the column simple effects are equal, and so testing that any of them, for example, within the second row, tests the column main effects. These tests have low power, however, because they use only part of the data—the data only in the first column for testing row effects and only in the first row for testing column effects.

Exercises for Chapter 9

Exercise 9.1*

(a) Expanding Equation 9.5, we have

$$\begin{aligned}
\mu_1 &= \mu + \alpha_1 \\
\mu_2 &= \mu + \alpha_2 \\
&\vdots \\
\mu_{m-1} &= \mu + \alpha_{m-1} \\
\mu_m &= \mu - (\alpha_1 + \alpha_2 + \cdots + \alpha_{m-1})
\end{aligned}$$

Then, adding up these equations,

$$\sum_{j=1}^{m} \mu_j = m\mu + \sum_{j=1}^{m-1} \alpha_j - \sum_{j=1}^{m-1} \alpha_j$$

So $\mu = \sum_{j=1}^{m} \mu_j/m = \mu_\cdot$, $\mu_j = \mu_\cdot + \alpha_j$ for $j = 1, \ldots, m-1$, and $\alpha_j = \mu_j - \mu_\cdot$ for $j = 1, \ldots, m-1$.

(b) A straightforward (if tedious) approach is to invert $\mathbf{X}_R$ and solve for $\boldsymbol{\beta} = \mathbf{X}_R^{-1}\boldsymbol{\mu}$:

$$\begin{bmatrix} \mu \\ \alpha_1 \\ \beta_1 \\ \beta_2 \\ \gamma_{11} \\ \gamma_{12} \end{bmatrix} = \begin{bmatrix} \frac{1}{6} & \frac{1}{6} & \frac{1}{6} & \frac{1}{6} & \frac{1}{6} & \frac{1}{6} \\ \frac{1}{6} & \frac{1}{6} & \frac{1}{6} & -\frac{1}{6} & -\frac{1}{6} & -\frac{1}{6} \\ \frac{1}{3} & -\frac{1}{6} & -\frac{1}{6} & \frac{1}{3} & -\frac{1}{6} & -\frac{1}{6} \\ -\frac{1}{6} & \frac{1}{3} & -\frac{1}{6} & -\frac{1}{6} & \frac{1}{3} & -\frac{1}{6} \\ \frac{1}{3} & -\frac{1}{6} & -\frac{1}{6} & -\frac{1}{3} & \frac{1}{6} & \frac{1}{6} \\ -\frac{1}{6} & \frac{1}{3} & -\frac{1}{6} & \frac{1}{6} & -\frac{1}{3} & \frac{1}{6} \end{bmatrix} \begin{bmatrix} \mu_{11} \\ \mu_{12} \\ \mu_{13} \\ \mu_{21} \\ \mu_{22} \\ \mu_{23} \end{bmatrix}$$

Then, writing out the result for each element on the left-hand side and rearranging:

$$\begin{aligned}
\mu &= \tfrac{1}{6}(\mu_{11} + \mu_{12} + \mu_{13} + \mu_{21} + \mu_{22} + \mu_{23}) \\
&= \mu_{\cdot\cdot} \\
\alpha_1 &= \tfrac{1}{6}(\mu_{11} + \mu_{12} + \mu_{13}) - \tfrac{1}{6}(\mu_{21} + \mu_{22} + \mu_{23}) \\
&= \tfrac{1}{2}(\mu_{1\cdot} - \mu_{2\cdot}) \\
&= \tfrac{1}{2}[2\mu_{1\cdot} - (\mu_{1\cdot} + \mu_{2\cdot})] \\
&= \mu_{1\cdot} - \mu_{\cdot\cdot} \\
\beta_1 &= \tfrac{1}{3}(\mu_{11} + \mu_{21}) - \tfrac{1}{6}(\mu_{12} + \mu_{13} + \mu_{22} + \mu_{23}) \\
&= \left(\tfrac{1}{3} + \tfrac{1}{6}\right)(\mu_{11} + \mu_{21}) - \tfrac{1}{6}(\mu_{11} + \mu_{12} + \mu_{13} + \mu_{21} + \mu_{22} + \mu_{23}) \\
&= \mu_{\cdot 1} - \mu_{\cdot\cdot} \\
\beta_2 &= \tfrac{1}{3}(\mu_{12} + \mu_{22}) - \tfrac{1}{6}(\mu_{11} + \mu_{13} + \mu_{21} + \mu_{23}) \\
&= \mu_{\cdot 2} - \mu_{\cdot\cdot} \\
\gamma_{11} &= \tfrac{1}{3}\mu_{11} - \tfrac{1}{6}\mu_{12} - \tfrac{1}{6}\mu_{13} - \tfrac{1}{3}\mu_{21} + \tfrac{1}{6}\mu_{22} + \tfrac{1}{6}\mu_{23} \\
&= \tfrac{6}{6}\mu_{11} - \tfrac{2}{6}(\mu_{11} + \mu_{12} + \mu_{13}) - \tfrac{3}{6}(\mu_{11} + \mu_{21}) + \tfrac{1}{6}(\mu_{11} + \mu_{12} + \mu_{13} + \mu_{21} + \mu_{22} + \mu_{23}) \\
&= \mu_{11} - \mu_{1\cdot} - \mu_{\cdot 1} + \mu_{\cdot\cdot} \\
\gamma_{21} &= -\tfrac{1}{6}\mu_{11} + \tfrac{1}{3}\mu_{12} - \tfrac{1}{6}\mu_{13} + \tfrac{1}{6}\mu_{21} - \tfrac{1}{3}\mu_{22} + \tfrac{1}{6}\mu_{23} \\
&= \mu_{12} - \mu_{1\cdot} - \mu_{\cdot 2} + \mu_{\cdot\cdot}
\end{aligned}$$

Exercise 9.3

I used R to invert $\mathbf{X}_B^{-1}$:

```
> XBInv <- matrix(c(
+   0, 0, 0, 1,
+   1, 0, 0,-1,
+   0, 1, 0,-1,
+   0, 0, 1,-1), 4, 4, byrow=TRUE)

> solve(XBInv)
     [,1] [,2] [,3] [,4]
[1,]    1    1    0    0
[2,]    1    0    1    0
[3,]    1    0    0    1
[4,]    1    0    0    0
```

Thus,

$$\begin{bmatrix}\mu_1\\ \mu_2\\ \mu_3\\ \mu_4\end{bmatrix} = \begin{bmatrix}1 & 1 & 0 & 0\\ 1 & 0 & 1 & 0\\ 1 & 0 & 0 & 1\\ 1 & 0 & 0 & 0\end{bmatrix}\begin{bmatrix}\mu\\ \gamma_1\\ \gamma_2\\ \gamma_3\end{bmatrix}$$

That is, $\mu_1 = \mu + \gamma_1, \mu_2 = \mu + \gamma_2, \mu_3 = \mu + \gamma_3, \mu_4 = \mu$, which is equivalent to 0/1 dummy coding.

Exercise 9.5

(a)* Multiplying the standardized regression equation through by $\mathbf{Z}_X'/(n-1)$, as suggested:

$$\frac{1}{n-1}\mathbf{Z}_X'\mathbf{z}_y = \frac{1}{n-1}\mathbf{Z}_X'\mathbf{Z}_X\mathbf{b}^* + \frac{1}{n-1}\mathbf{Z}_X'\mathbf{e}^*$$

Now examine the first element on the left-hand side of the equation, which multiplies the first row of $\mathbf{Z}_X'$ (i.e., the first column of $\mathbf{Z}_X$) into $\mathbf{z}_y$:

$$\begin{aligned}\frac{1}{n-1}\sum_{i=1}^{n} Z_{i1}Z_{iy} &= \frac{1}{n-1}\sum_{i=1}^{n}\frac{X_{i1}-\bar{X}_1}{S_1}\times\frac{Y_i-\bar{Y}}{S_y}\\ &= \frac{\frac{\sum(X_{i1}-\bar{X}_1)(Y_i-\bar{Y})}{n-1}}{S_1S_y}\\ &= \frac{S_{1y}}{S_1S_y}\\ &= r_{1y}\end{aligned}$$

Similarly, the remaining elements of $\frac{1}{n-1}\mathbf{Z}_X'\mathbf{z}_y$ are correlations between the rest of the Xs and Y, and $\frac{1}{n-1}\mathbf{Z}_X'\mathbf{Z}_X$ is the correlation matrix among the Xs. The second term on the right, $\frac{1}{n-1}\mathbf{Z}_X'\mathbf{e}^*$, isn't a vector of correlations because $\mathbf{e}^*$ isn't a standardized variable (recall that it's scaled by S_y rather than by S_E), but the term is 0 because, as a scalar multiple of the residuals $\mathbf{e}$, the scaled residuals $\mathbf{e}^*$ have 0 cross-products with the standardized explanatory variables. Thus

$$\begin{aligned}\mathbf{r}_{Xy} &= \mathbf{R}_{XX}\mathbf{b}^* + \mathbf{0}\\ \mathbf{b}^* &= \mathbf{R}_{XX}^{-1}\mathbf{r}_{Xy}\end{aligned}$$

(b) I did the computations in R:

```
> R <- matrix(0, 5, 5)
> R[upper.tri(R)] <- c(
```



```
+    .516,
+    .453, .438,
+    .332, .417, .538,
+    .322, .405, .596, .541
+ )

> R <- R + t(R)
> diag(R) <- 1
> R
      [,1]  [,2]  [,3]  [,4]  [,5]
[1,] 1.000 0.516 0.453 0.332 0.322
[2,] 0.516 1.000 0.438 0.417 0.405
[3,] 0.453 0.438 1.000 0.538 0.596
[4,] 0.332 0.417 0.538 1.000 0.541
[5,] 0.322 0.405 0.596 0.541 1.000

> b <- as.vector(solve(R[1:4, 1:4]) %*% R[1:4, 5])
> names(b) <- c("F's Ed", "F's Occ", "R's Ed", "R's 1st Job")
> b
     F's Ed     F's Occ      R's Ed R's 1st Job
-0.01394205  0.12052602  0.39830303  0.28108238
```

The slope for father's education is not only very small but negative! No, this doesn't necessarily imply that father's education is unimportant as a cause of respondent's occupational status, just that it isn't an important *direct* cause, holding father's occupational status, respondent's education, and the status of the respondent's first job constant. These other explanatory variables could very well be positively affected by father's education, which could therefore have an important *indirect* effect on respondent's occupational status.

(c)* Following the hint,

$$\begin{aligned}
\frac{1}{n-1}\mathbf{z}_y'\mathbf{z}_y &= \frac{1}{n-1}\mathbf{z}_y'\mathbf{Z}_X\mathbf{b}^* + \frac{1}{n-1}\mathbf{z}_y'\mathbf{e}^* \\
1 &= \mathbf{r}_{yX}'\mathbf{b}^* + \frac{1}{n-1}(\widehat{\mathbf{z}}_y + \mathbf{e}^*)'\mathbf{e}^* \\
1 &= \mathbf{r}_{yX}'\mathbf{b}^* + \frac{1}{n-1}\mathbf{e}^{*\prime}\mathbf{e}^*
\end{aligned}$$

I used the facts that the sum of cross products of standardized variables divided by $n-1$ is their correlation, and that the fitted values in least-squares regression are uncorrelated with the residuals.

The last line of the equation shows the proportional division of the variance of z_y^* into "explained" and residual components. We can alternatively write

$$\begin{aligned}
\frac{1}{n-1}\mathbf{z}_y'\mathbf{Z}_X\mathbf{b}^* &= \frac{1}{n-1}\mathbf{z}_y'\widehat{\mathbf{z}}_y \\
&= \frac{1}{n-1}(\widehat{\mathbf{z}}_y + \mathbf{e}^*)'\widehat{\mathbf{z}}_y \\
&= \frac{1}{n-1}\widehat{\mathbf{z}}_y'\widehat{\mathbf{z}}_y
\end{aligned}$$

The explained component, $\frac{1}{n-1}\widehat{\mathbf{z}}_y'\widehat{\mathbf{z}}_y = \mathbf{r}_{yX}'\mathbf{b}^*$ is the squared multiple correlation, R^2.

Applied to Blau and Duncan's regression,

```
> R[1:4, 5] %*% b
```



```
          [,1]
[1,] 0.4337779
```

That is, $R^2 = .434$.

Exercise 9.7*

The matrix $\mathbf{AX}$ is of order $(k+1 \times k+1)$. Let the first row of this matrix be given by $\mathbf{w}_0' \equiv [w_{10}, w_{11}, \ldots, w_{1k}]$. If $\boldsymbol{\beta} = [1, 0, \ldots, 0]'$, then $\mathbf{w}_0'\boldsymbol{\beta} = w_{10}$, and so the first element of $\mathbf{w}_0'$ must be 0. Similarly if $\boldsymbol{\beta} = [0, 1, \ldots, 0]'$, then $\mathbf{w}_0'\boldsymbol{\beta} = w_{11}$, and the second element of $\mathbf{w}_0'$ must be 0. Setting each remaining element of $\boldsymbol{\beta}$ to 0 in turn similarly implies that *all* of the elements of $\mathbf{w}_0'$ must be 0. But there's nothing special about $\mathbf{w}_0'$; we could repeat this argument for *each* subsequent row $\mathbf{w}_j'$ of $\mathbf{AX}$, $i = 1, \ldots, k$, and so $\mathbf{AX} = \mathbf{0}$

Exercise 9.9*

The likelihood is

$$L(\boldsymbol{\beta}, \sigma_\varepsilon^2) = \frac{1}{(2\pi\sigma_\varepsilon^2)^{n/2}} \exp\left[-\frac{(\mathbf{y} - \mathbf{X}\boldsymbol{\beta})'(\mathbf{y} - \mathbf{X}\boldsymbol{\beta})}{2\sigma_\varepsilon^2}\right]$$

The MLEs are $\widehat{\boldsymbol{\beta}} = (\mathbf{X}'\mathbf{X})^{-1}\mathbf{X}'\mathbf{y}$ and $\widehat{\sigma}_\varepsilon^2 = \mathbf{e}'\mathbf{e}/n$, where the residuals $\mathbf{e} = \mathbf{y} - \mathbf{X}\widehat{\boldsymbol{\beta}}$. Thus, at the MLEs,

$$\begin{aligned} L(\boldsymbol{\beta}, \sigma_\varepsilon^2) &= \frac{1}{\left(2\pi\frac{\mathbf{e}'\mathbf{e}}{n}\right)^{n/2}} \exp\left[-\frac{\mathbf{e}'\mathbf{e}}{\frac{2\mathbf{e}'\mathbf{e}}{n}}\right] \\ &= \left(2\pi\frac{\mathbf{e}'\mathbf{e}}{n}\right)^{-n/2} \exp\left(-\frac{n}{2}\right) \\ &= \left(2\pi e\frac{\mathbf{e}'\mathbf{e}}{n}\right)^{-n/2} \end{aligned}$$

Exercise 9.11*

A simple approach is to eliminate the intercept A by subtracting their means from Y and the Xs. Then letting $y_i^* \equiv Y_i - \bar{Y}$ and $x_{ij}^* \equiv x_{ij} - \bar{x}_j$, $j = 1, 2$:

$$y_i^* = B_1^* x_{ij}^* + B_2^* x_{i2}^* + E_i$$

and

$$\begin{aligned} \begin{bmatrix} B_1^* \\ B_2^* \end{bmatrix} = \mathbf{b}_1 &= (\mathbf{X}^{*\prime}\mathbf{X}^*)^{-1}\mathbf{X}^{*\prime}\mathbf{y}^* \\ &= \begin{bmatrix} \sum x_{i1}^{*2} & \sum x_{i1}^* x_{i2}^* \\ \sum x_{i1}^* x_{i2}^* & \sum x_{i2}^{*2} \end{bmatrix}^{-1} \begin{bmatrix} \sum x_{i1}^* y_i^* \\ \sum x_{i2}^* y_i^* \end{bmatrix} \end{aligned}$$

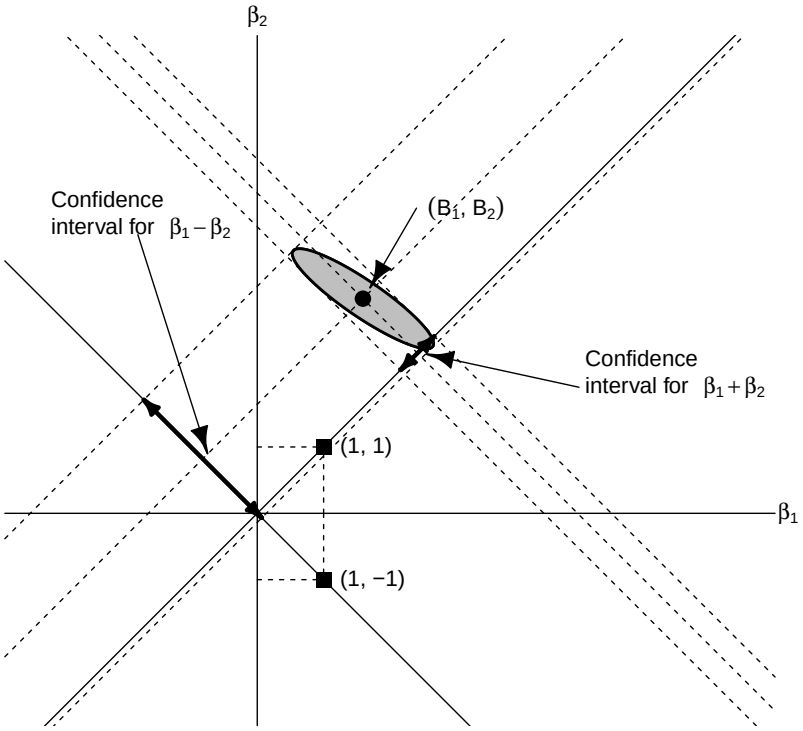
So, $\mathbf{V}_{11} = [(\mathbf{X}^{*\prime}\mathbf{X}^*)^{-1}]^{-1} = \mathbf{X}^{*\prime}\mathbf{X}^*$.

The generalization to any number of Xs is immediate, noting that

$$y_i^* = B_1^* x_{ij}^* + B_2^* x_{i2}^* + \cdots + B_k^* x_{ik}^* + E_i$$

Exercise 9.13

Here's a graph similar to Figure 9.2 showing both the confidence interval for $\beta_1 + \beta_2$ and for $\beta_1 - \beta_2$.



Given the positive correlation of X_1 and X_2, and the consequent negative tilt in the confidence ellipse for β_1 and β_2, the shadow of the ellipse on the line through $(1, 1)$, giving the confidence interval for $\beta_1 + \beta_2$ is narrower than the shadow on the line through $(1, -1)$ giving the confidence interval for $\beta_1 - \beta_2$. Thus $\beta_1 + \beta_2$ is estimated more precisely than $\beta_1 - \beta_2$.

If X_1 and X_2 were negatively correlated, then the confidence ellipse would have a positive tilt, and a larger shadow on the line through $(1, 1)$ than on the line through $(1, -1)$, implying that $\beta_1 - \beta_2$ is estimated more precisely than $\beta_1 + \beta_2$.

Exercise 9.15

(a) The row basis of the of the full-rank model matrix is

$$\mathbf{X}_B = \begin{bmatrix} 1 & 1 & 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

(b) I computed $\mathbf{X}_B^{-1}$ using R:

```
> X_B <- matrix(c(
+   1, 1, 1, 0, 1, 0,
+   1, 1, 0, 1, 0, 1,
+   1, 1, 0, 0, 0, 0,
+   1, 0, 1, 0, 0, 0,
```



```
+   1, 0, 0, 1, 0, 0,
+   1, 0, 0, 0, 0, 0
+ ), 6, 6, byrow=TRUE)

> X_B
     [,1] [,2] [,3] [,4] [,5] [,6]
[1,]    1    1    1    0    1    0
[2,]    1    1    0    1    0    1
[3,]    1    1    0    0    0    0
[4,]    1    0    1    0    0    0
[5,]    1    0    0    1    0    0
[6,]    1    0    0    0    0    0

> solve(X_B)
     [,1] [,2] [,3] [,4] [,5] [,6]
[1,]    0    0    0    0    0    1
[2,]    0    0    1    0    0   -1
[3,]    0    0    0    1    0   -1
[4,]    0    0    0    0    1   -1
[5,]    1    0   -1   -1    0    1
[6,]    0    1   -1    0   -1    1
```

Thus,

$$\boldsymbol{\beta}_F = \mathbf{X}_B^{-1}\boldsymbol{\mu}$$

$$\begin{bmatrix} \mu \\ \alpha_1 \\ \beta_1 \\ \beta_2 \\ \gamma_{11} \\ \gamma_{22} \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & -1 \\ 0 & 0 & 0 & 1 & 0 & -1 \\ 0 & 0 & 0 & 0 & 1 & -1 \\ 1 & 0 & -1 & -1 & 0 & 1 \\ 0 & 1 & -1 & 0 & -1 & 1 \end{bmatrix} \begin{bmatrix} \mu_{11} \\ \mu_{12} \\ \mu_{13} \\ \mu_{21} \\ \mu_{22} \\ \mu_{23} \end{bmatrix}$$

Consequently, $\alpha_1 = \mu_{13} - \mu_{23}$ and the corresponding null hypothesis is H_0: $\mu_{13} - \mu_{23}$. Similarly $\beta_1 = \mu_{21} - \mu_{23}$ and $\beta_2 = \mu_{22} - \mu_{23}$, so the corresponding null hypothesis is H_0: $\mu_{21} = \mu_{23}, \mu_{22} = \mu_{23}$. These hypotheses are sensible, in that they test for specific differences across the levels of each factor within the last level of the other factor, but they aren't what we would normally think of as tests of main effects.

(c) I did the computations in R, but rather than allowing the `lm()` function to construct regressors for factors automatically, as is normal, I instead created the requisite regressors in each case manually. Otherwise `lm()` resists fitting models that violate marginality.

```
> library("carData") # for Moore data set

> # simple function to compute incremental SSs
> SS <- function(model1, model2){
+   RSS1 <- sum(residuals(model1)^2)
+   RSS2 <- sum(residuals(model2)^2)
+   abs(RSS1 - RSS2)
+ }

> # using "deviation" ("effect") coding

> A1e <- with(Moore, ifelse(partner.status == "low", 1, -1))
> B1e <- with(Moore, ifelse(fcategory == "low", 1,
+                           ifelse (fcategory == "high", -1, 0)))
> B2e <- with(Moore, ifelse(fcategory == "medium", 1,
+                           ifelse (fcategory == "high", -1, 0)))
```



```
> C11e <- A1e*B1e
> C12e <- A1e*B2e

>   # SS(A | B)
> SS(lm(conformity ~ B1e + B2e, data=Moore),
+    lm(conformity ~ A1e + B1e + B2e, data=Moore))
[1] 212.2138

>   # SS(B | A)
> SS(lm(conformity ~ A1e, data=Moore),
+    lm(conformity ~ A1e + B1e + B2e, data=Moore))
[1] 11.6147

>   # SS(C | A, B)
> SS(lm(conformity ~ A1e + B1e + B2e, data=Moore),
+    lm(conformity ~ A1e + B1e + B2e + C11e + C12e, data=Moore))
[1] 175.4889

> # SS(A | B, C)
> SS(lm(conformity ~ B1e + B2e + C11e + C12e, data=Moore),
+    lm(conformity ~ A1e + B1e + B2e + C11e + C12e, data=Moore))
[1] 239.5624

> # SS(B | A, C)
> SS(lm(conformity ~ A1e + C11e + C12e, data=Moore),
+    lm(conformity ~ A1e + B1e + B2e + C11e + C12e, data=Moore))
[1] 36.01871

> # using 0, 1 "dummy" coding:
>
> A1d <- with(Moore, ifelse(partner.status == "low", 1, 0))
> B1d <- with(Moore, ifelse(fcategory == "low", 1, 0))
> B2d <- with(Moore, ifelse(fcategory == "medium", 1, 0))
> C11d <- A1d*B1d
> C12d <- A1d*B2d

>   # SS*(A | B) [matches SS(A | B)]
> SS(lm(conformity ~ B1d + B2d, data=Moore),
+    lm(conformity ~ A1d + B1d + B2d, data=Moore))
[1] 212.2138

>   # SS*(B | A) [matches SS(B | A)]
> SS(lm(conformity ~ A1d, data=Moore),
+    lm(conformity ~ A1d + B1d + B2d, data=Moore))
[1] 11.6147

>   # SS*(C | A, B) [matches SS(C | A, B)]
> SS(lm(conformity ~ A1d + B1d + B2d, data=Moore),
+    lm(conformity ~ A1d + B1d + B2d + C11d + C12d, data=Moore))
[1] 175.4889

>   # SS*(A | B, C) [doesn't match SS(A | B, C)]
> SS(lm(conformity ~ B1d + B2d + C11d + C12d, data=Moore),
+    lm(conformity ~ A1d + B1d + B2d + C11d + C12d, data=Moore))
[1] 2.20119

>   # SS*(B | A, C) [doesn't match SS(B | A, C)]
> SS(lm(conformity ~ A1d + C11d + C12d, data=Moore),
+    lm(conformity ~ A1d + B1d + B2d + C11d + C12d, data=Moore))
```



```
[1] 89.67408
```

First, the results match those in Table 8.4 (page 175), within rounding error. Second, the results demonstrate the equalities and inequalities specified in the question.

(d) There is a wide variety of statistical software capable of performing two-way ANOVA and different readers will have access to different software, so I'll leave this one to you!

Exercise 9.17

If the purpose of the study were to determine the effect of simply providing a voucher, rather than the effect of actual private-school attendance, then there's no need to use an instrumental variable, and because of randomization the experiment can answer the question directly. From a public-policy perspective, this question may indeed be of more direct interest, because a social program could provide vouchers to students but, presumably, couldn't make the voucher recipients attend private schools nor prevent non-recipients from attending them.

Exercise 9.19

We probably prefer to use Equation 9.30, for the *estimated* asymptotic covariance matrix of the coefficients, rather than Equation 9.29.

When $\mathbf{Z} = \mathbf{X}$ Equation 9.28 becomes

$$\mathbf{b}_{\text{IV}} = (\mathbf{X}'\mathbf{X})^{-1}\mathbf{X}'\mathbf{y}$$

which is the formula for the least-squares estimator.

Similarly, Equation 9.30 becomes

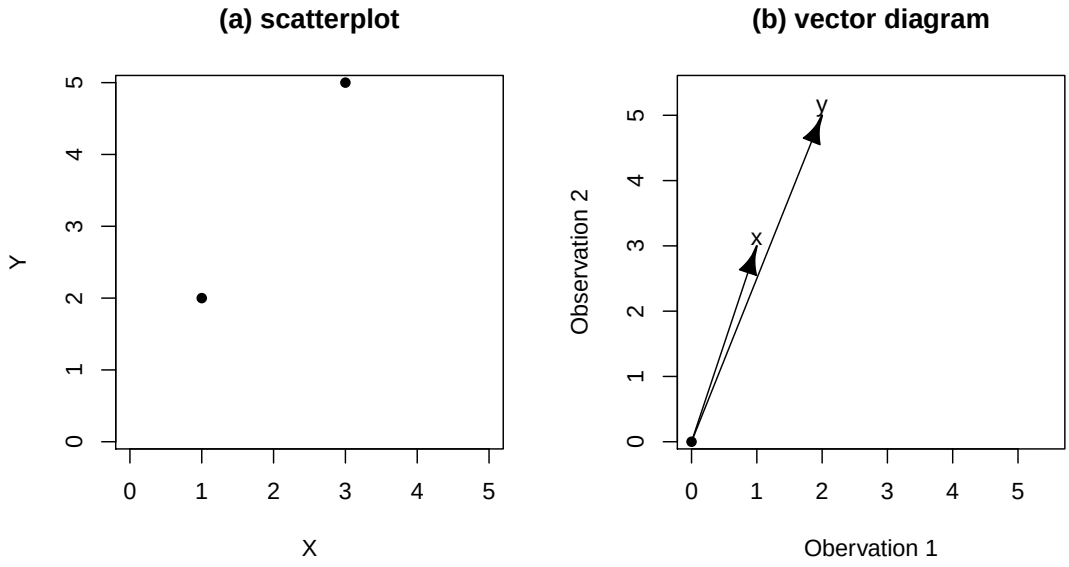
$$\begin{aligned}\widehat{\mathcal{V}}(\mathbf{b}_{\text{IV}}) &= \widehat{\sigma}^2_\varepsilon(\mathbf{X}'\mathbf{X})^{-1}(\mathbf{X}'\mathbf{X})(\mathbf{X}'\mathbf{X})^{-1} \\ &= \widehat{\sigma}^2_\varepsilon(\mathbf{X}'\mathbf{X})^{-1}\end{aligned}$$

This equation is the same as $V(\mathbf{b})$ for the least-squares estimator if we use $S^2_E = \sum E_i^2/(n-k-1)$ to estimate the error variance rather than the maximum-likelihood estimator $\widehat{\sigma}^2_\varepsilon = \sum E_i^2/n$.

Exercises for Chapter 10

Exercise 10.1

The scatterplot and the vector diagram:



Exercise 10.3*

That $\mathbf{e} \cdot \widehat{\mathbf{y}}$ is 0 is probably obvious from the geometry, but it is also easy to show:

$$\begin{aligned}\mathbf{e} \cdot \widehat{\mathbf{y}} &= \mathbf{e} \cdot (A\mathbf{1}_n + B\mathbf{x}) \\ &= A\mathbf{e} \cdot \mathbf{1}_n + B\mathbf{e} \cdot \mathbf{x} \\ &= A \times 0 + B \times 0 \\ &= 0\end{aligned}$$

Exercise 10.5

$$\begin{aligned}\mathbf{x}^* \cdot \mathbf{y}^* &= \mathbf{x}^* \cdot (\widehat{\mathbf{y}}^* + \mathbf{e}) \\ &= \mathbf{x}^* \cdot \widehat{\mathbf{y}}^* + \mathbf{x}^* \cdot \mathbf{e} \\ &= \mathbf{x}^* \cdot \widehat{\mathbf{y}}^* + 0 \\ &= \mathbf{x}^* \cdot \widehat{\mathbf{y}}^*\end{aligned}$$

Exercise 10.7

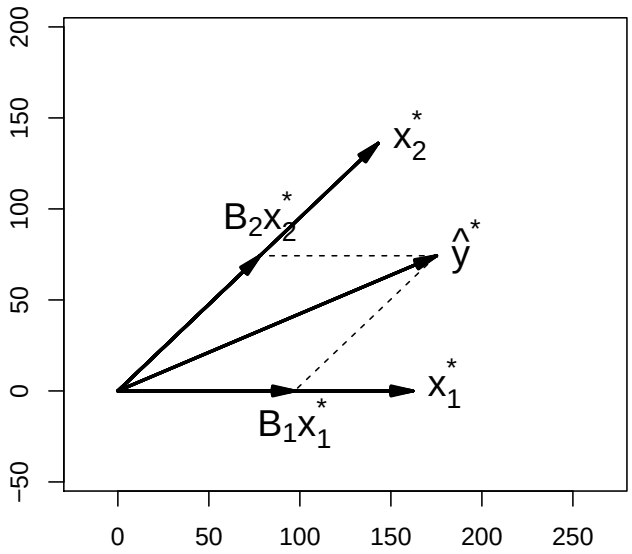
I used R to draw the vector diagrams.

(a) The $\{\mathbf{x}_1^*, \mathbf{x}_2^*\}$ plane:

```
> library("matlib")
> library("MASS")
> library("carData")

> m <- lm(prestige ~ income + education, data=Duncan)
> summary(m)

Call:
```

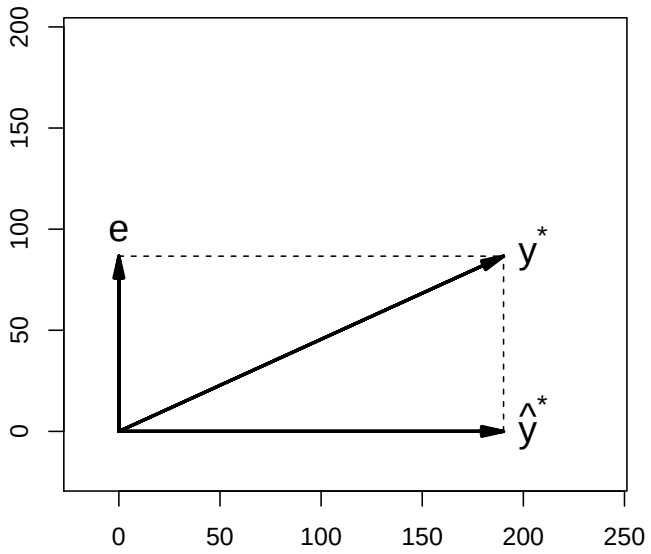


```
lm(formula = prestige ~ income + education, data = Duncan)

Residuals:
    Min      1Q  Median      3Q     Max
-29.538  -6.417   0.655   6.605  34.641

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept) -6.06466    4.27194  -1.420    0.163
income       0.59873    0.11967   5.003 1.05e-05 ***
education    0.54583    0.09825   5.555 1.73e-06 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 13.37 on 42 degrees of freedom
Multiple R-squared:  0.8282,    Adjusted R-squared:   0.82
F-statistic: 101.2 on 2 and 42 DF,  p-value: < 2.2e-16

> len <- function(x) sqrt(sum((x - mean(x))^2))

> me <- lm(education ~ income, data=Duncan)
> x1 <- c(x=len(Duncan$income), y=0)
> x2 <- c(x=len(fitted(me)), y=len(residuals(me)))
> all.equal(sqrt(sum(x2^2)), len(Duncan$education)) # check
[1] TRUE
> b <- coef(m)
> B1x1 <- b[2]*x1
> B2x2 <- b[3]*x2
> yhat <- B1x1 + B2x2

> Vectors <- rbind(
+   x1, x2, B1x1, B2x2, yhat
+ )

> eqscplot(c(0, 250), c(-50, 200), type="n", xlab="", ylab="")

> vectors(Vectors,
+         labels=c(expression(x[1]^"*"),
+                  expression(x[2]^"*"),
+                  expression(B[1]*x[1]^"*"),
+                  expression(B[2]*x[2]^"*"),
+                  expression(hat(y)^"*")),
+         pos.lab=c(4, 4, 1, 3, 4))
> lines(c(yhat[1], B1x1[1]), c(yhat[2], B1x1[2]), lty=2)
> lines(c(yhat[1], B2x2[1]), c(yhat[2], B2x2[2]), lty=2)

> # angle between income and education vectors in degrees
> with(Duncan, 180*acos(cor(income, education))/pi)
[1] 43.5717
> 180*acos(sum(x1*x2)/sqrt(sum(x1^2)*sum(x2^2)))/pi # check
[1] 43.5717
```

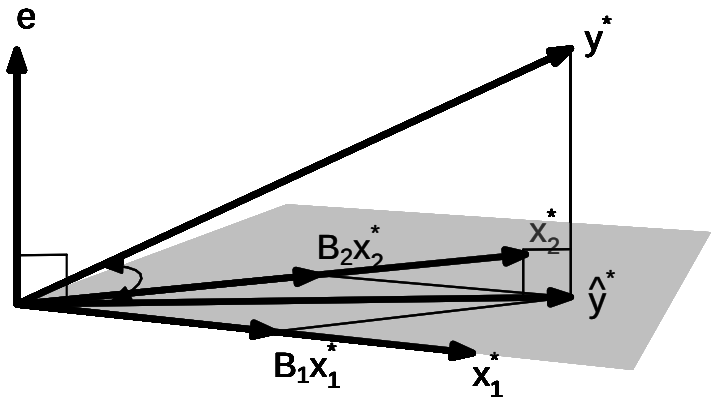



(b) The $\{\mathbf{y}^*, \widehat{\mathbf{y}}^*\}$ plane:

```
> yhat <- c(len(fitted(m)), 0)
> e <- c(0, len(residuals(m)))
> y <- c(len(fitted(m)), len(residuals(m)))
> Vectors <- rbind(yhat,
+                  e,
+                  y)
> eqscplot(c(0, 224), c(-25, 200), type="n", xlab="", ylab="")
> vectors(Vectors,
+         labels=c(expression(hat(y)^"*"),
+                  "e",
+                  expression(y^"*")),
+         pos.lab=c(4, 3, 4))
> lines(c(e[1], y[1]), c(e[2], y[2]), lty=2)
> lines(c(yhat[1], y[1]), c(yhat[2], y[2]), lty=2)
```

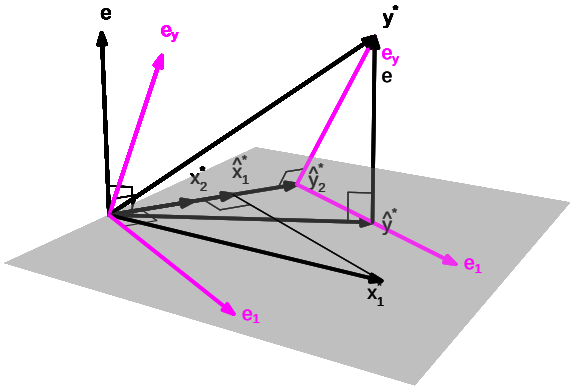



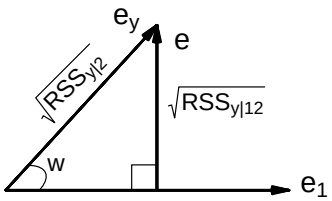
Here's a 3D vector diagram for Duncan's regression (not showing the R code):



Exercise 10.9

(a) Here is the vector diagram:





The upper panel shows the 3D vector space generated by the mean-deviation vectors $\mathbf{x}_1^*$, $\mathbf{x}_2^*$, and $\mathbf{y}^*$.

- The orthogonal projection of $\mathbf{y}^*$ onto the plane spanned by $\mathbf{x}_1^*$ and $\mathbf{x}_2^*$ is $\widehat{\mathbf{y}}^*$, and the residual vector for this regression (i.e., the residuals from the multiple regression of Y on both x_1 and x_2) is $\mathbf{e}$. A copy of this vector is moved from the origin so that its tail is at the tip of $\widehat{\mathbf{y}}^*$. (Recall that a vector is uniquely defined by its direction and length.)
- The orthogonal projection of $\mathbf{y}^*$ onto $\mathbf{x}_2^*$ alone is $\widehat{\mathbf{y}}_2^*$, and (as specified in the exercise) the residual vector for this regression is $\mathbf{e}_y$, shown in magenta. A copy of $\mathbf{e}_y$ is moved so that its tail is at the tip of $\widehat{\mathbf{y}}_2^*$.
- The orthogonal projection of $\mathbf{x}_1^*$ onto $\mathbf{x}_2^*$ is $\widehat{\mathbf{x}}_1^*$, and the residual vector for this regression (also as specified in the exercise) is $\mathbf{e}_1$, shown in magenta, which lies in the plane spanned by $\mathbf{x}_1^*$ and $\mathbf{x}_2^*$; for reasons to be made clear shortly, a copy of this residual vector is moved so that its tail is also at the tip of $\widehat{\mathbf{y}}_2^*$.

  Notice that when $\mathbf{e}_1$, which is orthogonal to $\mathbf{x}_2^*$, is positioned so that its tail is at the tip of $\widehat{\mathbf{y}}_2^*$, the vector goes through the tip of $\widehat{\mathbf{y}}^*$, because (as explained in the discussion of Figure 10.10 on page 255) the *simple regression* (i.e, orthogonal projection) of the multiple-regression fitted values $\widehat{\mathbf{y}}^*$ on $\mathbf{x}_2^*$ yields the *simple-regression slope* (say, B) for the regression of Y on x_2 alone.

The lower panel of the diagram shows the 2D subspace generated by the residual vectors $\mathbf{e}$, $\mathbf{e}_y$, and $\mathbf{e}_1$, which lie in a common plane. As depicted, the orthogonal projection of $\mathbf{e}_y$ onto $\mathbf{e}_1$ forms a right triangle with $\mathbf{e}_y$ as the hypotenuse and $\mathbf{e}$ as one side. The squared length of $\mathbf{e}$ is the residual sum of squares for the multiple regression, labeled $\text{RSS}_{y|12}$. The squared length of $\mathbf{e}_y$ is the residual sum of squares for the regression of Y on X_2 alone, $\text{RSS}_{y|2}$. By the Pythagorean theorem, the squared length of the third side of the triangle, $\text{RSS}_{y|2} - \text{RSS}_{y|12}$, is the incremental sum of squares due to the introduction of X_1 into the regression after X_2.

(b)* The cosine of the angle w separating the residual vectors $\mathbf{e}_y$ and $\mathbf{e}_1$ is the partial correlation $r_{y1|2}$.

Thus, the squared partial correlation is

$$\begin{aligned} r^2_{y1|2} &= \cos^2 w \\ &= \frac{\text{RSS}_{y|2} - \text{RSS}_{y|12}}{\text{RSS}_{y|2}} \\ &= 1 - \frac{\text{RSS}_{y|12}}{\text{RSS}_{y|2}} \end{aligned}$$

We then have

$$\begin{aligned} F_0 &= \frac{(n-k-1)r^2_{y1|2}}{1 - r^2_{y1|2}} \\ &= \frac{(n-3)\left(1 - \dfrac{\text{RSS}_{y|12}}{\text{RSS}_{y|2}}\right)}{\dfrac{\text{RSS}_{y|12}}{\text{RSS}_{y|2}}} \end{aligned}$$

Multiplying the numerator and denominator by $\text{RSS}_{y|2}$ and dividing both by $n-3$ produces the usual form of the incremental F-test statistic for X_1 after X_2:

$$F_0 = \frac{\text{RSS}_{y2} - \text{RSS}_{y|12}}{\dfrac{\text{RSS}_{y|12}}{n-3}}$$

This result is intuitively plausible because (as shown in Exercise 5.8(b)), the partial correlation $r_{y1|2}$ is nonzero only when the multiple regression coefficient B_1 of X_1 is nonzero.

Exercise 10.11*

We have

$$\begin{aligned} \underset{(n-k-1\times 1)}{\mathbf{z}} &= \underset{(n-k-1\times n)}{\mathbf{G}}\underset{(n\times 1)}{\mathbf{e}} \\ \mathbf{G}\mathbf{G}' &= \mathbf{I}_{n-k-1)} \quad (\mathbf{G} \text{ is orthonormal}) \\ \mathbf{G}\mathbf{X} &= \underset{(n-k-1\times k+1)}{\mathbf{0}} \quad (\mathbf{G} \text{ is orthogonal to } \mathbf{X}) \end{aligned}$$

It's simple to establish the properties in the exercise: First,

$$\begin{aligned} \mathbf{z} &= \mathbf{G}(\mathbf{y} - \mathbf{X}\mathbf{b}) \\ &= \mathbf{G}\mathbf{y} - \mathbf{G}\mathbf{X}\mathbf{b} \\ &= \mathbf{G}\mathbf{y} - \mathbf{0}\mathbf{b} \\ &= \mathbf{G}\mathbf{y} \end{aligned}$$

Next,

$$\begin{aligned} \underset{(n-k-1\times 1)}{E(\mathbf{z})} &= \underset{(n-k-1\times n)}{\mathbf{G}}\underset{(n\times 1)}{E(\mathbf{e})} \\ &= \mathbf{G}\mathbf{0} \\ &= \underset{(n-k-1\times 1)}{\mathbf{0}} \end{aligned}$$

And finally,

$$\begin{aligned}
\underset{(n-k-1\times n-k-1)}{V(\mathbf{z})} &= \underset{(n-k-1\times n)}{\mathbf{G}}\underset{(n\times n)}{V(\mathbf{e})}\underset{(n\times n-k-1)}{\mathbf{G}'} \\
&= \mathbf{G}\sigma_\varepsilon^2\mathbf{I}_n\mathbf{G}' \\
&= \sigma_\varepsilon^2\mathbf{G}\mathbf{G}' \\
&= \sigma_\varepsilon^2\mathbf{I}_{n-k-1}
\end{aligned}$$

Exercises for Chapter 11

Exercise 11.1*

In simple regression, the sum of squares and products matrix for X is

$$\mathbf{X}'\mathbf{X} = \begin{bmatrix} n & \sum X_j \\ \sum X_j & \sum X_j^2 \end{bmatrix}$$

with inverse

$$\begin{aligned}
(\mathbf{X}'\mathbf{X})^{-1} &= \frac{1}{n\sum X_j^2 - (\sum X_j)^2} \begin{bmatrix} \sum X_j^2 & -\sum X_j \\ -\sum X_j & n \end{bmatrix} \\
&= \frac{1}{n\sum X_j^2 - (\sum X_j)^2} \begin{bmatrix} \sum (X_j - \bar{X})^2 - n\bar{X}^2 & -n\bar{X} \\ -n\bar{X} & n \end{bmatrix} \\
&= \begin{bmatrix} \frac{1}{n} + \frac{\bar{X}^2}{\sum (X_j - \bar{X})^2} & -\frac{\bar{X}}{\sum (X_j - \bar{X})^2} \\ -\frac{\bar{X}}{\sum (X_j - \bar{X})^2} & \frac{1}{\sum (X_j - \bar{X})^2} \end{bmatrix}
\end{aligned}$$

Then

$$\begin{aligned}
\mathbf{x}_i'(\mathbf{X}'\mathbf{X})^{-1}\mathbf{x}_i &= [1, X_i] \begin{bmatrix} \frac{1}{n} + \frac{\bar{X}^2}{\sum (X_j - \bar{X})^2} & -\frac{\bar{X}}{\sum (X_j - \bar{X})^2} \\ -\frac{\bar{X}}{\sum (X_j - \bar{X})^2} & \frac{1}{\sum (X_j - \bar{X})^2} \end{bmatrix} \begin{bmatrix} 1 \\ X_i \end{bmatrix} \\
&= \frac{1}{n} + \frac{\bar{X}^2}{\sum (X_j - \bar{X})^2} - 2\frac{X_i\bar{X}}{n\sum (X_j - \bar{X})^2} + \frac{X_i^2}{\sum (X_j - \bar{X})^2} \\
&= \frac{1}{n} + \frac{\bar{X}^2 - 2X_i\bar{X} + X_i^2}{\sum (X_j - \bar{X})^2} \\
&= \frac{1}{n} + \frac{(X_i - \bar{X})^2}{\sum (X_j - \bar{X})^2}
\end{aligned}$$

Exercise 11.3*

The $\mathbf{X}$ matrix for one-way ANOVA is the full-rank model matrix $\mathbf{X}_F$ in Equation 9.4 on page 205. Labeling the rows and columns of the sum of squares and product matrix by the parameters to which they pertain, we have

$$\mathbf{X}_F'\mathbf{X}_F = \begin{bmatrix} & (\mu) & (\alpha_1) & (\alpha_2) & \cdots & (\alpha_{m-1}) \\ (\mu) & n & 0 & 0 & \cdots & 0 \\ (\alpha_1) & 0 & 2n' & n' & \cdots & n' \\ (\alpha_2) & 0 & n' & 2n' & \cdots & n' \\ \vdots & & & & & \\ (\alpha_{m-1}) & 0 & n' & n' & \cdots & 2n' \end{bmatrix}$$

Here, there are $n' \equiv n/m$ observations in each of the m groups.

Because there are equal numbers of observations in the groups, the columns in the model matrix $\mathbf{X}_F$, after the first column for the general mean μ, all sum to 0 and hence have 0 means, so if we subtract the mean from Y, all of the variables are in mean-deviation form, and the intercept is removed from the

model. The sum of squares and products corresponding to the mean-deviation model matrix $\mathbf{X}_F^*$ is thus

$$\mathbf{X}_F^{*'}\mathbf{X}_F^* = \begin{bmatrix} & (\alpha_1) & (\alpha_2) & \cdots & (\alpha_{m-1}) \\ (\alpha_1) & 2n' & n' & \cdots & n' \\ (\alpha_2) & n' & 2n' & \cdots & n' \\ \vdots & & & & \\ (\alpha_{m-1}) & n' & n' & \cdots & 2n' \end{bmatrix}$$
$$= n' \begin{bmatrix} & (\alpha_1) & (\alpha_2) & \cdots & (\alpha_{m-1}) \\ (\alpha_1) & 2 & 1 & \cdots & 1 \\ (\alpha_2) & 1 & 2 & \cdots & 1 \\ \vdots & & & & \\ (\alpha_{m-1}) & 1 & 1 & \cdots & 2 \end{bmatrix}$$

The inverse of $\mathbf{X}_F^{*'}\mathbf{X}_F^*$ is of the form

$$(\mathbf{X}_F^{*'}\mathbf{X}_F^*)^{-1} = \begin{bmatrix} & (\alpha_1) & (\alpha_2) & \cdots & (\alpha_{m-1}) \\ (\alpha_1) & a & b & \cdots & b \\ (\alpha_2) & b & a & \cdots & b \\ \vdots & & & & \\ (\alpha_{m-1}) & b & b & \cdots & a \end{bmatrix}$$

that is, with equal diagonal elements (say a) and equal off-diagonal elements (say b), where the values of a and b need not concern us.

We know, moreover (from page 290), that the hat-values for the original model $h_i = 1/n + h_i^*$, where the h_i^* are the hat-values for the model in mean-deviation form, and so if the latter are all equal, so are the former.

Now consider an observation i in the first group, for which $\mathbf{x}_i' = [1, 1, 0, \ldots, 0]$ and $\mathbf{x}_i'^* = [1, 0, \ldots, 0]$; then

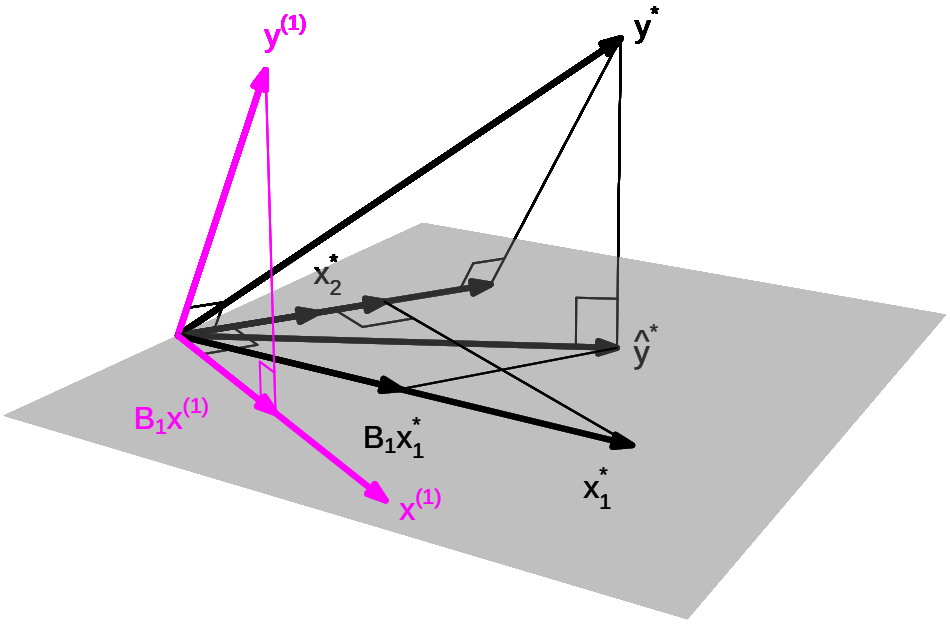
$$h_i^* = \mathbf{x}_i'^*(\mathbf{X}_F^{*'}\mathbf{X}_F^*)^{-1}\mathbf{x}_i^* = a$$

This pattern reproduces itself for observations in all the other groups—that is, for an observation i in any group, $h_i^* = a$—and thus all of the hatvalues are equal.

Perhaps the simplest way to see that this result applies to balanced (i.e., equal-cell-frequencies) ANOVA with any number of factors is to recall that the fitted values $\widehat{Y}$ for a two-way or higher-way ANOVA are just the cell means, and so, for example, a two-way ANOVA model is equivalent to the *cell-means model* $Y_{ijk} = \mu_{jk} + \varepsilon_{ijk}$, which, in turn is equivalent to a one-way ANOVA over the cells—in effect, "raveling" the two-way (or higher-way) table of means into a vector.

Exercise 11.5*

The following graph shows the vector geometry of the AV plot for X_1 in the regression of Y on X_1 and X_2:



The two residual vectors $\mathbf{y}^{(1)}$ and $\mathbf{x}^{(1)}$, respectively for the regressions of each of $\mathbf{y}$ and $\mathbf{x}_1$ on $\mathbf{x}_2$, are drawn in magenta. Embarrassingly, I don't see a direct geometric proof that the regression coefficient implied by the orthogonal projection of $\mathbf{y}^{(1)}$ onto $\mathbf{x}^{(1)}$ (the magenta vector $B_1\mathbf{x}^{(1)}$ in the diagram) is the multiple-regression slope B_1. Here, however, is a mostly algebraic proof, which uses some geometric ideas, and that has the advantage of generalizing to the next exercise:

First, form the normal equations for the regression of $\mathbf{y}^*$ on $\mathbf{x}_1^*$ and $\mathbf{x}_2^*$:

$$\begin{aligned}\mathbf{x}_1^{*\prime}\mathbf{x}_1^* B_1 + \mathbf{x}_1^{*\prime}\mathbf{x}_2^* B_2 &= \mathbf{x}_1^{*\prime}\mathbf{y}^* \\ \mathbf{x}_2^{*\prime}\mathbf{x}_1^* B_1 + \mathbf{x}_2^{*\prime}\mathbf{x}_2^* B_2 &= \mathbf{x}_2^{*\prime}\mathbf{y}^*\end{aligned}$$

Then solve the second normal equation for B_2:

$$B_2 = (\mathbf{x}_2^{*\prime}\mathbf{x}_2^*)^{-1}\mathbf{x}_2^{*\prime}\mathbf{y}^* - (\mathbf{x}_2^{*\prime}\mathbf{x}_2^*)^{-1}\mathbf{x}_2^{*\prime}\mathbf{x}_1^* B_1$$

Next, substitute for B_2 in the first normal equation:

$$\mathbf{x}_1^{*\prime}\mathbf{x}_1^* B_1 + \mathbf{x}_1^{*\prime}\mathbf{x}_2^*[(\mathbf{x}_2^{*\prime}\mathbf{x}_2^*)^{-1}\mathbf{x}_2^{*\prime}\mathbf{y}^* - (\mathbf{x}_2^{*\prime}\mathbf{x}_2^*)^{-1}\mathbf{x}_2^{*\prime}\mathbf{x}_1^* B_1] = \mathbf{x}_1^{*\prime}\mathbf{y}^*$$

After some rearrangement isolating B_1 on the left-hand side, this equation becomes

$$\mathbf{x}_1^{*\prime}[\mathbf{I}_n - \mathbf{x}_2^*(\mathbf{x}_2^{*\prime}\mathbf{x}_2^*)^{-1}\mathbf{x}_2^{*\prime}]\mathbf{x}_1^* B_1 = \mathbf{x}_1^{*\prime}[\mathbf{I}_n - \mathbf{x}_2^*(\mathbf{x}_2^{*\prime}\mathbf{x}_2^*)^{-1}\mathbf{x}_2^{*\prime}]\mathbf{y}^*$$

The matrix $\mathbf{H}_2^* \equiv \mathbf{x}_2^*(\mathbf{x}_2^{*\prime}\mathbf{x}_2^*)^{-1}\mathbf{x}_2^{*\prime}$ is the hat-matrix for the regression (i.e., orthogonal projection) of a mean-deviation variable (here $\mathbf{x}_1^*$ or $\mathbf{y}^*$) on $\mathbf{x}_2^*$. We can therefore re-express the last equation as

$$\mathbf{x}_1^{*\prime}(\mathbf{I}_n - \mathbf{H}_2^*)\mathbf{x}_1^* B_1 = \mathbf{x}_1^{*\prime}(\mathbf{I}_n - \mathbf{H}_2^*)\mathbf{y}^*$$

and so

$$B_1 = \frac{\mathbf{x}_1^{*\prime}(\mathbf{I}_n - \mathbf{H}_2^*)\mathbf{y}^*}{\mathbf{x}_1^{*\prime}(\mathbf{I}_n - \mathbf{H}_2^*)\mathbf{x}_1^*}$$

Similarly, $\mathbf{I}_n - \mathbf{H}_2^*$ projects a mean-deviation vector (again, here $\mathbf{x}_1^*$ or $\mathbf{y}^*$) onto the subspace of the $(n-1)$-dimensional mean-deviation vector space orthogonal to $\mathbf{x}_2^*$ and so produces residuals from a

regression on $\mathbf{x}_2^*$. Using the notation of the exercise, therefore, $\mathbf{y}^{(1)} = (\mathbf{I}_n - \mathbf{H}_2^*)\mathbf{y}^*$ (the residuals from the regression of $\mathbf{y}^*$ on $\mathbf{x}_2^*$), and $\mathbf{x}^{(1)} = (\mathbf{I}_n - \mathbf{H}_2^*)\mathbf{x}_1^*$ (the residuals from the regression of $\mathbf{x}_1^*$ on $\mathbf{x}_2^*$).

Thus,

$$\begin{aligned} B_1 &= \frac{\mathbf{x}_1^{*\prime}\mathbf{y}^{(1)}}{\mathbf{x}_1^{*\prime}\mathbf{x}^{(1)}} \\ &= \frac{(\widehat{\mathbf{x}}_1^{*\prime} + \mathbf{x}^{(1)\prime})\mathbf{y}^{(1)}}{(\widehat{\mathbf{x}}_1^{*\prime} + \mathbf{x}^{(1)\prime})\mathbf{x}^{(1)}} \\ &= \frac{\mathbf{x}^{(1)\prime}\mathbf{y}^{(1)}}{\mathbf{x}^{(1)\prime}\mathbf{x}^{(1)}} \end{aligned}$$

In the second line of the equation, the vector $\widehat{\mathbf{x}}_1^*$ represents the fitted values from the regression of $\mathbf{x}_1^*$ on $\mathbf{x}_2^*$; as a consequence $\widehat{\mathbf{x}}_1^*$ lies in the subspace spanned by $\mathbf{x}_2^*$, and it is orthogonal to the residual vectors $\mathbf{y}^{(1)}$ and $\mathbf{x}^{(1)}$, justifying its elimination from the last line of the equation (because $\widehat{\mathbf{x}}_1^{*\prime}\mathbf{y}^{(1)}$ and $\widehat{\mathbf{x}}_1^{*\prime}\mathbf{x}^{(1)}$ are both 0).

Finally, we recognize that the last line of the equation is the slope coefficient for the regression of $\mathbf{y}^{(1)}$ on $\mathbf{x}^{(1)}$, completing the proof.

Exercise 11.7*

We have from the text (on page 292) that

$$\mathbf{e}_0 = \mathbf{e} + \mathbf{X}(\mathbf{X}'\mathbf{X})^{-1}\mathbf{L}'\mathbf{u}$$

where

$$\mathbf{u} = [\mathbf{L}(\mathbf{X}'\mathbf{X})^{-1}\mathbf{L}']^{-1}\mathbf{L}\mathbf{b}$$

Note: The left square bracket is missing in the equation for $\mathbf{u}$ in the text (perhaps corrected in your printing of the book).

Consequently,

$$\mathbf{e}_0 - \mathbf{e} = \mathbf{X}(\mathbf{X}'\mathbf{X})^{-1}\mathbf{L}'[\mathbf{L}(\mathbf{X}'\mathbf{X})^{-1}\mathbf{L}']^{-1}\mathbf{L}\mathbf{b}$$

and

$$\begin{aligned} ||\mathbf{e}_0 - \mathbf{e}||^2 &= (\mathbf{e}_0 - \mathbf{e})'(\mathbf{e}_0 - \mathbf{e}) \\ &= \{\mathbf{X}(\mathbf{X}'\mathbf{X})^{-1}\mathbf{L}'[\mathbf{L}(\mathbf{X}'\mathbf{X})^{-1}\mathbf{L}']^{-1}\mathbf{L}\mathbf{b}\}'\mathbf{X}(\mathbf{X}'\mathbf{X})^{-1}\mathbf{L}'[\mathbf{L}(\mathbf{X}'\mathbf{X})^{-1}\mathbf{L}']^{-1}\mathbf{L}\mathbf{b} \\ &= \mathbf{b}'\mathbf{L}'[\mathbf{L}(\mathbf{X}'\mathbf{X})^{-1}\mathbf{L}']^{-1}\mathbf{L}(\mathbf{X}'\mathbf{X})^{-1}\mathbf{X}'\mathbf{X}(\mathbf{X}'\mathbf{X})^{-1}\mathbf{L}'[\mathbf{L}(\mathbf{X}'\mathbf{X})^{-1}\mathbf{L}']^{-1}\mathbf{L}\mathbf{b} \\ &= \mathbf{b}'\mathbf{L}'[\mathbf{L}(\mathbf{X}'\mathbf{X})^{-1}\mathbf{L}']^{-1}\mathbf{L}(\mathbf{X}'\mathbf{X})^{-1}\mathbf{I}_{k+1}\mathbf{L}'[\mathbf{L}(\mathbf{X}'\mathbf{X})^{-1}\mathbf{L}']^{-1}\mathbf{L}\mathbf{b} \\ &= \mathbf{b}'\mathbf{L}'[\mathbf{L}(\mathbf{X}'\mathbf{X})^{-1}\mathbf{L}']^{-1}\mathbf{I}_q\mathbf{L}\mathbf{b} \\ &= \mathbf{b}'\mathbf{L}'[\mathbf{L}(\mathbf{X}'\mathbf{X})^{-1}\mathbf{L}']^{-1}\mathbf{L}\mathbf{b} \end{aligned}$$

Exercises for Chapter 12

Exercise 12.1*

Consider Figure 10.8 on page 253. We know that $R^2 = r^2_{\mathbf{y}*,\widehat{\mathbf{y}}*} = \cos^2 W$. Similarly the squared correlation between the residuals and the response is $r^2_{\mathbf{e},\mathbf{y}*} = \cos^2(90° - W) = \sin^2 W$. Because $\cos^2 W + \sin^2 W = 1$, it follows that $r^2_{\mathbf{e},\mathbf{y}*} = 1 - R^2$ and that $r_{\mathbf{e},\mathbf{y}*} = \sqrt{1 - R^2}$.

Exercise 12.3*

Note: There are some corrections to this exercise in the errata for the text; these errors may be fixed in your printing of the book.

(a) The (correct) likelihood

$$L(\boldsymbol{\Sigma}, \sigma^2_\varepsilon) = \frac{1}{(2\pi)^{n/2}|\boldsymbol{\Sigma}|^{1/2}} \exp\left[-\frac{1}{2}(\mathbf{y} - \mathbf{X}\boldsymbol{\beta})'\boldsymbol{\Sigma}^{-1}(\mathbf{y} - \mathbf{X}\boldsymbol{\beta})\right]$$

follows directly from the observation that according to the regression model, $\mathbf{y}$ is multivariately normally distributed with mean vector $\mathbf{X}\boldsymbol{\beta}$ and covariance matrix $\boldsymbol{\Sigma}$. (See on-line Appendix D, Section D.3.5, for the formula of the multivariate-normal distribution.)

(b) We'll find the following facts useful:

- As stated in the problem, $\boldsymbol{\Sigma} = \sigma^2_\varepsilon \times \text{diag}\{1/w_1^2, \ldots, 1/w_n^2\} \equiv \sigma^2_\varepsilon \mathbf{W}^{-1}$.
- Thus $\boldsymbol{\Sigma}^{-1} = \frac{1}{\sigma^2_\varepsilon}\mathbf{W}$.
- Because $\boldsymbol{\Sigma}$ is diagonal, its determinant is simply the product of its diagonal elements, $|\boldsymbol{\Sigma}| = \prod_{i=1}^n \frac{\sigma^2_\varepsilon}{w_i^2}$, and, consequently, $\log_e |\boldsymbol{\Sigma}| = \sum_{i=1}^n \log_e \frac{\sigma^2_\varepsilon}{w_i^2}$. Although I could simplify the expression for the log-determinant, it will be helpful to have it in this form.

  The log-likelihood is

$$\begin{aligned}\log_e L(\boldsymbol{\beta}, \boldsymbol{\Sigma}) &= -\frac{n}{2}\log_e 2\pi - \frac{1}{2}\log_e |\boldsymbol{\Sigma}| - \frac{1}{2}(\mathbf{y} - \mathbf{X}\boldsymbol{\beta})'\boldsymbol{\Sigma}^{-1}(\mathbf{y} - \mathbf{X}\boldsymbol{\beta}) \\ &= -\frac{n}{2}\log_e 2\pi - \frac{1}{2}\sum_{i=1}^n \log_e \frac{\sigma^2_\varepsilon}{w_i^2} - \frac{1}{2\sigma^2_\varepsilon}(\mathbf{y} - \mathbf{X}\boldsymbol{\beta})'\mathbf{W}(\mathbf{y} - \mathbf{X}\boldsymbol{\beta})\end{aligned}$$

  I'll next differentiate the log-likelihood with respect to the regression coefficients $\boldsymbol{\beta}$:

$$\frac{\partial \log_e L(\boldsymbol{\beta}, \boldsymbol{\Sigma})}{\partial \boldsymbol{\beta}} = -2\left(-\frac{1}{2\sigma^2_\varepsilon}\right)(\mathbf{X}'\mathbf{W}\mathbf{X}\boldsymbol{\beta} - \mathbf{X}'\mathbf{W}\mathbf{y})$$

  Setting the vector partial derivative to $\mathbf{0}$ and dividing by σ^2_ε yields

$$\begin{aligned}\mathbf{X}'\mathbf{W}\mathbf{X}\widehat{\boldsymbol{\beta}} - \mathbf{X}'\mathbf{W}\mathbf{y} &= \mathbf{0} \\ \mathbf{X}'\mathbf{W}\mathbf{X}\widehat{\boldsymbol{\beta}} &= \mathbf{X}'\mathbf{W}\mathbf{y} \\ \widehat{\boldsymbol{\beta}} &= (\mathbf{X}'\mathbf{W}\mathbf{X})^{-1}\mathbf{X}'\mathbf{W}\mathbf{y}\end{aligned}$$

  as long as the weighted sum-of-squares-and-products matrix $\mathbf{X}'\mathbf{W}\mathbf{X}$ is nonsingular. The MLE of $\boldsymbol{\beta}$ is therefore the weighted-least-squares estimator.

  Now that $\widehat{\boldsymbol{\beta}}$ is known, we can write the last term in the maximized log-likelihood as

$$\begin{aligned}-\frac{1}{2\sigma^2_\varepsilon}(\mathbf{y} - \mathbf{X}\widehat{\boldsymbol{\beta}})'\mathbf{W}(\mathbf{y} - \mathbf{X}\widehat{\boldsymbol{\beta}}) &= -\frac{1}{2\sigma^2_\varepsilon}\mathbf{e}'\mathbf{W}\mathbf{e} \\ &= -\frac{1}{2\sigma^2_\varepsilon}\sum_{i=1}^n e_i^2 w_i^2\end{aligned}$$

where $\mathbf{e} \equiv \mathbf{y} - \mathbf{X}\widehat{\boldsymbol{\beta}}$ is the residual vector and the transition to the second line of the equation is justified by remembering the $\mathbf{W}$ is diagonal with diagonal elements w_i^2.

I next differentiate the rewritten log-likelihood with respect to σ_ε^2:

$$\frac{d \log_e L}{d\sigma_\varepsilon^2} = -\frac{n}{2\sigma_\varepsilon^2} + \frac{\sum_{i=1}^n e_i^2 w_i^2}{2(\sigma_\varepsilon^2)^2}$$

Setting the derivative to 0 and solving for $\widehat{\sigma}_\varepsilon^2$ produces the (correct) result, $\widehat{\sigma}_\varepsilon^2 = (\sum_{i=1}^n e_i^2 w_i^2)/n$.

(c) The only term in the log-likelihood that contributes to the MLE of $\boldsymbol{\beta}$ is the third term. Moreover, as we've seen, we can get rid of the error variance σ_ε^2 in the denominator of this term after setting the vector partial derivative with respect to $\boldsymbol{\beta}$ to $\mathbf{0}$, and we also already showed that the rest of the third term is $\sum w_i^2 e_i^2$. Thus, the MLE $\widehat{\boldsymbol{\beta}}$ minimizes the weighted sum of squares.

(d) We have $\widehat{\boldsymbol{\beta}} = (\mathbf{X}'\mathbf{W}\mathbf{X})^{-1}\mathbf{X}'\mathbf{W}\mathbf{y}$ and so, taking advantage of the symmetry of $\mathbf{W}$ and of $(\mathbf{X}'\mathbf{W}\mathbf{X})^{-1}$,

$$\begin{aligned} V(\widehat{\boldsymbol{\beta}}) &= (\mathbf{X}'\mathbf{W}\mathbf{X})^{-1}\mathbf{X}'\mathbf{W}V(\mathbf{y})[(\mathbf{X}'\mathbf{W}\mathbf{X})^{-1}\mathbf{X}'\mathbf{W}]' \\ &= (\mathbf{X}'\mathbf{W}\mathbf{X})^{-1}\mathbf{X}'\mathbf{W}\sigma_\varepsilon^2\mathbf{W}^{-1}\mathbf{W}\mathbf{X}(\mathbf{X}'\mathbf{W}\mathbf{X})^{-1} \\ &= \sigma_\varepsilon^2(\mathbf{X}'\mathbf{W}\mathbf{X})^{-1} \end{aligned}$$

The estimated asymptotic covariance matrix $\widehat{\mathcal{V}}(\widehat{\boldsymbol{\beta}})$ follows from substituting the MLE of the error variance $\widehat{\sigma}_\varepsilon^2$ for σ_ε^2.

Exercise 12.5*

(a) As explained in Section 6.1.2 (page 109), the OLS estimator of β is

$$\begin{aligned} B &= \sum_{i=1}^n m_i Y_i \\ m_i &= \frac{x_i - \bar{x}}{\sum_{j=1}^n (x_j - \bar{x})^2} \end{aligned}$$

so

$$\begin{aligned} V(B) &= \sum_{i=1}^n m_i^2 V(Y_i) \\ &= \sum_{i=1}^n \left[\frac{x_i - \bar{x}}{\sum_{j=1}^n (x_j - \bar{x})^2}\right]^2 \sigma_i^2 \\ &= \frac{\sum_{i=1}^n (x_i - \bar{x})^2 \sigma_i^2}{\sum_{i=1}^n (x_i - \bar{x})^2} \end{aligned}$$

Note: The question is slightly inconsistent in using x_i (i.e., fixed xs) in the model and X_i (random Xs) in the formula for $V(B)$. Here, I use x_i consistently.

Deriving the analogous formula for the WLS estimator $\widehat{\beta}$ is more tedious. One approach is to specialize $V(\widehat{\boldsymbol{\beta}}) = \sigma_\varepsilon^2(\mathbf{X}'\mathbf{W}\mathbf{X})^{-1}$ to simple regression. We have

$$\begin{aligned} \mathbf{X}'\mathbf{W}\mathbf{X} &= \begin{bmatrix} 1 & \cdots & 1 \\ x_1 & \cdots & x_n \end{bmatrix} \begin{bmatrix} w_1^2 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & w_n^2 \end{bmatrix} \begin{bmatrix} 1 & x_1 \\ \vdots & \vdots \\ 1 & x_n \end{bmatrix} \\ &= \begin{bmatrix} \sum w_i^2 & \sum w_i^2 x_i \\ \sum w_i^2 x_i & \sum w_i^2 x_i^2 \end{bmatrix} \end{aligned}$$

To find $V(\widehat{\beta})$ we need the second diagonal element of $(\mathbf{X}'\mathbf{W}\mathbf{X})^{-1}$, which is

$$(\mathbf{X}'\mathbf{W}\mathbf{X})^{-1}_{22} = \frac{\sum w_i^2}{(\sum w_i^2)(\sum w_i^2 x_i^2) - (\sum w_i^2 x_i)^2}$$

and so

$$\begin{aligned} V(\widehat{\beta}) &= \frac{\sigma_\varepsilon^2 \sum w_i^2}{(\sum w_i^2)(\sum w_i^2 x_i^2) - (\sum w_i^2 x_i)^2} \\ &= \frac{\sigma_\varepsilon^2}{\sum w_i^2 x_i^2 - \sum w_i^2 \left(\dfrac{\sum w_i^2 x_i}{\sum w_i^2}\right)^2} \end{aligned}$$

To complete the proof, I need to show that the denominator on the right-hand side of the last line of the equation is $\sum w_i^2 (x_i - \widetilde{x})^2$. Expanding the latter,

$$\begin{aligned} \sum w_i^2 (x_i - \widetilde{x})^2 &= \sum w_i^2 x_i^2 - 2\widetilde{x} \sum w_i^2 x_i + \widetilde{x}^2 \sum w_i^2 \\ &= \sum w_i^2 x_i^2 - 2\widetilde{x}^2 \sum w_i^2 + \widetilde{x}^2 \sum w_i^2 \\ &= \sum w_i^2 x_i^2 - \widetilde{x}^2 \sum w_i^2 \\ &= \sum w_i^2 x_i^2 - \left(\frac{\sum w_i^2 x_i}{\sum w_i^2}\right)^2 \sum w_i^2 \end{aligned}$$

(b) The simplest approach is to set the common error standard deviation parameter $\sigma = 1$, so that $\sigma_i = x_i$. I wrote a simple R program to compute the relative precision of the OLS estimator and then applied it to the various combinations of a and n values:

```
relPrecision <- function(a, n){
    x <- seq(1, a, length=n)
    sigma.sq <- x^2
    xbar <- mean(x)
    xtilde <- weighted.mean(x, x^2)
    vols <- sum(sigma.sq*(x - xbar)^2)/((sum((x - xbar)^2))^2)
    vwls <- 1/(sum((x^2)*(x - xtilde)^2))
    sqrt(vwls/vols)
}

as <- as.integer(c(2, 3, 5, 10))
ns <- as.integer(c(5, 10, 20, 50, 100))
rel.prec <- matrix(0, length(as), length(ns))
rownames(rel.prec) <- as.character(as)
colnames(rel.prec) <- as.character(ns)

for (a in as){
    for (n in ns){
        rel.prec[as.character(a), as.character(n)] <- relPrecision(a, n)
    }
}
```

The results, saved in `rel.prec`, are as follows:

	n				
a	5	10	20	50	100
2	0.452	0.449	0.448	0.448	0.448
3	0.259	0.255	0.255	0.254	0.254
5	0.117	0.115	0.114	0.114	0.114
10	0.035	0.034	0.034	0.034	0.034

The relative precision of the OLS estimator declines slightly, but only slightly, as sample size grows, but it declines dramatically as the ratio of the largest to smallest error standard deviation grows, from a little less than half when this ratio is 2 to less than 4% when the ratio is 10.

(c) First, the numeric results: As before, I wrote a simple R program to do the calculations and then applied it to the various combinations of ratios of largest to smallest error standard deviations and sample sizes:

```
relBias <- function(a, n){
    x <- seq(1, a, length=n)
    sigma.sq <- x^2
    sigma.sq.bar <- mean(sigma.sq)
    xbar <- mean(x)
    xtilde <- weighted.mean(x, x^2)
    true.var.ols <- sum(sigma.sq*(x - xbar)^2)/
        ((sum((x - xbar)^2))^2)
    E.usual.var.ols <- (sigma.sq.bar/sum((x - xbar)^2)) -
        (sum((x - xbar)^2*(sigma.sq - sigma.sq.bar)))/
        ((n - 2)*(sum((x - xbar)^2)^2))
    sqrt(E.usual.var.ols/true.var.ols)
}

rel.bias <- matrix(0, length(as), length(ns))
rownames(rel.bias) <- as.character(as)
colnames(rel.bias) <- as.character(ns)

for (a in as){
    for (n in ns){
        rel.bias[as.character(a), as.character(n)] <-  relBias(a, n)
    }
}
```

I called the program `relBias()` because the focus is on the relative bias of the usual OLS coefficient standard error. The results, saved in `rel.bias` are as follows:

	n				
a	5	10	20	50	100
2	0.976	0.982	0.984	0.985	0.986
3	0.951	0.962	0.966	0.969	0.970
5	0.922	0.938	0.945	0.949	0.951
10	0.895	0.916	0.926	0.931	0.932

Thus, the downwards bias (i.e., exaggerated precision) of the usual OLS standard error is greatest at small samples sizes and large ratios of biggest to smallest error variances, but under the conditions of this problem, the relative bias never gets very large.

Combining the results of parts (b) and (c) of the problem, under these circumstances, the robustness of efficiency of the OLS estimator is sensitive to non-constant error variance but its robustness of validity is not.

Now let's return to the formula for $E[\widehat{V}(B)]$. In stating the exercise, and in the R code given above, I used a result from Kmenta (1986, page 276–278). His derivation is fairly lengthy and dense, and rather than recapitulate it here, I'll instead derive a simpler, equivalent, formula.

The standard formula for the estimated variance of the simple-regression slope B, assuming

constant error variance, is (see Section 6.1.3, page 111)

$$\begin{aligned}\widehat{V}(B) &= \frac{S_E^2}{\sum(x_i - \bar{x})^2} \\ &= \frac{\sum E_i^2}{(n-2)\sum(x_i - \bar{x})^2}\end{aligned}$$

This estimator is unbiased when the error variance is in fact constant, but we now require the expectation of $\widehat{V}(B)$ when the *error variances differ*. First, because E_i has an expectation of 0, $E\left(E_i^2\right) = V(E_i)$. Next, the covariance matrix of the OLS residuals is

$$\begin{aligned}V(\mathbf{e}) &= (\mathbf{I}_n - \mathbf{H})\mathbf{\Sigma}(\mathbf{I}_n - \mathbf{H}) \\ &= \mathbf{Q}\mathbf{\Sigma}\mathbf{Q}\end{aligned}$$

where $\mathbf{\Sigma} = \text{diag}\left\{\sigma_i^2\right\}$ and $\mathbf{Q} \equiv \mathbf{I}_n - \mathbf{H}$, like $\mathbf{H}$, is symmetric and idempotent. Focus now on the ith diagonal entry of $V(\mathbf{e})$, that is, $V(E_i)$, and let $\mathbf{q}_i'$ represent the ith row of $\mathbf{Q}$, and hence $\mathbf{q}_i$ is its ith column:

$$\begin{aligned}V(E_i) &= \mathbf{q}_i' \text{diag}\left\{\sigma_1^2, \ldots, \sigma_i^2, \ldots, \sigma_n^2\right\} \mathbf{q}_i \\ &= \sum_{j=1}^{n} q_{ij}^2 \sigma_j^2\end{aligned}$$

Then

$$\begin{aligned}\sum_{i=1}^{n} V(E_i) &= \sum_{i=1}^{n}\sum_{j=1}^{n} q_{ij}^2 \sigma_j^2 \\ &= \sum_{j=1}^{n} q_{jj} \sigma_j^2\end{aligned}$$

because, by the symmetry and idempotency of $\mathbf{Q}$, $q_{jj} = \sum_{i=1}^{n} q_{ij}^2$. The expectation of the estimated variance of the OLS slope B is therefore

$$\begin{aligned}E[\widehat{V}(B)] &= \frac{\sum V(E_i)}{(n-2)\sum(x_i - \bar{x})^2} \\ &= \frac{\sum q_{ii}\sigma_i^2}{(n-2)\sum(x_i - \bar{x})^2} \\ &= \frac{\sum(1 - h_i)\sigma_i^2}{(n-2)\sum(x_i - \bar{x})^2}\end{aligned}$$

Recall, finally (from Section 11.2, page 2.70), that the hat-values in least-squares simple regression are

$$h_i = \frac{1}{n} + \frac{(x_i - \bar{x})^2}{\sum_{j=1}^{n}(x_j - \bar{x})^2}$$

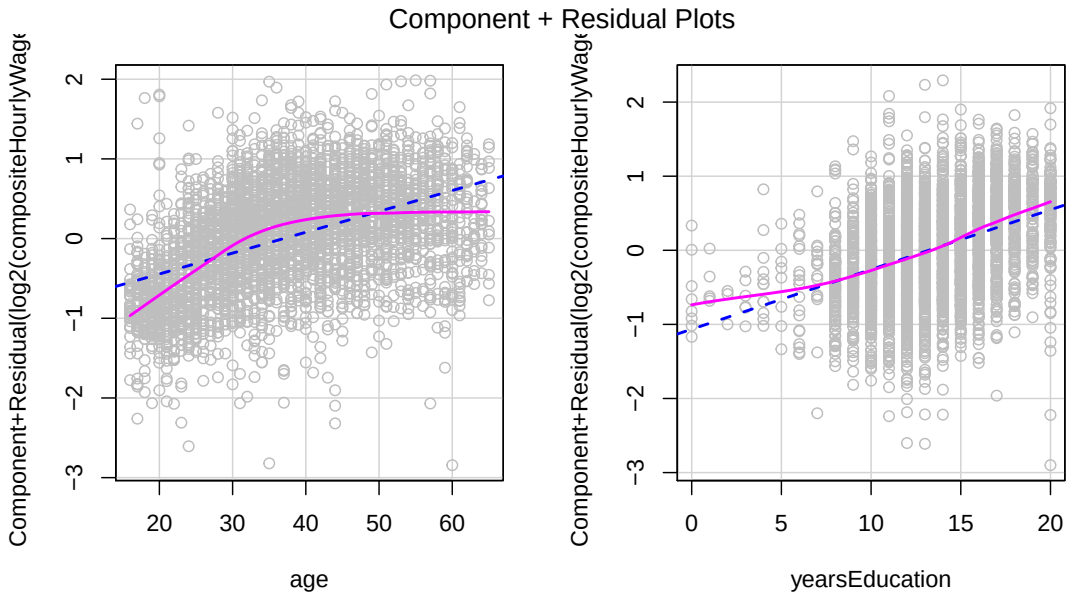
Exercise 12.7

I'll begin by duplicating the initial regression in the text, shown in Equation 12.2 (page 300) and the component-plus-residual plots in Figure 12.6 (page 310):

```
library("car") # for crPlots()

url <- paste("https://socialsciences.mcmaster.ca", "jfox", "Books",
```



```
            "Applied-Regression-3E", "datasets", "SLID-Ontario.txt",
            sep="/")
SLID <- read.table(url, header=TRUE)

m1 <- lm(log2(compositeHourlyWages) ~ sex + age + yearsEducation,
        data=SLID)
crPlots(m1, ~ age + yearsEducation, col="gray")
```

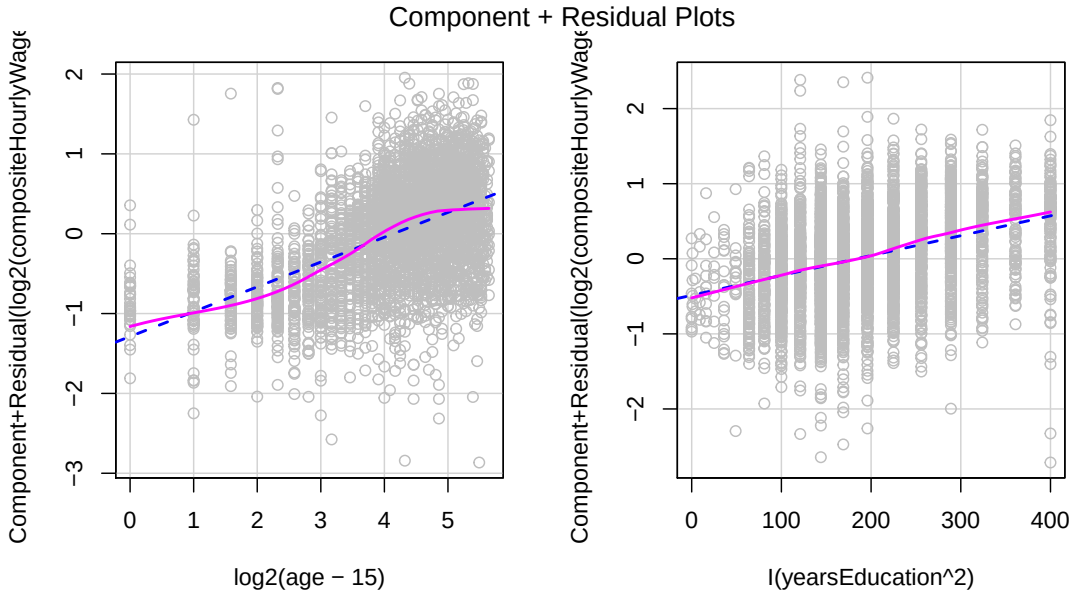


Squaring education does a reasonable job of straightening the partial relationship of log-wages to education, but the C+R plot for log(age − 15) reveals some lack of fit:

```
> m2 <- lm(log2(compositeHourlyWages) ~ sex + log2(age - 15)
+          + I(yearsEducation^2),
+          data=SLID)
> summary(m2)

Call:
lm(formula = log2(compositeHourlyWages) ~ sex + log2(age - 15) +
    I(yearsEducation^2), data = SLID)

Residuals:
    Min      1Q  Median      3Q     Max
-3.2855 -0.3611  0.0359  0.3746  2.5494

Coefficients:
                    Estimate Std. Error t value Pr(>|t|)
(Intercept)         1.847825   0.041403   44.63   <2e-16 ***
sexMale             0.318115   0.018243   17.44   <2e-16 ***
log2(age - 15)      0.310876   0.008267   37.61   <2e-16 ***
I(yearsEducation^2) 0.002646   0.000113   23.41   <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.5766 on 3993 degrees of freedom
Multiple R-squared:  0.3694,    Adjusted R-squared:  0.3689
F-statistic: 779.5 on 3 and 3993 DF,  p-value: < 2.2e-16
```



```
>
> crPlots(m2, ~ log2(age - 15)
+         + I(yearsEducation^2), col="gray")
```



Compare the C+R plot for age for this model with the corresponding C+R plot for quadratic model (lower-left of Figure 12.7 on page 312). The quadratic model, Equation 12.7 (page 310), however, has an extra parameter. One way to compare the two models is via the AIC or BIC (discussed in Section 22.1.1):

```
> # quadratic in age, see Equation 12.7
> m3 <- lm(log2(compositeHourlyWages) ~ sex + poly(age, 2, raw=TRUE)
+          + I(yearsEducation^2),
+          data=SLID)

> summary(m3)

Call:
lm(formula = log2(compositeHourlyWages) ~ sex + poly(age, 2,
    raw = TRUE) + I(yearsEducation^2), data = SLID)

Residuals:
     Min       1Q   Median       3Q      Max
-3.04688 -0.34263  0.02977  0.36354  2.56370

Coefficients:
                             Estimate Std. Error t value Pr(>|t|)
(Intercept)                 5.725e-01  8.338e-02   6.866 7.62e-12 ***
sexMale                     3.195e-01  1.796e-02  17.794  < 2e-16 ***
poly(age, 2, raw = TRUE)1   1.198e-01  4.598e-03  26.046  < 2e-16 ***
poly(age, 2, raw = TRUE)2 -1.230e-03  5.918e-05 -20.778  < 2e-16 ***
I(yearsEducation^2)         2.605e-03  1.135e-04  22.957  < 2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.5675 on 3992 degrees of freedom
Multiple R-squared:  0.3892,    Adjusted R-squared:  0.3886
```



```
F-statistic: 635.8 on 4 and 3992 DF,  p-value: < 2.2e-16

> AIC(m2)
[1] 6946.869

> AIC(m3)
[1] 6821.221

> BIC(m2)
[1] 6978.335

> BIC(m3)
[1] 6858.981
```

The quadratic model has a much lower AIC and BIC, indicating a better fit to the data, even after accounting for the extra parameter.

Exercise 12.9*

(a) The question suggests that you work from the distribution of the error vector $\boldsymbol{\varepsilon} = \mathbf{y} - \mathbf{X}\boldsymbol{\beta}$, but it's probably slightly more straightforward to work equivalently from the distribution of the response vector $\mathbf{y}$. The first part of the right-hand side of the equation given for $p(\mathbf{y})$ (and notice that $\mathbf{y}$ is a vector and so should be in boldface) is simply $p\left(\mathbf{y}^{(\lambda)}\right) = \prod_{i=1}^{n} p\left(y_i^{(\lambda)}\right)$ because the observations are independent, and because $Y_i^{(\lambda)} \sim N(\mathbf{x}_i'\boldsymbol{\beta}, \sigma_\varepsilon^2)$. That is, applying the formula for the normal distribution (see on-line Appendix D, Section D.3.1),

$$p\left(y_i^{(\lambda)}\right) = \frac{1}{\sigma_\varepsilon\sqrt{2\pi}} \exp\left[-\frac{\left(y_i^{(\lambda)} - \mathbf{x}_i'\boldsymbol{\beta}\right)^2}{2\sigma_\varepsilon^2}\right]$$

To get $p(y_i)$ from $p\left(y_i^{(\lambda)}\right)$, we need the Jacobian of the transformation,

$$\frac{dY_i^{(\lambda)}}{dY_i} = \frac{d\left(\dfrac{Y_i^\lambda - 1}{\lambda}\right)}{dY_i} = Y_i^{\lambda-1}$$

Then $p(y_i) = p\left(y_i^{(\lambda)}\right) Y_i^{\lambda-1}$ and $p(\mathbf{y}) = \prod_{i=1}^{n} p(y_i)$, which produces the desired result.

(b) We know that $Y_i^{(\lambda)} \sim N(\mathbf{x}_i'\boldsymbol{\beta}, \sigma_\varepsilon^2)$, so given the value of the transformation parameter λ, we can just perform an OLS regression of $Y^{(\lambda)}$ on the Xs to compute the MLE of $\boldsymbol{\beta}$.

To get the maximized log-likelihood conditional on λ, first use $p(\mathbf{y})$ given in part (a) of the exercise to get the general likelihood:

$$L(\boldsymbol{\beta}, \sigma_\varepsilon^2, \lambda) = (2\pi\sigma_\varepsilon^2)^{-n/2} \exp\left[-\frac{\sum\left(y_i^{(\lambda)} - \mathbf{x}_i'\boldsymbol{\beta}\right)^2}{2\sigma_\varepsilon^2}\right] \prod y_i^{\lambda-1}$$

Then, taking the log of L,

$$\log_e L(\boldsymbol{\beta}, \sigma_\varepsilon^2, \lambda) = -\frac{n}{2}\log_e(2\pi\sigma_\varepsilon^2) - \frac{\sum\left(y_i^{(\lambda)} - \mathbf{x}_i'\boldsymbol{\beta}\right)^2}{2\sigma_\varepsilon^2} + (\lambda - 1)\sum \log_e y_i$$

Now conditionally fix the value of λ. Then at the conditional MLE, $\widehat{\sigma}_\varepsilon^2(\lambda) = \sum E_i^2(\lambda)/n$ where the $E_i(\lambda) = y_i^{(\lambda)} - \mathbf{x}_i'\widehat{\boldsymbol{\beta}}(\lambda)$ are the least-squares residuals from the regression of $Y^{(\lambda)}$ on the Xs, and $\widehat{\boldsymbol{\beta}}(\lambda)$ is the estimated coefficient vector from this regression.

Finally, the maximized conditional log-likelihood is

$$\begin{aligned}\log_e L(\boldsymbol{\beta}, \sigma_\varepsilon^2|\lambda) &= -\frac{n}{2}\log_e[2\pi\widehat{\sigma}_\varepsilon^2(\lambda)] - \frac{\sum E_i^2(\lambda)}{2\widehat{\sigma}_\varepsilon^2(\lambda)} + (\lambda - 1)\sum \log_e Y_i \\ &= -\frac{n}{2}\log_e[2\pi\widehat{\sigma}_\varepsilon^2(\lambda)] - \frac{n\widehat{\sigma}_\varepsilon^2(\lambda)}{2\widehat{\sigma}_\varepsilon^2(\lambda)} + (\lambda - 1)\sum \log_e Y_i \\ &= -\frac{n}{2}(1 + \log_e 2\pi) - \frac{n}{2}\widehat{\sigma}_\varepsilon^2(\lambda) + (\lambda - 1)\sum \log_e Y_i\end{aligned}$$

as given in the text.

Exercise 12.11*

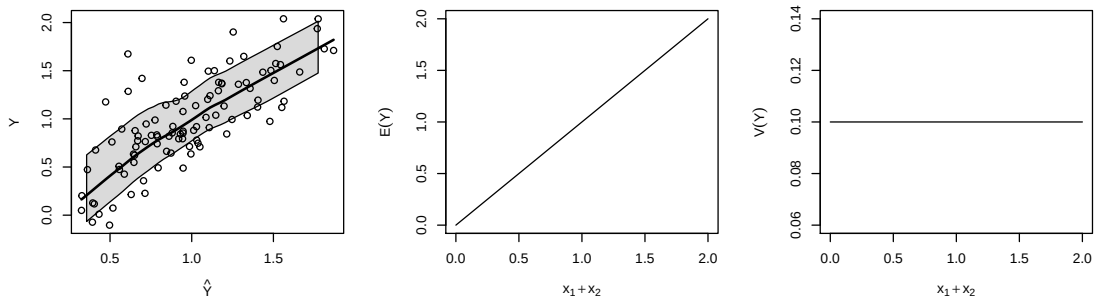
The instructions for the variance functions in this question are unclear. Here's a clearer version: For part (a), $V(Y|x_1, x_2) = 0.1$, and for the other parts of the question, $V(Y|x_1, x_2) = 0.1 \times |x_1 + x_2 - 1|$. In generating the error for each observation, sample a value from $\varepsilon \sim N(0, 1)$ and multiply it by the error standard deviation [i.e., $\sqrt{V(Y|x_1, x_2)}$] for that observation.

I worked this problem in R as follows:

(a)

```
library("car") # for loessLine()
set.seed(123) # for reproducibility
n <- 100

x1 <- runif(n, 0, 1)
x2 <- runif(n, 0, 1)
eps <- rnorm(n)
y <- x1 + x2 + eps*sqrt(0.1)
m <- lm(y ~ x1 + x2)

par(mfrow=c(1, 3))
plot(fitted(m), y, ylab="Y", xlab=expression(hat(Y)))
loessLine(fitted(m), y, col="black", var=TRUE,)

xx1 <- xx2 <- seq(0, 1, by=0.1)
X <- expand.grid(x1=xx1, x2=xx2)
X$x1.x2 <- with(X, x1 + x2)
X <- X[order(X$x1.x2), ]
X$Ey <- X$x1.x2
with(X, plot(x1 + x2, Ey, type="l",
             xlab=expression(x[1] + x[2]), ylab=expression(E(Y))))
with(X, plot(x1 + x2, rep(0.1, nrow(X)), type="l",
             xlab=expression(x[1] + x[2]), ylab=expression(V(Y))))
```

This code generates the following three-part figure:

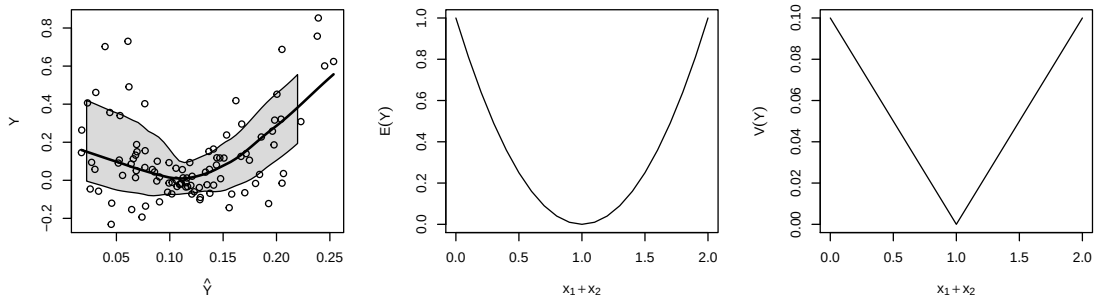


I loaded the **car** package for the `loessLine()` function, which draws a nonparametric regression line (showing how the conditional average of Y changes with $\widehat{Y}$) and also smooths the spread around the line (showing how the conditional variation of Y changes with $\widehat{Y}$).

The scatterplot of Y versus $\widehat{Y}$ clearly captures the the linearity of the mean function and the constant variance function.

(b)

```
y <- (x1 + x2 - 1)^2 + eps*sqrt(0.1*abs(x1 + x2 - 1))
m <- lm(y ~ x1 + x2)
par(mfrow=c(1, 3))
plot(fitted(m), y, ylab="Y", xlab=expression(hat(Y)))
loessLine(fitted(m), y, col="black", var=TRUE,)

X$Ey <- with(X, (x1 + x2 - 1)^2)
X$Vy <- with(X, 0.1*abs(x1 + x2 - 1))
with(X, plot(x1 + x2, Ey, type="l",
             xlab=expression(x[1] + x[2]), ylab=expression(E(Y))))
with(X, plot(x1 + x2, Vy, type="l",
             xlab=expression(x[1] + x[2]), ylab=expression(V(Y))))
```



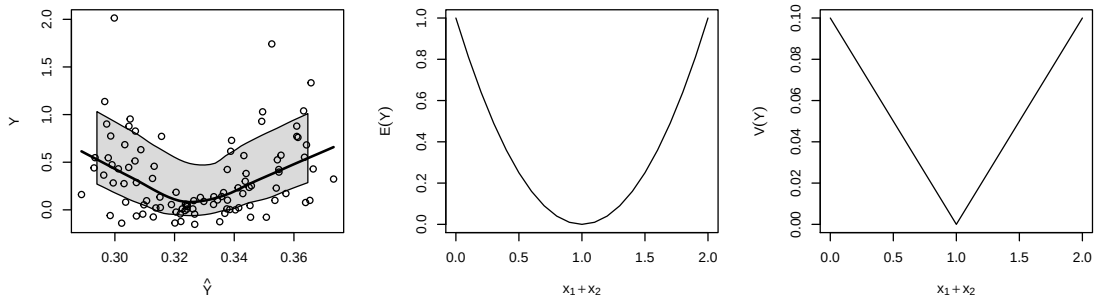
For this example, I reused the previous values of `x1` and `x2`, but generated new `y` values. The true mean function is nonlinear, and the variance function is not constant. The scatterplot of Y versus $\widehat{Y}$ does a reasonable job of revealing the nonlinearity in the mean function and the larger variance at the ends compared to the middle of the range of $x_1 + x_2$.

(c)

```
x2 <- x1 + 0.1*rnorm(n)
y <- (x1 + x2 - 1)^2 + eps*sqrt(0.1*abs(x1 + x2 - 1))
m <- lm(y ~ x1 + x2)
```



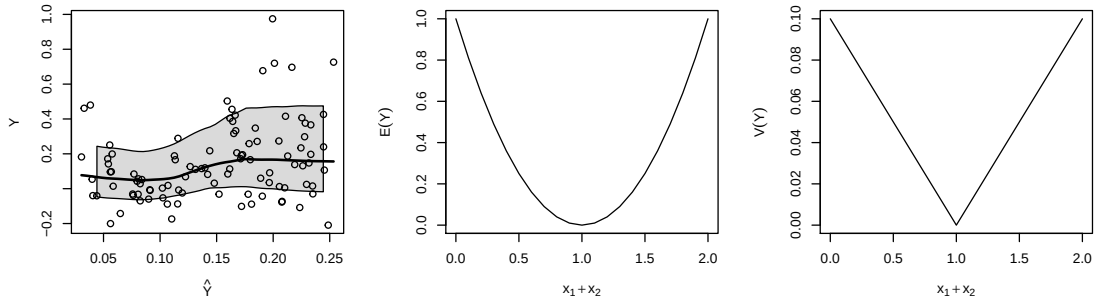
```
pdf("fig-ex-12.11-c.pdf", width=9, height=3)
par(mfrow=c(1, 3))
plot(fitted(m), y, ylab="Y", xlab=expression(hat(Y)))
loessLine(fitted(m), y, col="black", var=TRUE,)

X$Ey <- with(X, (x1 + x2 - 1)^2)
X$Vy <- with(X, 0.1*abs(x1 + x2 - 1))
with(X, plot(x1 + x2, Ey, type="l",
             xlab=expression(x[1] + x[2]), ylab=expression(E(Y))))
with(X, plot(x1 + x2, Vy, type="l",
             xlab=expression(x[1] + x[2]), ylab=expression(V(Y))))
```



As per the instructions, I generated new `x2` values correlated with `x1` and then generated new `y` values. The second and third plots, for the true mean and variance functions, are the same as in part (b). As in part (b), the scatterplot of Y versus $\widehat{Y}$ is informative about the mean and variance functions.

(d)

```
x2 <- abs(x1  - 0.5)
y <- (x1 + x2 - 1)^2 + eps*sqrt(0.1*abs(x1 + x2 - 1))
m <- lm(y ~ x1 + x2)
pdf("fig-ex-12.11-d.pdf", width=9, height=3)
par(mfrow=c(1, 3))
plot(fitted(m), y, ylab="Y", xlab=expression(hat(Y)))
loessLine(fitted(m), y, col="black", var=TRUE,)

X$Ey <- with(X, (x1 + x2 - 1)^2)
X$Vy <- with(X, 0.1*abs(x1 + x2 - 1))
with(X, plot(x1 + x2, Ey, type="l",
             xlab=expression(x[1] + x[2]), ylab=expression(E(Y))))
with(X, plot(x1 + x2, Vy, type="l",
             xlab=expression(x[1] + x[2]), ylab=expression(V(Y))))
```

As in (c), I generated new `x2` values, but now strongly (indeed, perfectly) nonlinearly related to `x1`, along with new corresponding `y` values. The true mean and variance functions are the same as in parts (b) and (c), as reflected in the second and third graphs. Now, however, the scatterplot of Y versus $\widehat{Y}$ fails to capture the mean and variance functions.

Exercises for Chapter 13

Exercise 13.1*

The question suggests introducing *two* Lagrange multipliers to find the second principal component, one for the normalizing constraint $\mathbf{a}_2'\mathbf{a}_2 = 1$ and one for the orthogonality constraint $\mathbf{a}_1'\mathbf{a}_2 = 0$, but the second Lagrange multiplier is unnecessary.

I proceed as in Section 13.1.1 (pages 349–350): The variance of the second principal component is $S^2_{W_2} = \mathbf{a}_2'\mathbf{R}_{XX}\mathbf{a}_2$. Define

$$F_2 \equiv \mathbf{a}_2'\mathbf{R}_{XX}\mathbf{a}_2 - L_2(\mathbf{a}_2'\mathbf{a}_2 - 1)$$

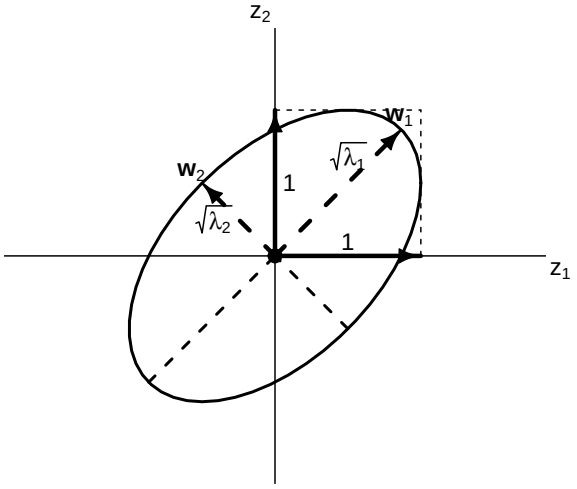
Finding the value of $\mathbf{a}_2$ along with the Langrange multiplier L_2 that maximize this function is the same problem as finding $\mathbf{a}_1$ and L_1, and the solution is therefore an eigenvector of $\mathbf{R}_{XX}$ and the associated eigenvalue. We want $L_2 = S^2_{W_2}$ to be as big as possible, but we also require that $\mathbf{a}_2$ be orthogonal to $\mathbf{a}_1$, and so L_2 must be the *second* largest eigenvalue of $\mathbf{R}_{XX}$.

A similar argument implies that the variances of the k principal components are the eigenvalues $L_1, L_2, \ldots, L_k$ of $\mathbf{R}_{XX}$ in descending order and that the principal-component coefficients in the k successive columns of $\mathbf{A}$ are the corresponding normalized eigenvectors.

That requiring that the $\mathbf{a}_j$s are mutually orthogonal is equivalent to requiring that the principal components, the $\mathbf{w}_j$s, are also mutually orthogonal, is proven in the text, where I show that the matrix of principal components $\mathbf{W} = [\mathbf{a}_1, \mathbf{a}_2, \ldots, \mathbf{a}_k]$ is an orthogonal matrix.

Exercise 13.3*

To make the linear algebra concrete, the geometry is shown in this graph, with the explanation to follow (where the graph is drawn for a positive correlation, specifically $r_{12} = .5$):



Because the covariance matrix for standardized variables is their correlation matrix, here $\mathbf{R}_{XX}$, the equation for the standard data ellipse takes the form $\mathbf{z}'\mathbf{R}_{XX}^{-1}\mathbf{z} = 1$, where in the general case $\mathbf{z}$ is a

$(k \times 1)$ vector. Here, $k = 2$, and so the equation for the ellipse is

$$
\begin{aligned}
[z_1, z_2] \begin{bmatrix} 1 & r_{12} \\ r_{12} & 1 \end{bmatrix}^{-1} \begin{bmatrix} z_1 \\ z_2 \end{bmatrix} &= 1 \\
\frac{1}{1 - r_{12}^2} [z_1, z_2] \begin{bmatrix} 1 & -r_{12} \\ -r_{12} & 1 \end{bmatrix} \begin{bmatrix} z_1 \\ z_2 \end{bmatrix} &= 1 \\
\frac{1}{1 - r_{12}^2} \left(z_1^2 - 2 r_{12} z_1 z_2 + z_2^2 \right) &= 1
\end{aligned}
$$

Because the variables are standardized, their means are 0 and their standard deviations are 1, so the ellipse is centered at the origin, and its vertical and horizontal shadows have half-length 1. The half-shadows are shown as vectors along the z_1 and z_2 axes in the diagram, not to be confused with the vector representation of regression discussed in Chapter 10.

Because of the symmetry induced by standardization, the axes of the ellipse are at $45°$ and $-45°$ angles. The major axis corresponds to the first principal component; the major half-axis is shown as a vector labeled $\mathbf{w}_1$. The length of this half-axis is the square-root of the larger eigenvalue L_1 of $\mathbf{R}_{XX}$ (whose eigenvalues are the inverses of the eigenvalues of $\mathbf{R}_{XX}^{-1}$), and its coordinates are given by the corresponding eigenvector scaled to be equal in length to $\sqrt{L_1}$ (rather than to length 1), that is $\mathbf{a}_1^* = [\sqrt{L_1/2}, \sqrt{L_1/2}]'$.

The story is similar for the minor axis of the ellipse, replacing the first eigenvalue of $\mathbf{R}_{XX}$ with the second (smaller) eigenvalue L_2, for which the coordinates of the half-axis vector are $\mathbf{a}_2^* = [-\sqrt{L_2/2}, \sqrt{L_2/2}]'$. The ordered eigenvalues of $\mathbf{R}_{XX}$ are $L_1 = 1 + r_{12}$ and $L_2 = 1 - r_{12}$ (for positive r_{12}, as in the diagram).

For $k = 3$, the standard data ellipse for standardized Xs becomes an ellipsoid whose half-shadows on the z_1, z_2, and z_3 axes are all of length 1, and whose axes, representing the principal components, correspond to the eigenvectors of $\mathbf{R}_{XX}$, each scaled by the square-root of the corresponding eigenvalue. These properties generalize to higher dimensions, where the standard data ellipsoid becomes a hyper-ellipsoid.

Exercise 13.5*

It's convenient (and sufficient) to work with the standardized regression coefficients, $\mathbf{b}^* = \mathbf{R}_{XX}^{-1} \mathbf{r}_{Xy}$, where $\mathbf{R}_{XX}$ is the correlation matrix for the regressors (with the exception of the constant regressor) and $\mathbf{r}_{Xy}$ is the vector of correlations between the regressors and the response.

The covariance matrix of the standardized regression coefficients is $\frac{\sigma_{\varepsilon^*}^2}{n-1} \mathbf{R}_{XX}^{-1}$. Because $\frac{\sigma_{\varepsilon^*}^2}{n-1}$ is a scalar constant multiplying all of the elements of $\mathbf{R}_{XX}^{-1}$, it affects all of the determinants proportionally and can be ignored.

For compactness, I'll write $\mathbf{R} \equiv \mathbf{R}_{XX}$ and $\mathbf{S} \equiv \mathbf{R}_{XX}^{-1}$. To convert $\mathbf{S}$ to a correlation matrix we can pre- and post-multiply it by a diagonal matrix of its inverse square-root diagonal elements. That is, let $\mathbf{s}$ contain the square-root diagonal elements of $\mathbf{S}$ (the standard errors of the standardized regression coefficients but ignoring the factor $\frac{\sigma_{\varepsilon^*}^2}{n-1}$). Then $\mathbf{S}^* \equiv \text{diag}\{\mathbf{s}\}^{-1} \, \mathbf{S} \, \text{diag}\{\mathbf{s}\}^{-1}$. Scaling the coefficient covariance matrix in this manner affects the numerator and denominator of the GVIF proportionally, and so it's sufficient to work with the unscaled covariance matrix $\mathbf{S}$ directly.

The result we want to establish follows from standard formulas for the inverse and determinant of a partitioned matrix. In particular, let's partition $\mathbf{R}$ as

$$
\mathbf{R} = \begin{bmatrix} \mathbf{R}_{11} & \mathbf{R}_{12} \\ \mathbf{R}_{21} & \mathbf{R}_{22} \end{bmatrix}
$$

where $\mathbf{R}_{11}$ is $(p \times p)$; $\mathbf{R}_{22}$ is $q \times q$; $\mathbf{R}_{12}$ is $p \times q$; $\mathbf{R}_{21} = \mathbf{R}_{12}'$ is $q \times p$; and $p + q = k$. Then the inverse of

$\mathbf{R}$ is

$$\begin{aligned}\mathbf{S} &= \begin{bmatrix}\mathbf{S}_{11} & \mathbf{S}_{12}\\ \mathbf{S}_{21} & \mathbf{S}_{22}\end{bmatrix}\\ &= \begin{bmatrix}(\mathbf{R}_{11}-\mathbf{R}_{12}\mathbf{R}_{22}^{-1}\mathbf{R}_{21})^{-1} & \mathbf{S}_{12}\\ \mathbf{S}_{21} & (\mathbf{R}_{22}-\mathbf{R}_{21}\mathbf{R}_{11}^{-1}\mathbf{R}_{12})^{-1}\end{bmatrix}\end{aligned}$$

where I've not bothered to expand $\mathbf{S}_{12}$ and $\mathbf{S}_{21} = \mathbf{S}_{12}'$ because we won't need them.

Likewise, we can express the determinant of $\mathbf{R}$ as

$$\begin{aligned}\det\mathbf{R} &= \det\mathbf{R}_{22}\det(\mathbf{R}_{11}-\mathbf{R}_{12}\mathbf{R}_{22}^{-1}\mathbf{R}_{21})\\ &= \det\mathbf{R}_{11}\det(\mathbf{R}_{22}-\mathbf{R}_{21}\mathbf{R}_{11}^{-1}\mathbf{R}_{12})\end{aligned}$$

from which

$$\begin{aligned}\det\mathbf{R}_{11} &= \frac{\det\mathbf{R}}{\det(\mathbf{R}_{22}-\mathbf{R}_{21}\mathbf{R}_{11}^{-1}\mathbf{R}_{12})}\\ \det\mathbf{R}_{22} &= \frac{\det\mathbf{R}}{\det(\mathbf{R}_{11}-\mathbf{R}_{12}\mathbf{R}_{22}^{-1}\mathbf{R}_{21})}\end{aligned}$$

Now, starting with the formula for the GVIF in Equation 13.7, using the results given above and the fact that the determinant of the inverse of a matrix is the inverse of its determinant,

$$\begin{aligned}\text{GVIF}_1 &= \frac{\det\mathbf{R}}{\det\mathbf{R}_{11}\det\mathbf{R}_{22}}\\ &= \frac{\det\mathbf{R}}{\frac{(\det\mathbf{R})^2}{\det(\mathbf{R}_{11}-\mathbf{R}_{12}\mathbf{R}_{22}^{-1}\mathbf{R}_{21})\det(\mathbf{R}_{22}-\mathbf{R}_{21}\mathbf{R}_{11}^{-1}\mathbf{R}_{12})}}\\ &= \frac{\det(\mathbf{R}_{11}-\mathbf{R}_{12}\mathbf{R}_{22}^{-1}\mathbf{R}_{21})\det(\mathbf{R}_{22}-\mathbf{R}_{21}\mathbf{R}_{11}^{-1}\mathbf{R}_{12})}{\det\mathbf{R}}\\ &= \frac{(\det\mathbf{S}_{11})^{-1}(\det\mathbf{S}_{22})^{-1}}{\det(\mathbf{S}^{-1})}\\ &= \frac{\det\mathbf{S}}{\det\mathbf{S}_{11}\det\mathbf{S}_{22}}\end{aligned}$$

as stated in footnote 25 the text.

Exercise 13.7

I did the computations in R, using the `stepAIC()` function in the **MASS** package. The function is slightly misleadingly named, in that it can perform selection by criteria other than the AIC. In particular, specifying the argument `k = log(n)`, where `n` is the sample size, uses the BIC as the criterion (still labeled "AIC" in the output, however). I made this specification because the BIC was the selection criterion in the all-subsets regressions performed in the text.

```
> library("carData") # for B. Fox data
> Bfox["1973", "tfr"] <- 1931 # correct error in data
> Bfox$time <- 1:30

> m <- lm(partic ~ ., data=Bfox)
> summary(m)

Call:
lm(formula = partic ~ ., data = Bfox)

Residuals:
```



```
     Min       1Q   Median       3Q      Max
-0.83213 -0.33438 -0.01621  0.36769  1.05048

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept)  1.680e+01  3.724e+00   4.512 0.000157 ***
tfr         -1.949e-06  5.011e-04  -0.004 0.996930
menwage     -2.919e-02  1.502e-01  -0.194 0.847660
womwage      1.984e-02  1.744e-01   0.114 0.910413
debt         6.397e-02  1.850e-02   3.459 0.002132 **
parttime     6.566e-01  8.205e-02   8.002 4.27e-08 ***
time         4.452e-03  1.107e-01   0.040 0.968272
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.5381 on 23 degrees of freedom
Multiple R-squared:  0.9935,    Adjusted R-squared:  0.9918
F-statistic: 587.3 on 6 and 23 DF,  p-value: < 2.2e-16

> library("MASS") # for stepAIC()

> # backward
> stepAIC(lm(partic ~ ., data=Bfox),
+         direction="backward",
+         scope=list(lower = ~ 1,
+                    upper= ~ tfr + menwage + womwage +
+                      debt + parttime + time),
+         k=log(30))
Start:  AIC=-21.35
partic ~ tfr + menwage + womwage + debt + parttime + time

           Df Sum of Sq     RSS     AIC
- tfr       1    0.0000  6.6586 -24.752
- time      1    0.0005  6.6590 -24.750
- womwage   1    0.0037  6.6623 -24.735
- menwage   1    0.0109  6.6695 -24.702
<none>                   6.6586 -21.350
- debt      1    3.4631 10.1217 -12.188
- parttime  1   18.5389 25.1974  15.174

Step:  AIC=-24.75
partic ~ menwage + womwage + debt + parttime + time

           Df Sum of Sq    RSS     AIC
- time      1    0.0006  6.659 -28.150
- womwage   1    0.0041  6.663 -28.134
- menwage   1    0.0114  6.670 -28.102
<none>                   6.659 -24.752
- debt      1    5.2403 11.899 -10.736
- parttime  1   27.0229 33.681  20.479

Step:  AIC=-28.15
partic ~ menwage + womwage + debt + parttime

           Df Sum of Sq    RSS     AIC
- womwage   1    0.0037  6.663 -31.535
- menwage   1    0.0188  6.678 -31.467
<none>                   6.659 -28.150
- debt      1    6.9403 13.599 -10.130
```



```
- parttime  1   27.2124 33.872  17.246

Step:  AIC=-31.53
partic ~ menwage + debt + parttime

           Df Sum of Sq    RSS     AIC
- menwage   1    0.0152  6.678 -34.868
<none>                   6.663 -31.535
- debt      1    9.3340 15.997  -8.661
- parttime  1   27.3305 33.993  13.953

Step:  AIC=-34.87
partic ~ debt + parttime

           Df Sum of Sq    RSS     AIC
<none>                   6.678 -34.868
- parttime  1    30.166 36.844  12.967
- debt      1    92.640 99.318  42.716

Call:
lm(formula = partic ~ debt + parttime, data = Bfox)

Coefficients:
(Intercept)         debt     parttime
   16.32501      0.06257      0.66133

> # forward:
> stepAIC(lm(partic ~ 1, data=Bfox),
+         direction="forward",
+         scope=list(lower = ~ 1,
+                    upper= ~ tfr + menwage + womwage +
+                      debt + parttime + time),
+         k=log(30)) # k = log(n) -> BIC
Start:  AIC=109.39
partic ~ 1

           Df Sum of Sq     RSS     AIC
+ debt      1    989.97   36.84  12.967
+ womwage   1    960.92   65.89  30.404
+ menwage   1    945.33   81.48  36.777
+ time      1    932.84   93.97  41.055
+ parttime  1    927.49   99.32  42.716
+ tfr       1    833.80  193.01  62.649
<none>                  1026.81 109.392

Step:  AIC=12.97
partic ~ debt

           Df Sum of Sq    RSS     AIC
+ parttime  1   30.1656  6.678 -34.868
+ tfr       1    9.4190 27.425   7.511
<none>                  36.844  12.967
+ menwage   1    2.8503 33.993  13.953
+ time      1    2.4229 34.421  14.328
+ womwage   1    0.0879 36.756  16.297

Step:  AIC=-34.87
partic ~ debt + parttime
```



```
          Df Sum of Sq    RSS     AIC
<none>                  6.6780 -34.868
+ menwage  1 0.0151670 6.6629 -31.535
+ time     1 0.0081014 6.6699 -31.503
+ tfr      1 0.0066785 6.6714 -31.496
+ womwage  1 0.0001297 6.6779 -31.467

Call:
lm(formula = partic ~ debt + parttime, data = Bfox)

Coefficients:
(Intercept)         debt     parttime
   16.32501      0.06257      0.66133

> # forward/backward:
> stepAIC(lm(partic ~ 1, data=Bfox),
+         direction="both",
+         scope=list(lower = ~ 1,
+                    upper= ~ tfr + menwage + womwage +
+                      debt + parttime + time),
+         k=log(30))
Start:  AIC=109.39
partic ~ 1

           Df Sum of Sq     RSS     AIC
+ debt      1    989.97   36.84  12.967
+ womwage   1    960.92   65.89  30.404
+ menwage   1    945.33   81.48  36.777
+ time      1    932.84   93.97  41.055
+ parttime  1    927.49   99.32  42.716
+ tfr       1    833.80  193.01  62.649
<none>                  1026.81 109.392

Step:  AIC=12.97
partic ~ debt

           Df Sum of Sq     RSS     AIC
+ parttime  1     30.17    6.68 -34.868
+ tfr       1      9.42   27.42   7.511
<none>                    36.84  12.967
+ menwage   1      2.85   33.99  13.953
+ time      1      2.42   34.42  14.328
+ womwage   1      0.09   36.76  16.297
- debt      1    989.97 1026.81 109.392

Step:  AIC=-34.87
partic ~ debt + parttime

           Df Sum of Sq    RSS     AIC
<none>                  6.678 -34.868
+ menwage   1     0.015  6.663 -31.535
+ time      1     0.008  6.670 -31.503
+ tfr       1     0.007  6.671 -31.496
+ womwage   1     0.000  6.678 -31.467
- parttime  1    30.166 36.844  12.967
- debt      1    92.640 99.318  42.716

Call:
lm(formula = partic ~ debt + parttime, data = Bfox)
```



```
Coefficients:
(Intercept)          debt      parttime
   16.32501       0.06257       0.66133
```

In this example, all three procedures select the same model, with debt and part-time work as explanatory variables. This is also the model with the smallest BIC among all subsets of explanatory variables, as revealed by Figure 13.8. If you look carefully, you'll see that the BIC for this model reported by `stepAIC()` is -34.9, while the BIC shown in Figure 13.8 is approximately -140. The BICs computed by `stepAIC()` and by the program used to produce Figure 13.8 differ by an additive constant; this difference is inessential because we only attend to *differences* in the values of the BIC for models to be compared.

Exercise 13.9*

Taking the hint, $\mathbf{b}_d^* = (\mathbf{R}_{XX} + d\mathbf{I}_k)^{-1}\frac{1}{n-1}\mathbf{Z}_X'\mathbf{z}_y$, and let $\mathbf{W} \equiv (\mathbf{R}_{XX} + d\mathbf{I}_k)^{-1}$. Then $\mathbf{b}_d^* = \mathbf{W}\frac{1}{n-1}\mathbf{Z}_X'\mathbf{z}_y$, and

$$
\begin{aligned}
V(\mathbf{b}_d^*) &= \frac{1}{(1-n)^2}\mathbf{W}\mathbf{Z}'V(\mathbf{z}_y)\mathbf{Z}\mathbf{W}' \\
&= \frac{1}{(n-1)^2}\mathbf{W}\mathbf{Z}'\sigma_\varepsilon^{*2}\mathbf{I}_n\mathbf{Z}\mathbf{W} \\
&= \frac{\sigma_\varepsilon^{*2}}{(n-1)^2}\mathbf{W}\mathbf{Z}'\mathbf{I}_n\mathbf{Z}\mathbf{W} \\
&= \frac{\sigma_\varepsilon^{*2}}{n-1}\mathbf{W}\frac{1}{n-1}\mathbf{Z}'\mathbf{Z}\mathbf{W} \\
&= \frac{\sigma_\varepsilon^{*2}}{n-1}\mathbf{W}\mathbf{R}_{XX}\mathbf{W} \\
&= \frac{\sigma_\varepsilon^{*2}}{n-1}(\mathbf{R}_{XX} + d\mathbf{I}_k)^{-1}\mathbf{R}_{XX}(\mathbf{R}_{XX} + d\mathbf{I}_k)^{-1}
\end{aligned}
$$

Exercises for Chapter 14

Exercise 14.1

π	$V(\varepsilon) = \pi(1-\pi)$
.001	.001
.01	.0099
.05	.0475
.1	.09
.3	.21
.5	.25
.7	.21
.9	.09
.95	.0475
.99	.0099
.999	.001

Heteroscedasticity becomes serious only when π gets close to 0 or 1; the error variance is nearly constant between $\pi = .3$ and $\pi = .7$.

Exercise 14.3*

Differentiating $\pi = 1/[1 + e^{-(\alpha+\beta X)}]$ with respect to X,

$$\begin{aligned}
\frac{d\pi}{dx} &= \frac{\beta e^{-(\alpha+\beta X)}}{[1 + e^{-(\alpha+\beta X)}]^2} \\
&= \beta \times \frac{e^{-(\alpha+\beta X)}}{1 + e^{-(\alpha+\beta X)}} \times \frac{1}{1 + e^{-(\alpha+\beta X)}} \\
&= \beta \times \left[1 - \frac{1}{1 + e^{-(\alpha+\beta X)}}\right] \times \frac{1}{1 + e^{-(\alpha+\beta X)}} \\
&= \beta(1-\pi)\pi
\end{aligned}$$

Exercise 14.5*

Dropping the subscript i for compactness,

$$\begin{aligned}
1 - \pi &= 1 - \frac{1}{1 + \exp[-(\alpha + \beta_1 X_1 + \cdots + \beta_k X_k)]} \\
&= \frac{1 + \exp[-(\alpha + \beta_1 X_1 + \cdots + \beta_k X_k)] - 1}{1 + \exp[-(\alpha + \beta_1 X_1 + \cdots + \beta_k X_k)]} \\
&= \frac{\exp(\alpha + \beta_1 X_1 + \cdots + \beta_k X_k) \times \exp[-(\alpha + \beta_1 X_1 + \cdots + \beta_k X_k)]}{\exp(\alpha + \beta_1 X_1 + \cdots + \beta_k X_k) \times \{1 + \exp[-(\alpha + \beta_1 X_1 + \cdots + \beta_k X_k)]\}} \\
&= \frac{1}{\exp(\alpha + \beta_1 X_1 + \cdots + \beta_k X_k) + 1}
\end{aligned}$$

Exercise 14.7*

We have

$$\begin{aligned}
-2\sigma_\varepsilon^2(l - l') &= -2\sigma_\varepsilon^2 \left\{ \left[-\frac{n}{2} \log_e(2\pi\sigma_\varepsilon^2) - \frac{\sum \varepsilon_i^2}{2\sigma_\varepsilon^2} \right] - \left[-\frac{n}{2} \log_e(2\pi\sigma_\varepsilon^2) \right] \right\} \\
&= -2\sigma_\varepsilon^2 \left(-\frac{\sum \varepsilon_i^2}{2\sigma_\varepsilon^2} \right) \\
&= \sum \varepsilon_i^2
\end{aligned}$$

The result follows from substituting the least-squares residuals E_i for the errors ε_i.

Exercise 14.9*

Substituting $\mathbf{y}^*$ into the formula for $\mathbf{b}$, multiplying out, and simplifying produces the desired result:

$$\begin{aligned}\mathbf{b} &= (\mathbf{X}'\mathbf{V}\mathbf{X})^{-1}\mathbf{X}'\mathbf{V}[\mathbf{X}\mathbf{b} + \mathbf{V}^{-1}(\mathbf{y} - \mathbf{p})] \\ &= (\mathbf{X}'\mathbf{V}\mathbf{X})^{-1}\mathbf{X}'\mathbf{V}\mathbf{X}\mathbf{b} + (\mathbf{X}'\mathbf{V}\mathbf{X})^{-1}\mathbf{X}'\mathbf{V}\mathbf{V}^{-1}(\mathbf{y} - \mathbf{p}) \\ &= \mathbf{b} + (\mathbf{X}'\mathbf{V}\mathbf{X})^{-1}\mathbf{X}'(\mathbf{y} - \mathbf{p})\end{aligned}$$

Exercise 14.11*

Using the chain rule to differentiate the log-likelihood (in the equation at the bottom of page 397),

$$\begin{aligned}\frac{\partial \log_e L}{\partial \boldsymbol{\gamma}_j} &= W_{ij}\mathbf{x}_i - \sum_{i=1}^{n}\left[1 + \sum_{l=1}^{m-1}\exp(\mathbf{x}_i'\boldsymbol{\gamma}_l)\right]^{-1}\exp(\mathbf{x}_i'\boldsymbol{\gamma}_j)\mathbf{x}_i \\ &= W_{ij}\mathbf{x}_i - \sum_{i=1}^{n}\frac{\exp(\mathbf{x}_i'\boldsymbol{\gamma}_j)\mathbf{x}_i}{1 + \sum_{l=1}^{m-1}\exp(\mathbf{x}_i'\boldsymbol{\gamma}_l)}\end{aligned}$$

The estimating equations follow from setting the derivatives to 0 and substituting the estimated coefficients $\mathbf{c}_j$ for the parameters $\boldsymbol{\gamma}_j$.

To get the components of the information matrix, differentiate the log-likelihood a second time, first with respect to $\boldsymbol{\gamma}_j$ and then with respect to $\boldsymbol{\gamma}_{j'}$ (i.e., the parameter vector for a category of the response *different from* j):

$$\begin{aligned}\frac{\partial \log_e L}{\partial \boldsymbol{\gamma}_j \partial \boldsymbol{\gamma}_j'} &= \sum_{i=1}^{n}\left\{\frac{\mathbf{x}_i\mathbf{x}_i'\exp(2\mathbf{x}_i'\boldsymbol{\gamma}_j)}{[1 + \sum_{l=1}^{m-1}\exp(\mathbf{x}_i'\boldsymbol{\gamma}_l)]^2} - \frac{\mathbf{x}_i\mathbf{x}_i'\exp(\mathbf{x}_i'\boldsymbol{\gamma}_j)}{1 + \sum_{l=1}^{m-1}\exp(\mathbf{x}_i'\boldsymbol{\gamma}_l)}\right\} \\ &= -\sum_{i=1}^{n}\frac{\mathbf{x}_i\mathbf{x}_i'\exp(\mathbf{x}_i'\boldsymbol{\gamma}_j)[1 + \sum_{l=1}^{m-1}\exp(\mathbf{x}_i'\boldsymbol{\gamma}_l) - \exp(\mathbf{x}_i'\boldsymbol{\gamma}_j)]}{[1 + \sum_{l=1}^{m-1}\exp(\mathbf{x}_i'\boldsymbol{\gamma}_l)]^2} \\ \frac{\partial \log_e L}{\partial \boldsymbol{\gamma}_j \partial \boldsymbol{\gamma}_{j'}'} &= \sum_{i=1}^{n}\frac{\mathbf{x}_i\mathbf{x}_i'\exp(\mathbf{x}_i'\boldsymbol{\gamma}_j)\exp(\mathbf{x}_i'\boldsymbol{\gamma}_{j'})}{[1 + \sum_{l=1}^{m-1}\exp(\mathbf{x}_i'\boldsymbol{\gamma}_l)]^2} \\ &= \sum_{i=1}^{n}\frac{\mathbf{x}_i\mathbf{x}_i'\exp[\mathbf{x}_i'(\boldsymbol{\gamma}_j + \boldsymbol{\gamma}_{j'})]}{[1 + \sum_{l=1}^{m-1}\exp(\mathbf{x}_i'\boldsymbol{\gamma}_l)]^2}\end{aligned}$$

These derivatives involve $\mathbf{x}$s and $\boldsymbol{\gamma}$s, none of which are random, and so the components of the information matrix are just the negatives of the second derivatives, as given in the text.

Exercise 14.13*

Taking the log of the likelihood given on page 412 produces (capitalizing Y_i for consistency with the notation used for the likelihood of the binary logit model)

$$\log_e L(\boldsymbol{\beta}) = \sum_{i=1}^{m}\left\{\log_e\binom{n_i}{Y_i} + Y_i\mathbf{x}_i'\boldsymbol{\beta} + n_i\log_e\left[\frac{1}{1 + \exp(\mathbf{x}_i'\boldsymbol{\beta})}\right]\right\}$$

Then differentiating the log-likelihood with respect to the regression coefficients,

$$\begin{aligned}\frac{\partial \log_e L(\boldsymbol{\beta})}{\partial \boldsymbol{\beta}} &= \sum_{i=1}^{m}Y_i\mathbf{x}_i - \sum_{i=1}^{m}\frac{n_i\exp(\mathbf{x}_i'\boldsymbol{\beta})}{1 + \exp(\mathbf{x}_i'\boldsymbol{\beta})}\mathbf{x}_i \\ &= \sum_{i=1}^{m}Y_i\mathbf{x}_i - \sum_{i=1}^{m}n_i\frac{1}{1 + \exp(-\mathbf{x}_i'\boldsymbol{\beta})}\mathbf{x}_i\end{aligned}$$

Setting the derivatives to 0 and substituting the MLE $\mathbf{b}$ for the parameters $\boldsymbol{\beta}$ produces the estimating equations

$$\sum_{i=1}^{m} n_i \frac{1}{1 + \exp(-\mathbf{x}_i'\mathbf{b})}\mathbf{x}_i = \sum_{i=1}^{m} Y_i \mathbf{x}_i$$

where we recognize that $P_i \equiv 1/[1 + \exp(-\mathbf{x}_i'\boldsymbol{\beta})]$ is the fitted probability.

Comparing these estimating equations for the binomial logit model with those for the binary logit model in Equation 14.16 (on page 389), we see the right-hand sides of the equations are the same, except that Y_i for the binary model for a single individual is either 0 or 1, while the response for the binomial model counts the number of "successes" (equivalent to the number of binary 1s) in n_i binomial trials. Similarly, the sum on the left-hand side of the estimating equations is the same for the binomial model as for the binary model, except that in the binary case, we multiply by the number of binomial trials n_i. Thus if we applied the binary logit model to binomial data resolved into $n = \sum_{i=1}^{m} n_i$ individual observations, we'd get exactly the same MLE of $\boldsymbol{\beta}$.

Exercises for Chapter 15

Exercise 15.1

In the corrected description of the negative-binomial model (see the errata for the text), the inverse of the scale parameter is the shape parameter $\psi = 1/\omega$. It's actually when the *scale parameter* $\omega = 0$ that the negative binomial model reduces to the Poisson model, because the negative-binomial variance of the response is then $V(Y_i) = \mu_i + \omega\mu_i^2 = \mu_i + 0 \times \mu_i^2 = \mu_i$—that is, the Poisson variance.

I'll begin by using R to reproduce the Poisson and negative-binomial regressions reported in the text; recall from later in the chapter that the specification of the linear predictor in these models is flawed in that assets should really be replaced by the log of assets:

```
> library("car") # for Ornstein Data
Loading required package: carData
> library("MASS") # for glm.nb()

>   # data management to correspond to text
> Ornstein$sector <- factor(Ornstein$sector,
+                           levels=c('WOD', 'TRN', 'FIN', 'MIN', 'HLD',
+                                    'MER', 'MAN', 'AGR', 'BNK', 'CON'))
> contrasts(Ornstein$sector) <- contr.treatment(levels(Ornstein$sector),
+                                               base=10)
> contrasts(Ornstein$nation) <- contr.treatment(levels(Ornstein$nation),
+                                               base=4)
> Ornstein$assets <- Ornstein$assets/1000

>   # Poisson and negative-binomial models as fit in the text:

> mod.ornstein.p <- glm(interlocks ~ assets + nation + sector,
+                       family=poisson, data=Ornstein)
> summary(mod.ornstein.p)

Call:
glm(formula = interlocks ~ assets + nation + sector, family = poisson,
    data = Ornstein)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-5.9908  -2.4767  -0.8582   1.3472   7.3610

Coefficients:
            Estimate Std. Error z value Pr(>|z|)
(Intercept) 0.879075   0.210058   4.185 2.85e-05 ***
assets      0.020851   0.001202  17.340  < 2e-16 ***
nationCAN   0.825933   0.048968  16.867  < 2e-16 ***
nationOTH   0.662727   0.075534   8.774  < 2e-16 ***
nationUK    0.248847   0.091932   2.707 0.006792 **
sectorWOD   1.331123   0.213065   6.247 4.17e-10 ***
sectorTRN   1.297399   0.213786   6.069 1.29e-09 ***
sectorFIN   1.296546   0.211464   6.131 8.72e-10 ***
sectorMIN   1.240637   0.208526   5.950 2.69e-09 ***
sectorHLD   0.828031   0.232934   3.555 0.000378 ***
sectorMER   0.797261   0.218188   3.654 0.000258 ***
sectorMAN   0.672169   0.213298   3.151 0.001625 **
sectorAGR   0.619571   0.211968   2.923 0.003467 **
sectorBNK   0.210389   0.253688   0.829 0.406922
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```



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

    Null deviance: 3737.0  on 247  degrees of freedom
Residual deviance: 1887.4  on 234  degrees of freedom
AIC: 2813.4

Number of Fisher Scoring iterations: 5

> mod.ornstein.nb <- glm.nb(interlocks ~ assets + nation + sector,
+                           data=Ornstein)
> summary(mod.ornstein.nb)

Call:
glm.nb(formula = interlocks ~ assets + nation + sector, data = Ornstein,
    init.theta = 1.312185376, link = log)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-2.6805  -1.0840  -0.2759   0.4387   2.0262

Coefficients:
             Estimate Std. Error z value Pr(>|z|)
(Intercept)  0.734694   0.470339   1.562  0.11828
assets       0.032663   0.005734   5.697 1.22e-08 ***
nationCAN    0.786248   0.143056   5.496 3.88e-08 ***
nationOTH    0.798014   0.246622   3.236  0.00121 **
nationUK     0.307807   0.253997   1.212  0.22557
sectorWOD    1.387476   0.500557   2.772  0.00557 **
sectorTRN    1.329927   0.509125   2.612  0.00900 **
sectorFIN    1.343301   0.500797   2.682  0.00731 **
sectorMIN    1.254662   0.470301   2.668  0.00764 **
sectorHLD    0.873148   0.579060   1.508  0.13159
sectorMER    0.939730   0.505574   1.859  0.06306 .
sectorMAN    0.812702   0.479040   1.697  0.08979 .
sectorAGR    0.733907   0.476568   1.540  0.12356
sectorBNK   -0.328478   0.719424  -0.457  0.64797
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for Negative Binomial(1.3122) family taken to be 1)

    Null deviance: 440.26  on 247  degrees of freedom
Residual deviance: 293.49  on 234  degrees of freedom
AIC: 1717.1

Number of Fisher Scoring iterations: 1


              Theta:  1.312
          Std. Err.:  0.143

 2 x log-likelihood:  -1687.104
```

Next, I'll compare the log-likelihoods for the models to compute the likelihood-ratio test statistic and its p-value:

```
> (G2 <- as.vector(2*(logLik(mod.ornstein.nb) - logLik(mod.ornstein.p))))
[1] 1098.317
> 0.5*pchisq(G2, df=1, lower.tail=FALSE)
```



```
[1] 3.834035e-241
```

The p-value for the test is effectively 0, suggesting overwhelming evidence for overdispersion.

Exercise 15.3

It's simple to perform the necessary computations in R:

```
> observed <- matrix(c(305, 126,
+                      405, 125,
+                      265,  49), 3, 2, byrow=TRUE)
> observed
     [,1] [,2]
[1,]  305  126
[2,]  405  125
[3,]  265   49
> expected <- outer(rowSums(observed), colSums(observed))/
+   sum(observed)
> expected
         [,1]      [,2]
[1,] 329.5882 101.41176
[2,] 405.2941 124.70588
[3,] 240.1176  73.88235
>
> # Pearson chi-square
> sum((observed - expected)^2/expected)
[1] 18.75532
>
> # LR chi-square
> 2*sum(observed*log(observed/expected))
[1] 19.42795
```

As is typically the case, the Pearson and likelihood-ratio chi-square test statistics are similar, and the value of the LR test statistic computed here agrees with the value reported in the text.

Exercise 15.5*

Making the indicated substitutions, and recalling that in the binomial context y is the observed proportion of successes in n binomial trials and μ is the probability of a success on an individual trial, we have

$$
\begin{aligned}
p(y) &= \exp\left\{ny \log_e \frac{\mu}{1-\mu} - n \log_e \left[1 + \exp\left(\log_e \frac{\mu}{1-\mu}\right)\right] + \log_e \binom{n}{ny}\right\} \\
&= \binom{n}{ny} \left(\frac{\mu}{1-\mu}\right)^{ny} \left[1 + \exp\left(\log_e \frac{\mu}{1-\mu}\right)\right]^{-n} \\
&= \binom{n}{ny} \left(\frac{\mu}{1-\mu}\right)^{ny} \left(1 + \frac{\mu}{1-\mu}\right)^{-n} \\
&= \binom{n}{ny} \left(\frac{\mu}{1-\mu}\right)^{ny} \left(\frac{1}{1-\mu}\right)^{-n} \\
&= \binom{n}{ny} \left(\frac{\mu}{1-\mu}\right)^{ny} (1-\mu)^n \\
&= \binom{n}{ny} \mu^{ny} (1-\mu)^{n(1-y)}
\end{aligned}
$$

which is the usual form for the binomial probability of ny successes and $n(1-y)$ failures.

Exercise 15.7*

- For the Gaussian family:

$$\begin{aligned} V(Y) &= \phi \times \frac{d^2(\theta^2)/2}{d\theta^2} \\ &= \phi \times \frac{d(\theta)}{d\theta} \\ &= \phi \end{aligned}$$

- For the binomial family (using $\theta = \log_e[\mu(1-\mu)]$ and hence $e^\theta = \mu/(1-\mu)$):

$$\begin{aligned} V(Y) &= \frac{1}{n} \times \frac{d^2[\log_e(1+e^\theta)]}{d\theta^2} \\ &= \frac{1}{n} \times \frac{d\left(\frac{e^\theta}{1+e^\theta}\right)}{d\theta} \\ &= \frac{1}{n} \times \frac{e^\theta}{(1+e^\theta)^2} \\ &= \frac{1}{n} \times \frac{\frac{\mu}{1-\mu}}{\left(1+\frac{\mu}{1-\mu}\right)^2} \\ &= \frac{1}{n} \times \frac{\frac{\mu}{1-\mu}}{\left(\frac{1}{1-\mu}\right)^2} \\ &= \frac{\mu(1-\mu)}{n} \end{aligned}$$

- For the Poisson family (using $\theta = \log_e \mu$ and hence $e^\theta = \mu$):

$$V(Y) = 1 \times \frac{d^2\left(e^\theta\right)}{d\theta^2} = e^\theta = \mu$$

- For the gamma family (using $\theta = -1/\mu$, that is, in this case the negative of the canonical link):

$$\begin{aligned} V(Y) &= \phi \times \frac{d^2[-\log_e(-\theta)]}{d\theta^2} \\ &= \phi \times \frac{d(-1/\theta)}{d\theta} \\ &= \phi \times \frac{1}{\theta^2} \\ &= \phi\mu^2 \end{aligned}$$

- For the inverse-Gaussian family (using $\theta = -1/(2\mu^2)$, in this case $-1/2$ times the canonical link)

$$\begin{aligned} V(Y) &= \phi \times \frac{d^2(-\sqrt{-2\theta})}{d\theta^2} \\ &= \phi \times \frac{d\left(\frac{1}{\sqrt{2\theta}}\right)}{d\theta} \\ &= \phi \times \frac{1}{2\sqrt{2}(-\theta)^{3/2}} \\ &= \phi \times \frac{1}{2\sqrt{2}\left(\frac{1}{2\mu^2}\right)^{3/2}} \\ &= \phi\mu^3 \end{aligned}$$

Exercise 15.9

To make the general formula for the deviance work properly for the gamma and inverse-Gaussian families, it's necessary to be more careful about the definition of the canonical link in each case. In particular, for the gamma family, we must take the canonical link as $g_c(\mu) = -1/\mu$ rather than, as is conventional in fitting GLMs, $1/\mu$; for the inverse-Gaussian family, we must take $g_c(\mu) = -1/(2\mu^2)$ rather than as $1/\mu^2$. This difference is inessential for the fit of the model, because the alternative canonical links are linear functions of the conventional links.

I'm grateful to Georges Monette of York University for clarifying this issue, and the point should have been addressed either in the exercise or in the text.

- For the Gaussian family, $a_i = 1$. Recalling that in the Gaussian case, the ML and least-squares estimates coincide, $\widehat{\mu}_i = \widehat{Y}_i$, and $\sum Y_i \widehat{Y}_i = \sum \widehat{Y}_i^2$,

$$\begin{aligned} D(\mathbf{y}; \widehat{\boldsymbol{\mu}}) &= 2 \sum \left[Y_i (Y_i - \widehat{Y}_i) - \frac{Y_i^2}{2} + \frac{\widehat{Y}_i^2}{2} \right] \\ &= 2 \sum \left(Y_i^2 - Y_i \widehat{Y}_i - \frac{Y_i^2}{2} + \frac{\widehat{Y}_i^2}{2} \right) \\ &= \sum \left(2Y_i^2 - 2\widehat{Y}_i^2 - Y_i^2 + \widehat{Y}_i^2 \right) \\ &= \sum \left(Y_i^2 - \widehat{Y}_i^2 \right) \\ &= \sum \left(Y_i - \widehat{Y}_i \right)^2 \end{aligned}$$

  because $\sum \left(Y_i^2 - \widehat{Y}_i^2 \right) = \sum Y_i^2 + \sum \widehat{Y}_i^2 - 2 \sum Y_i \widehat{Y}_i = \sum Y_i^2 + \sum \widehat{Y}_i^2 - 2 \sum \widehat{Y}_i^2$.

- For the binomial family, $a_i = 1/n_i$.

$$\begin{aligned} D(\mathbf{y}; \widehat{\boldsymbol{\mu}}) &= 2 \sum n_i \left\{ Y_i \left(\log_e \frac{Y_i}{1 - Y_i} - \log_e \frac{\widehat{\mu}_i}{1 - \widehat{\mu}_i} \right) \right. \\ &\qquad \left. - \log_e \left[1 + \exp \left(\log_e \frac{Y_i}{1 - Y_i} \right) \right] + \log_e \left[1 + \exp \left(\log_e \frac{\widehat{\mu}_i}{1 - \widehat{\mu}_i} \right) \right] \right\} \\ &= 2 \sum n_i \Big\{ Y_i \left[\log_e Y_i - \log_e (1 - Y_i) - \log_e \widehat{\mu}_i + \log_e (1 - \widehat{\mu}_i) \right] \\ &\qquad + \log_e (1 - Y_i) - \log_e (1 - \widehat{\mu}_i) \Big\} \\ &= 2 \sum n_i \big[Y_i \log_e Y_i - Y_i \log_e (1 - Y_i) - Y_i \log_e \widehat{\mu}_i + Y_i \log_e (1 - \widehat{\mu}_i) \\ &\qquad + \log_e (1 - Y_i) - \log_e (1 - \widehat{\mu}_i) \big] \\ &= 2 \sum n_i \left[Y_i \log_e Y_i - Y_i \log_e \widehat{\mu}_i + (1 - Y_i) \log_e (1 - Y_i) - (1 - Y_i) \log_e (1 - \widehat{\mu}_i) \right] \\ &= 2 \sum \left[n_i \log_e \frac{Y_i}{\widehat{\mu}_i} + n_i (1 - Y_i) \log_e \frac{1 - Y_i}{1 - \widehat{\mu}_i} \right] \end{aligned}$$

- For the Poisson family, $a_i = 1$.

$$\begin{aligned} D(\mathbf{y}; \widehat{\boldsymbol{\mu}}) &= 2 \sum \left[Y_i (\log_e Y_i - \log_e \widehat{\mu}_i) \right] - e^{\log Y_i} + e^{\log_e \widehat{\mu}_i} \\ &= 2 \sum \left[Y_i \log_e \frac{Y_i}{\widehat{\mu}_i} - (Y_i - \widehat{\mu}_i) \right] \end{aligned}$$

- For the gamma family, $a_i = 1$.

$$\begin{aligned} D(\mathbf{y}; \widehat{\boldsymbol{\mu}}) &= 2 \sum \left\{ Y_i \left(-\frac{1}{Y_i} + \frac{1}{\widehat{\mu}_i} \right) + \log_e \left(\frac{1}{Y_i} \right) - \log_e \left(\frac{1}{\widehat{\mu}_i} \right) \right\} \\ &= 2 \sum \left[-1 + \frac{Y_i}{\widehat{\mu}_i} - \log_e Y_i + \log_e \widehat{\mu}_i \right] \\ &= 2 \sum \left(-\log_e \frac{Y_i}{\widehat{\mu}_i} + \frac{Y_i - \widehat{\mu}_i}{\widehat{\mu}_i} \right) \end{aligned}$$

- For the inverse-Gaussian family, $a_i = 1$.

$$\begin{aligned} D(\mathbf{y}; \widehat{\boldsymbol{\mu}}) &= 2 \sum \left[Y_i \left(-\frac{1}{2Y_i^2} + \frac{1}{2\widehat{\mu}_i^2} \right) + \sqrt{2\frac{1}{2Y_i^2}} - \sqrt{2\frac{1}{2\widehat{\mu}_i^2}} \right] \\ &= 2 \sum \left(-\frac{Y_i}{2Y_i^2} + \frac{Y_i}{2\widehat{\mu}_i^2} + \frac{1}{Y_i} - \frac{1}{\widehat{\mu}_i} \right) \\ &= \sum \frac{-\widehat{\mu}_i^2 + Y_i^2 + 2\widehat{\mu}_i^2 - 2Y_i\widehat{\mu}_i}{Y_i\widehat{\mu}_i^2} \\ &= \sum \frac{(Y_i - \widehat{\mu}_i)^2}{Y_i\widehat{\mu}_i^2} \end{aligned}$$

Exercise 15.11

I did the computations with the `glm()` function in R, which accommodates custom link functions defined as a `"link-glm"` objects. For quasi-variances, the link is

```
> explink <- list(
+   linkfun = function(mu) exp(mu),
+   linkinv = function(eta) log(eta),
+   mu.eta = function(eta) 1/eta, # d mu / d eta
+   valideta = function(eta) eta > 0,
+   name = "exp"
+ )
> class(explink) <- "link-glm"
```

Most of the elements of the link object are essentially self-explanatory: The link function is $\eta = g(\mu) = \exp(\mu)$; the inverse link (i.e., the mean function) is $\mu = g^{-1}(\eta) = \log_e \eta$; and the value of the linear predictor η is constrained to be positive. For those with calculus, the `mu.eta` element of the link object is the derivative of the inverse link, $d\mu/d\eta = 1/\eta$.

Then applying this exponential link, along with the constant variance function, produces the following quasi-likelihood estimates of the quasi-variances:

```
> y <- log(c(2.771, 3.867, 2.514)^2)
> X <- matrix(c(1, 1, 0,
+               1, 0, 1,
+               0, 1, 1),
+             3, 3, byrow=TRUE)
>
> (m <- glm(y ~ X - 1, family=quasi(link=explink,
+                                   variance="constant")))

Call:  glm(formula = y ~ X - 1,
           family = quasi(link = explink, variance = "constant"))

Coefficients:
     X1        X2        X3
```



```
 8.1560  -0.4775   6.7977

Degrees of Freedom: 3 Total (i.e. Null);  0 Residual
Null Deviance:      Inf
Residual Deviance: 0    AIC: NA
```

Note the 0 residual deviance, indicating a perfect fit, as should be the case when there are only three categories. In the call to `glm()`, I included `-1` in the model formula to suppress the regression constant, which otherwise would automatically have been included in the model.

Finally, let's verify that we can recover the original variances of the differences:

```
> v <- coef(m)
> all.equal(v[1] + v[2], 2.771^2, check.attributes=FALSE)
[1] TRUE
> all.equal(v[1] + v[3], 3.867^2, check.attributes=FALSE)
[1] TRUE
> all.equal(v[2] + v[3], 2.514^2, check.attributes=FALSE)
[1] TRUE
```

Notice that I used the `all.equal()` function to check for equality within rounding error rather than for exact equality. As usual, when numerical computations on done on a digital computer, the results usually aren't exact, and so checking for exact quality (using the `==` operator) would be a bad idea.

Exercises for Chapter 16

Exercise 16.1*

(a) Under the model, $\mathbf{y} \sim N_n(\mathbf{X}\boldsymbol{\beta}, \boldsymbol{\Sigma}_{\varepsilon\varepsilon})$, so

$$p(\mathbf{y}|\mathbf{X}, \boldsymbol{\Sigma}_{\varepsilon\varepsilon}) = \frac{1}{(2\pi)^{n/2}\sqrt{\det \boldsymbol{\Sigma}_{\varepsilon\varepsilon}}} \exp\left[-\frac{1}{2}(\mathbf{y} - \mathbf{X}\boldsymbol{\beta})'\boldsymbol{\Sigma}_{\varepsilon\varepsilon}^{-1}(\mathbf{y} - \mathbf{X}\boldsymbol{\beta})\right]$$

The log-likelihood, given in the exercise, follows directly.

(b) The only term in the log-likelihood involving $\boldsymbol{\beta}$ is the last, and so differentiating the log-likelihood with respect to $\boldsymbol{\beta}$ yields

$$\begin{aligned}\frac{\partial \log_e L(\boldsymbol{\beta})}{\partial \boldsymbol{\beta}} &= -\frac{1}{2} \times 2 \times \mathbf{X}'\boldsymbol{\Sigma}_{\varepsilon\varepsilon}^{-1}(\mathbf{y} - \mathbf{X}\boldsymbol{\beta}) \\ &= \mathbf{X}'\boldsymbol{\Sigma}_{\varepsilon\varepsilon}^{-1}\mathbf{X}\boldsymbol{\beta} - \mathbf{X}'\boldsymbol{\Sigma}_{\varepsilon\varepsilon}^{-1}\mathbf{y}\end{aligned}$$

Setting the derivatives to zero, substituting $\mathbf{b}_{\text{GLS}}$ for $\boldsymbol{\beta}$, and solving for $\mathbf{b}_{\text{GLS}}$ produces the required result,

$$\mathbf{b}_{\text{GLS}} = (\mathbf{X}'\boldsymbol{\Sigma}_{\varepsilon\varepsilon}^{-1}\mathbf{X})^{-1}\mathbf{X}'\boldsymbol{\Sigma}_{\varepsilon\varepsilon}^{-1}\mathbf{y}$$

The covariance matrix of the GLS estimator is then

$$\begin{aligned}V(\mathbf{b}_{\text{GLS}}) &= (\mathbf{X}'\boldsymbol{\Sigma}_{\varepsilon\varepsilon}^{-1}\mathbf{X})^{-1}\mathbf{X}'\boldsymbol{\Sigma}_{\varepsilon\varepsilon}^{-1}V(\mathbf{y})\boldsymbol{\Sigma}_{\varepsilon\varepsilon}^{-1}\mathbf{X}(\mathbf{X}'\boldsymbol{\Sigma}_{\varepsilon\varepsilon}^{-1}\mathbf{X})^{-1} \\ &= (\mathbf{X}'\boldsymbol{\Sigma}_{\varepsilon\varepsilon}^{-1}\mathbf{X})^{-1}\mathbf{X}'\boldsymbol{\Sigma}_{\varepsilon\varepsilon}^{-1}\boldsymbol{\Sigma}_{\varepsilon\varepsilon}\boldsymbol{\Sigma}_{\varepsilon\varepsilon}^{-1}\mathbf{X}(\mathbf{X}'\boldsymbol{\Sigma}_{\varepsilon\varepsilon}^{-1}\mathbf{X})^{-1} \\ &= (\mathbf{X}'\boldsymbol{\Sigma}_{\varepsilon\varepsilon}^{-1}\mathbf{X})^{-1}\mathbf{X}'\boldsymbol{\Sigma}_{\varepsilon\varepsilon}^{-1}\mathbf{X}(\mathbf{X}'\boldsymbol{\Sigma}_{\varepsilon\varepsilon}^{-1}\mathbf{X})^{-1} \\ &= (\mathbf{X}'\boldsymbol{\Sigma}_{\varepsilon\varepsilon}^{-1}\mathbf{X})^{-1}\end{aligned}$$

(c) Remarkably, the proof for the GLS estimator is identical to that for the WLS given in Exercise 12.4, up to the last step, where we have (using our current notation)

$$V(\widehat{b}_j) = \mathbf{m}_j'\boldsymbol{\Sigma}_{\varepsilon\varepsilon}\mathbf{m}_j + \mathbf{a}_j'\boldsymbol{\Sigma}_{\varepsilon\varepsilon}\mathbf{a}_j$$

The error-covariance matrix $\boldsymbol{\Sigma}_{\varepsilon\varepsilon}$ is positive-definite, and so the term $\mathbf{a}_j'\boldsymbol{\Sigma}_{\varepsilon\varepsilon}\mathbf{a}_j$ is non-negative and can only be 0 if $\mathbf{a}_j = \mathbf{0}$. Thus, the matrix $\mathbf{A}$, giving the difference between the BLUE (best linear unbiased estimator) and the GLS estimator must be $\mathbf{0}$, showing that the GLS estimator *is* the BLUE.

Exercise 16.3

(a) The proof that $E(\mathbf{b}) = \boldsymbol{\beta}$ depends only on the assumption of linearity—that is, $E(\boldsymbol{\varepsilon}) = \mathbf{0}$ or equivalently $E(\mathbf{y}) = \mathbf{X}\boldsymbol{\beta}$—and not on the assumptions that the errors are independent.

(b) The variance of the OLS estimator is

$$\begin{aligned}V(\mathbf{b}) &= (\mathbf{X}'\mathbf{X})^{-1}\mathbf{X}'V(\mathbf{y})[(\mathbf{X}'\mathbf{X})^{-1}\mathbf{X}']' \\ &= (\mathbf{X}'\mathbf{X})^{-1}\mathbf{X}'\boldsymbol{\Sigma}_{\varepsilon\varepsilon}\mathbf{X}(\mathbf{X}'\mathbf{X})\end{aligned}$$

because $V(\mathbf{y}) = V(\boldsymbol{\varepsilon}) = \boldsymbol{\Sigma}_{\varepsilon\varepsilon}$.

(c) As usual, I used R to perform the necessary computations. I'll illustrate for $\rho = .5$ and then present the results for all three values of ρ. In the computations below, I set the variance of the shocks $\sigma_\nu^2 = 1$, which is inessential because it affects all of the coefficient variances proportionally.

First, I'll form the model matrix $\mathbf{X}$ and the covariance matrix of the errors, $\boldsymbol{\Sigma}_{\boldsymbol{\varepsilon\varepsilon}}$:


```
> library("MASS") # for fractions()
> (X <- cbind(1, 1:10))
      [,1] [,2]
 [1,]    1    1
 [2,]    1    2
 [3,]    1    3
 [4,]    1    4
 [5,]    1    5
 [6,]    1    6
 [7,]    1    7
 [8,]    1    8
 [9,]    1    9
[10,]    1   10
> rho <- .5
> Sigma <- matrix(0, 10, 10)
> for (i in 1:10){
+     for (j in 1:10){
+         Sigma[i, j] <- rho^abs(i - j)/(1 - rho^2)
+     }
+ }
> fractions(Sigma)
      [,1]  [,2]  [,3]  [,4]  [,5]  [,6]  [,7]  [,8]  [,9]  [,10]
 [1,]  4/3   2/3   1/3   1/6  1/12  1/24  1/48  1/96 1/192 1/384
 [2,]  2/3   4/3   2/3   1/3   1/6  1/12  1/24  1/48  1/96 1/192
 [3,]  1/3   2/3   4/3   2/3   1/3   1/6  1/12  1/24  1/48  1/96
 [4,]  1/6   1/3   2/3   4/3   2/3   1/3   1/6  1/12  1/24  1/48
 [5,] 1/12   1/6   1/3   2/3   4/3   2/3   1/3   1/6  1/12  1/24
 [6,] 1/24  1/12   1/6   1/3   2/3   4/3   2/3   1/3   1/6  1/12
 [7,] 1/48  1/24  1/12   1/6   1/3   2/3   4/3   2/3   1/3   1/6
 [8,] 1/96  1/48  1/24  1/12   1/6   1/3   2/3   4/3   2/3   1/3
 [9,] 1/192 1/96  1/48  1/24  1/12   1/6   1/3   2/3   4/3   2/3
[10,] 1/384 1/192 1/96  1/48  1/24  1/12   1/6   1/3   2/3   4/3
```

It's then a straightforward matter to compute the coefficient standard deviations (that is, the square-roots of the coefficient variances) for the OLS and the GLS estimators:

```
> XtXi <- solve(t(X) %*% X)
> V_OLS <- XtXi %*% t(X) %*% Sigma %*% X %*% XtXi
> sqrt(diag(V_OLS))
[1] 1.1283387 .1750022
> V_GLS <- solve(t(X) %*% solve(Sigma) %*% X)
> sqrt(diag(V_GLS))
[1] 1.0915536 .1684304
```

The results for all three values of ρ are shown in the answer to part (d).

(d)* To compute the standard deviations of the estimator dropping the first transformed observation, I'll first find the transformation matrix $\mathbf{\Gamma}$ and verify that if I use *all* 10 observations, I get the same GLS standard errors that I computed above:

```
> Gamma <- matrix(0, 10, 10)
> diag(Gamma) <- 1
> for (i in 2:10){
+     Gamma[i, i-1] <- -rho
+ }
> Gamma[1, 1] <- sqrt(1 - rho^2)
> round(Gamma, 3)
       [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10]
 [1,]  0.866  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0     0
 [2,] -0.500  1.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0     0
```



```
 [3,]  0.000 -0.5  1.0  0.0  0.0  0.0  0.0  0.0  0.0    0
 [4,]  0.000  0.0 -0.5  1.0  0.0  0.0  0.0  0.0  0.0    0
 [5,]  0.000  0.0  0.0 -0.5  1.0  0.0  0.0  0.0  0.0    0
 [6,]  0.000  0.0  0.0  0.0 -0.5  1.0  0.0  0.0  0.0    0
 [7,]  0.000  0.0  0.0  0.0  0.0 -0.5  1.0  0.0  0.0    0
 [8,]  0.000  0.0  0.0  0.0  0.0  0.0 -0.5  1.0  0.0    0
 [9,]  0.000  0.0  0.0  0.0  0.0  0.0  0.0 -0.5  1.0    0
[10,]  0.000  0.0  0.0  0.0  0.0  0.0  0.0  0.0 -0.5    1
> Xs <- Gamma %*% X
> V_GLS_t <- solve(t(Xs) %*% Xs)
> sqrt(diag(V_GLS_t)) # check
[1] 1.0915536 .1684304
```

Then, dropping the first observation,

```
> V_GLS_t1 <- solve(t(Xs[-1, ]) %*% Xs[-1, ])
> sqrt(diag(V_GLS_t1))
[1] 1.9264244 .2581989
```

The results for the standard deviation of β for all three values of ρ are

ρ	OLS	GLS	GLS_{-1}
0	.1101	.1101	.1291
.5	.1750	.1684	.2582
.9	.3005	.2821	1.2910

where GLS_{-1} indicates deleting the first transformed observation. Notice that when $\rho = 0$, the coefficient standard deviations for the OLS and GLS estimators are the same, as should be the case. For this configuration of X-values, GLS is only slightly better than OLS, and the small improvement for GLS depends crucially on *retaining* the first transformed observation, especially when ρ is large.

If there are more observations, then the effect of removing the first observation gets smaller. For example, I repeated the computations shown above but with $n = 100$ observations, repeating each of X-values 10 times, producing the following results:

ρ	OLS	GLS	GLS_{-1}
0	.03482	.03482	.03526
.5	.04685	.03997	.04048
.9	.05865	.03654	.03665

(e) I repeated the computations in part (d) but for $x_t = (t-5)^2$ rather than for $x_t = t$, $t = 1, 2, \ldots 9$, producing the following results:

ρ	OLS	GLS	GLS_{-1}
0	.05698	.05698	.06901
.5	.07390	.07131	.08559
.9	.08370	.07918	.08077

Now the advantage of the GLS estimator is even smaller and dropping the first observation makes little difference.

Exercise 16.5*

There's an error in the hints for the exercise: Actually, not $\boldsymbol{\Sigma}_{\varepsilon\varepsilon}$ but $\boldsymbol{\Sigma}_{\varepsilon\varepsilon}^{-1} = (1/\sigma_\nu^2)\boldsymbol{\Gamma}'\boldsymbol{\Gamma}$, and so $\det \boldsymbol{\Sigma}_{\varepsilon\varepsilon} = (\sigma_\nu^2)^n (1/\det \boldsymbol{\Gamma})^2$. Applying the (corrected) hints given in the exercise, and substituting into

Equation 16.1 in the text, we have

$$\begin{aligned}
\log_e(\boldsymbol{\beta}, \rho, \sigma_\nu^2) &= -\frac{n}{2}\log_e 2\pi - \frac{1}{2}\log_e\left[\left(\sigma_\nu^2\right)^n \frac{1}{1-\rho^2}\right] - \frac{1}{2}(\mathbf{y}-\mathbf{X}\boldsymbol{\beta})'\frac{1}{\sigma_\nu^2}\boldsymbol{\Gamma}'\boldsymbol{\Gamma}(\mathbf{y}-\mathbf{X}\boldsymbol{\beta}) \\
&= -\frac{n}{2}\log_e 2\pi - \frac{n}{2}\log_e \sigma_\nu^2 + \frac{1}{2}\log_e(1-\rho^2) - \frac{1}{2\sigma_\nu^2}(\boldsymbol{\Gamma}\mathbf{y}-\boldsymbol{\Gamma}\mathbf{X}\boldsymbol{\beta})'(\boldsymbol{\Gamma}\mathbf{y}-\boldsymbol{\Gamma}\mathbf{X}\boldsymbol{\beta}) \\
&= -\frac{n}{2}\log_e 2\pi - \frac{n}{2}\log_e \sigma_\nu^2 + \frac{1}{2}\log_e(1-\rho^2) - \frac{1}{2\sigma_\nu^2}(\mathbf{y}^*-\mathbf{X}^*\boldsymbol{\beta})'(\mathbf{y}^*-\mathbf{X}^*\boldsymbol{\beta})
\end{aligned}$$

Exercise 16.7

(a) I used R to fit the OLS regression, and functions in the **car** package for various regression diagnostics:

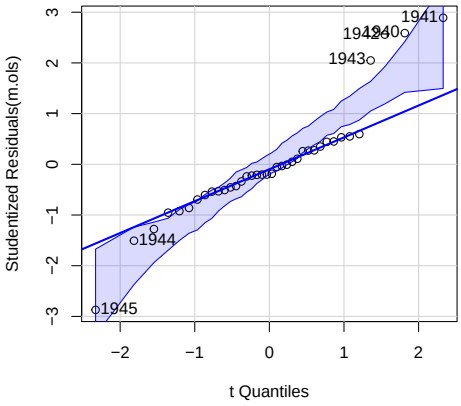
```
> library("car") # for data and diagnostics
Loading required package: carData
> rownames(Hartnagel) <- Hartnagel$year

> m.ols <- lm(fconvict ~ tfr + partic + degrees + mconvict,
+             data=Hartnagel)
> summary(m.ols) # matching Equation 16.18

Call:
lm(formula = fconvict ~ tfr + partic + degrees + mconvict,
          data = Hartnagel)

Residuals:
    Min      1Q  Median      3Q     Max
-42.964  -9.204  -3.566   6.149  48.385

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept) 127.639997  59.957044   2.129   0.0408 *
tfr          -0.046567   0.008033  -5.797 1.75e-06 ***
partic        0.253416   0.115132   2.201   0.0348 *
degrees      -0.212049   0.211454  -1.003   0.3232
mconvict      0.059105   0.045145   1.309   0.1995
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 19.19 on 33 degrees of freedom
Multiple R-squared:  0.6948,    Adjusted R-squared:  0.6578
F-statistic: 18.78 on 4 and 33 DF,  p-value: 3.905e-08
```

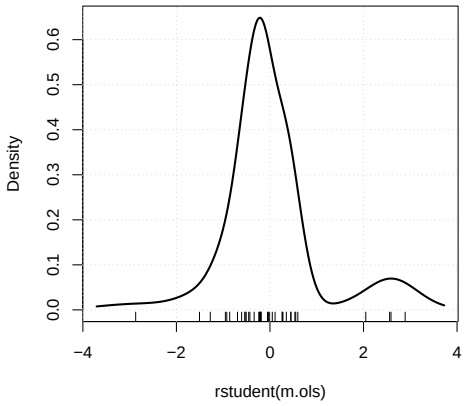
I performed a variety of regression diagnostics, including the following (with the resulting graphs shown following the R code):

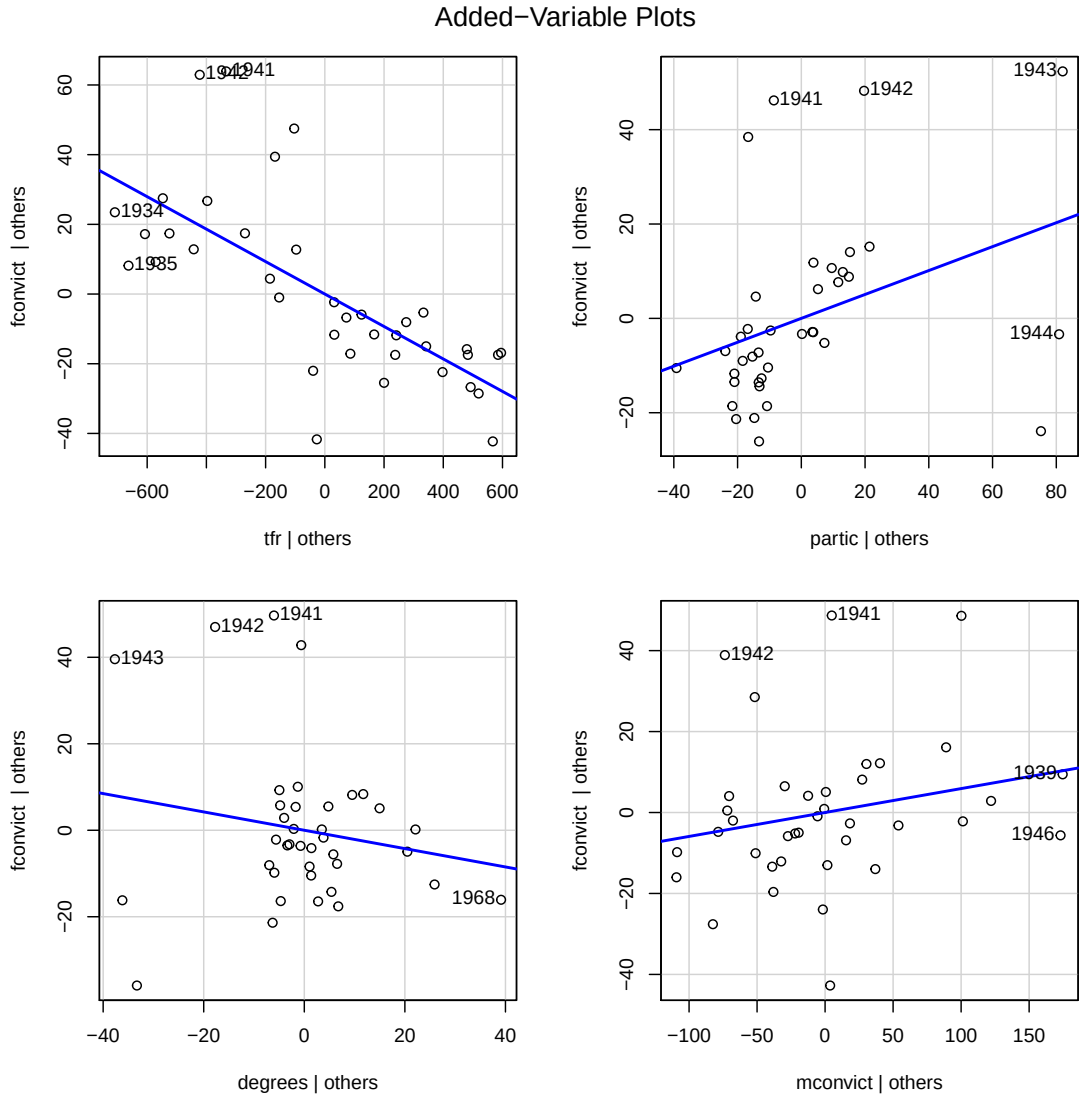
```
> qqPlot(m.ols, id=list(n=6), line="quartiles")
1940 1941 1942 1943 1944 1945
  10   11   12   13   14   15

> densityPlot(rstudent(m.ols))

> avPlots(m.ols)
> crPlots(m.ols)

> spreadLevelPlot(m.ols)
```

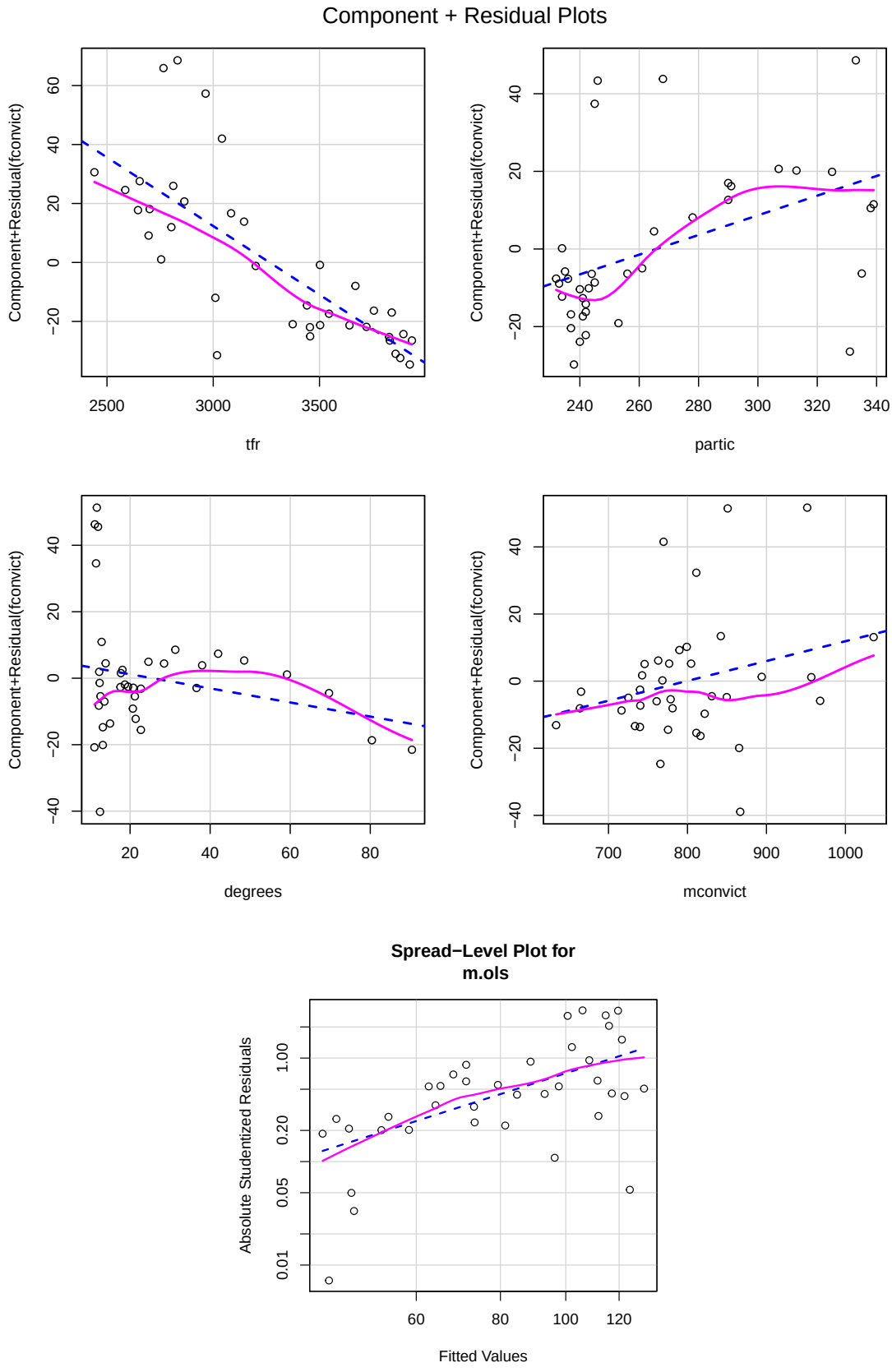


```
Suggested power transformation:  -1.080299

> ncvTest(m.ols)
Non-constant Variance Score Test
Variance formula: ~ fitted.values
Chisquare = 12.10765, Df = 1, p = 0.00050215
```









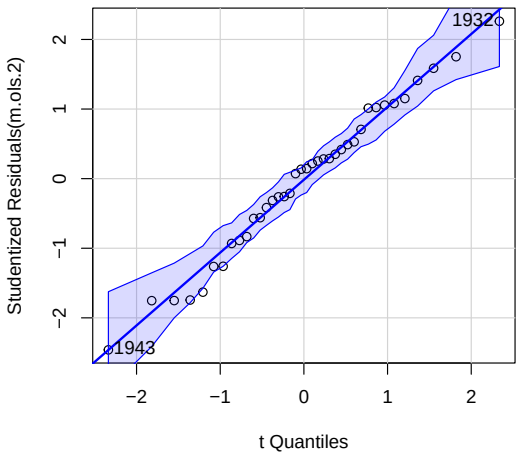
The most striking finding from the diagnostics is that there's something unusual going on during the early to mid-1940s—roughly the years of World War II—and that the years 1940–1943 appear similar to each other as do the years 1944 and 1945. The problem with the war years was also mentioned in the text; see in particular the time-series plot of residuals in Figure 16.6 (page 491).

The data set is small, and the risk of overfitting the data is high, but I decided to experiment with creating two dummy regressors corresponding to these two periods. The resulting model produces a noticeably different coefficient for post-secondary degrees, but in either case is the coefficient large relative to its standard error:

```
> Hartnagel$war1 <- with(Hartnagel, year >= 1940 & year <= 1943)
> Hartnagel$war2 <- with(Hartnagel, year >= 1944 & year <= 1945)
> m.ols.2 <- update(m.ols, . ~ . + war1 + war2)
> compareCoefs(m.ols, m.ols.2)
Calls:
1: lm(formula = fconvict ~ tfr + partic + degrees + mconvict,
         data = Hartnagel)
2: lm(formula = fconvict ~ tfr + partic + degrees + mconvict
         + war1 + war2, data = Hartnagel)

             Model 1  Model 2
(Intercept)    127.6     84.5
SE              60.0     27.8

tfr          -0.04657 -0.03775
SE            0.00803  0.00356

partic         0.2534   0.2584
SE             0.1151   0.0826

degrees       -0.2120  -0.0471
SE             0.2115   0.1452

mconvict       0.0591   0.0651
SE             0.0451   0.0190

war1TRUE                 50.65
SE                        5.71

war2TRUE                 -25.7
SE                        10.1
```

The diagnostics for the respecified model are improved. Here, for example, are the test for nonconstant error variance and the QQ plot of the studentized residuals:

```
> ncvTest(m.ols.2)
Non-constant Variance Score Test
Variance formula: ~ fitted.values
Chisquare = 0.467855, Df = 1, p = 0.49398

> qqPlot(m.ols.2)
1932 1943
   2   13
```

(b) Using the quadratic formula,

$$\beta = \frac{-(-1.068) \pm \sqrt{(-1.068)^2 - 4 \times 0.5507 \times 1}}{2 \times 0.5507}$$

The two roots resulting both have an imaginary component:

$$\beta_1 = 0.9697 - 0.9357i$$
$$\beta_2 = 0.9697 + 0.9357i$$

The roots have the same modulus, $\sqrt{0.9697^2 + 0.9357^2} = 1.347 > 1$.

(c) Unless you have access to an EGLS program, this problem requires nontrivial custom programming, and so I should probably have starred it. For example, I'm unaware of an EGLS function for R. To simplify the task, I wrote an R function for an AR(2) process for the errors, rather than for the general case of AR(p) errors. I also didn't try to make the program computationally efficient.

```
> egls2 <- function(X, y){
+   X <- cbind(1, X) # constant regressor
+   colnames(X)[1] <- "intercept"
+   n <- nrow(X)
+   ols <- lm(y ~ X - 1) # preliminary OLS fit
+   r <- as.vector(acf(ols$residuals, lag.max=2,
+                      plot=FALSE)$acf)[2:3]
+   P <- diag(2)
+   P[1, 2] <- P[2, 1] <- r[1]
+   phi <- solve(P, r) # AR(2) parameters by Yule-Walker
+   rho <- numeric(n) # autocorrelations of errors to lag n - 1
+   rho[1] <- 1 # rho_0
+   rho[2] <- phi[1]/(1 - phi[2]) # rho_1
+   for (i in 3:n){ # rho_3 to rho_(n - 1)
+     rho[i] <- phi[1]*rho[i - 1] + phi[2]*rho[i - 2]
+   }
+   P <- diag(n) # matrix of error autocorrelations
+   for (i in 1:(n - 1)){
+     P[i, (i + 1):n] <- rho[2:(n - i + 1)]
+   }
+   P[lower.tri(P)] <- rev(P[upper.tri(P)])
+   T <- chol(solve(P)) # GLS transformation
```



```
+   X <- T %*% X # transformed data
+   y <- T %*% y
+   result <- lm(y ~ X - 1) # GLS fit
+   result$phi <- phi
+   result
+ }
```

Applied to the Fox and Hartnagel regression, I get results very similar to those produced by maximum-likelihood (cf., Equation 16.19 on page 493):

```
> m.egls <- with(Hartnagel,
+                egls2(cbind(tfr, partic, degrees, mconvict),
+                      fconvict))
> coef(m.egls) # estimates
 Xintercept        Xtfr     Xpartic    Xdegrees   Xmconvict
85.10848214 -0.04002180  0.28338370 -0.20739153  0.07494638
> sqrt(diag(vcov(m.egls))) # standard errors
  Xintercept         Xtfr      Xpartic     Xdegrees    Xmconvict
59.746237062  0.009327557  0.112417552  0.207437858  0.035172106
> m.egls$phi # AR(2) parameter estimates
[1]  1.0566689 -0.5350862
```


Exercises for Chapter 17

Exercise 17.1*

Here's a table showing the several measures for each of the five models:

Model	Metric Effect $\partial Y/\partial X_1$	Proportional Change $X_1(\partial Y/\partial X_1)$	Rate of Return $(\partial Y/\partial X_1)/Y$	Point Elasticity $(\partial Y/\partial X_1)/ \times (X_1/Y)$
(a) $Y = \alpha + \beta_1 X_1 + \beta_2 X_2$	β_1	$\beta_1 X_1$	β_1/Y	$\beta_1 X_1/Y$
(b) $Y = \alpha + \beta_1 X_1 + \beta_2 X_1^2 + \beta_3 X_2$	$\beta_1 + 2\beta_2 X_1$	$\beta_1 X_1 + 2\beta_2 X_1^2$	$(\beta_1 + 2\beta_2 X_1)/Y$	$(\beta_1 X_1 + 2\beta_2 X_1^2)/Y$
(c) $Y = \alpha + \beta_1 X_1 + \beta_2 X_2 + \beta_3 X_1 X_2$	$\beta_1 + \beta_3 X_2$	$\beta_1 X_1 + \beta_3 X_1 X_2$	$(\beta_1 + \beta_3 X_2)/Y$	$(\beta_1 X_1 + \beta_3 X_1 X_2)/Y$
(d) $Y = \exp(\alpha + \beta_1 X_1 + \beta_2 X_2)$	$\beta_1 \exp(Y)$	$\beta_1 X_1 \exp(Y)$	β_1	$\beta_1 X_1$
(e) $Y = \alpha X_1^{\beta_1} X_2^{\beta 2}$	$\beta_1 Y/X_1$	$\beta_1 Y$	β_1/X_1	β_1

The simplest measure for model (a) is the metric effect; for (b) and (c) probably also the metric effect, though this changes with the level of X_1 in (b) and of X_2 in (c); for (d) the instantaneous rate of return; and for (e) the point elasticity.

The metric effect is interpretable as the instantaneous "effect" of increasing X_1 by one unit holding X_2 constant. The effect of proportional change in X_1 is the instantaneous effect of increasing X_1 by an amount equal to its size, holding X_2 constant. The rate of return is the instantaneous effect of increasing X_1 by one unit, holding X_2 constant, as a proportion of the size of Y. The point elasticity is the approximate percentage change in Y for a one-percent increment in X_1, holding X_2 constant. I hope that it's clear that the language of "effect," "change," and "holding constant" doesn't necessarily imply a causal interpretation of the regression coefficient β_1, but rather is adopted to avoid awkward language. As well, when the partial relationship of Y to X_1 is nonlinear holding X_2 constant, instantaneous effects are extrapolations based on the slope of the regression function in the direction of X_1 at a particular point.

Models (a), (b), and (c) are linear in the parameters and, assuming independent additive errors with equal variances, could be fit by OLS regression. Assuming a multiplicative error and positive Y, model (d) could be fit by OLS regression of $\log Y$ on X_1 and X_2. Assuming multiplicative errors and positive Y, model (e) could be fit by OLS regression of $\log Y$ on $\log X_1$ and $\log X_2$.

Exercise 17.3

I wrote a simple R function that uses the `wireframe()` function from the standard **lattice** package to draw 3D graphs of quadratic regression surfaces. My function takes the regression coefficients and the values of X_1 and X_2 over which to plot as arguments:

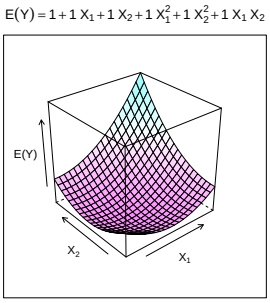
```
> f <- function(beta0=1, beta1=1, beta2=1, beta3=1, beta4=1, beta5=1,
+                x1=seq(-1, 1, length=20),
+                x2=seq(-1, 1, length=20),
+                color=TRUE, ...){
+   grid <- expand.grid(x1=x1, x2=x2)
+   grid$y <- with(grid, beta0 + beta1*x1 + beta2*x2 + beta3*x1^2 +
+                        beta4*x2^2 + beta5*x1*x2)
+   lattice::wireframe(y ~ x1 + x2, data=grid, xlab=expression(X[1]),
+             ylab=expression(X[2]), zlab="E(Y)", drape=color,
+             main=bquote(E(Y) == .(beta0) + .(beta1)~X[1] + .(beta2)~X[2]
+                          + .(beta3)~X[1]^2 + .(beta4)~X[2]^2
+                          + .(beta5)~X[1]~X[2]),
+             colorkey=FALSE, ...)
+ }
```

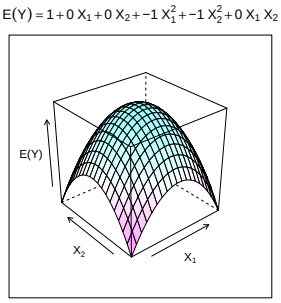
The question is open-ended and so I'll show three examples, for $E(Y) = 1 + X_1 + X_2 + X_1^2 + X_2^2 + X_1 X_2$; $E(Y) = 1 - X_1^2 - X_2^2$; and $E(Y) = 1 + X_1 + X_2 + X_1^2 + X_2^2 + 2X_1 X_2$. In the first two cases, X_1 and X_2 range from -1 to 1; in the third case, they range from 0 to 10. The equations appear (in slightly crude form) in the labels above the graphs:

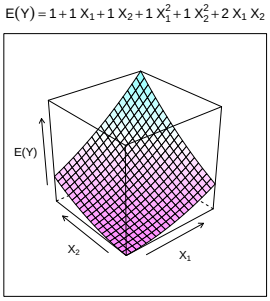
```
> f()
```



```
> f(beta1=0, beta2=0, beta3 = -1, beta4 = -1, beta5 = 0)
> f(beta5=2, x1=seq(0, 10, length=20), x2=seq(0, 10, length=20))
```







The graph of $E(Y) = 1 + X_1 + X_2 + X_1^2 + X_2^2 + X_1X_2$ is similar to the graph in Figure 17.1 (a). The graph of $E(Y) = 1 - X_1^2 - X_2^2$ is concave downwards ("spills water") rather than concave upwards, and the partial relationship of Y to each X doesn't change its shape with the value of the other X (that is the lines drawn on the regression surface in the direction of each X are parallel). The graph of $E(Y) = 1 + X_1 + X_2 + X_1^2 + X_2^2 + 2X_1X_2$ for non-negative values of X_1 and X_2 is monotone increasing in X_1 and X_2.

I encourage you to examine some other examples, either using my `f()` function or something else to draw the graphs.

Exercise 17.5*

Let's set X_2 to two different arbitrary values, x_2' and x_2''. Then

$$\mu' = \beta_0 + \beta_1 X_1 + \beta_2 x_2' + \beta_3 X_1 x_2'$$
$$\mu'' = \beta_0 + \beta_1 X_1 + \beta_2 x_2'' + +\beta_3 X_1 x_2''$$

The lines cross when $\mu' = \mu''$, that is for

$$\beta_0 + \beta_1 X_1 + \beta_2 x_2' + \beta_3 X_1 x_2' = \beta_0 + \beta_1 X_1 + \beta_2 x_2'' + \beta_3 X_1 x_2''$$

Subtracting $\beta_0 + \beta_1 X_1$ from both sides and factoring out the values of X_2, we have

$$x_2'(\beta_2 + \beta_3 X_1) = x_2''(\beta_2 + \beta_3 X_1)$$

The only way this can be true for all pairs of values x_2', x_2'' of X_2 is if $\beta_2 + \beta_3 X_1 = 0$, which implies that $x_1 = -\beta_3/\beta_2$ is the value of X_1 above which the lines cross.

Exercise 17.7

The Canadian interprovincial migration data are in the `Migration` data set in the **carData** package for R. This data set isn't in the tabular form of Tables 17.4 and 17.5, but rather has one row for each of the 90 migration streams:


```
> library("car") # for data and Tapply()
Loading required package: carData

> head(Migration)
  source destination migrants distance  pops66  pops71 popd66 popd71
1    PEI         NFLD      255      924  108535  111641 493396 522104
2     NS         NFLD     2380      952  756039  788960 493396 522104
3     NB         NFLD     1140     1119  616788  534557 493396 522104
4    QUE         NFLD     2145     1641 5780845 6027764 493396 522104
5    ONT         NFLD     6295     1996 6960870 7703106 493396 522104
6    MAN         NFLD      215     3159  963066  988247 493396 522104

> tail(Migration)
   source destination migrants distance  pops66  pops71  popd66  popd71
85     NB          BC     3115     3719  616788  534557 1873674 2184621
86    QUE          BC    16740     3197 5780845 6027764 1873674 2184621
87    ONT          BC    47395     3059 6960870 7703106 1873674 2184621
88    MAN          BC    26910     1679  963066  988247 1873674 2184621
89   SASK          BC    29920     1297  955344  926242 1873674 2184621
90   ALTA          BC    58915      987 1463203 1627874 1873674 2184621
```

I begin with some data management to average 1966 and 1971 provincial population, as suggested in the exercise, and to order the provinces (roughly) from west to east rather than alphabetically. Then, using `lm()`, I fit the gravity model to the data by least-squares:

```
> Migration$source <- factor(Migration$source, # west to east
+                            levels=c("BC", "ALTA", "SASK", "MAN", "ONT",
+                                     "QUE", "NB", "PEI", "NS", "NFLD"))
> Migration$destination <- factor(Migration$destination,
+                            levels=c("BC", "ALTA", "SASK", "MAN", "ONT",
+                                     "QUE", "NB", "PEI", "NS", "NFLD"))
> Migration$pops <- rowMeans(Migration[, c("pops66", "pops71")])
> Migration$popd <- rowMeans(Migration[, c("popd66", "popd71")])
> m <- lm(log(migrants) ~ log(pops) + log(popd) + I(-log(distance)),
+         data=Migration)
> summary(m)

Call:
lm(formula = log(migrants) ~ log(pops) + log(popd) + I(-log(distance)),
    data = Migration)

Residuals:
    Min      1Q  Median      3Q     Max
-1.8381 -0.6738  0.1105  0.4699  1.8691

Coefficients:
                 Estimate Std. Error t value Pr(>|t|)
(Intercept)       -8.32094    1.66914  -4.985 3.18e-06 ***
log(pops)          0.76064    0.07439  10.226  < 2e-16 ***
log(popd)          0.87620    0.07439  11.779  < 2e-16 ***
I(-log(distance))  0.88379    0.10483   8.431 7.08e-13 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.8098 on 86 degrees of freedom
Multiple R-squared:  0.7587,    Adjusted R-squared:  0.7503
F-statistic: 90.16 on 3 and 86 DF,  p-value: < 2.2e-16
```

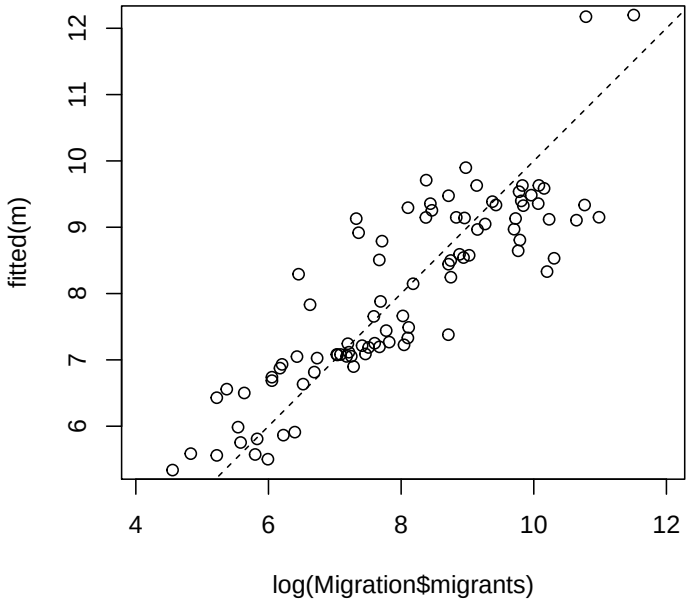

Thus, the estimated model is

$$
\begin{aligned}
Y_{ij} &= \exp(-8.321)\frac{P_i^{0.7606}P_j^{0.8762}}{D_{ij}^{0.8838}} \\
&= 0.0002433\frac{P_i^{0.7606}P_j^{0.8762}}{D_{ij}^{0.8838}}
\end{aligned}
$$

,

The R^2 of .76 may seem impressive, but for a model of this type, it indicates potentially important predictive failure. Here, for example, is a plot of fitted values (on the log scale) against log-migration (the response in the linearized model):

```
> library("MASS") # for eqscplot()
> eqscplot(log(Migration$migrants), fitted(m))
> abline(0, 1, lty=2)  # y = x
```



Some of the departures of the points from the line represent substantial failures to predict migration when expressed on the scale of numbers of migrants:

```
> Migration$fit <- exp(fitted(m)) # fitted values on population scale
> Migration$res <- with(Migration, migrants - fit) # residuals
>
> table <- round(Tapply(res ~ source + destination, c, data=Migration))
> table  <- cbind(table, sum=rowSums(table, na.rm=TRUE))
> table <- rbind(table, sum=colSums(table, na.rm=TRUE))
> names(dimnames(table)) <- c("source", "destination")
> table
      destination
source      BC  ALTA  SASK   MAN    ONT     QUE    NB   PEI    NS  NFLD     sum
  BC        NA 18637  1449  2496   8064   -5714     5    67  1056  -420   25640
  ALTA   49495    NA  2074  1804   3808   -7573   -60   -75   590  -377   49686
```



```
SASK   24848 32901    NA  8499   -114   -7709  -488  -113  -286  -435   57103
MAN    22759 11730  1592    NA   8570   -6932   247   155   502  -490   38133
ONT    36067 11983 -2556  6193     NA -145386  1123   933  7584  1387  -82672
QUE     7500 -1567 -5890 -5060 -99236      NA -5887 -1764 -6938 -2804 -121646
NB      1739   816  -530   124   3345  -11989    NA   154  3003   -42   -3380
PEI      231   152  -142   -51    114   -3356   -64    NA  -462  -143   -3721
NS      4472  1773  -532   296  11176  -12130  2506  -138    NA   675    8098
NFLD     464  -107  -386   -81  11179   -4336   541     7  1551    NA    8832
sum   147575 76318 -4921 14220 -53094 -205125 -2077  -774  6600 -2649  -23927
```

The model does a particularly poor job in some of the provinces, such as British Columbia and Quebec. Adding dummy variables for the provinces of origin or destination (or both) won't work, because provincial population and log-population would be perfectly collinear with either set of dummy regressors. A possibly better strategy would be to add explanatory variables based on characteristics of the provinces in addition to their population, such as unemployment rates, living costs, and wage rates.

Exercise 17.9

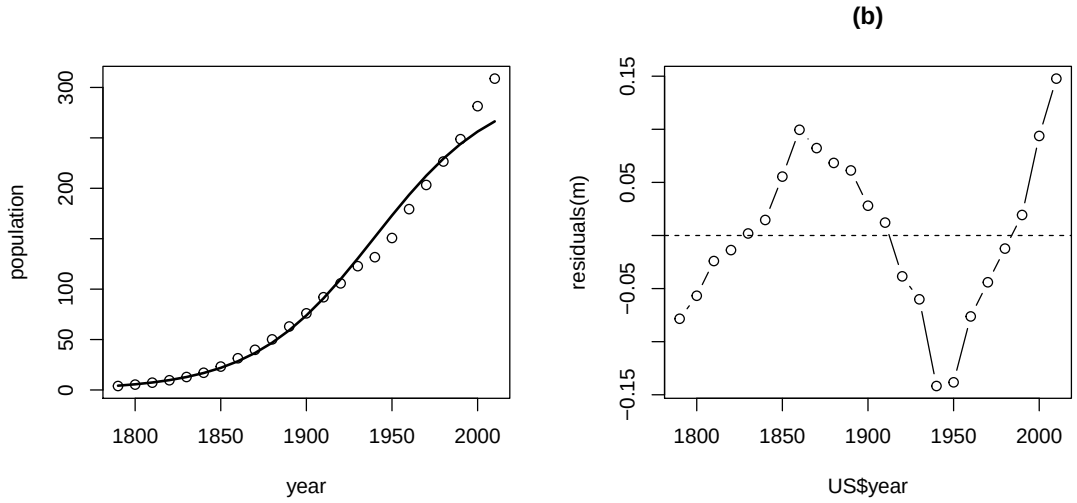
I'll use the data read in the preceding exercise, and again use the R `nls()` function to fit the model. To fit the logistic-growth model with multiplicative errors, I just take the log of both sides of the model:

```
> m <- nls(log(population) ~ log(beta1/(1 + exp(beta2 + beta3*decade))),
+          data=US, start=c(beta1=350, beta2=4.5, beta3=-0.3))
> summary(m)

Formula: log(population) ~ log(beta1/(1 + exp(beta2 + beta3 * decade)))

Parameters:
       Estimate Std. Error t value Pr(>|t|)
beta1 302.71023   16.63508   18.20  6.5e-14 ***
beta2   4.25184    0.05466   77.79  < 2e-16 ***
beta3  -0.28378    0.00665  -42.67  < 2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.07831 on 20 degrees of freedom

Number of iterations to convergence: 3
Achieved convergence tolerance: 3.241e-06
```

The estimated parameters are quite close to those produced by assuming additive errors; except for $\mathrm{SE}(\widehat{\beta}_1)$, the standard errors of the parameter estimates are also similar.

Here are plots similar to those in Figure 17.9 (page 520) in the text. The plot for population is on the scale of millions of people, while the residuals are plotted on the log scale:

```
> plot(population ~ year, data=US)
> lines(US$year, exp(fitted(m)), lwd=2, main="(a)")
> plot(US$year, residuals(m), type="b", main="(b)")
> abline(h=0, lty=2)

> print(acf(residuals(m), lag.max=5))

Autocorrelations of series 'residuals(m)', by lag

     0      1      2      3      4      5
 1.000  0.753  0.438  0.154 -0.133 -0.402
```

In comparison to the model with additive errors, the residuals are even more autocorrelated and the fitted model does a poorer job in later decades (but a better job in the earlier decades), substantially underestimating population at the last two Censuses. Neither model seems entirely adequate—U.S. population growth is more complicated than logistic growth.

Exercise 17.11

I'll begin by using the `boxTidwell()` function in the **car** package for R to reproduce the results in Section 12.5.2. For comparison with the results reported below, I also fit the regression by linear least-squares after transforming age and education:

```
> library("car") # for boxTidwell()
Loading required package: carData
> url <- paste("https://socialsciences.mcmaster.ca", "jfox", "Books",
+              "Applied-Regression-3E", "datasets", "SLID-Ontario.txt",
+              sep="/")
> SLID <- read.table(url, header=TRUE)

> boxTidwell(log(compositeHourlyWages) ~
+              I(age - 15) + I(yearsEducation + 1), ~ sex,
+            data=SLID)
                       MLE of lambda Score Statistic (z)  Pr(>|z|)
I(age - 15)                 0.050965            -20.3694 < 2.2e-16 ***
I(yearsEducation + 1)       1.893103              4.4761 7.601e-06 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

> summary(lm(log(compositeHourlyWages) ~
+              I((age - 15)^0.050965) +
+                I((yearsEducation + 1)^1.893103) + sex,
+            data=SLID))

Call:
lm(formula = log(compositeHourlyWages) ~ I((age - 15)^0.050965) +
    I((yearsEducation + 1)^1.893103) + sex, data = SLID)

Residuals:
     Min       1Q   Median       3Q      Max
-2.28198 -0.24886  0.02405  0.26063  1.76219
```



```
Coefficients:
                                  Estimate Std. Error t value Pr(>|t|)
(Intercept)                     -4.1428856  0.1666483  -24.86   <2e-16 ***
I((age - 15)^0.050965)           5.4113341  0.1433974   37.74   <2e-16 ***
I((yearsEducation + 1)^1.893103) 0.0024334  0.0001028   23.67   <2e-16 ***
sexMale                          0.2209610  0.0126427   17.48   <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.3996 on 3993 degrees of freedom
Multiple R-squared:  0.3696,    Adjusted R-squared:  0.3691
F-statistic: 780.4 on 3 and 3993 DF,  p-value: < 2.2e-16
```

Then, I'll use the `nls()` function to fit the Box-Tidwell model by general nonlinear least-squares, simultaneously estimating the transformation and regression parameters:

```
> SLID$male <- with(SLID, as.numeric(sex == "Male"))
> coef(lm(log(compositeHourlyWages) ~
+       I(age - 15) + I(yearsEducation + 1) + male, data=SLID))
          (Intercept)            I(age - 15) I(yearsEducation + 1)
            1.31546388             0.01815485             0.05587644
                  male
            0.22449593

> m.nls <- nls(log(compositeHourlyWages) ~ beta0 +
+                 beta1*(age - 15)^gamma1 +
+                 beta2*(yearsEducation + 1)^gamma2 +
+                 beta3*male, data=SLID,
+               start=c(beta0=1.3, beta1=0.018, beta2=0.056, beta3=0.22,
+                       gamma1=1, gamma2=1),
+               control=nls.control(maxiter=1000))
> summary(m.nls)

Formula: log(compositeHourlyWages) ~ beta0 + beta1 * (age - 15)^gamma1 +
    beta2 * (yearsEducation + 1)^gamma2 + beta3 * male

Parameters:
        Estimate Std. Error t value Pr(>|t|)
beta0  -4.142732   6.295615  -0.658    0.511
beta1   5.411181   6.270220   0.863    0.388
beta2   0.002433   0.002236   1.088    0.277
beta3   0.220961   0.012668  17.442  < 2e-16 ***
gamma1  0.050966   0.052556   0.970    0.332
gamma2  1.893101   0.284275   6.659 3.12e-11 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.3997 on 3991 degrees of freedom

Number of iterations to convergence: 91
Achieved convergence tolerance: 5e-06
```

In order for `nls()` to converge, I had to allow more than the default 50 iterations; as it turned out, 91 iterations were required. To provide start values for the Gauss-Newton algorithm employed by `nls()`, I performed a preliminary linear least-squares regression and started the power-transformation parameters at values of 1.

Within rounding error, I get the same estimated transformation parameters and regression coefficients as before, but the regression-coefficient standard errors for age and education produced by simultaneously

estimating the transformations and regression coefficients are much larger. Arguably, this doesn't really make sense, because the regression coefficients for age and education don't have defined scales until the transformations are selected, and the values of the coefficients thus naturally depend upon the values of the powers.

An advantage of nonlinear least-squares is that we get standard errors for the ML estimates of the transformation parameters. Applying nonlinear least-squares to the Box-Tidwell model in this example required much more computation, although "much more" is a relative term, in that on my computer Box and Tidwell's procedure took 0.02 seconds, while nonlinear least-squares took 0.41 seconds. For some problems, however, we might run into convergence issues in applying general nonlinear least-squares to the Box-Tidwell model.

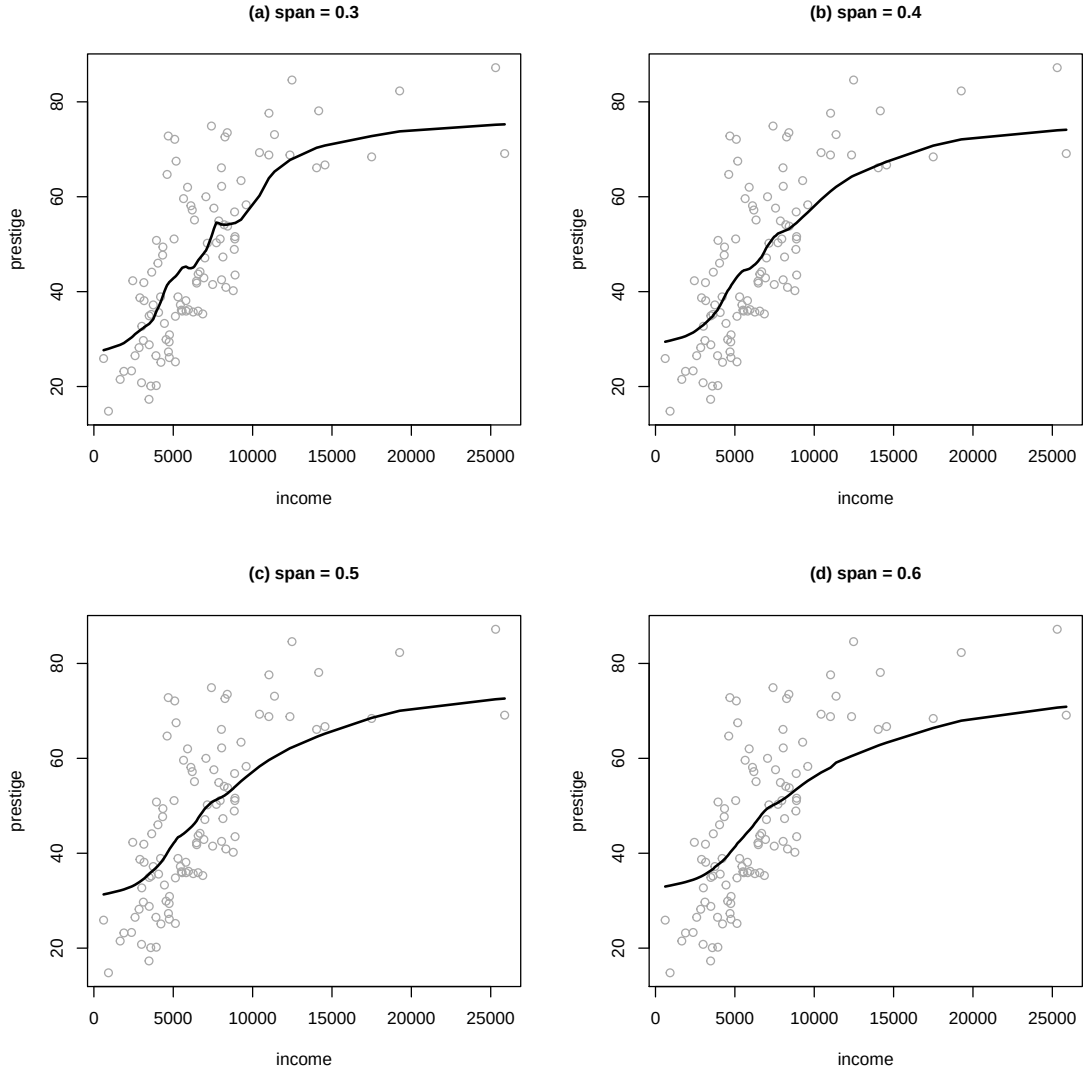
If we want standard errors for the transformation parameters, one approach would be to use Box and Tidwell's algorithm followed by linear least-squares to obtain the estimates, and then to employ these as start values for nonlinear least-squares. Convergence then should be nearly instantaneous.

Exercises for Chapter 18

Exercise 18.1

Kernel-regression programs that I'm aware of are somewhat more complicated than the simple version described in this chapter. I therefore wrote a simple kernel regression function for R that uses the tricube kernel, and I applied it to the Canadian occupational prestige data:

```
> kernelRegression <- function(x, y, span=0.5){
+   tricube <- function(z) { # weight function
+     ifelse(abs(z) < 1, (1 - (abs(z))^3)^3, 0)
+   }
+   n <- length(x)
+   n.span <- round(span*n) # no. within span
+   yhat <- numeric(n)
+   order <- order(x)
+   x <- x[order]
+   y <- y[order]
+   for (i in 1:n){   # kernel estimate at each x
+     x0 <- x[i]
+     dist <- abs(x - x0)
+     h <- sort(dist)[n.span] # window half-width
+     yhat[i] <- weighted.mean(y, tricube((x - x0)/h))
+   }
+   list(x=x, y=yhat)
+ }

> library("carData")  # for Prestige data set

> par(mfrow=c(2, 2))

> plot(prestige ~ income, data=Prestige, main="(a) span = 0.3",
+      cex.main=1, col="darkgray")
> with(Prestige,
+      lines(kernelRegression(income, prestige, span=0.3), lwd=2))

> plot(prestige ~ income, data=Prestige, main="(b) span = 0.4",
+      cex.main=1, col="darkgray")
> with(Prestige,
+      lines(kernelRegression(income, prestige, span=0.4), lwd=2))

> plot(prestige ~ income, data=Prestige, main="(c) span = 0.5",
+      cex.main=1, col="darkgray")
> with(Prestige,
+      lines(kernelRegression(income, prestige, span=0.5), lwd=2))

> plot(prestige ~ income, data=Prestige, main="(d) span = 0.6",
+      cex.main=1, col="darkgray")
> with(Prestige,
+      lines(kernelRegression(income, prestige, span=0.6), lwd=2))
```

All of the kernel estimates show boundary bias as the far left and right of the scatterplots, but a span of 0.4 or 0.5 seems reasonable. Figure 18.2 (d) on page 531 uses a span of 0.41, nearly identical to panel (b) in the figure above.

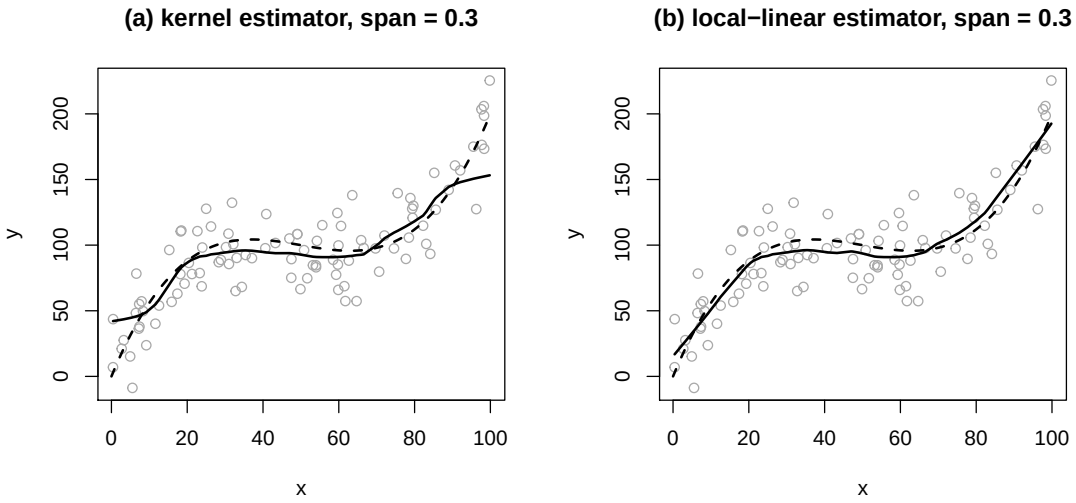
Exercise 18.3

I used the kernel regression and local-linear regression functions for R described in the preceding two exercises. I found that a span of 0.3 was a reasonable choice for both estimators:

```
> set.seed(753485254) # for reproducibility

> f <- function(x) 100 - 5*(x/10 - 5) + (x/10 - 5)^3

> n <- 100
> x <- runif(n, min=0, max=100)
> eps <- rnorm(n, mean=0, sd=20)
> y <- f(x) + eps

> par(mfrow=c(1, 2))

> plot(x, y, col="darkgray",
```



```
+      main="(a) kernel estimator, span = 0.3")
> x0 <- seq(0, 100, length=1000)
> lines(x0, f(x0), lwd=2, lty=2)
> lines(kernelRegression(x, y, span=0.3), lwd=2)

> plot(x, y, col="darkgray",
+      main="(b) local-linear estimator, span = 0.3")
> x0 <- seq(0, 100, length=1000)
> lines(x0, f(x0), lwd=2, lty=2)
> lines(localRegression(x, y, span=0.3), lwd=2)
```

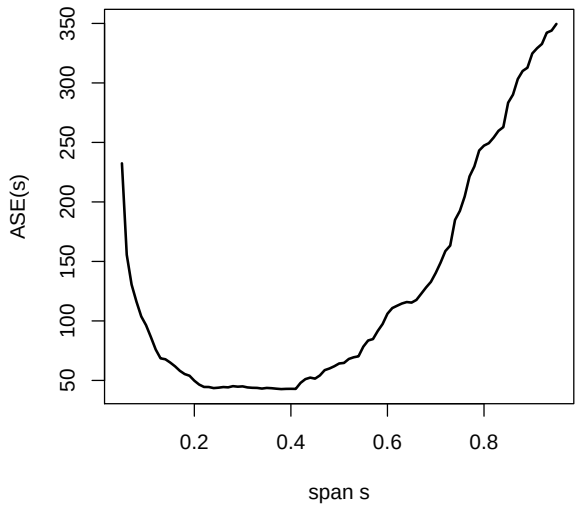


The greater "bias" of the kernel estimator is generally apparent, particularly near the boundaries of X.

Exercise 18.5*

Exercise 18.3 isn't starred but it's necessary to do it to generate the data and also to pick a span by visual trial-and-error. Once again, I used R for the computations:

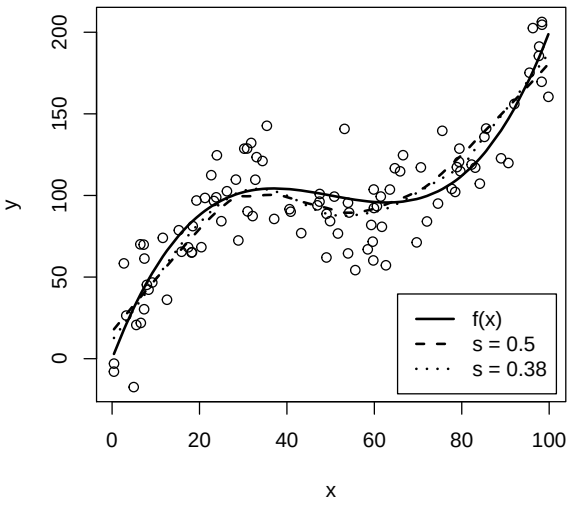
```
> f <- function(x) 100 - 5*(x/10 - 5) + (x/10 - 5)^3
> set.seed(753485254) # for reproducibility
> n <- 100
> x <- sort(runif(n, 0, 100))
> y <- f(x) + rnorm(n, mean=0, sd=20)
> mu <- f(x)
> spans <- seq(.05, .95, by=.01)
> ASEs <- rep(0, length(spans))
> names(ASEs) <- as.character(spans)
> for (span in spans){
+     yhat <- fitted(loess(y ~ x, span=span, degree=1, family="gaussian"))
+     ASEs[as.character(span)] <- sum((yhat - mu)^2)/n
+ }
> spans[which.min(ASEs)]
[1] .38
```

I used the `loess()` function to produce the local-linear fits to the data for spans s between .05 and .95 at an interval of .01. The minimum ASE is at $s = .38$.

Here's a scatterplot of the data showing the true regression function $f(x)$ along with the local-linear fits at spans of .38 and .5—the last is the value I picked by trial-and-error:

```
> plot(x, y)
> lines(x, mu, lwd=2)
> lines(loess.smooth(x, y, span=.5, family="gaussian"), lwd=2, lty=2)
> lines(loess.smooth(x, y, span=.38, family="gaussian"), lwd=2, lty=3)
> legend("bottomright", lty=1:3, lwd=2,
+        legend=c("f(x)", "s = .5", "s = .3"), inset=.02)
```

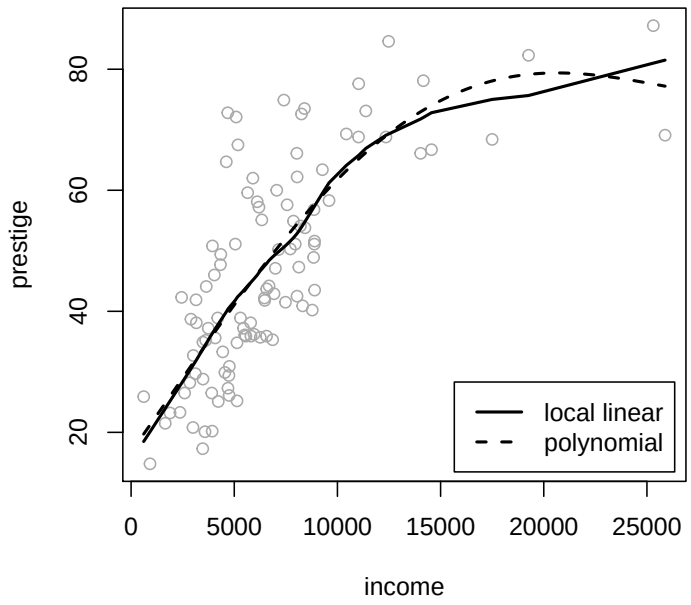


Exercise 18.7*

I performed the computations for this problem in R, using the `localRegression()` function written for Exercise 18.2 for the local-linear regression and the standard `lm()` function for the polynomial regressions.

(a) The Canadian occupational prestige data are in the **carData** package for R:

```
> library("carData") # for Prestige data set

> plot(prestige ~ income, data=Prestige, col="darkgray")
> with(Prestige, lines(localRegression(income, prestige, span=0.6),
+                      lwd=2))

> m <- lm(prestige ~ poly(income, 4), data=Prestige)
> x0 <- with(Prestige, seq(min(income), max(income), length=1000))
> lines(x0, predict(m, newdata=data.frame(income=x0)),
+       lty=2, lwd=2)

> legend("bottomright", inset=0.02, lty=1:2, lwd=2,
+        legend=c("local linear", "polynomial"))
```

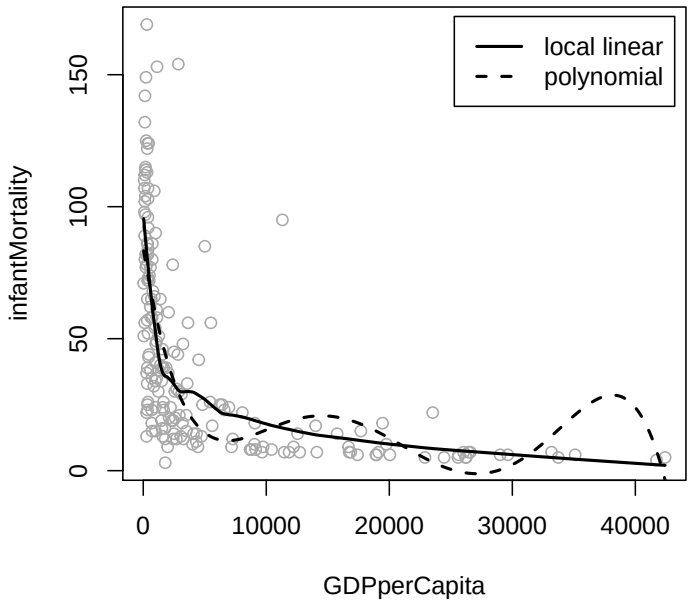


(b) The United Nations data are read from the website for the text:

```
> url <- paste("https://socialsciences.mcmaster.ca", "jfox", "Books",
+              "Applied-Regression-3E", "datasets", "UnitedNations.txt",
+              sep="/")
> UN <- read.table(url, header=TRUE)
> UN <- na.omit(UN[, c("infantMortality", "GDPperCapita")])

> plot(infantMortality ~ GDPperCapita, data=UN, col="darkgray")
> with(UN, lines(localRegression(GDPperCapita, infantMortality,
+                                span=0.5),
+                      lwd=2))
```



```
> m <- lm(infantMortality ~ poly(GDPperCapita, 5), data=UN)
> x0 <- with(UN, seq(min(GDPperCapita), max(GDPperCapita), length=1000))
> lines(x0, predict(m, newdata=data.frame(GDPperCapita=x0)),
+       lty=2, lwd=2)

> legend("topright", inset=0.02, lty=1:2, lwd=2,
+        legend=c("local linear", "polynomial"))
```



(c) The local-linear regression and fourth-order polynomial produce similar fits to the Canadian occupational prestige data, though the latter dips down slightly at the far right, which isn't plausible. The fifth-order polynomial fit to the UN data, however, produces a very wild fit at all but the lowest levels of GDP per capita.

Exercises for Chapter 19

Exercise 19.1*

We want to minimize $f(\widehat{\mu}) = \sum_{i=1}^{n} |Y_i - \widehat{\mu}|$. Differentiating $f(\widehat{\mu})$,

$$f'(\widehat{\mu}) = \sum \frac{d|Y_i - \widehat{\mu}|}{d\widehat{\mu}}$$

Setting the derivative to 0 and solving for $\widehat{\mu}$ minimizes the sum of absolute deviations.

The derivative of the absolute-value function, say $g(z) = |z|$, is

$$g'(z) = \begin{cases} -1 & \text{for } z < 0 \\ \text{undefined} & \text{for } z = 0 \\ 1 & \text{for } z > 0 \end{cases}$$

So $f'(\widehat{\mu}) = 0$ when there are equal numbers of negative and positive deviations from $\widehat{\mu}$.

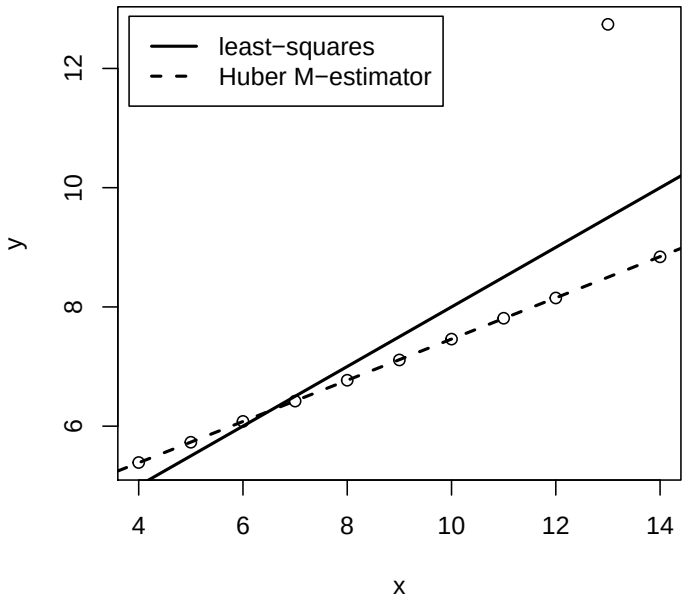
Suppose that n is even. Then we want a value of $\widehat{\mu}$ with $n/2$ observations below it and $n/2$ observations above it. That value, of course, is $\widehat{\mu} = \text{median}(Y)$, with the caveat that *any* value of Y between $Y_{(n/2)}$ and $Y_{(n/2+1)}$ will serve (where the parenthetical subscripts represent order statistics—that is, values in the *ordered* data); the median is conventionally taken as $\left(Y_{(n/2)} + Y_{(n/2+1)}\right)/2$. If we move $\widehat{\mu}$ below $Y_{(n/2)}$ or above $Y_{(n/2+1)}$, however, we'll increase either the number of positive deviations or the number of negative deviations, and the sum of the $+1$s and -1s will no longer be 0.

The situation is slightly different if n is odd. Then the median is $Y_{([n+1]/2)}$, with $n/2$ positive deviations, $n/2$ negative deviations, and one 0 deviation (for which the derivative is undefined). Again, if we move $\widehat{\mu}$ either left or right from the median, we increase either the number of positive deviations or the number of negative deviations.

Exercise 19.3

(a) and (b) I used the `lm()` and `rlm()` functions in R (the latter in the **MASS** package) to compute the least-squares and Huber M-estimator regression lines. The least-squares line is pulled towards the outlier, but the robust-regression line isn't affected by it at all. Anscombe's data sets are supplied by R in `anscombe`; this is the third of the three Anscombe data sets.

```
> x <- anscombe[, "x3"]
> y <- anscombe[, "y3"]

> plot(y ~ x)
> abline(lm(y ~ x), lwd=2)

> library("MASS")  # for rlm()
> abline(rlm(y ~ x), lty=2, lwd=2)

> legend("topleft", inset=0.02, lty=1:2, lwd=2,
+        legend=c("least-squares", "Huber M-estimator"))
```

(c) First, I'll verify the results stated in the exercise about what happens when the third point is omitted:

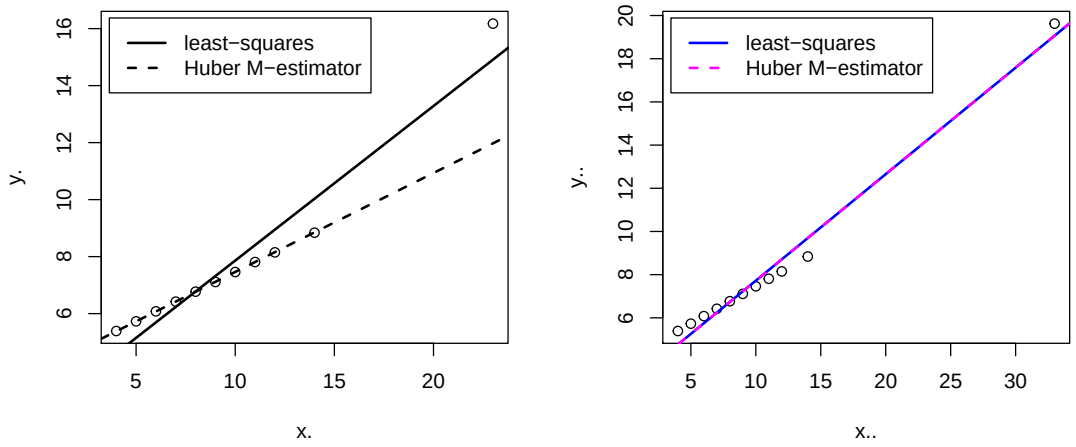
```
> (m <- lm(y ~ x, subset = -3))

Call:
lm(formula = y ~ x, subset = -3)

Coefficients:
(Intercept)            x
     4.0056       0.3454

> y[3] - predict(m, newdata=data.frame(x=x[3]))
       1
4.244286
```

Then I'll refit the least-squares and robust regressions with the two more high-leverage X-values and the corresponding outliers:

```
> y. <- y
> x. <- x
> x.[3] <- 23
> y.[3] <- 4 + 0.345*23 + 4.24

> par(mfrow=c(1, 2))

> plot(y. ~ x.)
> abline(lm(y. ~ x.), lwd=2)
> abline(rlm(y. ~ x., maxit=100), lty=2, lwd=2)
> legend("topleft", inset=0.02, lty=1:2, lwd=2,
+        legend=c("least-squares", "Huber M-estimator"))

> y.. <- y
```



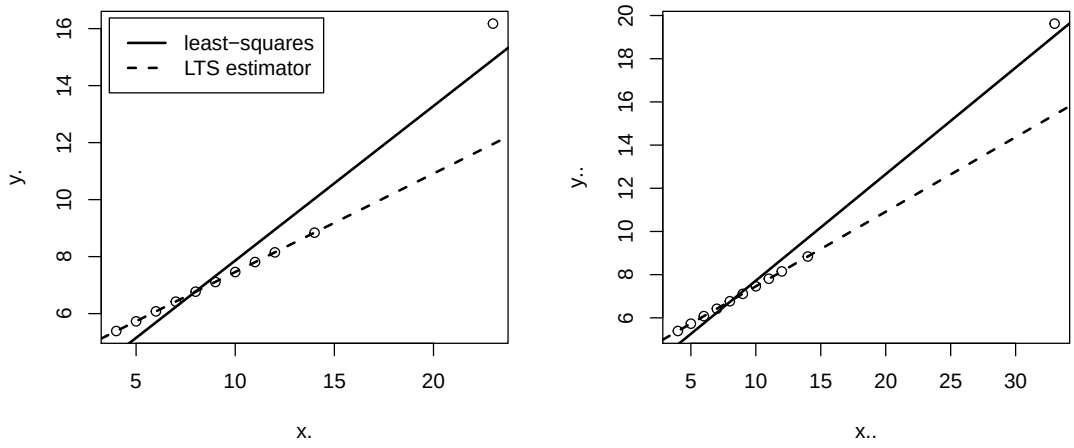
```
> x.. <- x
> x..[3] <- 33
> y..[3] <- 4 + 0.345*33 + 4.24

> plot(y.. ~ x..)
> abline(lm(y.. ~ x..), lwd=2, col="blue")
> abline(rlm(y.. ~ x.., maxit=100), lty=2, lwd=2, col="magenta")
> legend("topleft", inset=0.02, lty=1:2, lwd=2,
+        col=c("blue", "magenta"),
+        legend=c("least-squares", "Huber M-estimator"))
```



To get the Huber M-estimator to converge in both of these cases, I had to increase the maximum number of iterations, which defaults to 20. In the less extreme case (at the left), the robust regression estimator still ignores the outlier, but in the more extreme case, the robust regression is as bad as least-squares—the two regression lines coincide, and so I used different colors to differentiate them.

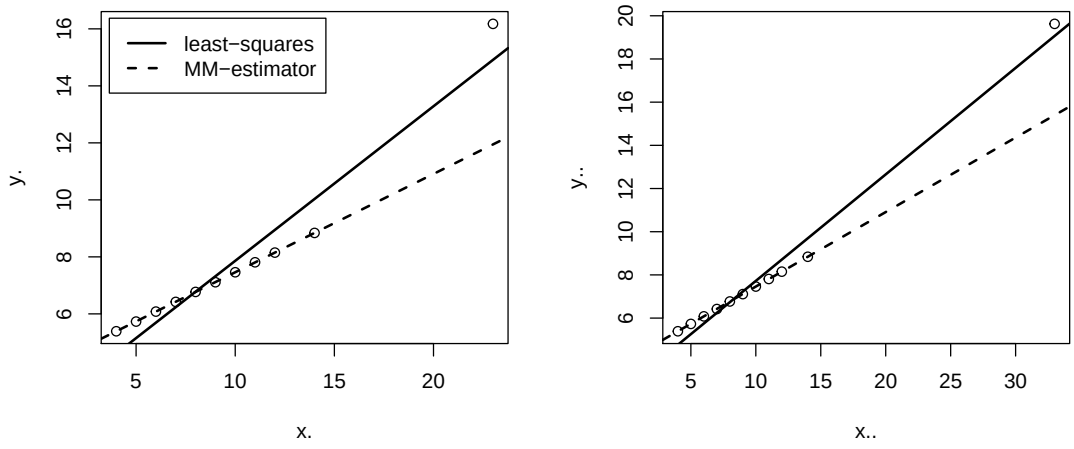
(d) I use the `ltsReg()` function in the **robustbase** R package to fit the LTS estimator:

```
> library("robustbase")

> par(mfrow=c(1, 2))

> plot(y. ~ x.)
> abline(lm(y. ~ x.), lwd=2)
> abline(ltsReg(y. ~ x.), lty=2, lwd=2)
> legend("topleft", inset=0.02, lty=1:2, lwd=2,
+        legend=c("least-squares", "LTS estimator"))

> plot(y.. ~ x..)
> abline(lm(y.. ~ x..), lwd=2)
> abline(ltsReg(y. ~ x.), lty=2, lwd=2)
```

The `rlm()` function can fit the MM estimator:

```
> par(mfrow=c(1, 2))

> plot(y. ~ x.)
> abline(lm(y. ~ x.), lwd=2)
> abline(rlm(y. ~ x., method="MM"), lty=2, lwd=2)
> legend("topleft", inset=0.02, lty=1:2, lwd=2,
+        legend=c("least-squares", "MM-estimator"))

> plot(y.. ~ x..)
> abline(lm(y.. ~ x..), lwd=2)
> abline(rlm(y.. ~ x.., method="MM"), lty=2, lwd=2)
```



Both the LTS and the MM estimator completely ignore the outlier, even in the higher-leverage case, and the MM estimator converges in fewer than 20 iterations in both cases.

Exercise 19.5

I start by using the standard R `glm()` function to fit the Poisson model by maximum likelihood, reproducing the results in Table 15.3, and refitting the model without the first observation:

```
> library("car") # for Ornstein data and compareCoefs()
Loading required package: carData

> m.ml <- glm(interlocks ~ ., data=Ornstein, family=poisson)
> m.ml.1 <- update(m.ml, subset=-1)
```

The coefficients from these regressions are shown below, in comparison to those produced by the LTS estimator.

A note on the comparison with Table 15.3: In the text, assets are scaled in 100s of millions of dollars, while here they are in millions of dollars, and so the reported coefficients differ by a factor of 100. As well, the baseline categories of the dummy regressors for the factors sector and nation of control are different in the two regressions. Because the focus here is on the comparison between the ML and LTS estimators, I haven't bothered to adjust for these inessential differences.

Next, I use the `glmrob()` function in the **robustbase** R package to compute the LTS estimator. I fit two versions of the LTS estimator: The first computes robustness weights based on residuals, similar to the M estimator, and the second additionally employs weights that are inversely related to hatvalues, down-weighting high-leverage observations; the two sets of weight are multiplied.

```
> library("robustbase") # for glmrob()

> m.rob <- glmrob(interlocks ~ ., data=Ornstein, family=poisson)

> m.rob.hat <- glmrob(interlocks ~ ., data=Ornstein, family=poisson,
+                     weights.on.x="hat")

> compareCoefs(m.ml, m.ml.1, m.rob, m.rob.hat)
Calls:
1: glm(formula = interlocks ~ ., family = poisson, data = Ornstein)
2: glm(formula = interlocks ~ ., family = poisson, data = Ornstein,
           subset = -1)
3: glmrob(formula = interlocks ~ ., family = poisson, data = Ornstein)
4: glmrob(formula = interlocks ~ ., family = poisson, data = Ornstein,
          weights.on.x = "hat")

             Model 1  Model 2  Model 3  Model 4
(Intercept)   2.3246   2.3119   2.0680   2.0576
SE            0.0519   0.0520   0.0604   0.0605

assets      2.08e-05 2.60e-05 2.04e-05 2.28e-05
SE          1.20e-06 1.43e-06 1.20e-06 1.24e-06

sectorBNK    -0.4092  -0.7487  -0.0351  -0.1919
SE            0.1560   0.1689   0.1582   0.1639

sectorCON     -0.620   -0.621   -1.091   -1.088
SE             0.212    0.212    0.298    0.296

sectorFIN     0.6770   0.6085   0.8817   0.8560
SE            0.0688   0.0697   0.0764   0.0767

sectorHLD      0.208    0.201    0.113    0.117
SE             0.119    0.119    0.144    0.144

sectorMAN     0.0526   0.0516  -0.3398  -0.3230
SE            0.0755   0.0756   0.0982   0.0980
```

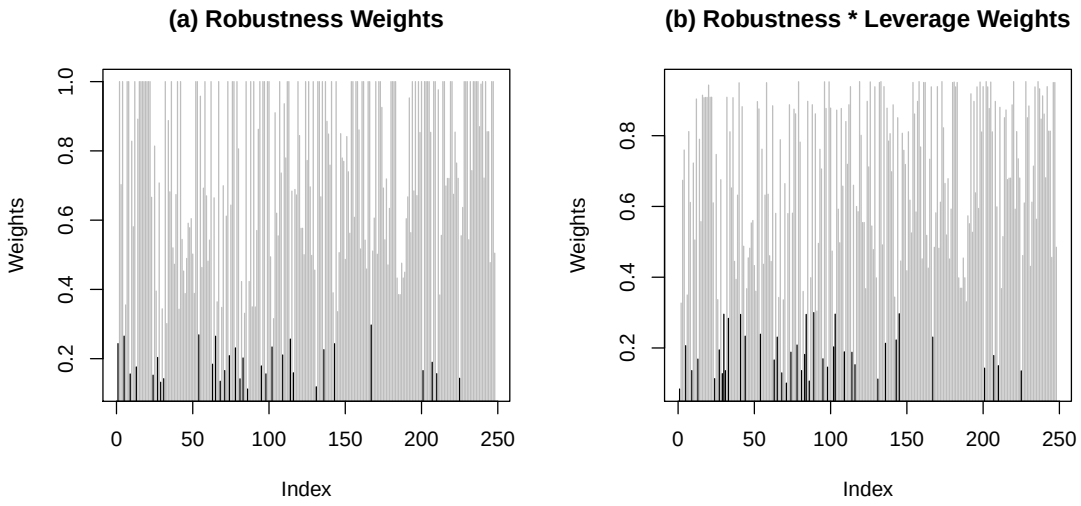


```
sectorMER     0.1777   0.1787   0.1405   0.1470
SE            0.0865   0.0866   0.1022   0.1023

sectorMIN     0.6211   0.6014   0.7182   0.7108
SE            0.0669   0.0670   0.0760   0.0762

sectorTRN     0.6778   0.6282   0.8274   0.7882
SE            0.0748   0.0754   0.0838   0.0845

sectorWOD     0.7116   0.7078   0.9021   0.9167
SE            0.0753   0.0753   0.0832   0.0830

nationOTH    -0.1632  -0.1470   0.0354   0.0554
SE            0.0736   0.0738   0.0777   0.0776

nationUK     -0.5771  -0.5625  -0.5487  -0.5291
SE            0.0890   0.0891   0.1005   0.0999

nationUS     -0.8259  -0.8164  -0.8270  -0.8285
SE            0.0490   0.0491   0.0564   0.0567
```

Without leverage-based weights, the LTS estimator of the assets coefficient is not very different from the ML estimator—recall that the first observation is at a high-leverage point. The coefficient of assets in the LTS regression with leverage-based weights is larger, but not as large as the ML estimator of this coefficient when the first case is omitted. There are differences in some of the dummy-regressor coefficients for both LTS estimators in comparison to the ML estimator, with or without the first observation removed. I don't want to push this example too hard because of the clear deficiencies of the model, mentioned in the exercise and discussed in Chapter 15. The reader may wish, for example, to repeat the ML and LTS regressions with assets log-transformed.

To get some insight into what the LTS estimators are doing, I plotted the observation weights that they computed:

```
> par(mfrow=c(1, 2))

> plot(m.rob$w.r, type="h", ylab="Weights",
+      col=ifelse(m.rob$w.r < 0.3, "black", "gray"),
+      main="(a) Robustness Weights")

> wt <- m.rob.hat$w.r*m.rob.hat$w.x
> plot(wt, type="h", ylab="Weights",
+      col=ifelse(wt < 0.3, "black", "gray"),
+      main="(b) Robustness * Leverage Weights")
```

To clarify these plots, I arbitrarily colored weights < 0.3 black. It's clear that the first observation gets relatively low weight in the first LTS regression, though some others have even smaller weights. In the second regression, the first observation gets the smallest weight, but there are others that have small weights as well, accounting for the differences between the leverage-based LTS estimator and the ML estimator with the first observation removed.

Exercises for Chapter 20

Exercise 20.1

(a) It's simple to use the `cor()` function in R for this computation:

```
> Data <- read.table(header=TRUE, text="
+ x1 x2 x3
+ 1 1 NA
+ 1 NA 1
+ -1 -1 NA
+ -1 NA -1
+ NA 1 -1
+ NA -1 1
+ 5 NA NA
+ ")

> cor(Data, use="pairwise.complete.obs")
   x1 x2 x3
x1  1  1  1
x2  1  1 -1
x3  1 -1  1
```

Thus, the available-cases correlations (called "pairwise-complete") are $r_{12} = 1$, $r_{13} = 1$, and $r_{23} = -1$. Clearly these are inconsistent, because if $r_{12} = 1$ and $r_{13} = 1$, then r_{23} should also be 1.

(b) I wrote a simple R function to do the computation, and then applied it to each pair of variables:

```
> myCor <- function(x, y){
+   mean.x <- mean(x, na.rm=TRUE)
+   mean.y <- mean(y, na.rm=TRUE)
+   sd.x <- sd(x, na.rm=TRUE)
+   sd.y <- sd(y, na.rm=TRUE)
+   n.complete <- sum(complete.cases(x, y))
+   cov.xy <- sum((x - mean.x)*(y - mean.y), na.rm=TRUE)/(n.complete - 1)
+   cov.xy/(sd.x*sd.y)
+ }

> with(Data, myCor(x1, x2))
[1] 0.7071068

> with(Data, myCor(x1, x3))
[1] 0.7071068

> with(Data, myCor(x2, x3))
[1] -1.5
```

This approach produces not only inconsistent correlations but also obvious nonsense, with $r_{23} < -1$!

(c)* The available-case correlation matrix has a negative eigenvalue:

```
> eigen(cor(Data, use="pairwise.complete.obs"))$values
[1]  2  2 -1
```

Exercise 20.3*

This is really a small project more than an exercise!

Functions for Sampling

I'll start by writing R functions to draw simulated data and then create missing data as MCAR, MAR, and MNAR. Both here and elsewhere in this exercise, I'm unconcerned with writing efficient R code because the small simulation study runs quickly. Instead the code is meant to be transparent. This is a very slightly edited version of the code I used when I originally wrote the chapter on missing data.

```
library(MASS) # for mvnorm(), to draw multivariate-normal samples

mu <- c(10, 20)
Sigma <- matrix(c(9, 8, 8, 16), 2, 2)

mcar <- function(n, mu, Sigma, f){
    nmiss <- round(f*n)
    X <- mvrnorm(n, mu=mu, Sigma=Sigma)
    imiss <- sample(n, nmiss, replace=FALSE)
    X[imiss, 2] <- NA
    X
}

mar <- function(n, mu, Sigma, a=-2/3, b=-2){
    X <- mvrnorm(n, mu=mu, Sigma=Sigma)
    pmiss <- 1/(1 + exp(-(a + b*(X[,1] - mu[1])/sqrt(Sigma[1,1]))))
    imiss <- pmiss > runif(n)
    X[imiss, 2] <- NA
    X
}

mnar <- function(n, mu, Sigma, a=-2/3, b=-2){
    X <- mvrnorm(n, mu=mu, Sigma=Sigma)
    pmiss <- 1/(1 + exp(-(a + b*(X[,2] - mu[2])/sqrt(Sigma[2,2]))))
    imiss <- pmiss > runif(n)
    X[imiss, 2] <- NA
    X
}
```

Functions to Estimate the Parameters

I'll next write functions to obtain estimates by complete-case analysis, mean imputation, regression imputation and multiple imputation, the last using the **mi** package, and to run the simulation for each of these cases. Each function returns the simulation results as a list.

```
completeCaseAnalysis <- function(reps, n, mu, Sigma, fn=mcar, ...){
    beta12 <- Sigma[1,2]/Sigma[2,2]
    beta21 <- Sigma[1,2]/Sigma[1,1]
    ci.mu1 <- rep(FALSE, reps)
    ci.mu2 <- rep(FALSE, reps)
    ci.beta12 <- rep(FALSE, reps)
    ci.beta21 <- rep(FALSE, reps)
```



```
    ci.mu1.width <- rep(0, reps)
    ci.mu2.width <- rep(0, reps)
    ci.beta12.width <- rep(0, reps)
    ci.beta21.width <- rep(0, reps)
    xbar1 <- rep(0, reps)
    xbar2 <- rep(0, reps)
    b12 <- rep(0, reps)
    b21 <- rep(0, reps)
    ngood <- rep(0, reps)
    for (rep in 1:reps){
        X <- na.omit(fn(n=n, mu=mu, Sigma=Sigma, ...))
        mod <- lm(X[,1] ~ X[,2])
        b <- coef(mod)[2]
        ci <- confint(mod)[2,]
        b12[rep] <- b
        ci.beta12[rep] <- (ci[1] < beta12) && (beta12 < ci[2])
        ci.beta12.width[rep] <-  ci[2] - ci[1]
        mod <- lm(X[,2] ~ X[,1])
        b <- coef(mod)[2]
        ci <- confint(mod)[2,]
        b21[rep] <- b
        ci.beta21[rep] <- (ci[1] < beta21) && (beta21 < ci[2])
        ci.beta21.width[rep] <-  ci[2] - ci[1]
        xbar1[rep] <- mean(X[,1])
        xbar2[rep] <- mean(X[,2])
        ci <- t.test(X[,2])$conf.int
        ci.mu2[rep] <- (ci[1] < mu[2]) && (mu[2] < ci[2])
        ci.mu2.width[rep] <- ci[2] - ci[1]
        ci <- t.test(X[,1])$conf.int
        ci.mu1[rep] <- (ci[1] < mu[1]) && (mu[1] < ci[2])
        ci.mu1.width[rep] <- ci[2] - ci[1]
        ngood[rep] <- nrow(X)
    }
    list(reps=reps, n=n, mu=mu, Sigma=Sigma, fn=fn, beta12=beta12, beta21=beta21,
         ci.mu1=sum(ci.mu1)/reps,
         ci.mu2=sum(ci.mu2)/reps,
         ci.beta12=sum(ci.beta12)/reps,
         ci.beta21=sum(ci.beta21)/reps,
         ci.mu1.width=mean(ci.mu1.width),
         ci.mu2.width=mean(ci.mu2.width),
         ci.beta12.width=mean(ci.beta12.width),
         ci.beta21.width=mean(ci.beta21.width),
         xbar1=mean(xbar1), xbar2=mean(xbar2), b12=mean(b12), b21=mean(b21),
         rmse.xbar1=sqrt(mean((xbar1 - mu[1])^2)),
         rmse.xbar2=sqrt(mean((xbar2 - mu[2])^2)),
         rmse.beta12=sqrt(mean((b12 - beta12)^2)),
```



```
        rmse.beta21=sqrt(mean((b21 - beta21)^2)),
        f=mean(ngood)/n)
}

meanImputation <- function(reps, n, mu, Sigma, fn=mcar, ...){
    beta12 <- Sigma[1,2]/Sigma[2,2]
    beta21 <- Sigma[1,2]/Sigma[1,1]
    ci.mu1 <- rep(FALSE, reps)
    ci.mu2 <- rep(FALSE, reps)
    ci.beta12 <- rep(FALSE, reps)
    ci.beta21 <- rep(FALSE, reps)
    ci.mu1.width <- rep(0, reps)
    ci.mu2.width <- rep(0, reps)
    ci.beta12.width <- rep(0, reps)
    ci.beta21.width <- rep(0, reps)
    xbar1 <- rep(0, reps)
    xbar2 <- rep(0, reps)
    b12 <- rep(0, reps)
    b21 <- rep(0, reps)
    ngood <- rep(0, reps)
    for (rep in 1:reps){
        X <- fn(n=n, mu=mu, Sigma=Sigma, ...)
        X[is.na(X[,2]), 2] <- mean(X[,2], na.rm=TRUE)
        mod <- lm(X[,1] ~ X[,2])
        b <- coef(mod)[2]
        ci <- confint(mod)[2,]
        b12[rep] <- b
        ci.beta12[rep] <- (ci[1] < beta12) && (beta12 < ci[2])
        ci.beta12.width[rep] <-  ci[2] - ci[1]
        mod <- lm(X[,2] ~ X[,1])
        b <- coef(mod)[2]
        ci <- confint(mod)[2,]
        b21[rep] <- b
        ci.beta21[rep] <- (ci[1] < beta21) && (beta21 < ci[2])
        ci.beta21.width[rep] <-  ci[2] - ci[1]
        xbar1[rep] <- mean(X[,1])
        xbar2[rep] <- mean(X[,2])
        ci <- t.test(X[,1])$conf.int
        ci.mu1[rep] <- (ci[1] < mu[1]) && (mu[1] < ci[2])
        ci.mu1.width[rep] <- ci[2] - ci[1]
        ci <- t.test(X[,2])$conf.int
        ci.mu2[rep] <- (ci[1] < mu[2]) && (mu[2] < ci[2])
        ci.mu2.width[rep] <- ci[2] - ci[1]
        ngood[rep] <- nrow(X)
    }
    list(reps=reps, n=n, mu=mu, Sigma=Sigma, fn=fn, beta12=beta12, beta21=beta21,
```



```
          ci.mu1=sum(ci.mu1)/reps,
          ci.mu2=sum(ci.mu2)/reps,
          ci.beta12=sum(ci.beta12)/reps,
          ci.beta21=sum(ci.beta21)/reps,
          ci.mu1.width=mean(ci.mu1.width),
          ci.mu2.width=mean(ci.mu2.width),
          ci.beta12.width=mean(ci.beta12.width),
          ci.beta21.width=mean(ci.beta21.width),
          xbar1=mean(xbar1), xbar2=mean(xbar2), b12=mean(b12), b21=mean(b21),
          rmse.xbar1=sqrt(mean((xbar1 - mu[1])^2)),
          rmse.xbar2=sqrt(mean((xbar2 - mu[2])^2)),
          rmse.beta12=sqrt(mean((b12 - beta12)^2)),
          rmse.beta21=sqrt(mean((b21 - beta21)^2)),
          f=mean(ngood)/n)
}

regressionImputation <- function(reps, n, mu, Sigma, fn=mcar, ...){
    beta12 <- Sigma[1,2]/Sigma[2,2]
    beta21 <- Sigma[1,2]/Sigma[1,1]
    ci.mu1 <- rep(FALSE, reps)
    ci.mu2 <- rep(FALSE, reps)
    ci.beta12 <- rep(FALSE, reps)
    ci.beta21 <- rep(FALSE, reps)
    ci.mu1.width <- rep(0, reps)
    ci.mu2.width <- rep(0, reps)
    ci.beta12.width <- rep(0, reps)
    ci.beta21.width <- rep(0, reps)
    xbar1 <- rep(0, reps)
    xbar2 <- rep(0, reps)
    b12 <- rep(0, reps)
    b21 <- rep(0, reps)
    ngood <- rep(0, reps)
    for (rep in 1:reps){
        X <- fn(n=n, mu=mu, Sigma=Sigma, ...)
        x1 <- X[,1]
        x2 <- X[,2]
        mod <- lm(x2 ~ x1)
        yhat <- predict(mod, data.frame(x1=x1[is.na(x2)]))
        X[is.na(X[,2]), 2] <- yhat
        mod <- lm(X[,1] ~ X[,2])
        b <- coef(mod)[2]
        ci <- confint(mod)[2,]
        b12[rep] <- b
        ci.beta12[rep] <- (ci[1] < beta12) && (beta12 < ci[2])
        ci.beta12.width[rep] <-  ci[2] - ci[1]
        mod <- lm(X[,2] ~ X[,1])
```



```
        b <- coef(mod)[2]
        ci <- confint(mod)[2,]
        b21[rep] <- b
        ci.beta21[rep] <- (ci[1] < beta21) && (beta21 < ci[2])
        ci.beta21.width[rep] <-  ci[2] - ci[1]
        xbar1[rep] <- mean(X[,1])
        ci <- t.test(X[,1])$conf.int
        ci.mu1[rep] <- (ci[1] < mu[1]) && (mu[1] < ci[2])
        ci.mu1.width[rep] <- ci[2] - ci[1]
        xbar2[rep] <- mean(X[,2])
        ci <- t.test(X[,2])$conf.int
        ci.mu2[rep] <- (ci[1] < mu[2]) && (mu[2] < ci[2])
        ci.mu2.width[rep] <- ci[2] - ci[1]
        ngood[rep] <- nrow(X)
    }
    list(reps=reps, n=n, mu=mu, Sigma=Sigma, fn=fn, beta12=beta12, beta21=beta21,
         ci.mu1=sum(ci.mu1)/reps,
         ci.mu2=sum(ci.mu2)/reps,
         ci.beta12=sum(ci.beta12)/reps,
         ci.beta21=sum(ci.beta21)/reps,
         ci.mu1.width=mean(ci.mu1.width),
         ci.mu2.width=mean(ci.mu2.width),
         ci.beta12.width=mean(ci.beta12.width),
         ci.beta21.width=mean(ci.beta21.width),
         xbar1=mean(xbar1), xbar2=mean(xbar2), b12=mean(b12), b21=mean(b21),
         rmse.xbar1=sqrt(mean((xbar1 - mu[1])^2)),
         rmse.xbar2=sqrt(mean((xbar2 - mu[2])^2)),
         rmse.beta12=sqrt(mean((b12 - beta12)^2)),
         rmse.beta21=sqrt(mean((b21 - beta21)^2)),
         f=mean(ngood)/n)
}

multipleImputation <- function(imps=5, steps=10, seed=1234567, reps, n, mu,
                               Sigma, fn=mcar, ...){
    require(norm)
    rngseed(seed)
    beta12 <- Sigma[1,2]/Sigma[2,2]
    beta21 <- Sigma[1,2]/Sigma[1,1]
    ci.mu1 <- rep(FALSE, reps)
    ci.mu2 <- rep(FALSE, reps)
    ci.beta12 <- rep(FALSE, reps)
    ci.beta21 <- rep(FALSE, reps)
    ci.mu1.width <- rep(0, reps)
    ci.mu2.width <- rep(0, reps)
    ci.beta12.width <- rep(0, reps)
    ci.beta21.width <- rep(0, reps)
```



```
xbar1 <- rep(0, reps)
xbar2 <- rep(0, reps)
b12 <- rep(0, reps)
b21 <- rep(0, reps)
ngood <- rep(0, reps)
for (rep in 1:reps){
    X <- fn(n=n, mu=mu, Sigma=Sigma, ...)
    ngood[rep] <- nrow(na.omit(X))
    colnames(X) <- c("x1", "x2")
    X.summary <- prelim.norm(X)
    X.em <- em.norm(X.summary, showits=FALSE)
    imputed.data <- as.list(1:imps)
    for (imp in 1:imps){
        X.da <- da.norm(X.summary, X.em, steps=steps)
        imputed.data[[imp]] <- imp.norm(X.summary, X.da, X)
    }
    results <- lapply(imputed.data,
                      function(data) {
                          mod <- lm(x1 ~ x2,
                                    data=as.data.frame(data))
                          res <- list()
                          res$coef <- coef(mod)
                          res$se <- sqrt(diag(vcov(mod)))
                          res
                      }
    )
    results <- mi.inference(lapply(results, function(x) x$coef),
                            lapply(results, function(x) x$se))
    b12[rep] <- results$est[2]
    ci.beta12[rep] <- (results$lower[2] < beta12) && (beta12 < results$upper[2])
    ci.beta12.width[rep] <-  results$upper[2] - results$lower[2]
    results <- lapply(imputed.data,
                      function(data) {
                          mod <- lm(x2 ~ x1,
                                    data=as.data.frame(data))
                          res <- list()
                          res$coef <- coef(mod)
                          res$se <- sqrt(diag(vcov(mod)))
                          res
                      }
    )
    results <- mi.inference(lapply(results, function(x) x$coef),
                            lapply(results, function(x) x$se))
    b21[rep] <- results$est[2]
    ci.beta21[rep] <- (results$lower[2] < beta21) && (beta21 < results$upper[2])
    ci.beta21.width[rep] <-  results$upper[2] - results$lower[2]
```



```
    results <- lapply(imputed.data,
                      function(data) {
                          mod <- lm(x1 ~ 1,
                                    data=as.data.frame(data))
                          res <- list()
                          res$coef <- coef(mod)
                          res$se <- sqrt(diag(vcov(mod)))
                          res
                      }
    )
    results <- mi.inference(lapply(results, function(x) x$coef),
                            lapply(results, function(x) x$se))
    xbar1[rep] <- results$est[1]
    ci.mu1[rep] <- (results$lower[1] < mu[1]) && (mu[1] < results$upper[1])
    ci.mu1.width[rep] <-  results$upper[1] - results$lower[1]
    results <- lapply(imputed.data,
                      function(data) {
                          mod <- lm(x2 ~ 1,
                                    data=as.data.frame(data))
                          res <- list()
                          res$coef <- coef(mod)
                          res$se <- sqrt(diag(vcov(mod)))
                          res
                      }
    )
    results <- mi.inference(lapply(results, function(x) x$coef),
                            lapply(results, function(x) x$se))
    xbar2[rep] <- results$est[1]
    ci.mu2[rep] <- (results$lower[1] < mu[2]) &&
        (mu[2] < results$upper[1])
    ci.mu2.width[rep] <-  results$upper[1] - results$lower[1]
}
list(imps=imps, steps=steps, seed=seed,
     reps=reps, n=n, mu=mu, Sigma=Sigma, fn=fn, beta12=beta12, beta21=beta21,
     ci.mu1=sum(ci.mu1)/reps,
     ci.mu2=sum(ci.mu2)/reps,
     ci.beta12=sum(ci.beta12)/reps,
     ci.beta21=sum(ci.beta21)/reps,
     ci.mu1.width=mean(ci.mu1.width),
     ci.mu2.width=mean(ci.mu2.width),
     ci.beta12.width=mean(ci.beta12.width),
     ci.beta21.width=mean(ci.beta21.width),
     xbar1=mean(xbar1), xbar2=mean(xbar2), b12=mean(b12), b21=mean(b21),
     rmse.xbar1=sqrt(mean((xbar1 - mu[1])^2)),
     rmse.xbar2=sqrt(mean((xbar2 - mu[2])^2)),
     rmse.beta12=sqrt(mean((b12 - beta12)^2)),
```



```
        rmse.beta21=sqrt(mean((b21 - beta21)^2)),
        f=mean(ngood)/n)
}
```

Running the Simulations

Next, I'll run the simulations, setting the seed for R's random-number generator for replicability, and using the same seed for each case so that I obtain the same samples across different methods. The **mi** package uses its own random-number generator, and I set the seed for that as well to a common value. There are 12 sets of simulations, crossing the 3 kinds of missing data with the 4 methods:

```
# MCAR

set.seed(87610826)
cc.mcar <- completeCaseAnalysis(reps=1000, n=250, mu=mu, Sigma=Sigma,
                                fn=mcar, f=0.4)

set.seed(87610826)
ms.mcar <- meanImputation(reps=1000, n=250, mu=mu, Sigma=Sigma,
                          fn=mcar, f=0.4)

set.seed(87610826)
rs.mcar <- regressionImputation(reps=1000, n=250, mu=mu, Sigma=Sigma,
                                fn=mcar, f=0.4)

set.seed(87610826)
mi.mcar <- multipleImputation(imps=5, steps=20, seed=44996878, reps=1000, n=250,
                              mu=mu, Sigma=Sigma, fn=mcar, f=0.4)

Loading required package: norm

# MAR

set.seed(87610826)
cc.mar <- completeCaseAnalysis(reps=1000, n=250, mu=mu, Sigma=Sigma,
                               fn=mar)

set.seed(87610826)
ms.mar <- meanImputation(reps=1000, n=250, mu=mu, Sigma=Sigma,
                         fn=mar)

set.seed(87610826)
rs.mar <- regressionImputation(reps=1000, n=250, mu=mu, Sigma=Sigma,
                               fn=mar)

set.seed(87610826)
mi.mar <- multipleImputation(imps=5, steps=20, seed=44996878, reps=1000,
```



```
                                n=250, mu=mu, Sigma=Sigma, fn=mar)

# MNAR

set.seed(87610826)
cc.mnar <- completeCaseAnalysis(reps=1000, n=250, mu=mu, Sigma=Sigma,
                                    fn=mnar)

set.seed(87610826)
ms.mnar <- meanImputation(reps=1000, n=250, mu=mu, Sigma=Sigma,
                                fn=mnar)

set.seed(87610826)
rs.mnar <- regressionImputation(reps=1000, n=250, mu=mu, Sigma=Sigma,
                                    fn=mnar)

set.seed(87610826)
mi.mnar <- multipleImputation(imps=5, steps=20, seed=44996878, reps=1000,
                                n=250, mu=mu, Sigma=Sigma, fn=mnar)
```

Results of the Simulations

Finally, here are summaries of the results of these simulations, starting with MAR, which is the case summarized in Table 20.2 in the text. It would have been neater to write an R function to create the summary tables, but I decided that it would be quicker simply to assemble the tables directly from the 12 lists returned by the simulation functions.

Data Missing At Random (MAR)

You'll see that the results differ slightly from those in Table 20.2 because different random numbers were sampled—I was unable to duplicate the results in the text exactly, even though I saved the random-number generator seed that I used, probably because of changes to R and perhaps the **MASS** package since 2007, when I originally ran the simulation.

Average parameter estimates (recall that the parameters are $\mu_1 = 10, \mu_2 = 20, \beta_{12} = 0.5, \beta_{21} = 0.889$):

```
mean.MAR <- matrix(0, 4, 4)
rownames(mean.MAR) <- c("mu_1", "mu_2", "beta_12", "beta_21")
colnames(mean.MAR) <- c("CC", "Mean Imp", "Regr Imp", "Mult Imp")
mean.MAR["mu_1", "CC"] <- cc.mar$xbar1
mean.MAR["mu_2", "CC"] <- cc.mar$xbar2
mean.MAR["beta_12", "CC"] <- cc.mar$b12
mean.MAR["beta_21", "CC"] <- cc.mar$b21
mean.MAR["mu_1", "Mean Imp"] <- ms.mar$xbar1
mean.MAR["mu_2", "Mean Imp"] <- ms.mar$xbar2
mean.MAR["beta_12", "Mean Imp"] <- ms.mar$b12
mean.MAR["beta_21", "Mean Imp"] <- ms.mar$b21
mean.MAR["mu_1", "Regr Imp"] <- rs.mar$xbar1
```



```
mean.MAR["mu_2", "Regr Imp"] <- rs.mar$xbar2
mean.MAR["beta_12", "Regr Imp"] <- rs.mar$b12
mean.MAR["beta_21", "Regr Imp"] <- rs.mar$b21
mean.MAR["mu_1", "Mult Imp"] <- mi.mar$xbar1
mean.MAR["mu_2", "Mult Imp"] <- mi.mar$xbar2
mean.MAR["beta_12", "Mult Imp"] <- mi.mar$b12
mean.MAR["beta_21", "Mult Imp"] <- mi.mar$b21
round(mean.MAR, 3)
```

```
           CC Mean Imp Regr Imp Mult Imp
mu_1    11.453    9.995    9.995    9.995
mu_2    21.297   21.297   19.998   20.000
beta_12  0.391    0.391    0.642    0.496
beta_21  0.891    0.356    0.891    0.890
```

Root-mean-square error of the estimates:

```
RMSE.MAR <- matrix(0, 4, 4)
rownames(RMSE.MAR) <- c("mu_1", "mu_2", "beta_12", "beta_21")
colnames(RMSE.MAR) <- c("CC", "Mean Imp", "Regr Imp", "Mult Imp")
RMSE.MAR["mu_1", "CC"] <- cc.mar$rmse.xbar1
RMSE.MAR["mu_2", "CC"] <- cc.mar$rmse.xbar2
RMSE.MAR["beta_12", "CC"] <- cc.mar$rmse.beta12
RMSE.MAR["beta_21", "CC"] <- cc.mar$rmse.beta21
RMSE.MAR["mu_1", "Mean Imp"] <- ms.mar$rmse.xbar1
RMSE.MAR["mu_2", "Mean Imp"] <- ms.mar$rmse.xbar2
RMSE.MAR["beta_12", "Mean Imp"] <- ms.mar$rmse.beta12
RMSE.MAR["beta_21", "Mean Imp"] <- ms.mar$rmse.beta21
RMSE.MAR["mu_1", "Regr Imp"] <- rs.mar$rmse.xbar1
RMSE.MAR["mu_2", "Regr Imp"] <- rs.mar$rmse.xbar2
RMSE.MAR["beta_12", "Regr Imp"] <- rs.mar$rmse.beta12
RMSE.MAR["beta_21", "Regr Imp"] <- rs.mar$rmse.beta21
RMSE.MAR["mu_1", "Mult Imp"] <- mi.mar$rmse.xbar1
RMSE.MAR["mu_2", "Mult Imp"] <- mi.mar$rmse.xbar2
RMSE.MAR["beta_12", "Mult Imp"] <- mi.mar$rmse.beta12
RMSE.MAR["beta_21", "Mult Imp"] <- mi.mar$rmse.beta21
round(RMSE.MAR, 3)
```

```
           CC Mean Imp Regr Imp Mult Imp
mu_1    1.467    0.196    0.196    0.196
mu_2    1.332    1.332    0.328    0.340
beta_12 0.118    0.118    0.149    0.043
beta_21 0.101    0.536    0.101    0.107
```

Confidence-interval coverage (nominally 95%):

```
coverage.MAR <- matrix(0, 4, 4)
rownames(coverage.MAR) <- c("mu_1", "mu_2", "beta_12", "beta_21")
colnames(coverage.MAR) <- c("CC", "Mean Imp", "Regr Imp", "Mult Imp")
```



```
coverage.MAR["mu_1", "CC"] <- cc.mar$ci.mu1
coverage.MAR["mu_2", "CC"] <- cc.mar$ci.mu2
coverage.MAR["beta_12", "CC"] <- cc.mar$ci.beta12
coverage.MAR["beta_21", "CC"] <- cc.mar$ci.beta21
coverage.MAR["mu_1", "Mean Imp"] <- ms.mar$ci.mu1
coverage.MAR["mu_2", "Mean Imp"] <- ms.mar$ci.mu2
coverage.MAR["beta_12", "Mean Imp"] <- ms.mar$ci.beta12
coverage.MAR["beta_21", "Mean Imp"] <- ms.mar$ci.beta21
coverage.MAR["mu_1", "Regr Imp"] <- rs.mar$ci.mu1
coverage.MAR["mu_2", "Regr Imp"] <- rs.mar$ci.mu2
coverage.MAR["beta_12", "Regr Imp"] <- rs.mar$ci.beta12
coverage.MAR["beta_21", "Regr Imp"] <- rs.mar$ci.beta21
coverage.MAR["mu_1", "Mult Imp"] <- mi.mar$ci.mu1
coverage.MAR["mu_2", "Mult Imp"] <- mi.mar$ci.mu2
coverage.MAR["beta_12", "Mult Imp"] <- mi.mar$ci.beta12
coverage.MAR["beta_21", "Mult Imp"] <- mi.mar$ci.beta21
round(coverage.MAR, 3)
```

```
          CC Mean Imp Regr Imp Mult Imp
mu_1    0.000    0.950    0.950    0.950
mu_2    0.007    0.001    0.811    0.944
beta_12 0.304    0.612    0.053    0.949
beta_21 0.947    0.000    0.654    0.948
```

Average confidence-interval width:

```
width.MAR <- matrix(0, 4, 4)
rownames(width.MAR) <- c("mu_1", "mu_2", "beta_12", "beta_21")
colnames(width.MAR) <- c("CC", "Mean Imp", "Regr Imp", "Mult Imp")
width.MAR["mu_1", "CC"] <- cc.mar$ci.mu1.width
width.MAR["mu_2", "CC"] <- cc.mar$ci.mu2.width
width.MAR["beta_12", "CC"] <- cc.mar$ci.beta12.width
width.MAR["beta_21", "CC"] <- cc.mar$ci.beta21.width
width.MAR["mu_1", "Mean Imp"] <- ms.mar$ci.mu1.width
width.MAR["mu_2", "Mean Imp"] <- ms.mar$ci.mu2.width
width.MAR["beta_12", "Mean Imp"] <- ms.mar$ci.beta12.width
width.MAR["beta_21", "Mean Imp"] <- ms.mar$ci.beta21.width
width.MAR["mu_1", "Regr Imp"] <- rs.mar$ci.mu1.width
width.MAR["mu_2", "Regr Imp"] <- rs.mar$ci.mu2.width
width.MAR["beta_12", "Regr Imp"] <- rs.mar$ci.beta12.width
width.MAR["beta_21", "Regr Imp"] <- rs.mar$ci.beta21.width
width.MAR["mu_1", "Mult Imp"] <- mi.mar$ci.mu1.width
width.MAR["mu_2", "Mult Imp"] <- mi.mar$ci.mu2.width
width.MAR["beta_12", "Mult Imp"] <- mi.mar$ci.beta12.width
width.MAR["beta_21", "Mult Imp"] <- mi.mar$ci.beta21.width
round(width.MAR, 3)
```

```
          CC Mean Imp Regr Imp Mult Imp
```



```
mu_1    0.789    0.746    0.746    0.742
mu_2    1.191    0.710    0.879    1.425
beta_12 0.174    0.244    0.140    0.176
beta_21 0.396    0.221    0.192    0.466
```

Data Missing Completely At Random (MAR)

Average parameter estimates:

```
mean.MCAR <- matrix(0, 4, 4)
rownames(mean.MCAR) <- c("mu_1", "mu_2", "beta_12", "beta_21")
colnames(mean.MCAR) <- c("CC", "Mean Imp", "Regr Imp", "Mult Imp")
mean.MCAR["mu_1", "CC"] <- cc.mcar$xbar1
mean.MCAR["mu_2", "CC"] <- cc.mcar$xbar2
mean.MCAR["beta_12", "CC"] <- cc.mcar$b12
mean.MCAR["beta_21", "CC"] <- cc.mcar$b21
mean.MCAR["mu_1", "Mean Imp"] <- ms.mcar$xbar1
mean.MCAR["mu_2", "Mean Imp"] <- ms.mcar$xbar2
mean.MCAR["beta_12", "Mean Imp"] <- ms.mcar$b12
mean.MCAR["beta_21", "Mean Imp"] <- ms.mcar$b21
mean.MCAR["mu_1", "Regr Imp"] <- rs.mcar$xbar1
mean.MCAR["mu_2", "Regr Imp"] <- rs.mcar$xbar2
mean.MCAR["beta_12", "Regr Imp"] <- rs.mcar$b12
mean.MCAR["beta_21", "Regr Imp"] <- rs.mcar$b21
mean.MCAR["mu_1", "Mult Imp"] <- mi.mcar$xbar1
mean.MCAR["mu_2", "Mult Imp"] <- mi.mcar$xbar2
mean.MCAR["beta_12", "Mult Imp"] <- mi.mcar$b12
mean.MCAR["beta_21", "Mult Imp"] <- mi.mcar$b21
round(mean.MCAR, 3)
```

```
           CC Mean Imp Regr Imp Mult Imp
mu_1    9.999    9.998    9.998    9.998
mu_2   19.993   19.993   19.992   19.995
beta_12 0.500    0.500    0.645    0.500
beta_21 0.890    0.531    0.890    0.890
```

Root-mean-square error of the estimates:

```
RMSE.MCAR <- matrix(0, 4, 4)
rownames(RMSE.MCAR) <- c("mu_1", "mu_2", "beta_12", "beta_21")
colnames(RMSE.MCAR) <- c("CC", "Mean Imp", "Regr Imp", "Mult Imp")
RMSE.MCAR["mu_1", "CC"] <- cc.mcar$rmse.xbar1
RMSE.MCAR["mu_2", "CC"] <- cc.mcar$rmse.xbar2
RMSE.MCAR["beta_12", "CC"] <- cc.mcar$rmse.beta12
RMSE.MCAR["beta_21", "CC"] <- cc.mcar$rmse.beta21
RMSE.MCAR["mu_1", "Mean Imp"] <- ms.mcar$rmse.xbar1
RMSE.MCAR["mu_2", "Mean Imp"] <- ms.mcar$rmse.xbar2
RMSE.MCAR["beta_12", "Mean Imp"] <- ms.mcar$rmse.beta12
RMSE.MCAR["beta_21", "Mean Imp"] <- ms.mcar$rmse.beta21
RMSE.MCAR["mu_1", "Regr Imp"] <- rs.mcar$rmse.xbar1
```



```
RMSE.MCAR["mu_2", "Regr Imp"] <- rs.mcar$rmse.xbar2
RMSE.MCAR["beta_12", "Regr Imp"] <- rs.mcar$rmse.beta12
RMSE.MCAR["beta_21", "Regr Imp"] <- rs.mcar$rmse.beta21
RMSE.MCAR["mu_1", "Mult Imp"] <- mi.mcar$rmse.xbar1
RMSE.MCAR["mu_2", "Mult Imp"] <- mi.mcar$rmse.xbar2
RMSE.MCAR["beta_12", "Mult Imp"] <- mi.mcar$rmse.beta12
RMSE.MCAR["beta_21", "Mult Imp"] <- mi.mcar$rmse.beta21
round(RMSE.MCAR, 3)
```

```
          CC Mean Imp Regr Imp Mult Imp
mu_1    0.248    0.191    0.191    0.191
mu_2    0.330    0.330    0.295    0.302
beta_12 0.046    0.046    0.150    0.042
beta_21 0.080    0.364    0.080    0.083
```

Confidence-interval coverage:

```
coverage.MCAR <- matrix(0, 4, 4)
rownames(coverage.MCAR) <- c("mu_1", "mu_2", "beta_12", "beta_21")
colnames(coverage.MCAR) <- c("CC", "Mean Imp", "Regr Imp", "Mult Imp")
coverage.MCAR["mu_1", "CC"] <- cc.mcar$ci.mu1
coverage.MCAR["mu_2", "CC"] <- cc.mcar$ci.mu2
coverage.MCAR["beta_12", "CC"] <- cc.mcar$ci.beta12
coverage.MCAR["beta_21", "CC"] <- cc.mcar$ci.beta21
coverage.MCAR["mu_1", "Mean Imp"] <- ms.mcar$ci.mu1
coverage.MCAR["mu_2", "Mean Imp"] <- ms.mcar$ci.mu2
coverage.MCAR["beta_12", "Mean Imp"] <- ms.mcar$ci.beta12
coverage.MCAR["beta_21", "Mean Imp"] <- ms.mcar$ci.beta21
coverage.MCAR["mu_1", "Regr Imp"] <- rs.mcar$ci.mu1
coverage.MCAR["mu_2", "Regr Imp"] <- rs.mcar$ci.mu2
coverage.MCAR["beta_12", "Regr Imp"] <- rs.mcar$ci.beta12
coverage.MCAR["beta_21", "Regr Imp"] <- rs.mcar$ci.beta21
coverage.MCAR["mu_1", "Mult Imp"] <- mi.mcar$ci.mu1
coverage.MCAR["mu_2", "Mult Imp"] <- mi.mcar$ci.mu2
coverage.MCAR["beta_12", "Mult Imp"] <- mi.mcar$ci.beta12
coverage.MCAR["beta_21", "Mult Imp"] <- mi.mcar$ci.beta21
round(coverage.MCAR, 3)
```

```
          CC Mean Imp Regr Imp Mult Imp
mu_1    0.947    0.948    0.948    0.946
mu_2    0.956    0.739    0.852    0.947
beta_12 0.952    0.974    0.042    0.948
beta_21 0.958    0.000    0.779    0.955
```

Confidence-interval width:

```
width.MCAR <- matrix(0, 4, 4)
rownames(width.MCAR) <- c("mu_1", "mu_2", "beta_12", "beta_21")
colnames(width.MCAR) <- c("CC", "Mean Imp", "Regr Imp", "Mult Imp")
```



```
width.MCAR["mu_1", "CC"] <- cc.mcar$ci.mu1.width
width.MCAR["mu_2", "CC"] <- cc.mcar$ci.mu2.width
width.MCAR["beta_12", "CC"] <- cc.mcar$ci.beta12.width
width.MCAR["beta_21", "CC"] <- cc.mcar$ci.beta21.width
width.MCAR["mu_1", "Mean Imp"] <- ms.mcar$ci.mu1.width
width.MCAR["mu_2", "Mean Imp"] <- ms.mcar$ci.mu2.width
width.MCAR["beta_12", "Mean Imp"] <- ms.mcar$ci.beta12.width
width.MCAR["beta_21", "Mean Imp"] <- ms.mcar$ci.beta21.width
width.MCAR["mu_1", "Regr Imp"] <- rs.mcar$ci.mu1.width
width.MCAR["mu_2", "Regr Imp"] <- rs.mcar$ci.mu2.width
width.MCAR["beta_12", "Regr Imp"] <- rs.mcar$ci.beta12.width
width.MCAR["beta_21", "Regr Imp"] <- rs.mcar$ci.beta21.width
width.MCAR["mu_1", "Mult Imp"] <- mi.mcar$ci.mu1.width
width.MCAR["mu_2", "Mult Imp"] <- mi.mcar$ci.mu2.width
width.MCAR["beta_12", "Mult Imp"] <- mi.mcar$ci.beta12.width
width.MCAR["beta_21", "Mult Imp"] <- mi.mcar$ci.beta21.width
round(width.MCAR, 3)
```

```
          CC Mean Imp Regr Imp Mult Imp
mu_1    0.966    0.748    0.748    0.744
mu_2    1.289    0.770    0.879    1.230
beta_12 0.181    0.209    0.139    0.171
beta_21 0.323    0.221    0.192    0.355
```

Data Missing Not at Random

Average parameter estimates

```
mean.MNAR <- matrix(0, 4, 4)
rownames(mean.MNAR) <- c("mu_1", "mu_2", "beta_12", "beta_21")
colnames(mean.MNAR) <- c("CC", "Mean Imp", "Regr Imp", "Mult Imp")
mean.MNAR["mu_1", "CC"] <- cc.mnar$xbar1
mean.MNAR["mu_2", "CC"] <- cc.mnar$xbar2
mean.MNAR["beta_12", "CC"] <- cc.mnar$b12
mean.MNAR["beta_21", "CC"] <- cc.mnar$b21
mean.MNAR["mu_1", "Mean Imp"] <- ms.mnar$xbar1
mean.MNAR["mu_2", "Mean Imp"] <- ms.mnar$xbar2
mean.MNAR["beta_12", "Mean Imp"] <- ms.mnar$b12
mean.MNAR["beta_21", "Mean Imp"] <- ms.mnar$b21
mean.MNAR["mu_1", "Regr Imp"] <- rs.mnar$xbar1
mean.MNAR["mu_2", "Regr Imp"] <- rs.mnar$xbar2
mean.MNAR["beta_12", "Regr Imp"] <- rs.mnar$b12
mean.MNAR["beta_21", "Regr Imp"] <- rs.mnar$b21
mean.MNAR["mu_1", "Mult Imp"] <- mi.mnar$xbar1
mean.MNAR["mu_2", "Mult Imp"] <- mi.mnar$xbar2
mean.MNAR["beta_12", "Mult Imp"] <- mi.mnar$b12
mean.MNAR["beta_21", "Mult Imp"] <- mi.mnar$b21
round(mean.MNAR, 3)
```



```
           CC Mean Imp Regr Imp Mult Imp
mu_1    10.960    9.995    9.995    9.995
mu_2    21.945   21.945   21.270   21.265
beta_12  0.501    0.501    0.735    0.551
beta_21  0.699    0.357    0.699    0.699
```

Root-mean-square error of the estimates:

```
RMSE.MNAR <- matrix(0, 4, 4)
rownames(RMSE.MNAR) <- c("mu_1", "mu_2", "beta_12", "beta_21")
colnames(RMSE.MNAR) <- c("CC", "Mean Imp", "Regr Imp", "Mult Imp")
RMSE.MNAR["mu_1", "CC"] <- cc.mnar$rmse.xbar1
RMSE.MNAR["mu_2", "CC"] <- cc.mnar$rmse.xbar2
RMSE.MNAR["beta_12", "CC"] <- cc.mnar$rmse.beta12
RMSE.MNAR["beta_21", "CC"] <- cc.mnar$rmse.beta21
RMSE.MNAR["mu_1", "Mean Imp"] <- ms.mnar$rmse.xbar1
RMSE.MNAR["mu_2", "Mean Imp"] <- ms.mnar$rmse.xbar2
RMSE.MNAR["beta_12", "Mean Imp"] <- ms.mnar$rmse.beta12
RMSE.MNAR["beta_21", "Mean Imp"] <- ms.mnar$rmse.beta21
RMSE.MNAR["mu_1", "Regr Imp"] <- rs.mnar$rmse.xbar1
RMSE.MNAR["mu_2", "Regr Imp"] <- rs.mnar$rmse.xbar2
RMSE.MNAR["beta_12", "Regr Imp"] <- rs.mnar$rmse.beta12
RMSE.MNAR["beta_21", "Regr Imp"] <- rs.mnar$rmse.beta21
RMSE.MNAR["mu_1", "Mult Imp"] <- mi.mnar$rmse.xbar1
RMSE.MNAR["mu_2", "Mult Imp"] <- mi.mnar$rmse.xbar2
RMSE.MNAR["beta_12", "Mult Imp"] <- mi.mnar$rmse.beta12
RMSE.MNAR["beta_21", "Mult Imp"] <- mi.mnar$rmse.beta21
round(RMSE.MNAR, 3)
```

```
           CC Mean Imp Regr Imp Mult Imp
mu_1    0.989    0.196    0.196    0.196
mu_2    1.964    1.964    1.297    1.294
beta_12 0.055    0.055    0.241    0.073
beta_21 0.206    0.535    0.206    0.207
```

Confidence-interval converage:

```
coverage.MNAR <- matrix(0, 4, 4)
rownames(coverage.MNAR) <- c("mu_1", "mu_2", "beta_12", "beta_21")
colnames(coverage.MNAR) <- c("CC", "Mean Imp", "Regr Imp", "Mult Imp")
coverage.MNAR["mu_1", "CC"] <- cc.mnar$ci.mu1
coverage.MNAR["mu_2", "CC"] <- cc.mnar$ci.mu2
coverage.MNAR["beta_12", "CC"] <- cc.mnar$ci.beta12
coverage.MNAR["beta_21", "CC"] <- cc.mnar$ci.beta21
coverage.MNAR["mu_1", "Mean Imp"] <- ms.mnar$ci.mu1
coverage.MNAR["mu_2", "Mean Imp"] <- ms.mnar$ci.mu2
coverage.MNAR["beta_12", "Mean Imp"] <- ms.mnar$ci.beta12
coverage.MNAR["beta_21", "Mean Imp"] <- ms.mnar$ci.beta21
coverage.MNAR["mu_1", "Regr Imp"] <- rs.mnar$ci.mu1
```

```
coverage.MNAR["mu_2", "Regr Imp"] <- rs.mnar$ci.mu2
coverage.MNAR["beta_12", "Regr Imp"] <- rs.mnar$ci.beta12
coverage.MNAR["beta_21", "Regr Imp"] <- rs.mnar$ci.beta21
coverage.MNAR["mu_1", "Mult Imp"] <- mi.mnar$ci.mu1
coverage.MNAR["mu_2", "Mult Imp"] <- mi.mnar$ci.mu2
coverage.MNAR["beta_12", "Mult Imp"] <- mi.mnar$ci.beta12
coverage.MNAR["beta_21", "Mult Imp"] <- mi.mnar$ci.beta21
round(coverage.MNAR, 3)
```

```
          CC Mean Imp Regr Imp Mult Imp
mu_1    0.013    0.950    0.950    0.950
mu_2    0.000    0.000    0.000    0.013
beta_12 0.949    0.984    0.007    0.836
beta_21 0.329    0.000    0.105    0.423
```

Confidence-interval width:

```
width.MNAR <- matrix(0, 4, 4)
rownames(width.MNAR) <- c("mu_1", "mu_2", "beta_12", "beta_21")
colnames(width.MNAR) <- c("CC", "Mean Imp", "Regr Imp", "Mult Imp")
width.MNAR["mu_1", "CC"] <- cc.mnar$ci.mu1.width
width.MNAR["mu_2", "CC"] <- cc.mnar$ci.mu2.width
width.MNAR["beta_12", "CC"] <- cc.mnar$ci.beta12.width
width.MNAR["beta_21", "CC"] <- cc.mnar$ci.beta21.width
width.MNAR["mu_1", "Mean Imp"] <- ms.mnar$ci.mu1.width
width.MNAR["mu_2", "Mean Imp"] <- ms.mnar$ci.mu2.width
width.MNAR["beta_12", "Mean Imp"] <- ms.mnar$ci.beta12.width
width.MNAR["beta_21", "Mean Imp"] <- ms.mnar$ci.beta21.width
width.MNAR["mu_1", "Regr Imp"] <- rs.mnar$ci.mu1.width
width.MNAR["mu_2", "Regr Imp"] <- rs.mnar$ci.mu2.width
width.MNAR["beta_12", "Regr Imp"] <- rs.mnar$ci.beta12.width
width.MNAR["beta_21", "Regr Imp"] <- rs.mnar$ci.beta21.width
width.MNAR["mu_1", "Mult Imp"] <- mi.mnar$ci.mu1.width
width.MNAR["mu_2", "Mult Imp"] <- mi.mnar$ci.mu2.width
width.MNAR["beta_12", "Mult Imp"] <- mi.mnar$ci.beta12.width
width.MNAR["beta_21", "Mult Imp"] <- mi.mnar$ci.beta21.width
round(width.MNAR, 3)
```

```
          CC Mean Imp Regr Imp Mult Imp
mu_1    0.892    0.746    0.746    0.742
mu_2    1.053    0.628    0.727    1.122
beta_12 0.222    0.270    0.180    0.216
beta_21 0.310    0.191    0.170    0.344
```

The conclusions to be drawn from the MCAR and MNAR data are consistent with the discussion in the text.

Exercise 20.5* To be completed.

Exercise 20.7

Here is a table of $\mathrm{RE}(\widetilde{\beta}_j)$ as a function of g and γ_j,

	γ_j					
g	0.05	0.1	0.2	0.5	0.9	0.99
1	0.952	0.909	0.833	0.667	0.526	0.503
2	0.976	0.952	0.909	0.800	0.690	0.669
3	0.984	0.968	0.938	0.857	0.769	0.752
5	0.990	0.980	0.962	0.909	0.847	0.835
10	0.995	0.990	0.980	0.952	0.917	0.910
20	0.998	0.995	0.990	0.976	0.957	0.953

and a table of $\sqrt{\mathrm{RE}(\widetilde{\beta}_j)}$,

	γ_j					
g	0.05	0.1	0.2	0.5	0.9	0.99
1	0.976	0.953	0.913	0.816	0.725	0.709
2	0.988	0.976	0.953	0.894	0.830	0.818
3	0.992	0.984	0.968	0.926	0.877	0.867
5	0.995	0.990	0.981	0.953	0.921	0.914
10	0.998	0.995	0.990	0.976	0.958	0.954
20	0.999	0.998	0.995	0.988	0.978	0.976

We do very well with g as small as 5, even for very high rates of missing information.

Exercise 20.9

(a) Here are the mean and variance as a function of the left-truncation point a:

	a				
$E(Y)$	0.055	0.288	0.798	1.525	2.373
$V(Y)$	0.886	0.630	0.363	0.199	0.114

(b)* We can take advantage of the symmetry of a normal distribution about its mean. Thus

$$
\begin{aligned}
E(Y) &= E(\xi|\xi \le a) = \mu - \sigma m(-z_a) \\
V(Y) &= V(\xi|\xi \le a) = \sigma^2[1 - d(-z_a)]
\end{aligned}
$$

As the threshold a moves to the left, so does the mean, and the variance decreases (as the distribution is increasingly squeezed).

Exercise 20.11*

(a) We can take advantage of the symmetry of the normal distribution to adapt the formulas for the left-censored case. First, define

$$
\begin{aligned}
\Phi'(z) &\equiv 1 - \Phi(z) \\
m'(z) &\equiv \frac{\phi(z)}{\Phi'(-z)} \\
d'(z) &\equiv m'(z)[m'(z) + z]
\end{aligned}
$$

Then if Y is derived from $\xi \sim N(\mu, \sigma^2)$ censored to the right at b, and $z_b \equiv (b - \mu)/\sigma$,

$$
\begin{aligned}
E(Y) &= b\Phi'(z_b) + [\mu - \sigma m'(z_b)][1 - \Phi'(z_b)] \\
V(Y) &= \sigma^2[1 - \Phi'(z_b)]\left\{1 - d'(z_b) + [z_b + m'(z_b)]^2\Phi'(z_b)\right\}
\end{aligned}
$$

(b) Now suppose that Y is derived from $\xi \sim N(\mu, \sigma^2)$ censored to the left at a and to the right at b. Then, combining the results in the text for left censoring with those in part (a),

$$\begin{aligned} E(Y) = & a\Phi(z_a) + b\Phi'(z_b) + \mu[1 - \Phi(z_a) - \Phi'(z_b)] \\ & + \sigma m(z_a)[1 - \Phi(z_a)] - \sigma m'(z_b)[1 - \Phi'(z_b)] \end{aligned}$$

$V(Y)$ To be completed.

Exercise 20.13*

This is similar to the previous problem in that it follows from the application of the general equation for $V(Y)$ in Equations 2.17, but now letting Y_i play the role of Y. Then

$$\begin{aligned} V(Y_i) &= \sigma^2_{\xi_i}[1 - \rho^2_{\xi_i\zeta_i} d(z_{-\psi_i})] \\ &= \sigma^2_\varepsilon \left\{1 - \rho^2_{\varepsilon\delta} m(-\psi_i)[m(-\psi_i) - (-\psi_i)]\right\} \\ &= \sigma^2_\varepsilon[1 - \rho^2_{\varepsilon\delta}\lambda_i(\lambda_i + \psi_i)] \end{aligned}$$

The transition to the second line of the equation is justified by noting that ε_i is the error associated with the latent response ξ_i and hence has the same, constant, conditional variance, σ^2_ε, as ξ_i; δ_i is the error associated with the selection response ζ_i; and $\rho_{\varepsilon\delta}$ is the constant correlation between the two errors and hence between the corresponding latent responses.

Exercise 20.15*

Recall that White's coefficient-variance estimator takes the form

$$\widetilde{V}(\mathbf{b}) = (\mathbf{X}'\mathbf{X})^{-1}\mathbf{X}'\widehat{\mathbf{\Sigma}}\mathbf{X}(\mathbf{X}'\mathbf{X})^{-1}$$

where $\widehat{\mathbf{\Sigma}} \equiv \text{diag}\{E_i^2\}$. In the current context, we can replace E_i^2 with the estimated conditional variance of Y_i, that is $\widehat{\sigma}^2_\varepsilon[1 - \widehat{\rho}^2_{\varepsilon\delta}\widehat{\lambda}_i(\widehat{\lambda}_i + \widehat{\psi}_i)]$.

To make this work, we need estimates of the various quantities. We already have $\widehat{\lambda}_i$ and $\widehat{\psi}_i$ from the first-step probit regression. We can simplify the notation a bit by writing $d_i \equiv d(-\widehat{\psi}_i) = \widehat{\lambda}_i(\widehat{\lambda}_i + \widehat{\psi}_i)$ and $\widehat{\rho} \equiv \widehat{\rho}_{\varepsilon\delta}$. Then $\widehat{V}(Y_i) = \widehat{\sigma}^2_\varepsilon(1 - \widehat{\rho}^2 d_i)$.

The variance of the least-squares residuals $\widehat{\nu}_i$ from the second-step regression estimates the average of these case-wise conditional variances. That is, $\sum \widehat{\nu}_i^2/n = \widehat{\sigma}^2_\varepsilon(1 - \widehat{\rho}^2\bar{d})$ where $\bar{d}$ is the mean of the d_i. The regression coefficient b_λ for $\widehat{\lambda}_i$ in the second step estimates $\rho\sigma_\varepsilon$, and so we can take $\widehat{\sigma}^2_\varepsilon = \sum \widehat{\nu}_i^2/n + \bar{d}b^2_\lambda$, and then $\widehat{\rho} = b_\lambda/\widehat{\sigma}_\varepsilon$.

Exercises for Chapter 21

Exercise 21.1*

Here is a simple approach to the problem that doesn't require the hint: Bootstrap sampling is just independent random sampling from the discrete population constituted by the original sample. Thus, the results in this exercise follow directly from the sampling distribution of the mean of an independent sample, as typically discussed in a basic statistics course.

In particular, each value $Y_{i'}$, $i' = 1, \ldots, n$, is selected with equal probability $p_{i'} = 1/n$ for each draw Y^*_{ib}, $i = 1, \ldots, n$ of the bootstrap sample. (It's true that some values $Y_{i'}$ might be duplicated, say $k_{i'}$ times, but then the probability of drawing that duplicated value is $k_{i'}\frac{1}{n}$, and so we can just as well treat the duplicated values individually, as I have done.) Each of the n draws therefore has expectation $E^*(Y^*_{ib}) = \sum_{i'=1}^{n} \frac{1}{n} Y_{i'} = \bar{Y}$, and thus $E^*(\bar{Y}^*) = \frac{1}{n}\sum_{i=1}^{n} E^*(Y^*_{ib}) = \frac{1}{n} \times n \times \bar{Y} = \bar{Y}$, as required.

Similarly, from the sampling distribution for the mean of an independent sample, the variance of the bootstrap mean is the variance of the bootstrapped values divided by n. The variance of each bootstrap draw is $V^*(Y_{ib}) = \sum_{i=1}^{n} \frac{1}{n}(Y_i - \bar{Y})^2$. Then $V^*(\bar{Y}^*) = \frac{1}{n} V^*(Y_{ib}) = \frac{1}{n^2}\sum_{i=1}^{n}(Y_i - \bar{Y})^2 = \frac{1}{n} S_n^2$ where S_n^2 is the sample variance of Y using n rather than the more typical $n - 1$ in the denominator; I denote the latter simply S^2. The standard deviation of the bootstrap means is the square root of their bootstrap variance. There is therefore a small error in the question: The bootstrap standard error of $\bar{Y}^*$ is $\mathrm{SE}^*(\bar{Y}^*) = \frac{\sqrt{n-1}S}{n}$, not $\frac{S}{\sqrt{n-1}}$, as given in the question.

Exercise 21.3

The `Boot()` function in the **car** package for R performs a more complex kind of fixed-X resampling than the version described in the text, and so I wrote simple R functions for random and fixed-X resampling to use for this problem:

```
> bootRandom <- function(model, B=1000){
+   y <- model.response(model.frame(model))
+   X <- model.matrix(model)
+   n <- nrow(X)
+   coef <- matrix(0, B, ncol(X))
+   colnames(coef) <- colnames(X)
+   residuals <- fitted <- matrix(0, B, n)
+   colnames(residuals) <- paste0("e", 1:n)
+   colnames(fitted) <- paste0("yhat", 1:n)
+   for (b in 1:B){
+     cases <- sample(n, n, replace=TRUE)
+     m <- lm(y[cases] ~ X[cases, ] - 1)
+     coef[b, ] <- coef(m)
+     residuals[b, ] <- residuals(m)
+     fitted[b, ] <- fitted(m)
+   }
+   list(coef=coef, fitted=fitted, residuals=residuals)
+ }

> bootFixed <- function(model, B=1000){
+   yhat <- fitted(model)
+   e <- residuals(model)
+   X <- model.matrix(model)
+   n <- nrow(X)
+   coef <- matrix(0, B, ncol(X))
+   colnames(coef) <- colnames(X)
+   residuals <- fitted <- matrix(0, B, n)
+   colnames(residuals) <- paste0("e", 1:n)
+   colnames(fitted) <- paste0("yhat", 1:n)
+   for (b in 1:B){
+     cases <- sample(n, n, replace=TRUE)
```



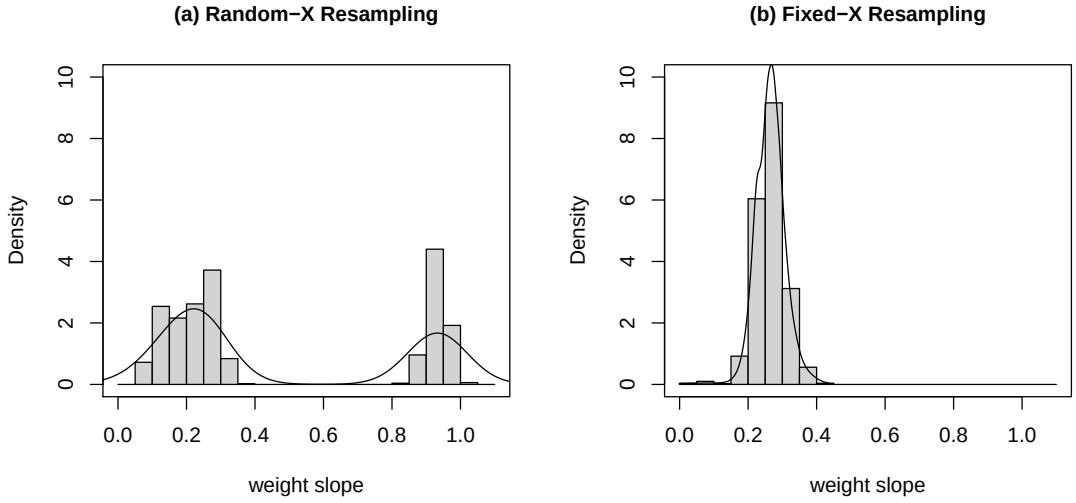
```
+     y <- yhat + e[cases]
+     m <- lm(y ~ X - 1)
+     coef[b, ] <- coef(m)
+     residuals[b, ] <- residuals(m)
+     fitted[b, ] <- fitted(m)
+   }
+   list(coef=coef, fitted=fitted, residuals=residuals)
+ }
```

(a) I drew the two histograms and density estimates on the same scale to facilitate comparison:

```
> library("car") # for Davis data and adaptiveKernel()
Loading required package: carData

>   # remove NAs
> Females <- na.omit(Davis[Davis$sex == "F", c("repwt", "weight")])
> m.davis <- lm(repwt ~ weight, data=Females)
> summary(m.davis)

Call:
lm(formula = repwt ~ weight, data = Females)

Residuals:
     Min       1Q   Median       3Q      Max
-29.2230  -3.0746  -0.1325   3.3386  15.5783

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept) 41.32276    2.72086  15.187  < 2e-16 ***
weight       0.26446    0.04558   5.802  7.9e-08 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 5.85 on 99 degrees of freedom
Multiple R-squared:  0.2537,    Adjusted R-squared:  0.2462
F-statistic: 33.66 on 1 and 99 DF,  p-value: 7.9e-08

> set.seed(2308345) # for reproducibility
> boot.random <- bootRandom(m.davis)
> boot.fixed <- bootFixed(m.davis)

>   # bootstrap SEs:

> apply(boot.random$coef, 2, sd)
(Intercept)      weight
 20.0951786   0.3535695

> apply(boot.fixed$coef, 2, sd)
(Intercept)      weight
 2.65407014  0.04438204

> par(mfrow=c(1, 2))

> hist(boot.random$coef[, 2], freq=FALSE, xlab="weight slope",
+      main="(a) Random-X Resampling", cex.main=1,
+      breaks=seq(0, 1.1, by=0.05), ylim=c(0, 10))
> box()
> lines(adaptiveKernel(boot.random$coef[, 2]))
```



```
> hist(boot.fixed$coef[, 2], freq=FALSE, xlab="weight slope",
+      main="(b) Fixed-X Resampling", cex.main=1,
+      breaks=seq(0, 1.1, by=0.05), ylim=c(0, 10))
> box()
> lines(adaptiveKernel(boot.fixed$coef[, 2]))
```

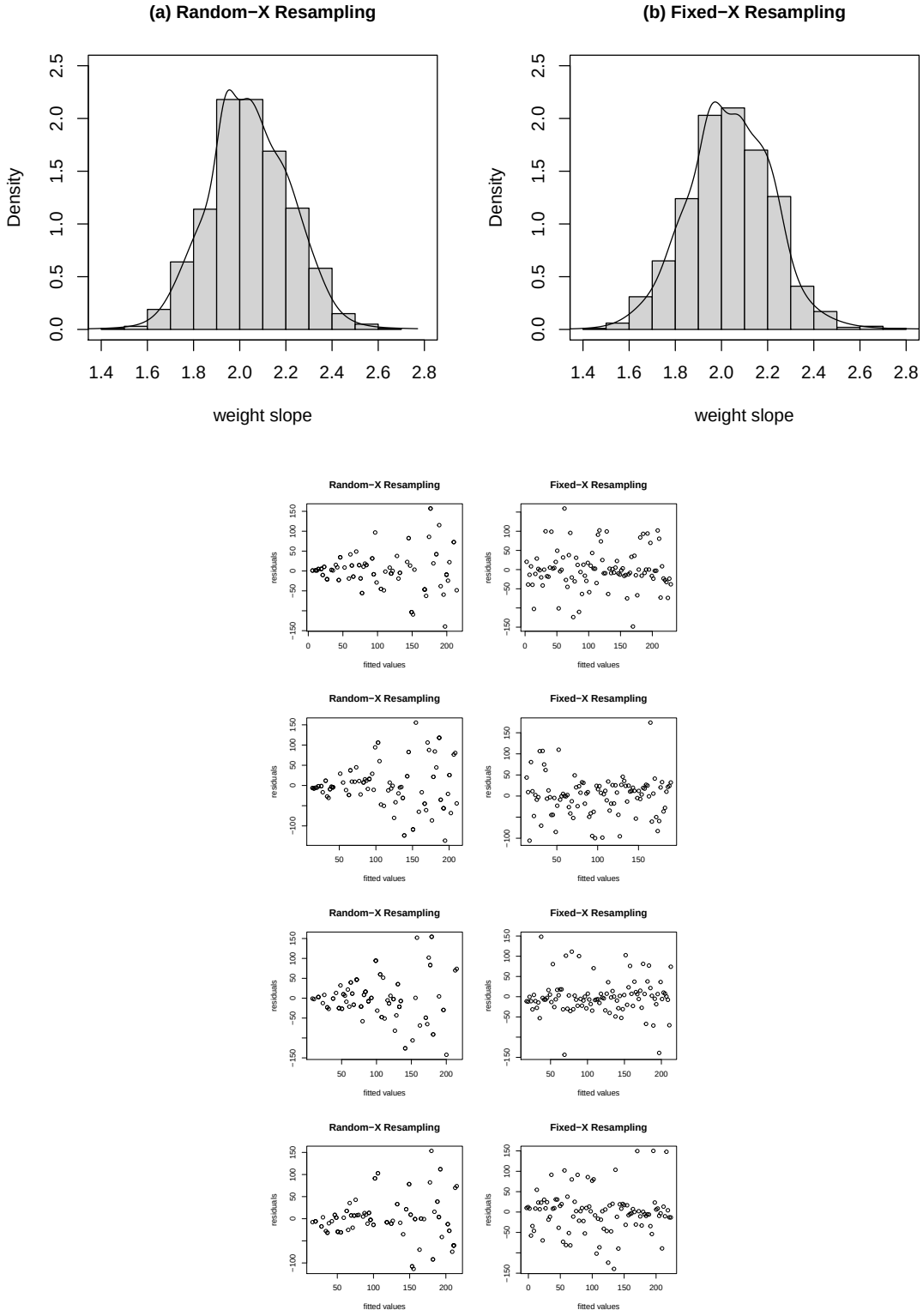


In random-X resampling, the results differ drastically depending on whether the outlier appears in a bootstrap sample (possibly more than once) or not, producing a bimodal bootstrap distribution of the slope. In fixed-X resampling, the large residual for the outlier moves around the data and its effect on the results isn't apparent. As well, the bootstrap standard error of the slope is much larger in fixed-X resampling than in random-X resampling.

(b)
```
> set.seed(54375921) # for reproducibility
> x <- 1:100
> y <- 5 + 2*x + rnorm(100, 0, sd=x)
> Data <- data.frame(x, y)
> m <- lm(y ~ x, data=Data)
> summary(m)

Call:
lm(formula = y ~ x, data = Data)

Residuals:
     Min       1Q   Median       3Q      Max
-139.341  -24.386   -1.884   16.641  156.620

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)   9.5300    10.4175   0.915    0.363
x             2.0414     0.1791  11.398   <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 51.7 on 98 degrees of freedom
Multiple R-squared:   0.57,     Adjusted R-squared:  0.5656
F-statistic: 129.9 on 1 and 98 DF,  p-value: < 2.2e-16

> boot.random <- bootRandom(m)
> boot.fixed <- bootFixed(m)
```



```
> # bootstrap SEs:

> apply(boot.random$coef, 2, sd)
(Intercept)           x
  6.1739242   0.1776632

> apply(boot.fixed$coef, 2, sd)
(Intercept)           x
  10.596893    0.183641

> par(mfrow=c(1, 2))

# bootstrap distributions of the slope:

> hist(boot.random$coef[, 2], freq=FALSE, xlab="weight slope",
+      main="(a) Random-X Resampling", cex.main=1,
+      xlim=c(1.4, 2.8), ylim=c(0, 2.5))
> box()
> lines(adaptiveKernel(boot.random$coef[, 2]))

> hist(boot.fixed$coef[, 2], freq=FALSE, xlab="weight slope",
+      main="(b) Fixed-X Resampling", cex.main=1,
+      xlim=c(1.4, 2.8), ylim=c(0, 2.5))
> box()
> lines(adaptiveKernel(boot.fixed$coef[, 2]))

> par(mfrow=c(4, 2))

> residuals.random <- boot.random$residuals
> fitted.random <- boot.random$fitted
> residuals.fixed <- boot.fixed$residuals
> fitted.fixed <- boot.fixed$fitted

>    # residuals vs. fitted values for the first 4 bootstrap replications:
> for (i in 1:4){
+   plot(fitted.random[i, ], residuals.random[i, ],
+        xlab="fitted values", ylab="residuals",
+        main="Random-X Resampling")
+   plot(fitted.fixed[i, ], residuals.fixed[i, ],
+        xlab="fitted values", ylab="residuals",
+        main="Fixed-X Resampling")
+ }
```

In this case, the bootstrap distributions and standard errors are quite similar for random and fixed-X resampling, but the plots of residuals against fitted values (which I drew for the first four bootstrap samples produced by each method) are quite different, with the plots for the random-X bootstrap replications capturing the pattern of nonconstant error variance while the plots for the fixed-X bootstrap

replications miss the pattern. The difference arises because fixed-X resampling spreads the residuals randomly among the observations, while fixed-X resampling keeps each X-value with the corresponding Y-value.

(c) It's apparent from these examples that fixed-X resampling depends more on the correctness of the structure of the model and can fail to detect problems when the model is defective in some manner. That's particularly apparent in the part (a), where fixed-X resampling fails to reflect the impact of the outlier.

Exercise 21.5*

I'll use the `rlm()` (robust linear model) function from the **MASS** package for R to obtain the Huber M-estimator, the `linearHypothesis()` function from the **car** package to compute test statistics for the hypothesis, and the `Boot()` function, also from the **car** package, to perform bootstrap resampling. Loading the **car** package gives us direct access to the Duncan data set.

As you'll see, `linearHypothesis()` computes an F-test rather than a χ^2 test; although I could force it to compute a χ^2 test, it's probably not worth the extra programming effort to do so, particularly because we'd get the same bootstrapped p-value for both test statistics (which are monotone functions of each other), and the traditional F-test (as opposed to the bootstrap test) is likely better than the χ^2 test in a small sample .

Here are the estimated model and test for the original data set:

```
> library("car")  # for data, linearHypothesis(), and Boot()
Loading required package: carData
> library("MASS") # for rlm()
> m <- rlm(prestige ~ income + education, data=Duncan)
> summary(m)

Call: rlm(formula = prestige ~ income + education, data = Duncan)
Residuals:
    Min      1Q  Median      3Q     Max
-3.120  -6.889   1.291   4.592  38.603

Coefficients:
            Value   Std. Error t value
(Intercept) -7.1107  3.8813    -1.8320
income       .7014   .1087     6.4516
education    .4854   .0893     5.4380

Residual standard error: 9.892 on 42 degrees of freedom
> linearHypothesis(m, c("income", "education"))
Linear hypothesis test

Hypothesis:
income = 0
education = 0

Model 1: restricted model
Model 2: prestige ~ income + education

  Res.Df Df      F    Pr(>F)
1     44
2     42  2 128.43 < 2.2e-16 ***
---
Signif. codes:  0'***' .001'**' .01'*' .05'.' .1' ' 1
```

The p-value for the test is effectively 0 (i.e., $p < 2.2 \times 10^{-16}$).

Next, I'll write a small function to extract the test statistic from the object returned by `linearHypothesis()`, to be used with the `Boot()` function. Recall that for the bootstrap test, the hypothesis should test whether the estimated coefficients are equal to their values in the original data set. Applied to the model fit to the original data set, we should therefore get a test statistic of $F = 0$, within rounding error (in our case $F = 1.15 \times 10^{-29} \approx 0$):

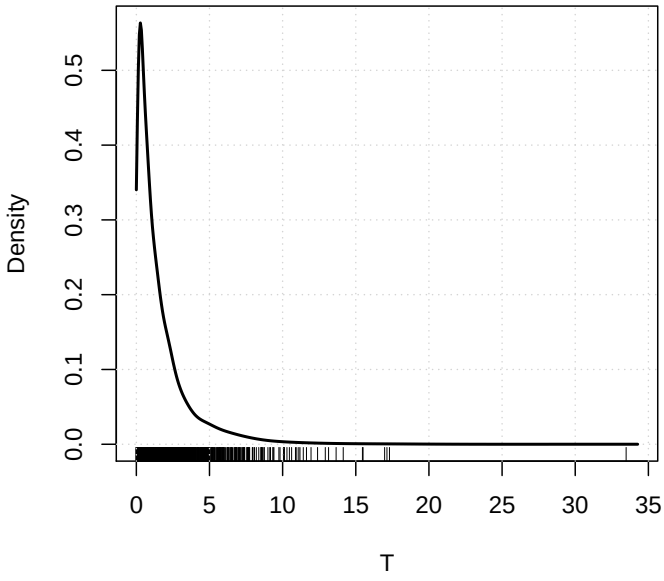
```
> (b <- coef(m))
(Intercept)      income   education
 -7.1107028   .7014493   .4854390
> T <- function(model){
+     linearHypothesis(model, c(paste("income=", b["income"]),
+                               paste("education=", b["education"])))$F[2]
+ }
> T(m)
[1] 1.148245e-29
```

I'll base the bootstrap test on 2,000 replications:

```
> set.seed(6858046) # for reproducibility
> boots <- Boot(m, f=T, R=2000)
Loading required namespace: boot
There were 37 warnings (use warnings() to see them)
> warnings()[1]
Warning message:
In rlm.default(x, y, weights, method = method, wt.method = wt.method,  ... :
  'rlm' failed to converge in 20 steps
> sum(boots$t > 128.43)
[1] 0
```

I've hit a small snag: the Huber M-estimator failed to converge in a few of the bootstrap samples. Again, I could program around this problem, for example, trapping the convergence failures, but it's probably not worth the effort because the result is unambiguous: The obtained bootstrap F-statistics exceed the original test statistic in *none* of the bootstrap samples, and so the boostrapped p-value is 0 (or more pedantically, $p < 1/2000$). Here's a density estimate (with rug plot) of the 2000 bootstrapped Fs:

```
> densityPlot(boots$t, from=0, xlab="T")
```

It's clear from the graph that none of the bootstrapped test statistics gets near the observed $F = 128$.

Exercise 21.7*

We treat the X-values as fixed to preserve the time-series structure of the data. If we simply resampled cases, we'd lose the sequence of the observations.

It's straightforward to implement the time-series bootstrapping procedure described in this exercise in R. I'll first fit the model with AR(1) errors by ML using the `gls()` (generalized least squares) function in the **nlme** package. The data reside conveniently in the `Hartnagel` data frame in the **carData** package:

```
> library("carData")
> library("nlme")
> m.ar1 <- gls(fconvict ~ tfr + partic + degrees + mconvict,
+              data=Hartnagel, correlation=corAR1(), method="ML")
> summary(m.ar1)
Generalized least squares fit by maximum likelihood
  Model: fconvict ~ tfr + partic + degrees + mconvict
  Data: Hartnagel
       AIC      BIC    logLik
  312.4234 323.8865 -149.2117

Correlation Structure: AR(1)
 Formula: ~1
 Parameter estimate(s):
      Phi
.8015954

Coefficients:
               Value Std.Error    t-value p-value
(Intercept) 152.20280  81.40131  1.8697833  .0704
tfr          -.03169   .01532 -2.0686521  .0465
partic        .05400   .12694  .4254100  .6733
degrees       .01047   .30897  .0338871  .9732
mconvict      .02666   .03896  .6842862  .4986
```



```
 Correlation:
          (Intr) tfr    partic degres
tfr       -.822
partic    -.623  .293
degrees   -.247  .347 -.177
mconvict -.529  .155  .125 -.004

Standardized residuals:
       Min         Q1        Med         Q3        Max
-1.1156221 -.7326178 -.2830969  .3014207  2.7527218

Residual standard error: 2.25772
Degrees of freedom: 38 total; 33 residual
```

I'll then write a function to generate the estimates of the errors ν_i (i.e., the V_is in the question) and another function to draw a single bootstrap sample of the Y_is:

```
> nuAR1 <- function(e, rho){
+     n <- length(e)
+     nu <- numeric(n)
+     nu[1] <- e[1]
+     for (i in 2:n){
+         nu[i] <- e[i] - rho*e[i - 1]
+     }
+     nu
+ }
>
> bootstrapSampleAR1 <- function(yhat, nu, rho){
+     n <- length(nu)
+     nus <- sample(nu, n, replace=TRUE)
+     es <- numeric(n)
+     es[1] <- nus[1]
+     for (i in 2:n){
+         es[i] <- rho*es[i - 1] + nus[i]
+     }
+     yhat + es
+ }
```

Next, I'll generate $B = 1000$ bootstrap samples of the Ys, redo the regression for each bootstrap sample, and save the resulting regression coefficients:

```
> B <- 1000
> b.coefs <- matrix(0, B, length(coef(m.ar1)))
> nu <- nuAR1(e=residuals(m.ar1), rho=.8015954)
> yhat <- fitted(m.ar1)
> set.seed(4308467) # for reproducibility
> for (b in 1:B){
+     y.b <- bootstrapSampleAR1(yhat, nu, .8015954)
+     b.coefs[b, ] <- coef(update(m.ar1, y.b ~ .))
+ }
```

Finally, the standard deviations of the bootstrapped regression coefficients are the bootstrap standard errors:

```
> apply(b.coefs, 2, sd)
[1] 8.78248934  .01511508  .12953714  .31772396  .03887480
```

In this case, the bootstrapped standard errors are very similar to the conventional asymptotic standard errors for the ML estimates of the regression coefficients:


```
> sqrt(diag(vcov(m.ar1)))
(Intercept)         tfr      partic     degrees    mconvict
81.40130852  .01532105  .12694397  .30896543  .03895725
```

This procedure is easily adapted to an AR(2) model for the errors. First, I'll use `gls()` to reproduce the results reported in the text (Equation 16.19 on page 493):

```
> m.ar2 <- gls(fconvict ~ tfr + partic + degrees + mconvict, data=Hartnagel,
+              correlation=corARMA(p=2), method="ML")
> summary(m.ar2)
Generalized least squares fit by maximum likelihood
  Model: fconvict ~ tfr + partic + degrees + mconvict
  Data: Hartnagel
       AIC      BIC    logLik
  305.4145 318.5152 -144.7073

Correlation Structure: ARMA(2,0)
 Formula: ~1
 Parameter estimate(s):
      Phi1       Phi2
 1.0683473 -.5507269

Coefficients:
               Value Std.Error   t-value p-value
(Intercept) 83.34028  59.47084  1.401364  .1704
tfr         -.03999   .00928 -4.308632  .0001
partic       .28761   .11201  2.567653  .0150
degrees     -.20984   .20658 -1.015757  .3171
mconvict     .07569   .03501  2.161899  .0380

 Correlation:
         (Intr) tfr    partic degres
tfr      -.773
partic   -.570  .176
degrees   .093  .033 -.476
mconvict -.689  .365  .047  .082

Standardized residuals:
       Min         Q1        Med         Q3        Max
-2.4991516 -.3716988 -.1494540  .3372409  2.9094711

Residual standard error: 17.70228
Degrees of freedom: 38 total; 33 residual
```

As before, I'll write functions to estimate the ν_is and to generate a bootstrap sample of the Y_is, and then use these to generate $B = 1000$ bootstrap samples and the corresponding regression coefficients:

```
> nuAR2 <- function(e, phi1, phi2){
+     n <- length(e)
+     nu <- numeric(n)
+     nu[1:2] <- e[1:2]
+     for (i in 3:n){
+         nu[i] <- e[i] - phi1*e[i - 1] - phi2*e[i - 2]
+     }
+     nu
+ }
>
> bootstrapSampleAR2 <- function(yhat, nu, phi1, phi2){
+     n <- length(nu)
```



```
+     nus <- sample(nu, n, replace=TRUE)
+     es <- numeric(n)
+     es[1:2] <- nus[1:2]
+     for (i in 3:n){
+         es[i] <- phi1*es[i - 1] + phi2*es[i - 2] + nus[i]
+     }
+     yhat + es
+ }
>
> B <- 1000
> b.coefs <- matrix(0, B, length(coef(m.ar2)))
> nu <- nuAR2(e=residuals(m.ar2), phi1=1.0683473 , phi2=-.5507269)
> yhat <- fitted(m.ar2)
> set.seed(8438573) # for reproducibility
> for (b in 1:B){
+     y.b <- bootstrapSampleAR2(yhat, nu, phi1=1.0683473,
+                               phi2=-.5507269)
+     b.coefs[b, ] <- coef(update(m.ar2, y.b ~ .))
+ }
>
> apply(b.coefs, 2, sd)
[1] 57.387591723  .009163869  .113360143  .202188110  .033093102
> sqrt(diag(vcov(m.ar2)))
 (Intercept)          tfr       partic      degrees     mconvict
59.470835291  .009280671  .112013488  .206580991  .035009033
```

The bootstrap standard errors are once again similar to the conventional coefficient standard errors.

Exercises for Chapter 22

Exercise 22.1

(a) I used R to generate and analyze the data:

```
> set.seed(137472394)  # for reproducibility
> Data <- as.data.frame(matrix(rnorm(500*101), 500, 101))
> mod.1 <- lm(V1 ~ ., data=Data)
> summary(mod.1)

Call:
lm(formula = V1 ~ ., data = Data)

Residuals:
     Min       1Q   Median       3Q      Max
-2.21776 -0.61594  0.00312  0.52684  2.52022

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept)  0.0138874  0.0484708   0.287   0.7746
V2          -0.0396278  0.0515072  -0.769   0.4421
V3           0.0872461  0.0490917   1.777   0.0763 .
V4           0.0652411  0.0478351   1.364   0.1734
V5           0.0326468  0.0515921   0.633   0.5272
V6          -0.0593439  0.0490481  -1.210   0.2270
V7           0.0267640  0.0476731   0.561   0.5748
V8          -0.1100395  0.0499235  -2.204   0.0281 *
V9           0.0475599  0.0498315   0.954   0.3405
V10         -0.0459965  0.0481835  -0.955   0.3404
V11         -0.0054681  0.0482714  -0.113   0.9099
V12         -0.0760712  0.0453468  -1.678   0.0942 .
V13         -0.0156193  0.0472535  -0.331   0.7412
V14          0.0159363  0.0488267   0.326   0.7443
V15         -0.0091271  0.0502167  -0.182   0.8559
V16         -0.0382370  0.0474418  -0.806   0.4207
V17          0.0690735  0.0471310   1.466   0.1436
V18         -0.0261989  0.0500812  -0.523   0.6012
V19         -0.0933564  0.0497180  -1.878   0.0611 .
V20         -0.0023633  0.0514322  -0.046   0.9634
V21         -0.0147989  0.0487386  -0.304   0.7616
V22         -0.0332344  0.0455643  -0.729   0.4662
V23         -0.0152634  0.0478530  -0.319   0.7499
V24         -0.0625352  0.0484450  -1.291   0.1975
V25          0.0234108  0.0519230   0.451   0.6523
V26         -0.0958155  0.0469544  -2.041   0.0419 *
V27         -0.0609216  0.0477549  -1.276   0.2028
V28          0.0282179  0.0482901   0.584   0.5593
V29         -0.0262295  0.0465917  -0.563   0.5738
V30         -0.0059334  0.0487864  -0.122   0.9033
V31          0.0652734  0.0502204   1.300   0.1944
V32          0.0506495  0.0485431   1.043   0.2974
V33         -0.0358884  0.0487992  -0.735   0.4625
V34         -0.0073339  0.0515772  -0.142   0.8870
V35         -0.0163436  0.0491378  -0.333   0.7396
V36          0.0364931  0.0500981   0.728   0.4668
V37         -0.0595755  0.0486485  -1.225   0.2214
V38         -0.0635972  0.0479555  -1.326   0.1855
V39         -0.0257037  0.0528781  -0.486   0.6272
V40         -0.0373742  0.0468585  -0.798   0.4256
```



```
V41          0.0167876  0.0497118   0.338   0.7358
V42         -0.0030788  0.0515829  -0.060   0.9524
V43         -0.0573513  0.0494979  -1.159   0.2473
V44         -0.0174783  0.0494978  -0.353   0.7242
V45          0.0304971  0.0487878   0.625   0.5323
V46         -0.0420323  0.0496042  -0.847   0.3973
V47          0.0082395  0.0482725   0.171   0.8646
V48         -0.0768555  0.0466368  -1.648   0.1001
V49         -0.0136467  0.0457084  -0.299   0.7654
V50         -0.0199305  0.0480899  -0.414   0.6788
V51          0.0247131  0.0477458   0.518   0.6050
V52         -0.0373444  0.0462600  -0.807   0.4200
V53         -0.0119503  0.0495916  -0.241   0.8097
V54         -0.0738473  0.0478947  -1.542   0.1239
V55         -0.0615077  0.0469668  -1.310   0.1911
V56         -0.0161867  0.0448362  -0.361   0.7183
V57         -0.0040034  0.0493313  -0.081   0.9354
V58         -0.0144516  0.0502976  -0.287   0.7740
V59         -0.0186445  0.0498004  -0.374   0.7083
V60          0.0468476  0.0484895   0.966   0.3346
V61          0.0284392  0.0502595   0.566   0.5718
V62         -0.0830301  0.0489702  -1.696   0.0908 .
V63          0.0594251  0.0485027   1.225   0.2212
V64         -0.0102879  0.0486041  -0.212   0.8325
V65          0.0388134  0.0487871   0.796   0.4268
V66          0.0279078  0.0488903   0.571   0.5684
V67         -0.0512207  0.0506632  -1.011   0.3126
V68          0.0587382  0.0468937   1.253   0.2111
V69          0.0051568  0.0508331   0.101   0.9192
V70          0.0413269  0.0506641   0.816   0.4152
V71         -0.0566214  0.0485982  -1.165   0.2447
V72          0.0021439  0.0505229   0.042   0.9662
V73         -0.0206714  0.0510546  -0.405   0.6858
V74         -0.0421982  0.0490304  -0.861   0.3899
V75         -0.0153635  0.0491446  -0.313   0.7547
V76          0.0227442  0.0464805   0.489   0.6249
V77         -0.0275705  0.0464025  -0.594   0.5527
V78         -0.0082844  0.0492279  -0.168   0.8664
V79          0.0467985  0.0489607   0.956   0.3397
V80         -0.0737237  0.0497805  -1.481   0.1394
V81          0.0252105  0.0480291   0.525   0.5999
V82         -0.0606381  0.0531426  -1.141   0.2545
V83          0.0618214  0.0504182   1.226   0.2209
V84         -0.0461935  0.0509764  -0.906   0.3654
V85          0.0004865  0.0456825   0.011   0.9915
V86         -0.0274232  0.0501451  -0.547   0.5848
V87         -0.0152845  0.0478293  -0.320   0.7495
V88          0.0072633  0.0495654   0.147   0.8836
V89          0.0515427  0.0499349   1.032   0.3026
V90         -0.0787349  0.0502769  -1.566   0.1181
V91         -0.0665580  0.0478359  -1.391   0.1649
V92          0.0802798  0.0483334   1.661   0.0975 .
V93          0.0071514  0.0468604   0.153   0.8788
V94          0.0744670  0.0470358   1.583   0.1142
V95          0.0570803  0.0489585   1.166   0.2444
V96          0.0059961  0.0467765   0.128   0.8981
V97          0.0544637  0.0488747   1.114   0.2658
V98          0.0365336  0.0509547   0.717   0.4738
V99         -0.0397640  0.0499706  -0.796   0.4267
```



```
V100         -0.0175037  0.0500740  -0.350   0.7269
V101         -0.0275651  0.0496512  -0.555   0.5791
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.9818 on 399 degrees of freedom
Multiple R-squared:  0.1857,    Adjusted R-squared:  -0.01835
F-statistic: 0.9101 on 100 and 399 DF,  p-value: 0.7115
```

The omnibus F-test produces a p-value of .71, which is not "statistically significant." Two of the 100 slope coefficients, for predictors 8 and 26, are "statistically significant" at the .05 level for a two-sided test. This is more or less what I expected: The true values of the β_js are all 0, and so the null hypotheses are all correct. I'd expect each null hypothesis to produce a Type-I error (rejecting a true H_0) 5 percent of the time. The omnibus test didn't produce one of these unlucky results. I'd expect about 5 of the 100 coefficients to be "statistically significant" by chance, and as it turned out, I observed only 2 such Type-I errors.

(b) The three predictors with the largest t-values are 8, 26, and 19. Here's the regression with just these three predictors:

```
> mod.2 <- lm(V1 ~ V8 + V26 + V19, data=Data)
> summary(mod.2)

Call:
lm(formula = V1 ~ V8 + V26 + V19, data = Data)

Residuals:
    Min      1Q  Median      3Q     Max
-2.3113 -0.6741 -0.0195  0.5894  3.4283

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept)  0.0003916  0.0431845   0.009   0.9928
V8          -0.0925030  0.0433772  -2.133   0.0335 *
V26         -0.0808518  0.0421630  -1.918   0.0557 .
V19         -0.0817653  0.0430856  -1.898   0.0583 .
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.9645 on 496 degrees of freedom
Multiple R-squared:  0.02311,   Adjusted R-squared:  0.0172
F-statistic: 3.911 on 3 and 496 DF,  p-value: 0.008853
```

In this regression, only predictor 8 is "statistically significant," and the p-values are about the same as in the initial regression, but now the omnibus F-test has a small p-value of .00885.

(c) I used the `step()` function in R for variable selection by backward elimination. I elided most of the lengthy output, showing only the last few steps:

```
> step(mod.1, k=1000, direction="backward")

. . .

Step:  AIC=5949.05
V1 ~ V8 + V19 + V48 + V54 + V97

       Df Sum of Sq    RSS    AIC
- V48   1    3.2209 454.78 4952.6
- V19   1    3.7169 455.27 4953.1
- V54   1    4.6031 456.16 4954.1
```



```
- V97    1     4.6357 456.19 4954.2
- V8     1     5.6474 457.21 4955.3
<none>               451.56 5949.0

Step:  AIC=4952.6
V1 ~ V8 + V19 + V54 + V97

       Df Sum of Sq    RSS    AIC
- V19    1     3.5457 458.32 3956.5
- V54    1     4.8318 459.61 3957.9
- V97    1     5.3093 460.09 3958.4
- V8     1     5.3841 460.16 3958.5
<none>               454.78 4952.6

Step:  AIC=3956.48
V1 ~ V8 + V54 + V97

       Df Sum of Sq    RSS    AIC
- V54    1     4.2131 462.54 2961.1
- V8     1     5.5074 463.83 2962.5
- V97    1     5.6111 463.94 2962.6
<none>               458.32 3956.5

Step:  AIC=2961.06
V1 ~ V8 + V97

       Df Sum of Sq    RSS    AIC
- V8     1     4.7994 467.34 1966.2
- V97    1     5.5337 468.07 1967.0
<none>               462.54 2961.1

Step:  AIC=1966.22
V1 ~ V97

       Df Sum of Sq    RSS     AIC
- V97    1     5.0246 472.36  971.57
<none>               467.34 1966.22

Step:  AIC=971.57
V1 ~ 1


Call:
lm(formula = V1 ~ 1, data = Data)

Coefficients:
(Intercept)
   0.003213
```

Note: Setting `k=1000` forces `step()` to run to completion; as a consequence, the criterion labeled "`AIC`" in the output isn't the usual AIC. The "best" model with three predictors has variables 8, 54, and 97:

```
> mod.3 <- lm(V1 ~ V8 + V54 + V97, data=Data)
> summary(mod.3)

Call:
lm(formula = V1 ~ V8 + V54 + V97, data = Data)
```



```
Residuals:
     Min       1Q   Median       3Q      Max
-2.27764 -0.67990 -0.04784  0.61895  3.08590

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept) -0.002034   0.043046  -0.047   0.9623
V8          -0.105993   0.043416  -2.441   0.0150 *
V54         -0.089611   0.041967  -2.135   0.0332 *
V97          0.105810   0.042939   2.464   0.0141 *
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.9613 on 496 degrees of freedom
Multiple R-squared:  0.02972,   Adjusted R-squared:  0.02385
F-statistic: 5.064 on 3 and 496 DF,  p-value: 0.001836
```

In this model, all three slope coefficients are "statistically significant," as is the omnibus F-test. The results seem more "promising" than those in part (a)

(d) This part of the exercise is a bit open-ended. I'll perform stepwise regression by backwards elimination, using the AIC as the criterion for model selection, which is the default in `step()`, showing the final model:

```
> step(mod.1, direction="backward")

. . .

Step:  AIC=-44.07
V1 ~ V3 + V4 + V6 + V8 + V9 + V12 + V17 + V19 + V24 + V26 + V37 +
    V48 + V54 + V62 + V79 + V80 + V91 + V92 + V94 + V97

       Df Sum of Sq    RSS     AIC
<none>              420.93 -44.067
- V80   1    1.7054 422.64 -44.045
- V79   1    1.7509 422.68 -43.991
- V37   1    1.7707 422.70 -43.968
- V92   1    1.7838 422.72 -43.952
- V12   1    1.8251 422.76 -43.904
- V4    1    1.8416 422.77 -43.884
- V91   1    1.8949 422.83 -43.821
- V6    1    1.9220 422.86 -43.789
- V24   1    2.0455 422.98 -43.643
- V17   1    2.1563 423.09 -43.512
- V9    1    2.4208 423.35 -43.200
- V3    1    2.8510 423.78 -42.692
- V26   1    2.8872 423.82 -42.649
- V97   1    3.2766 424.21 -42.190
- V94   1    3.3747 424.31 -42.074
- V62   1    3.5268 424.46 -41.895
- V19   1    3.5369 424.47 -41.883
- V48   1    3.8507 424.78 -41.514
- V54   1    5.2984 426.23 -39.813
- V8    1    5.7447 426.68 -39.289

Call:
lm(formula = V1 ~ V3 + V4 + V6 + V8 + V9 + V12 + V17 + V19 +
    V24 + V26 + V37 + V48 + V54 + V62 + V79 + V80 + V91 + V92 +
    V94 + V97, data = Data)
```



```
> mod.4 <- lm(V1 ~ V3 + V4 + V6 + V8 + V9 + V12 + V17 + V19 +
+   V24 + V26 + V37 + V48 + V54 + V62 + V79 + V80 + V91 + V92 +
+   V94 + V97, data = Data)
> summary(mod.4)

Call:
lm(formula = V1 ~ V3 + V4 + V6 + V8 + V9 + V12 + V17 + V19 +
    V24 + V26 + V37 + V48 + V54 + V62 + V79 + V80 + V91 + V92 +
    V94 + V97, data = Data)

Residuals:
    Min      1Q  Median      3Q     Max
-2.5332 -0.6464 -0.0126  0.5522  2.8187

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept)  0.002817   0.042346   0.067   0.9470
V3           0.077306   0.042919   1.801   0.0723 .
V4           0.061354   0.042382   1.448   0.1484
V6          -0.063069   0.042646  -1.479   0.1398
V8          -0.109620   0.042874  -2.557   0.0109 *
V9           0.071904   0.043323   1.660   0.0976 .
V12         -0.057588   0.039961  -1.441   0.1502
V17          0.065668   0.041921   1.566   0.1179
V19         -0.084801   0.042270  -2.006   0.0454 *
V24         -0.064758   0.042446  -1.526   0.1278
V26         -0.075418   0.041608  -1.813   0.0705 .
V37         -0.060812   0.042840  -1.419   0.1564
V48         -0.084568   0.040399  -2.093   0.0368 *
V54         -0.102098   0.041580  -2.455   0.0144 *
V62         -0.084401   0.042131  -2.003   0.0457 *
V79          0.059836   0.042391   1.412   0.1587
V80         -0.059121   0.042440  -1.393   0.1642
V91         -0.062161   0.042331  -1.468   0.1426
V92          0.059266   0.041598   1.425   0.1549
V94          0.080883   0.041274   1.960   0.0506 .
V97          0.082363   0.042654   1.931   0.0541 .
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.9374 on 479 degrees of freedom
Multiple R-squared:  0.1089,    Adjusted R-squared:  0.07167
F-statistic: 2.926 on 20 and 479 DF,  p-value: 2.676e-05
```

Using the AIC nominates a model with 20 predictors, and fitting this model produces five "statistically significant" slope coefficients and a very small p-value for the omnibus F-test. Because all of the regression coefficients, including the intercept, are really 0, the true model generating the data is just independent random draws from the standard-normal distribution, $Y_i \sim N(0, 1)$. Thus the "results" obtained in parts (b), (c), and (d) are illusions.

(e) Refitting the models in parts (b), (c), and (d) using new data:

```
> set.seed(34753274)  # for reproducibility
> Data2 <- as.data.frame(matrix(rnorm(500*101), 500, 101))

> mod.2v <- update(mod.2, data=Data2)
> summary(mod.2v)
```



```
Call:
lm(formula = V1 ~ V8 + V26 + V19, data = Data2)

Residuals:
     Min       1Q   Median       3Q      Max
-2.86661 -0.68360  0.04837  0.64533  2.42016

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept) -0.003801   0.044590  -0.085    0.932
V8          -0.030919   0.043813  -0.706    0.481
V26         -0.042188   0.042924  -0.983    0.326
V19         -0.037894   0.046519  -0.815    0.416

Residual standard error: 0.9955 on 496 degrees of freedom
Multiple R-squared:  0.004234,  Adjusted R-squared:  -0.001789
F-statistic: 0.703 on 3 and 496 DF,  p-value: 0.5506

> mod.3v <- update(mod.3, data=Data2)
> summary(mod.3v)

Call:
lm(formula = V1 ~ V8 + V54 + V97, data = Data2)

Residuals:
     Min       1Q   Median       3Q      Max
-2.96913 -0.66961  0.05356  0.68018  2.39876

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept) -0.01174    0.04444  -0.264   0.7918
V8          -0.02946    0.04358  -0.676   0.4994
V54          0.09449    0.04460   2.118   0.0346 *
V97          0.04113    0.04280   0.961   0.3371
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.9917 on 496 degrees of freedom
Multiple R-squared:  0.01181,   Adjusted R-squared:  0.005833
F-statistic: 1.976 on 3 and 496 DF,  p-value: 0.1167

> mod.4v <- update(mod.4, data=Data2)
> summary(mod.4v)

Call:
lm(formula = V1 ~ V3 + V4 + V6 + V8 + V9 + V12 + V17 + V19 +
    V24 + V26 + V37 + V48 + V54 + V62 + V79 + V80 + V91 + V92 +
    V94 + V97, data = Data2)

Residuals:
    Min      1Q  Median      3Q     Max
-2.8875 -0.6954  0.0474  0.6877  2.4615

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept) -0.008795   0.045543  -0.193   0.8469
V3           0.035369   0.044631   0.792   0.4285
V4           0.038398   0.044435   0.864   0.3879
V6           0.010029   0.042441   0.236   0.8133
```



```
V8          -0.036782   0.044270  -0.831   0.4065
V9           0.013851   0.042873   0.323   0.7468
V12         -0.001996   0.044429  -0.045   0.9642
V17         -0.022253   0.044146  -0.504   0.6144
V19         -0.034578   0.047310  -0.731   0.4652
V24         -0.011356   0.047433  -0.239   0.8109
V26         -0.044122   0.043529  -1.014   0.3113
V37         -0.047897   0.046271  -1.035   0.3011
V48          0.049033   0.046126   1.063   0.2883
V54          0.092212   0.045183   2.041   0.0418 *
V62          0.022604   0.042857   0.527   0.5981
V79          0.004610   0.045302   0.102   0.9190
V80          0.005371   0.045500   0.118   0.9061
V91          0.028117   0.044783   0.628   0.5304
V92          0.036477   0.044491   0.820   0.4127
V94         -0.082788   0.045622  -1.815   0.0702 .
V97          0.045992   0.043694   1.053   0.2931
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.9986 on 479 degrees of freedom
Multiple R-squared:  0.03241,   Adjusted R-squared:  -0.007993
F-statistic: 0.8022 on 20 and 479 DF,  p-value: 0.712
```

None of the selected models produce "statistically significant" omnibus F-tests when applied to the validation data, and in the three regressions, only two individual coefficients are "statistically significant." This exercise illustrates how easy it is to produce nonsensical results by model selection without properly accounting for chance "findings."

(f) I'll leave repeating the exercise—which gives a concrete sense of how the detailed results change randomly—to the reader.

Exercise 22.3

We can eliminate the additive constant 1 from $\widetilde{R}^2$ along with the multiplicative constant $(n-1)/\text{TSS}$ without changing the order in which it ranks models. Removing the minus sign reverses the rank order of models, producing the equivalent criterion $\text{RSS}/(n-s)$, where small values are better, and so the only difference in comparison to the GCV is the division by $n-s$ rather than by $(n-s)^2$(along with omission of the constant factor n in the GCV). Because squaring $n-s$ implies a greater relative parsimony penalty for the number of parameters s, the GCV and $\widetilde{R}^2$ need not rank models identically.

Exercise 22.5

I began by reading the baseball-salary data from the website for the text and performing data-management tasks to create the data set used for the example:

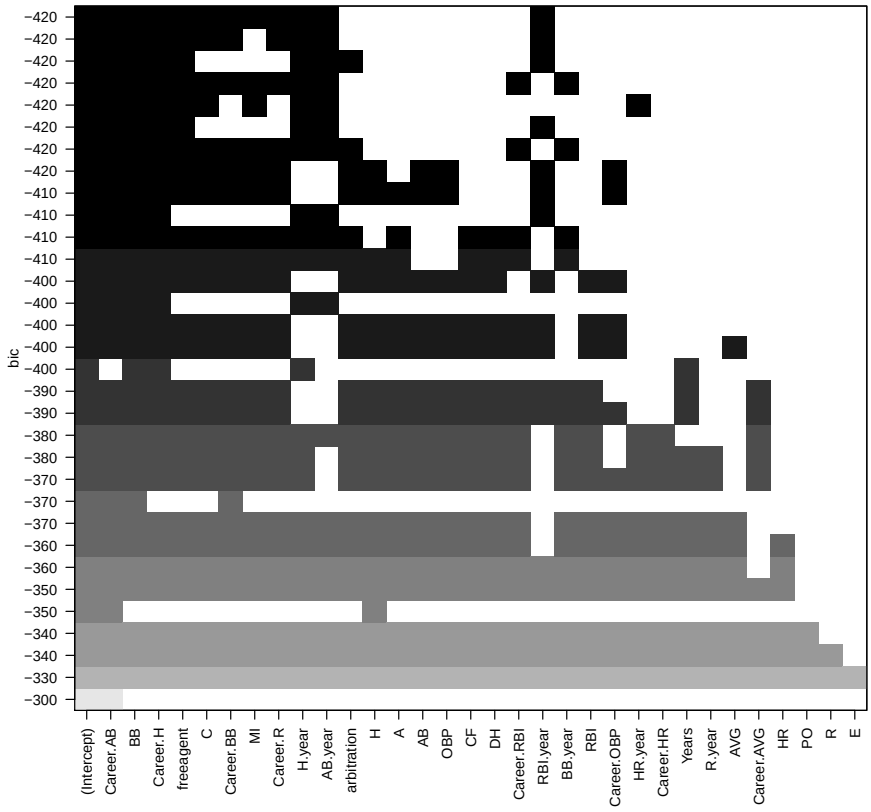
```
> library("car")  # for recode() and subsets()
Loading required package: carData

> url <- paste0(c("https://socialsciences.mcmaster.ca", "jfox", "Books",
+         "Applied-Regression-3E", "datasets", "BaseballHitters.txt"),
+       collapse="/")
> Baseball <- read.table(url, header=TRUE)
>   # data management:
> rownames(Baseball) <- with(Baseball,
+                            paste(firstName, lastName, sep="."))
> Baseball <- Baseball[, -(1:2)] # remove names
>   # remove Pete Rose
> Baseball <- Baseball[-which(rownames(Baseball)=="Pete.Rose"),]
> Baseball <- na.omit(Baseball) # remove cases with missing data
```



```
> Baseball$middle.infield <- recode(Baseball$position,
+   " c('2B', 'SS', '2S') = 1; else=0 ", as.factor=FALSE)
> Baseball$center.field <- recode(Baseball$position,
+   " 'CF' = 1; else=0 ", as.factor=FALSE)
> Baseball$catcher <- recode(Baseball$position,
+    " 'C' = 1; else=0 ", as.factor=FALSE)
> Baseball$dh <-  recode(Baseball$position,
+   " 'DH' = 1; else=0 ", as.factor=FALSE)
> Baseball <- Baseball[, -c(14:17, 22:23)] # remove unused variables
> Baseball$atbats.year <- Baseball$careerAB/Baseball$years
> Baseball$hits.year <- Baseball$careerH/Baseball$years
> Baseball$homers.year <- Baseball$careerHR/Baseball$years
> Baseball$runs.year <- Baseball$careerR/Baseball$years
> Baseball$rbis.year <- Baseball$careerRBI/Baseball$years
> Baseball$walks.year <- Baseball$careerW/Baseball$years
> Baseball$average <- Baseball$H86/Baseball$AB86
> Baseball$career.average <- Baseball$careerH/Baseball$careerAB
> Baseball$onbase <- 100*(Baseball$H86 + Baseball$W86)/
+   (Baseball$AB86 + Baseball$W86)
> Baseball$career.onbase <- 100*
+   (Baseball$careerH + Baseball$careerW)/
+   (Baseball$careerAB + Baseball$careerW)
> Baseball$arbitration <- as.numeric((Baseball$years >= 3) &
+                                    (Baseball$years < 6))
> Baseball$freeagent <- as.numeric(Baseball$years >= 6)
> Baseball$years <- log(Baseball$years)
> Baseball$careerAB <- log(Baseball$careerAB)
> vnames <- scan(what="", text='
+ AB H HR R RBI BB Years Career.AB Career.H
+ Career.HR Career.R Career.RBI
+ Career.BB PO A E salary MI CF C DH
+ AB.year H.year HR.year R.year RBI.year BB.year
+ AVG Career.AVG OBP Career.OBP arbitration freeagent
+ ')
Read 33 items
>   # modify variable names to conform to text
> names(Baseball) <- vnames
>   # order as in Fig. 22.1
> vnames <- scan(what="", text='
+ salary Career.AB BB Career.H freeagent C Career.BB MI Career.R
+ H.year AB.year arbitration H A AB OBP CF DH Career.RBI RBI.year
+ BB.year RBI Career.OBP HR.year Career.HR Years R.year AVG
+ Career.AVG HR PO R E
+ ')
Read 33 items
> Baseball <- Baseball[, vnames]
> dim(Baseball)
[1] 262  33
```

I proceeded to try to reproduce the results summarized in Figure 22.1 (page 683):

```
> library("leaps") # for regsubsets()

> mods <- regsubsets(log(salary) ~ ., data=Baseball, nbest=1, nvmax=32)

>   # reproduce Fig. 22.1: Plot of BIC for best model of each size
> par(cex=0.75) # make text a bit smaller
> plot(mods, digits=3, scale="bic", mar=c(10, 4, 2, 2) + 0.1)
```

The careful reader will notice that this graph is *almost*, but not exactly, the same as the one in the text. Because I used the same R code for the computations as for the text, I can't account for the small discrepancies.

I did these computations for the "best" model of each size (and decided not to bother with the best 10 or 15 regardless of size). Within a *particular* size—that is, number of coefficients—the several model-selection criteria order the models identically, and so agree on which model is best, but the various criteria penalize complexity differently and so need not agree across *different* sizes.

Here is some information about the models nominated by the criteria computed by `regsubsets()`:

```
> sumry.mods <- summary(mods)
> mods.criteria <- with(sumry.mods, cbind(bic, cp, adjr2, rsq))
> which.min(mods.criteria[, "bic"])
[1] 11
> which.min(mods.criteria[, "cp"])
[1] 17
> which.max(mods.criteria[, "adjr2"])
[1] 19
> which.max(mods.criteria[, "rsq"])
[1] 32

> rownames(mods.criteria) <- 1:32
> round(mods.criteria, 4)
         bic       cp  adjr2    rsq
1  -297.7685 249.2568 0.6912 0.6924
2  -348.1138 153.7742 0.7496 0.7515
```



```
3   -366.5561 119.8904 0.7707 0.7733
4   -395.0088  76.3603 0.7978 0.8009
5   -401.4233  63.6805 0.8061 0.8098
6   -410.1768  48.9940 0.8157 0.8199
7   -417.8572  36.3490 0.8241 0.8288
8   -419.1331  31.0687 0.8279 0.8332
9   -417.9075  28.5469 0.8301 0.8360
10  -419.6567  23.0952 0.8341 0.8405
11  -419.7769  19.4445 0.8370 0.8439
12  -418.3749  17.3829 0.8390 0.8464
13  -416.9079  15.4473 0.8408 0.8487
14  -416.2705  12.7965 0.8432 0.8516
15  -412.4925  13.1295 0.8436 0.8526
16  -409.6798  12.5857 0.8446 0.8541
17  -407.3096  11.6668 0.8458 0.8559
18  -403.2350  12.3156 0.8461 0.8567
19  -400.1475  12.0886 0.8469 0.8581
20  -395.6546  13.1297 0.8469 0.8586
21  -390.9770  14.3385 0.8468 0.8591
22  -385.9963  15.8179 0.8465 0.8594
23  -381.0404  17.2766 0.8462 0.8598
24  -376.5576  18.3204 0.8462 0.8603
25  -371.5423  19.8347 0.8459 0.8606
26  -366.2824  21.5643 0.8454 0.8608
27  -360.8946  23.4062 0.8448 0.8609
28  -355.5242  25.2329 0.8443 0.8610
29  -350.0854  27.1196 0.8437 0.8611
30  -344.6027  29.0447 0.8431 0.8611
31  -339.0814  31.0036 0.8424 0.8611
32  -333.5171  33.0000 0.8417 0.8611

> xnames <- mods$xnames
> which.x <- sumry.mods$which

> xnames[which.x[11, ]] # by bic
 [1] "(Intercept)" "Career.AB"   "BB"          "Career.H"
 [5] "freeagent"   "C"           "Career.BB"   "MI"
 [9] "Career.R"    "H.year"      "AB.year"     "RBI.year"

> xnames[which.x[17, ]] # by cp
 [1] "(Intercept)" "Career.AB"   "BB"          "Career.H"
 [5] "freeagent"   "C"           "Career.BB"   "MI"
 [9] "Career.R"    "H.year"      "AB.year"     "arbitration"
[13] "H"           "A"           "CF"          "DH"
[17] "Career.RBI"  "BB.year"

> xnames[which.x[19, ]] # by bic
 [1] "(Intercept)" "Career.AB"   "BB"          "Career.H"
 [5] "freeagent"   "C"           "Career.BB"   "MI"
 [9] "Career.R"    "arbitration" "H"           "A"
[13] "AB"          "OBP"         "CF"          "DH"
[17] "Career.RBI"  "RBI.year"    "RBI"         "Career.OBP"
```

The BIC prefers a more parsimonious model, with 11 predictors (plus the intercept), than C_p, with 17 predictors, or the adjusted $\widetilde{R}^2$, with 19. That said, almost all of the predictors in the model selected by the BIC are also in the models selected by the other criteria. The unadjusted R^2, of course, prefers the largest model, with all 31 predictors.

Exercise 22.7*

Start with Equation 22.12 (page 680), and let $j = 1$ and $j' = 2$. Then

$$2 \times \log \frac{p(\mathbf{y}|M_1)}{p(\mathbf{y}|M_2)} = \text{BIC}_2 - \text{BIC}_1$$

If the prior probabilities for the two models are equal, then the posterior probabilities are proportional to the likelihoods and

$$\begin{aligned} 2 \times \log \frac{p(M_1|\mathbf{y})}{p(M_2|\mathbf{y})} &= \text{BIC}_2 - \text{BIC}_1 \\ \log \frac{p(M_1|\mathbf{y})}{p(M_2|\mathbf{y})} &= \tfrac{1}{2}\text{BIC}_2 - \tfrac{1}{2}\text{BIC}_1 \\ \log \frac{p(M_1|\mathbf{y})}{1 - p(M_1|\mathbf{y})} &= \tfrac{1}{2}\text{BIC}_2 - \tfrac{1}{2}\text{BIC}_1 \end{aligned}$$

We recognize that this is the posterior log-odds (i.e., logit) for models 1 and 2, and we know how to convert a logit into the corresponding probability: $\Pr = e^{\text{logit}}/(1 + e^{\text{logit}})$. Applying this equation,

$$\begin{aligned} p(M_1|\mathbf{y}) &= \frac{\exp(\frac{1}{2}\text{BIC}_2 - \frac{1}{2}\text{BIC}_1)}{1 + \exp(\frac{1}{2}\text{BIC}_2 - \frac{1}{2}\text{BIC}_1)} \\ &= \frac{\exp(\frac{1}{2}\text{BIC}_2)\exp(-\frac{1}{2}\text{BIC}_1)}{1 + \exp(\frac{1}{2}\text{BIC}_2)\exp(-\frac{1}{2}\text{BIC}_1)} \\ &= \frac{\exp(-\frac{1}{2}\text{BIC}_1)}{\frac{1}{\exp(\frac{1}{2}\text{BIC}_2)} + \exp(-\frac{1}{2}\text{BIC}_1)} \\ &= \frac{\exp(-\frac{1}{2}\text{BIC}_1)}{\exp(-\frac{1}{2}\text{BIC}_2) + \exp(-\frac{1}{2}\text{BIC}_1)} \end{aligned}$$

We can extend this result to m models $\{M_1, \ldots, M_m\}$ by using a set of logits (as in Section 14.2.1 on polytomous logistic regression), comparing each model to an arbitrarily selected model (say, the last). We then have

$$\text{logit}_j \equiv \log \frac{p(M_j|\mathbf{y})}{p(M_m|\mathbf{y})} = \tfrac{1}{2}\text{BIC}_m - \tfrac{1}{2}\text{BIC}_j$$

and

$$p(M_j|\mathbf{y}) = \frac{\text{logit}_j}{1 + \sum_{j'=1}^{m-1} \text{logit}_{j'}}$$

which (in very much the same manner as the two-model case considered above) simplifies to

$$p(M_j|\mathbf{y}) = \frac{\exp(-\frac{1}{2}\text{BIC}_j)}{\sum_{j'=1}^{m} \exp(-\frac{1}{2}\text{BIC}_{j'})}$$

Exercises for Chapter 23

Exercise 23.1

Referring to Equations 23.9 and 23.10, the fitted fixed-effects part of the model is

$$\widehat{\text{mathach}} = (\widehat{\beta}_1 + \widehat{\beta}_2 \times \overline{\text{ses}} + \widehat{\beta}_3 \text{sector}) \\ (\widehat{\beta}_4 + \widehat{\beta}_5 \times \overline{\text{ses}} + \widehat{\beta}_6 \times \overline{\text{ses}}^2 + \widehat{\beta}_7 \times \text{sector}) \times \text{cses}$$

with the terms rearranged to show the intercept and slope for cses at fixed values of $\overline{\text{ses}}$ (school-mean SES) and sector (coded 0 for Public and 1 for Catholic).

To get the equations of the six regression lines shown in Figure 23.6, we can just use the values of the $\widehat{\beta}$s in the table on page 715, along with the six combinations of values 0 and 1 for sector and -0.7 (low), 0 (medium), and 0.7 (high) for $\overline{\text{ses}}$. It's perfectly possible to do these computations on a calculator, but it was more convenient for me to do them in R:

```
> beta <- c(12.128, 5.337, 1.225, 3.140, 0.755, -1.647, -1.516)

> fixedEffects <- function(beta, meanses, sector){
+   intercept <- beta[1] + beta[2]*meanses + beta[3]*sector
+   slope <- beta[4] + beta[5]*meanses + beta[6]*meanses^2 +
+     beta[7]*sector
+   c(intercept=intercept, slope=slope)
+ }

> fixedEffects(beta, meanses=-0.7, sector=0) # Public, low mean SES
intercept     slope
  8.39210   1.80447

> fixedEffects(beta, meanses=0, sector=0) # Public, medium mean SES
intercept     slope
   12.128     3.140

> fixedEffects(beta, meanses=0.7, sector=0) # Public, low mean SES
intercept     slope
 15.86390   2.86147

> fixedEffects(beta, meanses=-0.7, sector=1) # Catholic, low mean SES
intercept     slope
  9.61710   0.28847

> fixedEffects(beta, meanses=0, sector=1) # Catholic, medium mean SES
intercept     slope
   13.353     1.624

> fixedEffects(beta, meanses=0.7, sector=1) # Catholic, low mean SES
intercept     slope
 17.08890   1.34547
```

These intercepts and slopes are consistent with the lines drawn in Figure 23.6.

Exercise 23.3*

Following the hint and thinking about AV plots, in Model 1 the coefficient $\widehat{\beta}_2^{(1)}$ is formed from two sets of residuals: (1) residuals $E_Y^{(1)}$ from the regression of Y_{ij} on the j dummy variables for groups (and no intercept); and (2) $E_X^{(1)}$ from the regression of $X_{ij} - \bar{X}_{i\cdot}$ on the j dummy variables for groups. The fitted values for these regressions are just the group means, $\bar{Y}_{i\cdot}$ in the first case and 0 in the second case, so the two sets of residuals are respectively $Y_{ij} - \bar{Y}_{i\cdot}$ and $X_{ij} - \bar{X}_{i\cdot}$, and $\widehat{\beta}_2^{(1)}$ is the least-squares

coefficient from the simple regression of the first set of residuals on the second (with no intercept, because both sets of residuals have means of 0).

Now consider Model 5. The residuals $E_Y^{(5)}$ for the regression of Y_{ij} on $\bar{X}_{i\cdot}$ and an intercept are *not* the same as the residuals $E_Y^{(1)}$. In contrast, the residuals $E_X^{(5)}$ for the regression of $X_{ij} - \bar{X}_{i\cdot}$, on $\bar{X}_{i\cdot}$ and an intercept are still the same as $E_X^{(1)}$; that is, the fitted values for this regression are still all 0 and thus the residuals are still just $X_{ij} - \bar{X}_{i\cdot}$. Moreover, the difference $E_Y^{(5)} - E_Y^{(1)}$ is orthogonal to $X_{ij} - \bar{X}_{i\cdot}$, and so the coefficient $\widehat{\beta}_2^{(5)}$ for the regression of $E_Y^{(5)}$ on $E_X^{(5)}$ is the same as $\widehat{\beta}_2^{(1)}$ for the regression of $E_Y^{(1)}$ on $E_X^{(1)}$.

Exercise 23.5*

The simplest way I know to derive these estimating equations is to use a strategy from Stroup (2013, Section 4.4) that decomposes the probability density for $\mathbf{y}$ as $p(\mathbf{y}) = p(\mathbf{y}|\boldsymbol{\delta}) \times p(\boldsymbol{\delta})$. The response $\mathbf{y}$ conditional on the random effects $\boldsymbol{\delta}$ is multivariate normal with mean vector $\mathbf{X}\boldsymbol{\beta} + \mathbf{Z}\boldsymbol{\delta}$ and covariance matrix $\sigma_\varepsilon^2\boldsymbol{\Lambda}$, while $\boldsymbol{\delta}$ is multivariate normal with mean vector $\mathbf{0}$ and covariance matrix $\boldsymbol{\Psi}^*$. Using the formula for the multivariate-normal density (see on-line Appendix D, Section D.3.5),

$$
\begin{aligned}
p(\mathbf{y}) &= p(\mathbf{y}|\boldsymbol{\delta}) \times p(\boldsymbol{\delta}) \\
&= \frac{1}{(2\pi)^{n/2}\sqrt{\sigma_\varepsilon^2 \det \boldsymbol{\Lambda}}} \exp\left[-\tfrac{1}{2}(\mathbf{y} - \mathbf{X}\boldsymbol{\beta} - \mathbf{Z}\boldsymbol{\delta})' \frac{1}{\sigma_\varepsilon^2}\boldsymbol{\Lambda}^{-1}(\mathbf{y} - \mathbf{X}\boldsymbol{\beta} - \mathbf{Z}\boldsymbol{\delta})\right] \\
&\quad \times \frac{1}{(2\pi)^{mq/2}\sqrt{\det \boldsymbol{\Psi}^*}} \exp\left(-\tfrac{1}{2}\boldsymbol{\delta}'\boldsymbol{\Psi}^{*-1}\boldsymbol{\delta}\right)
\end{aligned}
$$

The next step is to convert this density to a log-likelihood for $\boldsymbol{\beta}$ and $\boldsymbol{\delta}$. We know that we can ignore the constants $1/(2\pi)^{n/2}$ and $1/(2\pi)^{mq/2}$. Our experience with least-squares and generalized-least-squares regression tells us that for purposes of maximizing the likelihood, we can also ignore the terms involving the determinants of the covariance matrices and simply concentrate on the exponents. Then,

$$
\log_e L(\boldsymbol{\beta}, \boldsymbol{\delta}) = -\tfrac{1}{2}(\mathbf{y} - \mathbf{X}\boldsymbol{\beta} - \mathbf{Z}\boldsymbol{\delta})' \frac{1}{\sigma_\varepsilon^2}\boldsymbol{\Lambda}^{-1}(\mathbf{y} - \mathbf{X}\boldsymbol{\beta} - \mathbf{Z}\boldsymbol{\delta}) - \tfrac{1}{2}\boldsymbol{\delta}'\boldsymbol{\Psi}^{*-1}\boldsymbol{\delta}
$$

Differentiating the log-likelihood with respect to $\boldsymbol{\beta}$ and $\boldsymbol{\delta}$,

$$
\begin{aligned}
\frac{\partial \log_e L(\boldsymbol{\beta}, \boldsymbol{\delta})}{\partial \boldsymbol{\beta}} &= \frac{2}{2\sigma_\varepsilon^2}\left[-\mathbf{X}'\boldsymbol{\Lambda}^{-1}(\mathbf{y} - \mathbf{X}\boldsymbol{\beta} - \mathbf{Z}\boldsymbol{\delta})\right] \\
\frac{\partial \log_e L(\boldsymbol{\beta}, \boldsymbol{\delta})}{\partial \boldsymbol{\delta}} &= \frac{2}{2\sigma_\varepsilon^2}\mathbf{Z}'\boldsymbol{\Lambda}^{-1}(\mathbf{y} - \mathbf{X}\boldsymbol{\beta} - \mathbf{Z}\boldsymbol{\delta}) - \frac{2}{2}\boldsymbol{\Psi}^{*-1}\boldsymbol{\delta}
\end{aligned}
$$

Setting the derivatives to $\mathbf{0}$ produces estimating equations for $\boldsymbol{\beta}$ and $\boldsymbol{\delta}$. We can eliminate σ_ε^2 from the first such equation, obtaining after rearrangement

$$
\mathbf{X}'\boldsymbol{\Lambda}^{-1}\mathbf{X}\widehat{\boldsymbol{\beta}} + \mathbf{X}'\boldsymbol{\Lambda}^{-1}\mathbf{Z}\widehat{\boldsymbol{\delta}} = \mathbf{X}'\boldsymbol{\Lambda}^{-1}\mathbf{y}
$$

We can simplify the second equation a bit by multiplying it by σ_ε^2, to get, again after rearrangement,

$$
\mathbf{Z}'\boldsymbol{\Lambda}^{-1}\mathbf{X}\widehat{\boldsymbol{\beta}} + \mathbf{Z}'\boldsymbol{\Lambda}^{-1}\mathbf{Z}\widehat{\boldsymbol{\delta}} + \sigma_\varepsilon^2\boldsymbol{\Psi}^{*-1}\widehat{\boldsymbol{\delta}} = \mathbf{Z}'\boldsymbol{\Lambda}^{-1}\mathbf{y}
$$

To verify that these two estimating are equivalent to the partitioned-matrix Equation 23.19 in the text, simply multiply out the latter.

The remainder of this exercise is very tedious, and I should not have casually posed the question as I did. The details are given, for example, by Stroup (2013) (though with different notation). The result, however, is of interest, because the formula $\widehat{\boldsymbol{\beta}} = (\mathbf{X}'\boldsymbol{\Theta}^{-1}\mathbf{X})^{-1}\mathbf{X}'\boldsymbol{\Theta}^{-1}\mathbf{y}$ for the fixed effects explains why this is called the GLS estimator. That is, the weight matrix $\boldsymbol{\Theta}^{-1}$ is the inverse of the covariance matrix of $\mathbf{y}$, as shown in Exercise 23.4.

Exercises for Chapter 24

Exercise 24.1

(a) The graph of fixed effects in Figure 24.1 is based on the simplified model with parameter estimates given in the table on page 747:

$$\widehat{\text{logit}(\pi_{ij})} = \widehat{\beta}_1 + \widehat{\beta}_2 M_{1i} + \widehat{\beta}_3 M_{2i} + \widehat{\beta}_4 P_{ij} + \widehat{\beta}_6 \sqrt{T_{1ij}}$$

where M_1 is a dummy regressor coded 1 for subjects in the reduced medication group and 0 otherwise; M_2 is a dummy regressor coded 1 for subjects in the continuing medication group and 0 otherwise; P is a dummy regressor coded 1 posttreatment and 0 before; and T_1 is time (in days) post-treatment, coded 0 during the pretreatment period.

To produce Figure 24.1, let time run from -29 to 99 days, corresponding to values of T_1 from 0 (for time ≤ 0) through 99 (for time > 0), and values of P of 0 (for time ≤ 0) and 1 (for time > 0); let the dummy regressors M_1 and M_2 respectively take on the values 0 and 0 for the no-medication group, 1 and 0 for the reduced-medication group, and 0 and 1 for the continuing medication group. Substitute these combinations of values for the regressors into the fitted fixed-effects equation to compute the fitted logit in each case. Finally, convert the logits to fitted probabilities in the usual manner, $\widehat{\Pr(\text{headache})} = 1/[1 + \exp(-\text{logit})]$.

(b) After reading the data set from the website for the text, I do some data management, and then fit the model reported in Figure 24.1 and the table on page 747 in the text, using the `glmer()` function in the **lme4** package for R:

```
> url <- paste(c("https://socialsciences.mcmaster.ca",
+                "jfox", "Books",
+                "Applied-Regression-3E", "datasets",
+                "Migraines.txt"),
+              collapse="/")
> Migraines <- read.table(url, header=TRUE, stringsAsFactors=TRUE)

>   # data management:

> Migraines$treatment <- factor(with(Migraines,
+                       ifelse(time > 0, "yes", "no")))
> Migraines$pretreat <- with(Migraines, ifelse(time > 0, 0, time))
> Migraines$posttreat <- with(Migraines, ifelse(time > 0, time, 0))
>     # reorder levels
> Migraines$medication <- factor(Migraines$medication,
+                       levels=c("None", "Reduced", "Continuing"))

> library("car") # for brief()
Loading required package: carData

> brief(Migraines, c(25, 10))
4152 x 7 data.frame (4117 rows omitted)
      id headache time medication treatment pretreat posttreat
     [i]      [f]  [i]        [f]       [f]      [n]       [n]
1     1      yes  -11 Continuing       no       -11         0
2     1      yes  -10 Continuing       no       -10         0
3     1      yes   -9 Continuing       no        -9         0
4     1      yes   -8 Continuing       no        -8         0
5     1      yes   -7 Continuing       no        -7         0
6     1      yes   -6 Continuing       no        -6         0
7     1      yes   -5 Continuing       no        -5         0
8     1      yes   22 Continuing       yes        0        22
9     1      yes   23 Continuing       yes        0        23
```



```
10      1      yes   24 Continuing       yes        0        24
11      1      yes   25 Continuing       yes        0        25
12      1      yes   26 Continuing       yes        0        26
13      1      no    28 Continuing       yes        0        28
14      1      yes   29 Continuing       yes        0        29
15      1      yes   30 Continuing       yes        0        30
16      1      yes   31 Continuing       yes        0        31
17      1      yes   32 Continuing       yes        0        32
18      1      yes   33 Continuing       yes        0        33
19      1      yes   34 Continuing       yes        0        34
20      1      yes   35 Continuing       yes        0        35
21      2      yes    1 Continuing       yes        0         1
22      2      yes    2 Continuing       yes        0         2
23      2      no     3 Continuing       yes        0         3
24      2      yes    4 Continuing       yes        0         4
25      2      yes    5 Continuing       yes        0         5
. . .
4143 133       yes   37 Continuing       yes        0        37
4144 133       yes   40 Continuing       yes        0        40
4145 133       yes   41 Continuing       yes        0        41
4146 133       no    42 Continuing       yes        0        42
4147 133       yes   43 Continuing       yes        0        43
4148 133       yes   44 Continuing       yes        0        44
4149 133       no    45 Continuing       yes        0        45
4150 133       no    46 Continuing       yes        0        46
4151 133       no    47 Continuing       yes        0        47
4152 133       yes   48 Continuing       yes        0        48

> library("lme4") # for glmer()

> m.mig.1 <- glmer(headache ~ medication + treatment + sqrt(posttreat)
+                  + (1 + treatment  + sqrt(posttreat) | id),
+                  data=Migraines, family=binomial,
+                  control=glmerControl(optimizer="nlminbwrap"))
> summary(m.mig.1)
Generalized linear mixed model fit by maximum likelihood (Laplace
  Approximation) [glmerMod]
 Family: binomial  ( logit )
Formula:
headache ~ medication + treatment + sqrt(posttreat) + (1 + treatment +
    sqrt(posttreat) | id)
   Data: Migraines
Control: glmerControl(optimizer = "nlminbwrap")

     AIC      BIC   logLik deviance df.resid
  4369.0   4438.6  -2173.5   4347.0     4141

Scaled residuals:
    Min      1Q  Median      3Q     Max
-5.1816 -0.6463  0.2600  0.5801  3.6904

Random effects:
 Groups Name             Variance Std.Dev. Corr
 id     (Intercept)      1.70114  1.3043
        treatmentyes     1.71268  1.3087   -0.12
        sqrt(posttreat) 0.05708  0.2389    0.11 -0.66
Number of obs: 4152, groups:  id, 133

Fixed effects:
```



```
                    Estimate Std. Error z value Pr(>|z|)
(Intercept)          -0.24585    0.34377  -0.715  0.47451
medicationReduced     2.05011    0.46784   4.382 1.18e-05 ***
medicationContinuing  1.15530    0.38372   3.011  0.00261 **
treatmentyes          1.06085    0.24386   4.350 1.36e-05 ***
sqrt(posttreat)      -0.26844    0.04486  -5.984 2.18e-09 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Correlation of Fixed Effects:
            (Intr) mdctnR mdctnC trtmnt
medictnRdcd -0.674
mdctnCntnng -0.828  0.656
treatmentys -0.215 -0.053 -0.049
sqrt(psttr)  0.016 -0.009 -0.002 -0.685
```

These results are *almost* identical to those in the text, with some parameters differing in the last decimal place reported in the output. The very small discrepancies are almost surely due to changes in the **lme4** package since the computations were done for the text; to get the computation to converge, I had to use a non-default optimizer for the `glmer()` function, which wasn't originally necessary. That suggests that the random effects are nearly too complicated to be reliably estimated from the data.

Indeed, when I proceeded to specify an alternative model using a natural regression spline with 4 degrees of freedom, I found that I couldn't get the model to converge. This model is more complex than the model fit in the text, which uses 1 degree of freedom for time posttreatment, adding many additional covariance components. I therefore eliminated the spline term from the random effects, fitting the following GLMM:

```
> library("splines")  # for ns()
>
> m.mig.2 <- glmer(headache ~ medication + treatment +
+                    ns(posttreat, df=4)
+                  + (1 + treatment | id),
+                  data=Migraines, family=binomial,
+                  control=glmerControl(optimizer="nlminbwrap"))
> summary(m.mig.2)
Generalized linear mixed model fit by maximum likelihood (Laplace
  Approximation) [glmerMod]
 Family: binomial  ( logit )
Formula:
headache ~ medication + treatment + ns(posttreat, df = 4) + (1 +
    treatment | id)
   Data: Migraines
Control: glmerControl(optimizer = "nlminbwrap")

     AIC      BIC   logLik deviance df.resid
  4382.3   4451.9  -2180.1   4360.3     4141

Scaled residuals:
    Min      1Q  Median      3Q     Max
-6.6486 -0.6627  0.2640  0.6017  3.4805

Random effects:
 Groups Name          Variance Std.Dev. Corr
 id     (Intercept)  1.7305   1.3155
        treatmentyes 0.8882   0.9424   -0.05
Number of obs: 4152, groups:  id, 133
```



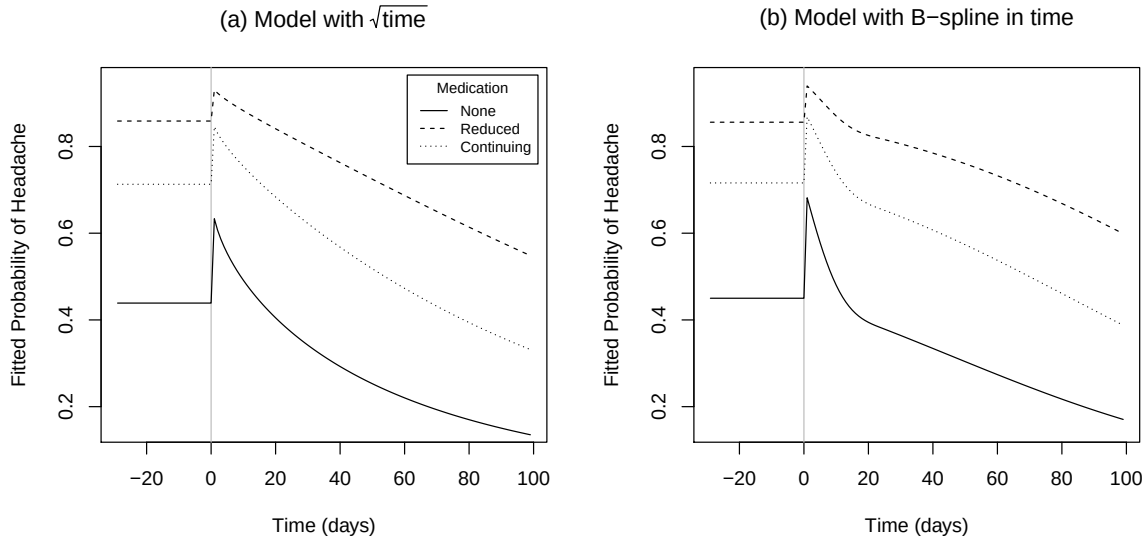
```
Fixed effects:
                        Estimate Std. Error z value Pr(>|z|)
(Intercept)              -0.2006     0.3431  -0.585 0.558695
medicationReduced         1.9826     0.4636   4.277 1.90e-05 ***
medicationContinuing      1.1249     0.3845   2.925 0.003439 **
treatmentyes              1.0849     0.3143   3.452 0.000556 ***
ns(posttreat, df = 4)1   -1.2369     0.2616  -4.728 2.26e-06 ***
ns(posttreat, df = 4)2   -1.3702     0.3054  -4.486 7.25e-06 ***
ns(posttreat, df = 4)3   -2.5820     0.6315  -4.089 4.34e-05 ***
ns(posttreat, df = 4)4   -2.1902     0.3745  -5.848 4.98e-09 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Correlation of Fixed Effects:
            (Intr) mdctnR mdctnC trtmnt n(,d=4)1 n(,d=4)2 n(,d=4)3
medictnRdcd -0.675
mdctnCntnng -0.828  0.652
treatmentys -0.153 -0.037 -0.042
ns(ps,d=4)1 -0.001 -0.002  0.004 -0.756
ns(ps,d=4)2  0.003 -0.017 -0.001 -0.460  0.285
ns(ps,d=4)3 -0.003 -0.009  0.008 -0.789  0.753    0.585
ns(ps,d=4)4 -0.021 -0.001  0.034 -0.213  0.328   -0.075    0.393
```

Finally, I plot the fixed effects from the two models, with the graph on left reproducing Figure 24.1 in the text:

```
>   # create graphs for fixed effects

> new.1 <- expand.grid(treatment="yes", posttreat=1:99,
+                      medication=c("Reduced", "Continuing", "None"))
> new.1$treatment <- factor("yes", levels=c("no", "yes"))
> new.2 <- expand.grid(treatment="no", posttreat=-29:0,
+                      medication=c("Reduced", "Continuing", "None"))
> new.2$posttreat <- 0
> new.2$treatment <- factor("no", levels=c("no", "yes"))
> new <- rbind(new.2, new.1)
> new$medication <- factor(new$medication,
+                          levels=c("None", "Reduced", "Continuing"))
> new$time <- c(rep(-29:0, 3),rep(1:99, 3))
> brief(new)
387 x 4 data.frame (382 rows omitted)
    treatment posttreat medication time
          [f]       [n]        [f]  [i]
1         no          0    Reduced  -29
2         no          0    Reduced  -28
3         no          0    Reduced  -27
. . .
386       yes        98    None      98
387       yes        99    None      99

> new$fit.1 <- predict(m.mig.1, newdata=new, re.form=NA)
> new$p.1 <- 1/(1 + exp(-new$fit.1))

> new$fit.2 <- predict(m.mig.2, newdata=new, re.form=NA)
> new$p.2 <- 1/(1 + exp(-new$fit.2))

> par(mfrow=c(1, 2))

> plot(p.1 ~ time, type="n", data=new, ylim=c(.15, .95),
```



```
+      xlab="Time (days)", ylab="Fitted Probability of Headache",
+      main=expression("(a) Model with" ~ sqrt(time)))
> abline(v=0, col="gray")
> lines(p.1 ~ time, subset = medication == "None", data=new, lty=1)
> lines(p.1 ~ time, subset = medication == "Reduced", data=new, lty=2)
> lines(p.1 ~ time, subset = medication == "Continuing", data=new, lty=3)
> legend("topright", lty=1:3, legend=c("None", "Reduced", "Continuing"),
+        title="Medication", inset=.02, cex=0.75)

> plot(p.2 ~ time, type="n", data=new, ylim=c(.15, .95),
+      xlab="Time (days)", ylab="Fitted Probability of Headache",
+      main=expression("(b) Model with B-spline in time"))
> abline(v=0, col="gray")
> lines(p.2 ~ time, subset = medication == "None", data=new, lty=1)
> lines(p.2 ~ time, subset = medication == "Reduced", data=new, lty=2)
> lines(p.2 ~ time, subset = medication == "Continuing", data=new, lty=3)
```



The two fits are very similar.

References

A. C. Atkinson. *Plots, Transformations, and Regression: An Introduction to Graphical Methods of Diagnostic Regrtession Analysis*. Clarendon Press, Oxford, 1985.

J. Fox and S. Weisberg. *An R Companion to Applied Regression*. Sage, Thousand Oaks CA, third edition, 2019.

W. Greene. Accounting for excess zeros and sample selection in Poisson and negative binomial regression models. Working papers, New York University, Leonard N. Stern School of Business, Department of Economics, 1994. URL https://EconPapers.repec.org/RePEc:ste:nystbu:94-10.

J. Kmenta. *Elements of Econometrics*. Macmillan, New York, second edition, 1986.

R Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria, 2021. URL https://www.R-project.org/.

P. J. Ribeiro Jr, P. J. Diggle, M. Schlather, R. Bivand, and B. Ripley. *geoR: Analysis of Geostatistical Data*, 2020. URL https://CRAN.R-project.org/package=geoR. R package version 1.8-1.

W. W. Stroup. *Generalized Linear Mixed Models: Modern Concepts, Methods and Applications*. CRC Press, Boca Raton FL, 2013.

R. J. Wonnacott and T. H. Wonnacott. *Econometrics*. Wiley, New York, second edition, 1979.

